



Information and Communication Technology G.C.E. (A/L) – Grade 13

Resource Book for Teachers

Department of Information Technology
National Institute of Education
Maharagama
Sri Lanka

www.nie.lk

Information and Communication Technology
Grade 13 –Resource Book for Teachers’

©National Institute of Education
First Print

Department of Information Technology
Faculty of Science and Technology
National Institute of Education
Maharagama

www.nie.lk

Printed by: Printing Department
National Institute of Education

Message from the Director General

The National Institute of Education takes opportune steps from time to time for the development of quality in education. Preparation of supplementary books for respective subjects is one such initiative.

The additional reading book has been composed by the National Institute of Education to implement grade 13 syllabus and Teachers' Guide successfully in the classroom.

It is our behalf that provision of essential staff relating to respective syllabus by this additional book will facilitate learning the relevant subject stream.

I wish to express my gratitude to NIE staff and external experts who made their academic contribution to make this material available to you.

Dr. Sunil Jayantha Nawaratne
Director General
National Institute of Education
Maharagama.

Message from the Deputy Director General

Education from the past has been constantly changing and forging forward. In recent years, these changes have become quite rapid. The Past two decades have witnessed a high surge in teaching methodologies as well as in the use of technological tools and in the field of knowledge creation.

Accordingly, the National Institute of Education is in the process of taking appropriate and timely steps with regard to the education reforms of 2015.

It is with immense pleasure that this Resource Book where the new curriculum has been planned based on a thorough study of the changes that have taken place in the global context adopted in terms of local needs based on a student-centered learning-teaching approach, is presented to you teachers who serve as the pilots of the schools system.

An instructional manual of this nature is provided to you with the confidence that, you will be able to make a greater contribution using this.

I trust that through the careful study of this Resource Book provided to you, you will act with commitment in the generation of a greatly creative set of students capable of helping Sri Lanka move socially as well as economically forward.

This Resource Book is the outcome of the expertise and unflagging commitment of a team of subject teachers and academics in the field Education.

While expressing my sincere appreciation for this task performed for the development of the education system, my heartfelt thanks go to all of you who contributed your knowledge and skills in making this document such a landmark in the field.

K. R. Pathmasiri

Deputy Director General

Faculty of Science and Technology

Message of the Director

The subject of Information and Communication Technology (ICT) is fast changing; revising subject and it is applying in all subject areas. The subject of Information and Communication Technology is applying for all subject areas such as engineering science, medicine science, economics, accounting, mathematics, chemistry, physics, music, dancing and art. New applications, new technological methods have added for each subject area. Therefore, everyone should have earned certain level of literacy of Information and Communication Technology.

In such situation, there are lots of job opportunities for people who have broad knowledge of Information and Communication Technology. There is a lack to have such amount of ICT personals. It is expected that students focus to learn this subject also helpful to increase the number of such experts.

Several subject areas were added to the syllabus when the syllabus revised recently. Specially, inserting of new application of Information and Communication Technology, IOT – Internet of Things is a remarkable revision. Operations of information was exists in the Internet and now operations of things are handle though Internet. Such IOT area was included with practical activities to the syllabus and this resource book. It was great risk because all teachers had to train with practical activities. That teacher training was successfully completed by the Department of IT and practical activities for that unit have included in this book with explanations.

Most of theoretical concepts have explained clearly in this book. Practical activities with explanations were added to this resource book, instead of adding over loaded theoretical contents. Completion of these activities in this book will helpful to learn the theoretical contents easily. Recently, new ICT syllabi were added from grade 6 to 9 students. Therefore, some revisions have to be involved for ICT syllabi from grade 10 to 13. It is expected to add more attractive practical activities, when it start the new syllabus revision since 2020

My gratitude thanks was given who have successfully involved completing this resource book. There were teams of resource persons, language editors and staff of the Department of IT. I thank all of them for their devotion to success the preparing this resource book.

D. Anura Jayalal
Director
Department of Information Technology
National Institute of Education

**Curriculum Committee
Guidance and Approval**

Subject Coordinator
Mr.S.Shanmugalingam

Resource Contributions
Mr. D.Anura Jayalal

Mr.S.Shanmugalingam

Dr. Keerthi Wijayasiriwardhane

Dr. Chathura Rajapakshe

Dr.K.Thabotharan

Dr. P.M.T.P. Sandirigama

Mr.S.Sarveswaran

Mr.S.Jayakanth

Ms. A.P.N.De Silva

Ms.P.H.Sirani

Academic Affairs Board
National Institute of Education

Senior Lecturer
Department of IT
National Institute of Education

Director
Department of IT
National Institute of Education

Senior Lecturer
Department of IT
National Institute of Education

Senior Lecturer
Faculty of Science
University of Kelaniya

Senior Lecturer
Faculty of Science
University of Kelaniya

Senior Lecturer
Faculty of Science
University of Jaffna

Senior Lecturer
Faculty of Engineering
University of Peradeniya

Principal
J/Vaddu Hindu College,
Jaffna

Computer Instructor
Zonal Computer Resource Centre
HZ/Ginigathena Central College,
Ginigathena

Teacher
MR/Godapitiya Maha Vidyalaya
Akuressa

Teacher
MR/Athuraliya Maha Vidyalaya
Akuressa

Content	Page No
Message from the Director General	ii
Message from the Deputy Director General	iii
Message of the Director	iv
Curriculum Committee	v
Contents	vi
Embedded Systems	1
Python Programming	31
Website Development	79
E-Commerce	126
Trends and Future Directions of ICT	141
References	150
Glossary	153

EMBEDDED SYSTEM AND INTERNET OF THINGS

Competency 11:

Explores IoT and

Identify the building blocks of embedded systems to develop simple applications.

Competency Level 11.1: Acquires the knowledge of basic building blocks of embedded systems

Learning Outcomes:

- Identifies and lists microcontroller based development boards
- Describes available features on a microcontroller based development board
- Identifies necessary software and download them from the Internet to develop a microcontroller based embedded system
- Develops simple applications using a microcontroller based development board
 - Switch a LED on/off on ambient light intensity
 - Run a fan on room temperature
 - Detect door opening /closing using a read switch

EMBEDDED SYSTEM

Embedded system is a computer system embedded into some other system such as a refrigerator, washing machine, car, etc. It also follows Input, Process and Output (IPO) model. Sensors capture the state of the physical world like heat, speed and light as inputs. Processor processes them according to a program and produces outputs. Outputs drive actuators and change the state like heat, speed and light of the physical world. Embedded systems, therefore, do physical computing.

IEEE Definition for EMBEDDED SYSTEM

“A computer system that is part of a larger system and performs some of the requirements of that system; for example, a computer system used in an aircraft or rapid transit system.”
[IEEE, 1992]

Figure 1 and 2 show the components of an embedded system.

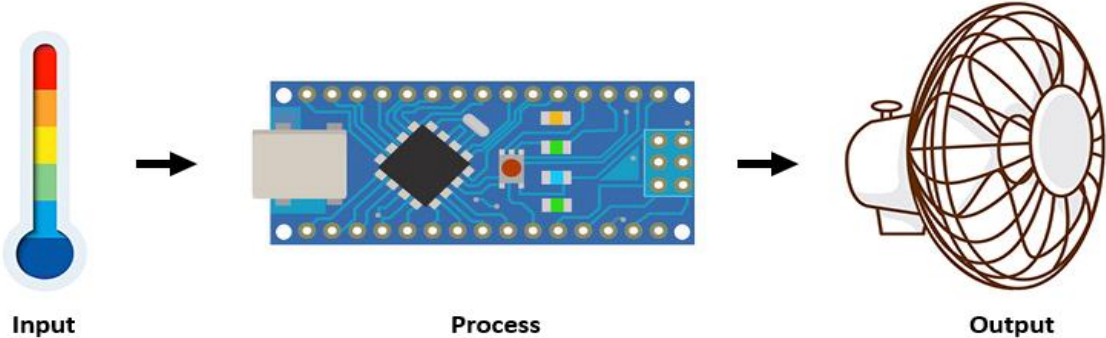


Figure 1: Embedded system model of physical world

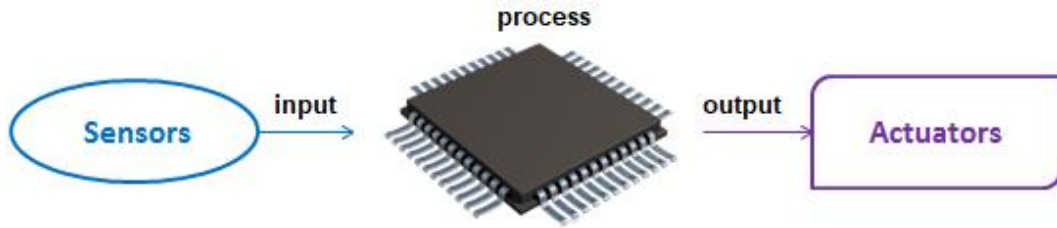


Figure2: Components of an Embedded System

Embedded systems are developed using either microprocessors optimized for the purpose called embedded processors or microcontrollers. The microcontroller is a single chip containing a CPU, memory, I/O ports, and other peripherals. In microprocessor-based embedded systems, except CPU, other components are external to the microprocessor chip. A majority of embedded systems are microcontroller based because they do not require expensive, powerful microprocessors to implement their basic functionalities.

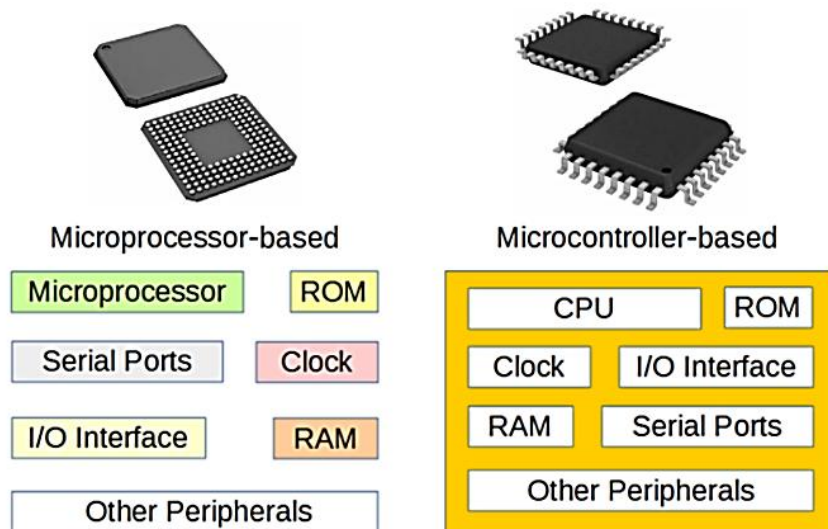


Figure3: Microprocessor-based and Microcontroller-based **Hardware and Software for Embedded System**

Embedded system development requires the knowledge of both hardware and software components. They include a microcontroller-based development board, sensors, actuators, and an Integrated Development Environment (IDE).

Microcontroller-based Development Boards

There are a number of microcontroller-based development boards such as Arduino, micro: bit, etc.

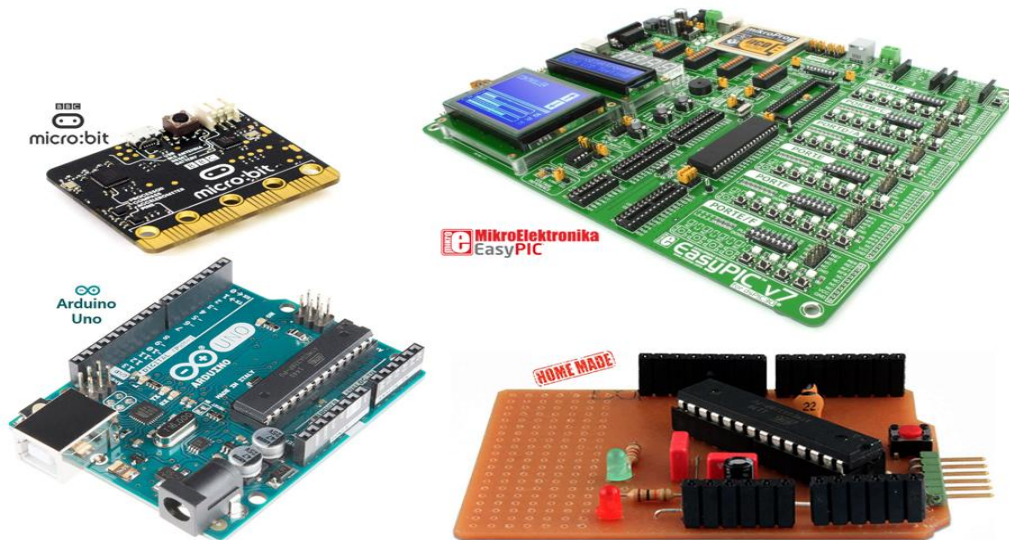


Figure 4: Micro-based Development Boards

There are different microcontroller-based embedded system development platforms. Among them, we use Arduino platform due the following reasons:

- Free and open source hardware designs and software tool chain
- Cross platform support
- Extensive official and community support
- Extensive availability of software libraries
- Extensive hardware extendibility with extension shields
- Extreme user friendliness due to the use of wrapper functions (basic programming knowledge with any programming language is sufficient)

Arduino

Arduino is an open-source, low cost, easy-to-use hardware, and software platform. There are different kinds of Arduino boards available. They can be chosen according to our requirements and affordability.

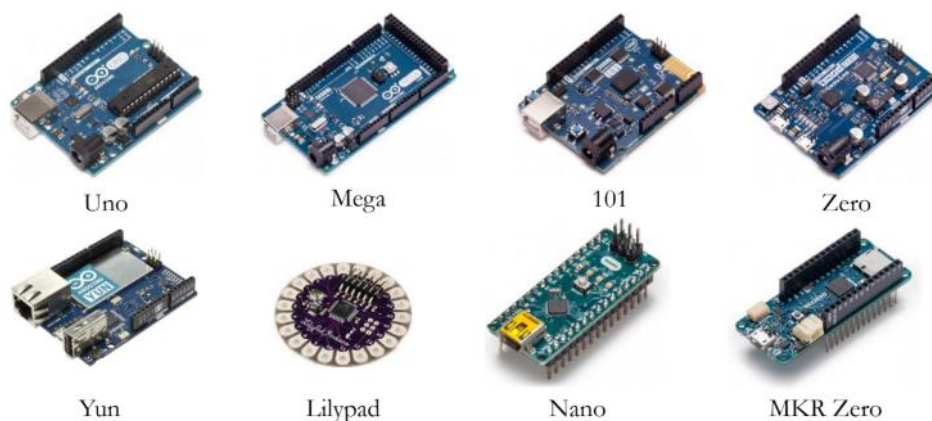


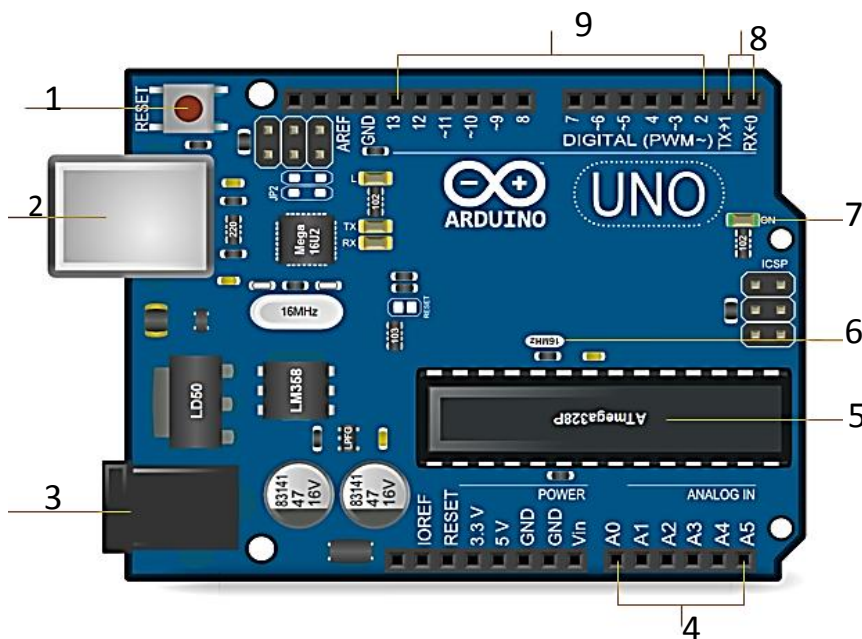
Figure5: Arduino Boards

Among the different Arduino boards available, we use Arduino Uno due to the following reasons:

- Inexpensive
- Most widely used, heavily documented, extensively library supported development board
- 493 extension shield support
- 5V compatible I/O ports (most sensors and actuators use 5V)
- Do It Yourself (DIY) supported hardware design (using through hole electronic components)

Arduino Uno Board

Figure 6 shows the features of the Arduino Uno microcontroller-based development board.



- | | | | |
|---|-------------------|---|-----------------------------|
| 1 | Reset button | 6 | Oscillator |
| 2 | USB port | 7 | Power indicator |
| 3 | Power supply jack | 8 | Transmit & receive pins |
| 4 | Analog input pins | 9 | Digital input / output pins |
| 5 | Microcontroller | | |

Figure 6: Features of Arduino Uno

In Figure 6, **Analog input pins** feed analog inputs to the microcontroller. **Digital I/O pins** feed digital inputs as well as deliver digital outputs. **Transmit and receive pins** transmit and receive data over serial communication to and from an externally connected device. Arduino Uno comes with an ATmega328P microcontroller. **The oscillator** provides clock pulses for the microcontroller to operate. The **USB port** connects a computer to the development board. It uploads the firmware into the microcontroller, sends and receives data between the computer and the board and also supplies DC 5 volts to the board. **The power supply jack** provides

supplementary power when the board is not connected to a USB Port. **The power indicator** indicates the status of power. **Reset button** resets the microcontroller.

Arduino Integrated Development Environment (IDE)

An Integrated Development Environment is essential for firmware development. An IDE for embedded system consists of an uploader as well as a code editor and a compiler. The uploader is used to upload the machine code into a microcontroller. Arduino provides a free and open-source IDE called **Arduino IDE** available at: <https://www.arduino.cc/en/main/software>.

Figure 7 shows key components of Arduino IDE

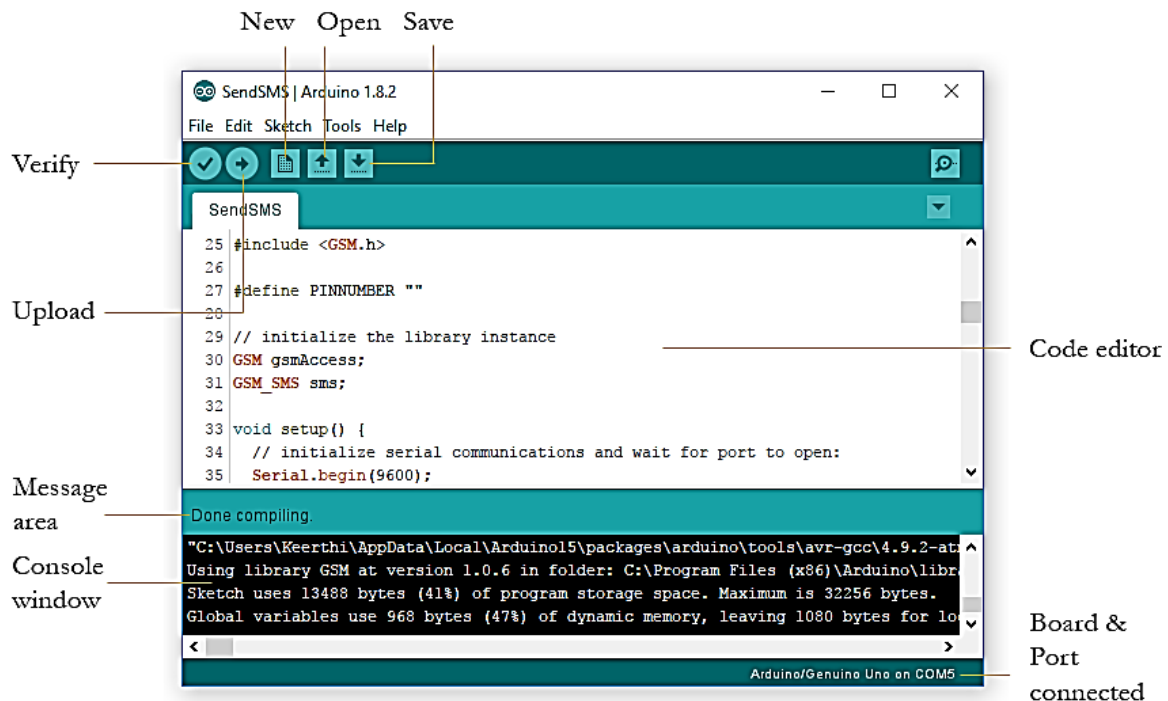


Figure 7: Key components of Arduino IDE

Programs written using Arduino IDE are called sketches. In Figure 7:

- **New** button creates a new sketch in the **Code editor**.
- **Open** button loads an existing sketch from secondary storage.
- **Save** button saves the current sketch in the code editor.
- **Verify** button checks the source code for any syntax errors. If any errors found, they are reported in the console window.
- **Upload** button compiles source code into machine code and uploads it into the microcontroller.
- **The message area** displays the status of IDE.
- **The console window** shows error messages and other information.
- **Board and port connected** show the board in use and the connected port.

EMBEDDED SYSTEM DEVELOPMENT

Let's develop the following embedded systems:

1. Blinker System
2. Auto Light System
3. Auto Fan System
4. Door Alarm System


SYSTEM 1: Blinker

This system is to turn on and off a Light Emitting Diode (LED) periodically.

Required components

- 1 × Arduino Uno microcontroller-based development board (to control the Blinker system)
- 1 × LED (to emit the light periodically)
- 1 × 220Ω Resistor (to drop the voltage and limit the current to LED as required)

Light Emitting Diode (LED)

LED a semiconductor device with two leads named, anode and cathode. The anode lead is usually longer than the cathode lead. When the current flows from anode to cathode, the LED emits light. LEDs are symbolically represented using  in schematic diagrams. Figure 8 shows an LED.

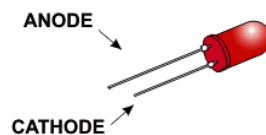
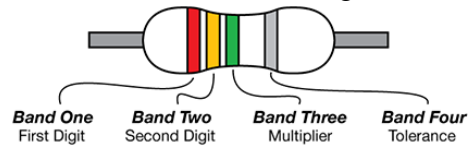


Figure 8: Leads of an LED

LEDs have specific current (2 - 20mA) and voltage (1.8 - 3.3V) ratings. Since a microcontroller I/O pin typically supplies 5V, it is necessary to use an appropriate resistor to drop the voltage and limit the required current flow through the LED. In blinker, a 220Ω resistor is used for this purpose.

Resistor

A resistor is an electronic component that resists to the flow of current in a circuit. Resistors are symbolically represented using $\sim\sim\sim$ in schematic diagrams. The resistance is measured in Ohms (Ω) and represented using a set of color bands. The bands are identified from left to right. Figure 9 shows resistor color codes. The value of the resistance could be varied according to given percentage in the tolerance column in the following table.



Color	Digit	Multiplier	Tolerance (%)
Black	0	10^0	
Brown	1	10^1	1
Red	2	10^2	2
Orange	3	10^3	
Yellow	4	10^4	
Green	5	10^5	0.5
Blue	6	10^6	0.25
Violet	7	10^7	0.1
Grey	8	10^8	
White	9	10^9	
Gold		10^{-1}	5
Silver		10^{-2}	10
(none)			20

Figure 9: Resistor Color Codes

Accordingly, a 220Ω resistor is represented with Red, Red, and Brown giving the value of 22×10^1 Ohms

To construct an embedded system, it is essential to follow some steps as given below:

- Construct the schematic diagram and assemble hardware
- Design firmware
- Develop firmware
- Compile firmware and Upload machine code

Construct Schematic Diagram and Assemble Hardware

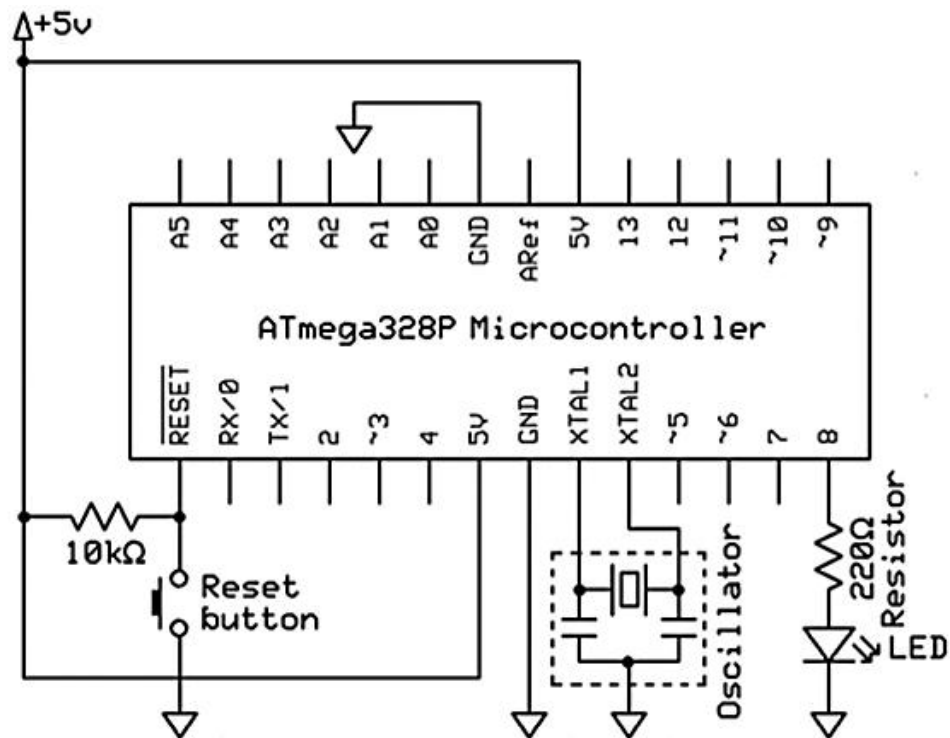


Figure 10: Schematic Diagram of Blinker

The schematic diagram depicts an electronic circuit using symbols. Figure 10 shows the schematic diagram of blinker system.

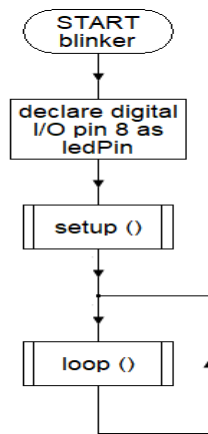
In Figure 10, the microcontroller is supplied with power through 5V and GND (ground) pins. The reset pin is connected to the ground through the reset switch. When pressed, it becomes logic low and the microcontroller resets. To keep it logic high during all other time, it is usually connected to positive power supply line through a resistor. XTAL1 and XTAL2 pins are connected with an external oscillator. In Arduino Uno, the above components are pre-connected. This makes the embedded system development easier. However, we are required to connect LED to be blinked with a resistor to an I/O pin.

Design Firmware

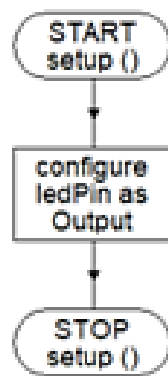
Firmware means a permanent software for a hardware device. Therefore, after assembling the hardware, the microcontroller is required to be programmed. The firmware development usually starts with designing an algorithm using a flowchart.

Flowchart

Unlike programs developed for general-purpose computer systems, the firmware for an embedded system typically does not have any execution endpoint. This is because when the program completes its execution, there is no operating system to take control and thereby the program itself has to plan for the next step. As a result, the program is designed to execute itself over and over again using an endless loop. Figure 11 shows a flowchart of the blinker.



The flowchart of setup() function is shown below:



The flowchart of loop() function is shown below:

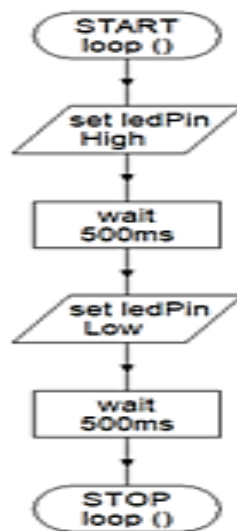


Figure 11: Flowcharts of Blinker

An Arduino sketch has two main functions, namely, setup() and loop(). The setup() runs only once at the beginning of the program and is typically used to perform any initializations/configurations. The loop() runs over and over again.

According to the flowchart in Figure 11, loop() first sets ledPin logic high. The microcontroller then sets digital I/O pin 8 to 5V and turns on the LED. The algorithm then waits for 500ms and

sets ledPin logic low. The microcontroller then sets digital I/O pin 8 to 0V and turns off the LED. The algorithm then waits for another 500ms. As loop () runs over and over again, LED is turned on and off every ½ a second.

Develop Firmware

Figure 12 shows the source code of firmware written based on the above flowchart using the Arduino IDE.

```
// blinks an LED every ½ a second
const int ledPin = 8;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  digitalWrite(ledPin, HIGH);
  delay(500);
  digitalWrite(ledPin, LOW);
  delay(500);
}
```

Figure 12: Source code of Blinker

In an Arduino sketch, //is used for comments. In source code, the ledPin has been declared as an integer constant. The *pinMode(pin, mode)*function configures the specified pin as an INPUT or an OUTPUT pin. The *digitalWrite(pin, state)*function sets the state of the specified digital pin either logic HIGH or LOW. The *delay(ms)* function holds execution for an amount of time specified in milliseconds.

Compile firmware and Upload machine code

The source code of firmware can be verified for any syntax errors using the verify button on IDE. After verification, the development board is connected to the computer via a USB port. Then source code is compiled into machine code and it is uploaded into the microcontroller using the upload button. Then the LED would start to blink every ½ a second drawing power via USB port.

SYSTEM2: AutoLight

In system 1, we simply blinked the LED periodically. In this system, we are going to turn on and off a LED depending on the ambient light intensity using a light sensor.

Required components

- 1 × Arduino Uno microcontroller-based development board (to control the Auto Light system)
- 1 × LED (to emit the light)
- 1 × 220Ω Resistor (to drop the voltage and limit the current to LED as required)
- 1 × Light Dependent Resistor (LDR) (to capture the ambient intensity as an input)
- 1 × 10Ω Resistor (to drop the voltage and limit current flow to ground line)

Light Dependent Resistor (LDR)


LDR is a resistor that changes its resistance depending on the intensity of light falls upon. LDRs are symbolically represented using  in schematic diagrams. Figure13 shows an LDR.



Figure 13: LDR

Construct Schematic Diagram and Assemble Hardware

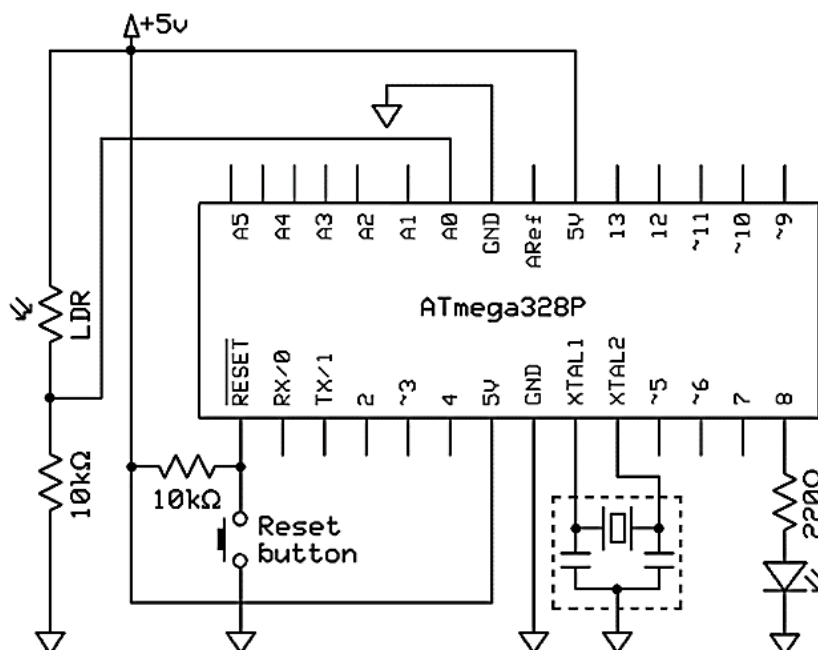


Figure 14: Schematic Diagram of AutoLight

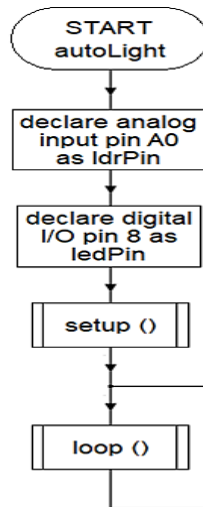
Figure 14 depicts the schematic diagram of Auto Light system. Arduino Uno comes with most of the required components pre-connected. Only connections to be made are an LED connected through a resistor to an I/O pin, an LDR and a 10kΩ resistor interconnected in series between 5V and ground lines forming a voltage divider and the LDR and 10kΩ resistor interconnection to an analog input pin. The next section discusses firmware design.

Design Firmware

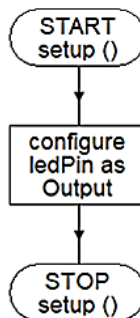
As in the system 1, the firmware development starts with designing an algorithm using a flowchart.

Flowchart

Figure 15 shows the flowchart for the AutoLight.



The flowchart of setup() function is shown below:



The flowchart of loop() function is shown below:

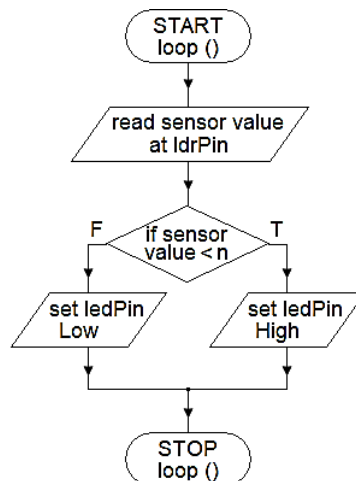


Figure 15: Flowchart of AutoLight

As loop() runs over and over again, LED is turned on or off automatically depending on light intensity.

Develop Firmware

Figure 16 shows the source code of firmware written based on the above flowchart using the Arduino IDE.

```
// switches an LED on and off depending on light intensity
const int ldrPin = A0;
const int ledPin = 8;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  int sensorValue = analogRead(ldrPin);
  if (sensorValue < 150)
    digitalWrite(ledPin, HIGH);
  else
    digitalWrite(ledPin, LOW);
}
```

Figure 16: Source code of AutoLight

The *analogRead(pin)* function reads voltage at the specified analog pin and returns a number between 0 and 1023. This number represents the amount of light intensity that falls upon the LDR. It is then compared with a pre-determined value and the LED is turned on/off depending on the light intensity. In practice, a suitable pre-determined value is used to turn on the LED at the nightfall.

Compile firmware and Uploading machine code

As in system 1, compiling firmware and uploading the machine code have to be done.

SYSTEM3: AutoFan

We have developed a simple embedded system to turn on and off a light periodically and another embedded system to sense the state of physical world and to activate an actuator according to that input. Now let's construct an AutoFan system to turn on and off a motor of a fan depending on the room temperature.

Required components

- 1 × Arduino Uno microcontroller-based development board
(to control the AutoFan system)
- 1 × 9 Volts DC Motor (to function the fan)
- 1 × LM35 Temperature Sensor
(to capture temperature as an input)
- 1 × BC547 Transistor (to supply sufficient current to drive the motor)
- 1 × 1kΩ Resistor (to limit the current flow to transistor)
- 1 × 1N4001 Rectifier Diode
(to protect transistor from flyback current from motor)

Temperature Sensor

LM35 is an Integrated Circuit temperature sensor that changes the voltage at its Vout pin depending on the temperature sensed. Its V+ and GND pins require to be connected to positive and ground power lines, respectively. Figure 17 shows the pinout of an LM35.



Figure 17: Pinout of LM35 Sensor

The temperature sensor has sensed can be determined in Celsius using the following equation.
Temperature = Vout × 100 (1)

Motor

Motor is an actuator used to convert electrical energy into kinetic energy. A motor draws more current than an I/O pin of a microcontroller can deliver. A transistor is therefore used to drive the motor and an I/O pin is used to switch the transistor. DC motors are symbolically represented using \textcircled{M} in schematic diagrams. Figure 18 shows a 9V DC Motor.



Figure 18: 9V DC Motor

Transistor

A transistor is a semiconductor device used to switch or control electrical power electronically. It has three pins, named Base (B), Collector (C) and Emitter (E). When a small current flows in its Base-Emitter circuit, the transistor allows a relatively larger current in its Collector-Emitter Circuit.

The NPN type transistors are symbolically represented using  in schematic diagrams. Figure 19 shows the pinout of a BC547 transistor.



Figure 19: Pinout of BC547 Transistor

Diode


Diode is also a semiconductor device. It has two leads, namely, anode and cathode. Diode lets current flow only from anode to cathode. Diodes are symbolically represented using  symbol in schematic diagrams. Figure 20 shows a 1N4001 rectifier diode.



Figure 20: 1N4001 Rectifier Diode

Construct Schematic Diagram and Assemble Hardware

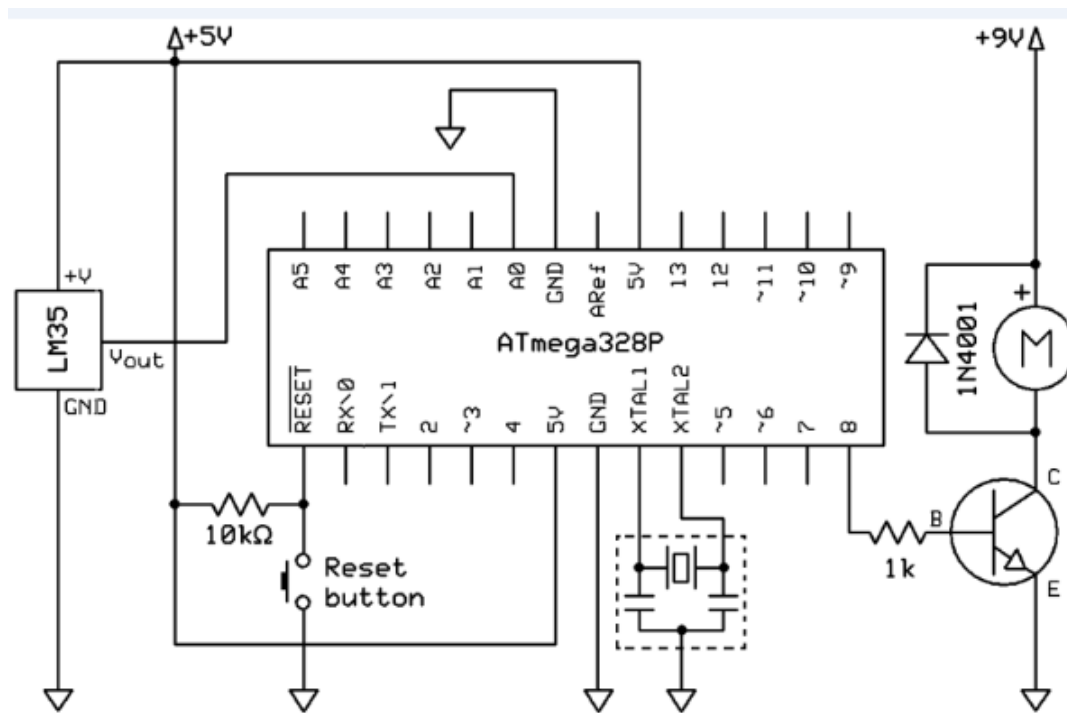


Figure 21: Schematic diagram of AutoFan

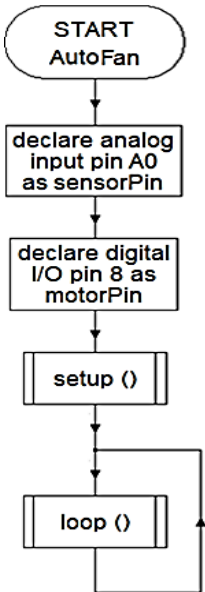
Figure 21 depicts the schematic diagram of Auto Fan system. As in the previous systems, Arduino Uno comes with most of the required components pre-connected. Only connections to be made are the LM35 temperature sensor's V+ and GND pins to 5V and ground power supply lines, respectively and Vout pin to an analog input pin. In addition, a digital I/O pin is required to be connected to the Base (B) of a transistor via a resistor. A motor is then connected between Collector (C) of the transistor and separate 9V power supply line. The Emitter (E) is directly connected to the common ground line. A rectifier diode is connected in parallel to the motor with its cathode to the 9V line. The next section discusses firmware design of Auto Fan system.

Design Firmware

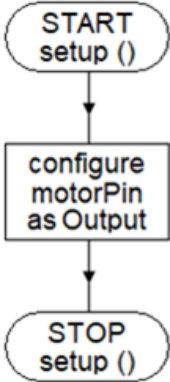
As in the previous systems, the firmware development starts with designing an algorithm using a flowchart.

Flowchart

Figure 22 shows the flowcharts for the Auto Fan.



The flowchart of setup() function is shown below:



The flowchart of loop() function is shown below:

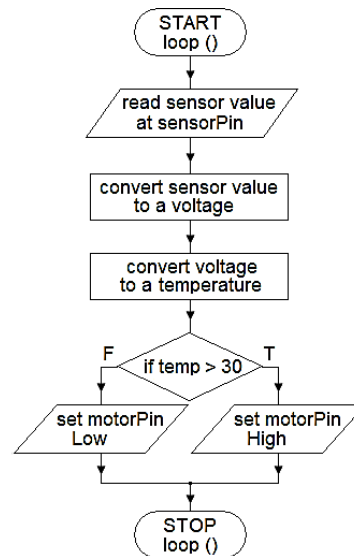


Figure 22: Flowchart of AutoFan

As loop() is called over and over again, the motor is turned on or off automatically depending on the room temperature.

Develop Firmware

Figure 23 shows the source code of firmware written based on the above flowchart using the Arduino IDE.

```
// switches a motor of a fan on and off depending on room temperature
const int sensorPin = A0;
const int motorPin = 8;

void setup()
{
  pinMode(motorPin, OUTPUT);
}

void loop()
{
  int sensorValue = analogRead(sensorPin);
  float voltage = value * 5.0 / 1024;
  float temp = voltage * 100;
  if (temp > 30)
    digitalWrite(motorPin, HIGH);
  else
    digitalWrite(motorPin, LOW);
}
```

Figure 23: Source code of Auto-Fan

The *analogRead(pin)* function reads voltage at the specified analog pin and returns a number between 0 and 1024. First, this number is converted into a voltage between 0 and 5V and then into a temperature in Celsius. The determined temperature is then compared with a pre-defined value and the motor is turned on/off depending on the room temperature.

Compile firmware and Upload machine code

As in the previous systems, compiling firmware and uploading the machine code have to be done.

SYSTEM4: Door-Alarm

In the above systems 2 and 3, we used an LDR and an LM35 to capture inputs and drive the actuators, respectively. In this system, we would construct a door alarm system to trigger an alarm when a door is opened. The system uses a reed switch as a sensor to capture the input.

Required components

- 1 × Arduino Uno microcontroller-based development board
(to control door alarm system)
- 1 × Piezo Buzzer (to generate alarm as output)
- 1 × Reed Switch (to detect whether door is open)
- 1 × 10kΩ Resistor (to drop the voltage and limit current flow to ground line)

Reed Switch

Reed switch is an electrical switch operated when a magnetic field applied. It is normally open and closes its contacts when a magnetic field is present. If a magnet and a Reed switch are fixed to a door and door frame, respectively, the switch will keep its contacts closed when the door is closed and will open the contacts when the door is opened. Figure 24 shows a Reed switch.



Figure 24: Reed switch

Piezo Buzzer

A Piezo buzzer is an actuator that uses the piezoelectric effect for generating sounds. Figure 25 shows a piezo buzzer.



Figure 25: Piezo buzzer

Construct Schematic Diagram and Assemble Hardware

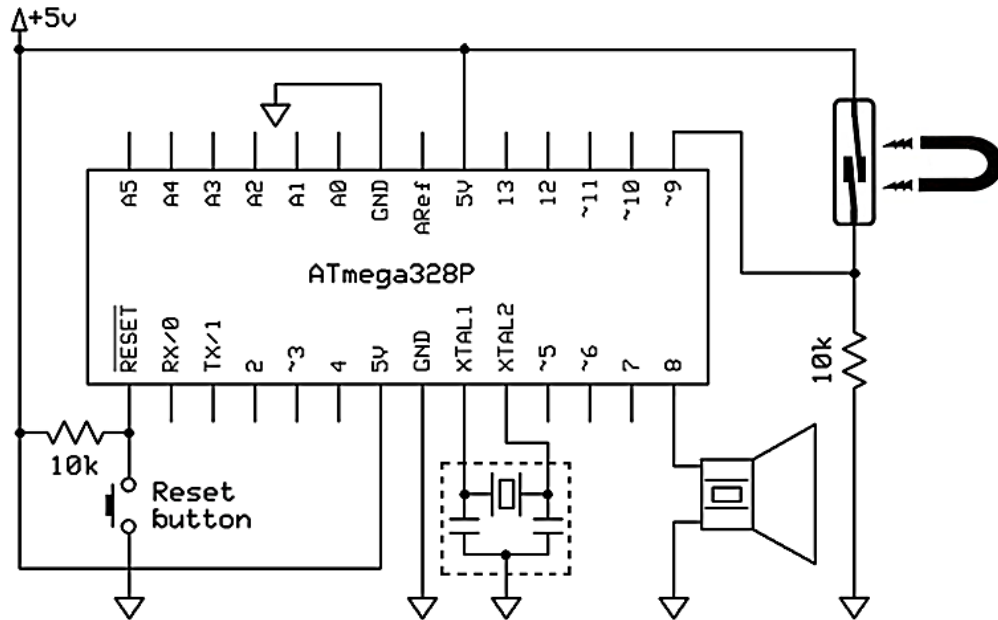


Figure 26: Schematic diagram of DoorAlarm

Figure 26 depicts the schematic diagram of Door Alarm system. As in the previous systems, Arduino Uno comes with most of the required components pre-connected. The additional components to be connected are a Piezobuzzer to a digital I/O pin and a Reed switch and a 10k Ω resistor in series between 5V and ground lines. Reed switch and 10k Ω resistor interconnection are connected to a digital I/O pin.

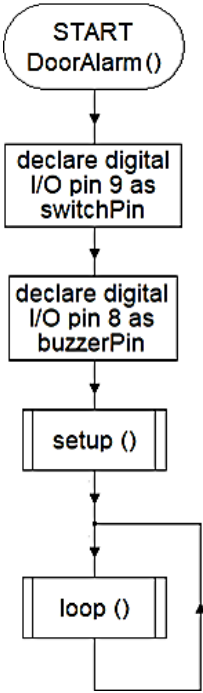
When the door is opened, as the magnetic field moves away, the Reed switch opens its contacts and leaves the digital I/O pin connected only to ground line through the resistor. This changes state of digital I/O pin 9 logic low. As a result, the microcontroller drives the Piezo buzzer via the digital I/O pin 8. The next section discusses firmware design for this purpose.

Design Firmware

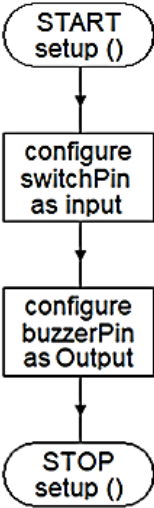
As in the previous systems, the firmware development starts with designing an algorithm using a flowchart.

Flowchart

Figure 27 shows the flowcharts for the Door-Alarm.



The flowchart of setup() function is shown below:



The flowchart of loop() function is shown below:

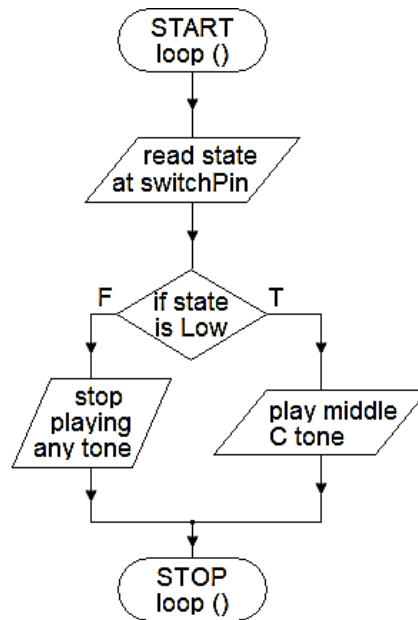


Figure 27: Flowchart of Door Alarm

As loop() runs over and over again, the buzzer would play a tone when the door is opened.

Develop Firmware

Figure 28 shows the source code of firmware written based on the above flowchart using the Arduino IDE.

```
// triggers an alarm when a door is opened
const int switchPin = 9;
const int buzzerPin = 8;

void setup()
{
  pinMode(switchPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
}

void loop()
{
  int switchState = digitalRead(switchPin);
  if (switchState == LOW)
    tone(buzzerPin, 262);
  else
    noTone(buzzerPin);
}
```

Figure 28: Source code of DoorAlarm

The *tone(pin, frequency)* function generates a square wave with the specified frequency on the given digital I/O pin. The generated tone plays continuously until the *noTone(pin)* function is called.

Compile firmware and Upload machine code

As in the previous systems, compiling firmware and uploading the machine code have been done.

The Internet of Things (IoT)

Competency Level 11.2: Explores the Internet of Things (IoT) to create a simple application

Learning Outcomes:

- Defines IoT (Internet of Thing)
- Identifies the needs of IoT to make day to day life smart
- Discusses the various applications of IoT
- Identifies the enabling technologies for IoT
- Designs and implements an IoT application to remotely control a device over Internet
Example: switching a LED on and off over the Internet
- Uses IoT based systems while knowing the social and security consequences of IoT

Smart World

IoT can create a "Smart World" where life is full of autonomous interconnected smart systems. For example, what if your smart alarm clock not only wakes up you in the morning but also notifies your electric kettle to boil water for your morning tea? How about smart refrigerator monitors the level of food items in it and send you a shopping list to your smartphone in the weekend? The whole idea is to make our life more convenient and comfortable.

Things are everyday objects from small wristwatches to large cars and buildings. When added computing power, things become smart and are called embedded systems. Embedded systems can sense the state of the physical world and change the state of the physical world. When we connect the embedded systems to the Internet, they start interacting with themselves and users and create an Internet of Things.

Enabling Technologies

Many technologies have enabled the Internet of Things. They include the Internet Protocol (IPv6) with an extremely large address space, increasing bandwidth and decreasing the cost of networking technology, increasing functionality and decreasing size and cost of sensor technology and increasing performance and capacity and decreasing power consumption, size and cost of processor and storage technologies, etc.

Applications of IoT

IoT can be applied to any physical system such as wearables, appliances, homes, transport, and agriculture, health, etc. Consumer IoT applications includes smart wearables with monitoring and home appliances with remote accessing capabilities. Commercial IoT applications include smart medical and health care systems with remote monitoring, emergency notification and assistive capabilities, smart building and home automation applications with capabilities to monitor and control lighting, climate, entertainment, security and other systems in a building, transportation applications such as smart traffic lights, smart parking, vehicle control, and safety and roadside assistance systems etc. Industrial IoT applications include smart manufacturing applications such as process controls and predictive maintenance systems and agriculture applications such as rainfall, humidity, pest infestation, and soil content monitoring systems and smart irrigation and fertilization systems, etc.

Challenges of IoT

IoT opens the door to a lot of possibilities and opportunities but also to many challenges. They include social concerns such as social isolation and security and privacy concerns such as unauthorized control of IoT devices and access to personal and sensitive information. What if someone hacks into your smart alarm clock and then gets access to your entire network of things? Therefore, it is very important to understand not just the opportunities but also the challenges of IoT.

IoT Based System Development

IoT based systems are developed to automate the intercommunication of smart systems.

SYSTEM5: Smart Light

This system explains how to turn on and off a light remotely over the Internet. When HTTP requests *http://host/?1* and *http://host/?0* are made using a web browser running on remote computer connected to the Internet, the Smart Light would turn on and off an LED, respectively.

Required components

- 1 × Arduino Uno microcontroller-based development board
(to control smart light system)
- 1 × Arduino Ethernet Shield
(to extend the functionality of Arduino Uno to connect to the Internet)
- 1 × LED (to emit the light)
- 1 × 220Ω Resistor (to drop the voltage and limit current flow to the LED as required)

Arduino Ethernet Shield

Arduino Ethernet Shield is a pre-assembled add-on module to the main microcontroller-based development board. Once added to the Arduino Uno board, the Ethernet Shield extends the functionality of Arduino Uno and facilitates it to connect to the Internet. Figure 25 shows the latest version of Arduino Ethernet Shield.

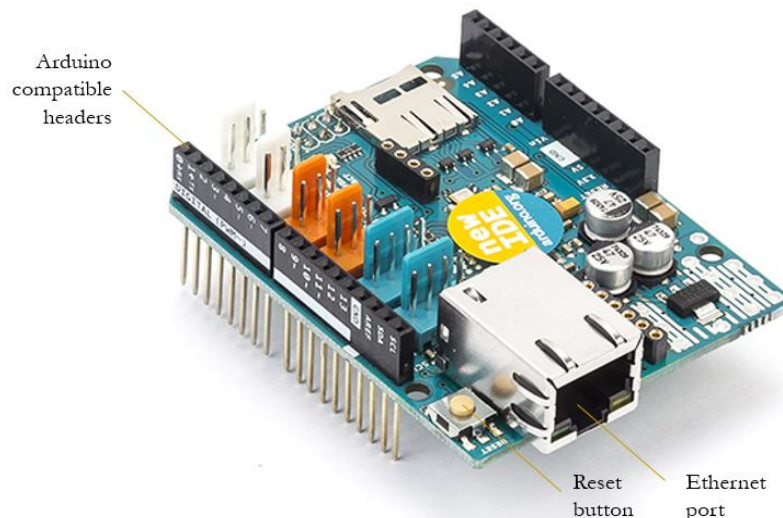


Figure 29: Arduino Ethernet Shield

Construct Schematic Diagram and Assemble Hardware

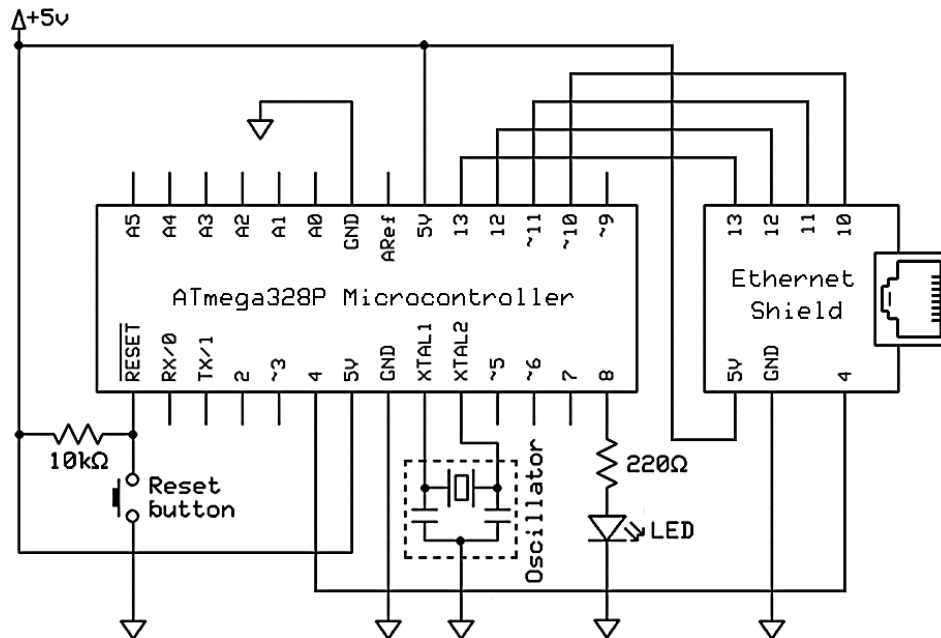


Figure 30: Schematic diagram of Smart Light

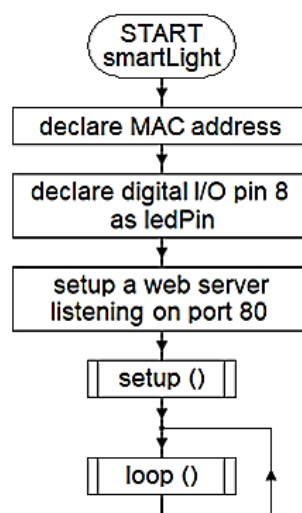
Figure 30 depicts the schematic diagram of Smart Light system. As in the previous systems, Arduino Uno comes with most of the required components pre-connected. The additional components to be connected are the Ethernet Shield and an LED to be controlled over the Internet with a resistor to a digital I/O pin. When stacked on Arduino Uno, the Ethernet Shield communicates with the microcontroller in the Arduino Uno board using the digital I/O pins 4, 10, 11, 12 and 13. Therefore, they cannot be used as general-purpose I/O pins.

Design Firmware

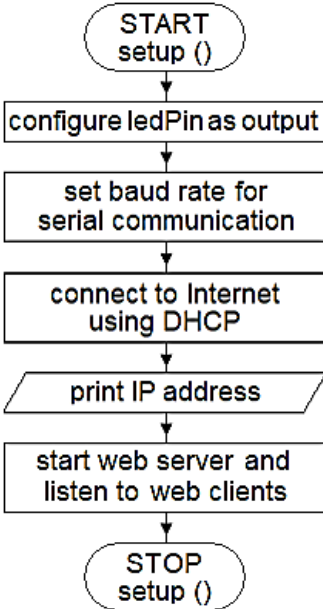
As in the previous systems, the firmware development starts with designing an algorithm using a flowchart.

Flowchart

Figure 31 shows the flowcharts for the Smart Light.



The flow chart of setup() function is shown below:



The flow chart of loop() function is shown below:

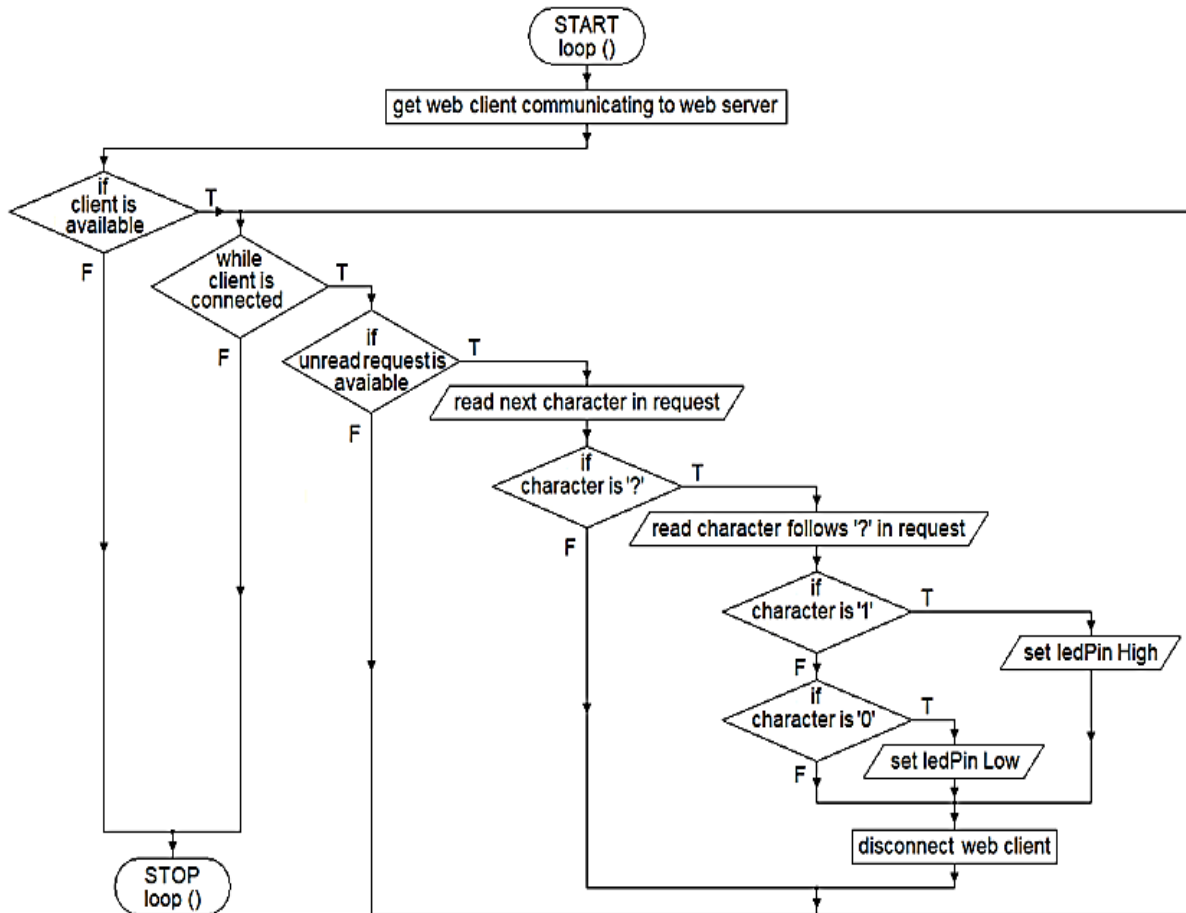


Figure 31: Flowchart of Smart Light

In the above algorithm, first, the hardware MAC address and digital I/O pin 8 are declared. Then a webserver is setup for listing any web client communication over port 80. In the setup() function, first the ledPin is configured as an output. Then the data communication rate for serial communication between the Arduino Uno board and the connected computer is set. This is followed by an attempt to connect to the Internet using DHCP. In the next step of setup() function, the IP address assigned by DHCP is printed on the serial monitor of Arduino IDE running in the computer connected. Then the previously setup webserver is started for listening to any incoming communication from web clients.

In each iteration of the loop() function, first it gets any web client communicating to the webserver. If such a web client is available, a loop is executed while the web client is connected to the webserver. In the body of this loop, it checks whether any unread HTTP request is available from the web client. If so, the request is read character by character. If the character is '?', then it checks whether the next character in the request is '1' or '0'. If so, the logic state of ledPin is set either high or low and the connected LED is turned on or off, respectively. Finally, the web client is disconnected from the webserver.

Develop Firmware

Figures 32 and 33 show the source code of firmware written based on the above flowchart using the Arduino IDE.

```
// turns on and off an LED over the Internet

#include <Ethernet2.h>

byte mac[] = {0x##, 0x##, 0x##, 0x##, 0x##, 0x##};
const int ledPin = 8;
EthernetServer webServer = EthernetServer(80);

void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
  Ethernet.begin(mac);
  Serial.println(Ethernet.localIP());
  webServer.begin();
}
```

Figure 32: Source code of Smart Light - Part 1

`#include<Ethernet2.h>` directive adds the *Ethernet2.h* library file required to work with Arduino Ethernet Shield 2 to the source code. The *mac[]* is an byte array containing the MAC address of the Ethernet Shield. Its *0x##* should be replaced with the hexadecimal numbers of the actual MAC address that comes with Ethernet Shield. The next line setups a webserver for listing any web client communication over port 80.

In the `setup()` function, `Serial.begin(baudRate)` function sets the data communication rate for serial communication between the Arduino Uno board and the connected computer. `Ethernet.begin(mac)` function connects the Ethernet Shield to the Internet using DHCP. In the next line in source code, `Ethernet.localIP()` function gets the dynamic IP address assigned by DHCP and `Serial.println(value)` function prints it on the serial monitor of Arduino IDE. The `begin()` function of the previously setup webserver starts the webserver and begins listening to any incoming communication from web clients.

```

void loop()
{
  EthernetClient webClient = webServer.available();
  if (webClient)
  {
    while(webClient.connected())
    {
      if(webClient.available())
      {
        char character = webClient.read();
        Serial.print(character);
        if (character == '?')
        {
          character = webClient.read();
          Serial.println(character);
          if (character == '1')
            digitalWrite(ledPin, HIGH);
          else if (character == '0')
            digitalWrite(ledPin, LOW);
          webClient.stop();
        }
      }
    }
  }
}

```

Figure 33: Source code of Smart Light - Part 2

In each iteration of the `loop()` function, the `available()` function of the webserver returns any web client communicating to the webserver. If such a web client is available, then a while loop is executed till the `connected()` function of web client returns false. In the body of this loop, `available()` function of web client checks whether any unread HTTP request is available from the web client. If so, `read()` function of web client reads next character in the request to a character variable. If the character is '?', then `read()` function of web client reads the character that follows '?' in the request to the character variable. The next two conditions check whether this character is '1' or '0'. If so, `digitalWrite(pin, state)` function sets the state of specified digital I/O pin logic HIGH or LOW and turns on or off the connected LED, respectively. Finally, the `stop()` function of web client disconnects the web client from the webserver.

Compile firmware and Upload machine code

Now the Smart Light system requires to be connected to a network with a connection to the Internet using a network cable and the Ethernet port of the Ethernet Shield. As in the previous systems, then compiling firmware and uploading the machine code are done. Once uploaded the machine code, it requires open the serial monitor of Arduino IDE and gets printed the IP address of the webserver running. When HTTP requests `http://host/?1` and `http://host/?0` are made using a web browser running on another computer connected to the Internet, the Smart Light would turn on and off the LED, respectively. In the above URLs, the host means the IP address of the webserver running on the Smart Light system. Further, IoT based systems can also be developed using WiFi wireless communication as well.

Review Questions

1. What is an Embedded System?
2. Like general-purpose computer systems, do embedded systems also follow Input, Process and Output (IPO) model? Explain using an example.
3. Consider the following statement, "Embedded systems perform physical computing." Explain using an example.
4. Discuss the pros and cons of microprocessors-based and microcontrollers-based embedded systems developments.
5. Why do we need to code an infinite loop when writing firmware for microcontrollers? Explain.
6. What is the Internet of Things (IoT)?
7. How does IPv6 affect the development and implementation of the IoT?
8. Discuss the impact of IoT on our lives.

PROGRAMMING

Competency 9: Develops algorithms to solve problems and uses python programming language to encode algorithms

Competency Level 9.1: Uses problem-solving process

Competency Level 9.2: Explores the top down and stepwise refinement methodologies in solving problems

Competency Level 9.3: Uses algorithmic approach to solve problems

Learning Outcomes:

- Describes the steps of problem solving process
- Uses stepwise refinement methodology to solve problems
- Draws structures charts to illustrate a solution for a system
- Describes algorithms briefly
- Identifies the standard symbols used to draw flow charts
- Draws flow charts to illustrate solutions to a given problem
- Writes pseudo codes to illustrate solutions to a given problem
- Uses hand traces to verify the solutions

Algorithm Development

An Algorithm is a finite sequence (a finite number of elements) of well-defined (clear and unambiguous) instructions, typically to solve a problem.

The problem solving process involves the steps depicted in the following figure in cyclic way.



The problem solving process is important not only in solving mathematical problems but also in real life problems.

Examples:

Switching on an air conditioner as the atmospheric temperature rises is solving a mathematical inequality depending on a physical papermaker temperature.

Finding a suitable course to follow is solving a mathematical comparison of cost and benefits of different courses.

For all the examples like the ones mentioned above, a computer program can be written based on an Algorithm.

Modularization (Decomposition)

To understand the term Algorithm, let us look at the problem-solving process in general. The problem-solving process can be itemized as follows:

- Identifying the problem
- Identifying the subproblems (causes of the main problem)
- Finding (sub) solutions to subproblems
- Connecting the sub-solutions logically to solve the main problem

For example, take a simple day to day situation of making a cup of tea. The process involves taking the ingredients as input, the process of mixing the ingredients to produce the output of a cup of tea.

In some problems, it is necessary to break the big problem into small subproblems and consider the solution as a collection of (sub) solutions to different subproblems.

Example:

Main problem: inefficient administration in an organization

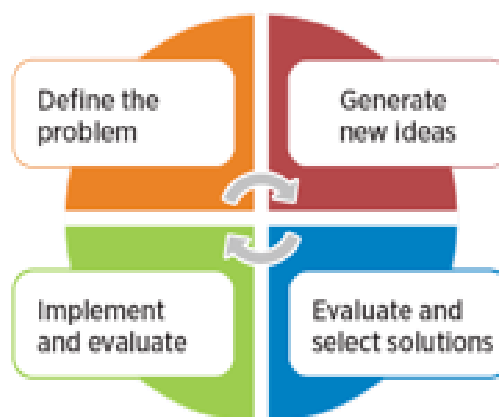
Sub Problem: Problems in financial division, problems in human resource division, problems in technical division, etc..

Note: This is what is done in the system design process.

The same methodology applies even in solving mathematical problems.

The above strategy for solving a problem is called Modularization or Decomposition. Modularization means breaking a big problem into logical subproblems. In our examples, we broke or modularized the big problem into small problems that cannot be further broken logically. In the case of making a cup of tea, the modularization is boiling water, mixing the ingredients, etc.

In a real-world system design, in the first attempt, we may not get the best possible modularization. It is necessary to refine the initial model to make it better. In the refinement process, we may have to combine some subproblems or break into further subproblems to make the system more sensible. The process can be depicted graphically as follows.



Once the subproblems (or subsystems) are identified, they have to be documented in a visually comprehensible way. The best visually comprehensible way of writing is by using a figure.

Suppose that our problem is to check whether a person is eligible to vote at an election:

- If the age is greater than or equal to 18, then eligible to vote
- If the age is less than 18, then not eligible to vote

The following is the representation of the above problem and solution using what is known as a block diagram

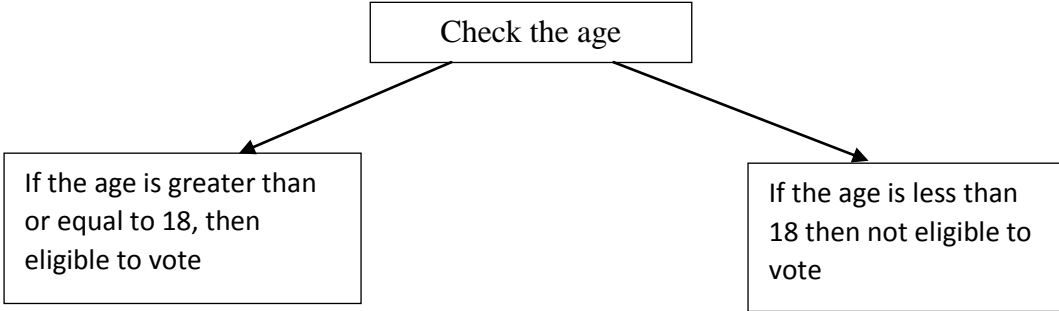


Figure No 2.1- Visually Comprehensible Representation

Algorithmic approach to solve problems

Algorithms can be represented as follows:

- Graphically : Flow chart
- Textually: Pseudocode

Flow chart / Pseudocode are converted into source code.

Graphically:

Once the system block diagram is done, it should be developed into a flow chart. Flow chart literally means the flow of actions to realize the components depicted in the block diagram.

- **Flow chart**

The flow chart uses the following standard symbol to describe a solution.

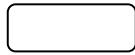
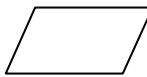
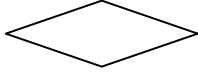
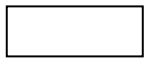
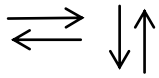


Start / End	
Input/ Output	
Decision Making	
Process (Action)	
Flow Lines	
Joining Paths (Connector)	
Subroutine	

Table 2.1 – Common Flowchart Symbols

Using the above symbols, we can draw the following flow chart to the problem discussed above.

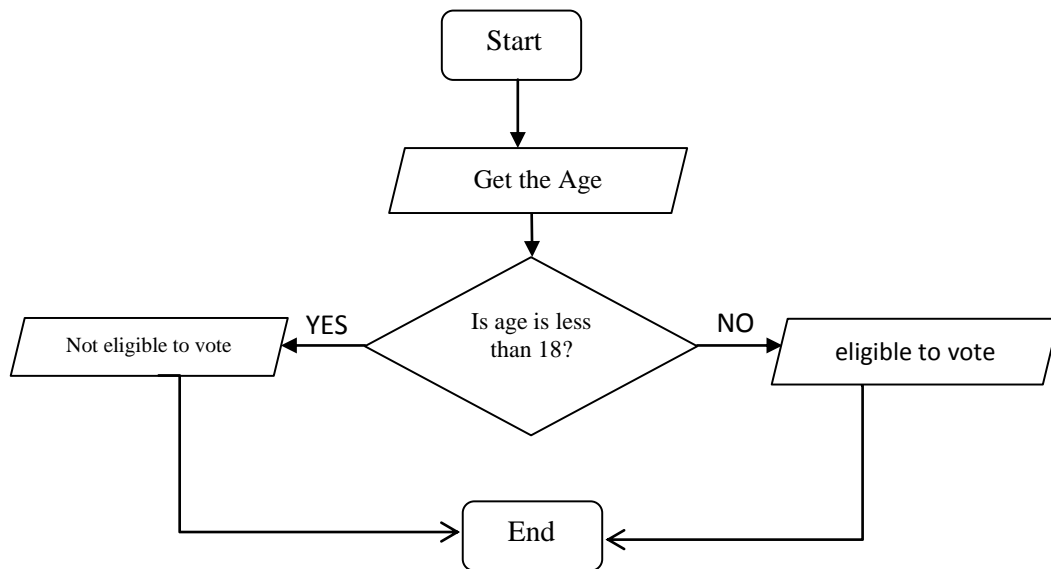


Figure 2.2: Flowchart for checking eligibility to vote

Textually:

- **Pseudocodes**

The flow chart then needs to be represented as a Pseudocode. Pseudocode is written as a statement in English that is closer to computer programming languages but independent of any particular language. We do this because when the pseudocode is written, converting it into a suitable computer programming language is easy.

The pseudocode for our example looks as below:

```
Begin
    Read age
    If age <18 then
        Display “not eligible to vote”
    Else
        Display “eligible to vote”
    Endif
End
```

- **Hand traces**

Once the pseudocode is written, a manual check is done before converting it into a programming language. This is done by working with an appropriate example situation and line by line. In our example, we could enter or input any age value and see what happens in the process. This is done in order to minimize possible errors when executing the program.

Competency Level 9.4: Compares and Contrasts different programming paradigms

Learning Outcomes:

- Describes the evolution of programming language in terms of generations
- Compares and contrasts imperative, declarative, object oriented languages

Evolution of programming languages

The programming language is a set of commands written in a specific format and grammar, and it has to be learnt just like learning another language to understand a foreign language to communicate with a foreigner. C, C++, Python, Pascal, Java, etc. are some common programming languages.

Once the pseudocode which is general and not specific to any programming language is written, it should be converted into a specific programming language of choice and based on the type of application and expertise.

The programming language is the language understandable to humans and to computers once compiled. There are many different languages that are suitable for different applications: scientific calculations, database management, business applications etc.

Programming paradigms

Computer programming languages can be categorized into two broad categories (depending on how they work), they are known as programming paradigms.

1. Imperative languages

This is the most common type. In this type, the problem is solved (the program is written) as a sequence of commands (imperatives).

Further, these languages can be categorized into three types:

- Procedural. e.g: C
- Object-oriented. e.g: Java, C++
- Parallel processing.e.g: Java

Declarative languages

This type is less common. In this type, there is no specific sequence and statements (declarations) can have a different flow.

Further, these languages can be categorized into three types:

- Logic. e.g: Prolog
- Functional data flow. e.g: Lisp
- Database. e.g: SQL

Competency Level 9.5: Explores the need of program translation and the type of program translators

Learning Outcomes:

- Describes the need of translation of a program
- Compares the source and object program
- Lists and briefly describes the types of program translators
- Briefly describes the function of linkers

Program translation and the type of program translators

Source Program and Object Program need to be translated into machine code.

Source code

A program written in a high-level language like C, Python, etc. can be understood by humans who know the particular language.

Object code

Source code which is not understandable by computers directly without being converted to what is known as the machine language, should be converted into machine language using an intermediate program. After converting into machine language, it is called as object code.

The machine language understandable by computers is made of digital commands in Boolean logic. This is something humans cannot learn and memorize due to the enormous amounts of digits and the vulnerability of mistakes.

To overcome the above-mentioned difficulty, once the human-understandable source code programming is done, an intermediate program converts it into a corresponding object code containing binary values.

There are two ways – interpreting and compiling - of conversion.

○ **Interpreter**

The interpreter converts the source code into an object code line by line. Due to its' nature, this process has to be repeated every time the program is run, and therefore, the execution time is high. Some languages like BASIC, FORTRAN, and Python use this type of conversion.

○ **Compilers**

A compiler converts the whole of the source code into object code at once. After the process, a permanent binary code known as object code is generated. Every time the program runs, the object code is executed and therefore, the execution time is low. Languages like Pascal, C use this approach. If a change is made to the source program, the compilation has to be done again, and new object code is generated.

○ **Hybrid approach**

When there are two or more ways of doing something, there are relative advantages and disadvantages, and to balance them, hybrid approaches are taken. An example is a hybridelectric car where there is a battery for town running, and an engine to run outside towns. Similarly, Interpreting and Compilation can be combined to improve the efficiency.

- **Linker**

Other than interpretation and compilation, there is another process known as Linking. Linking is connecting user commands with standard library functions. For example, the program Input and Output commands are linked to the main program during the linking process. For example, when a user writes *printf* the linker links the program with the printing subprogram stored in the Linker.

The following figure depicts the hierarchical position of Hardware, Machine language, Assembly language and High level languages.

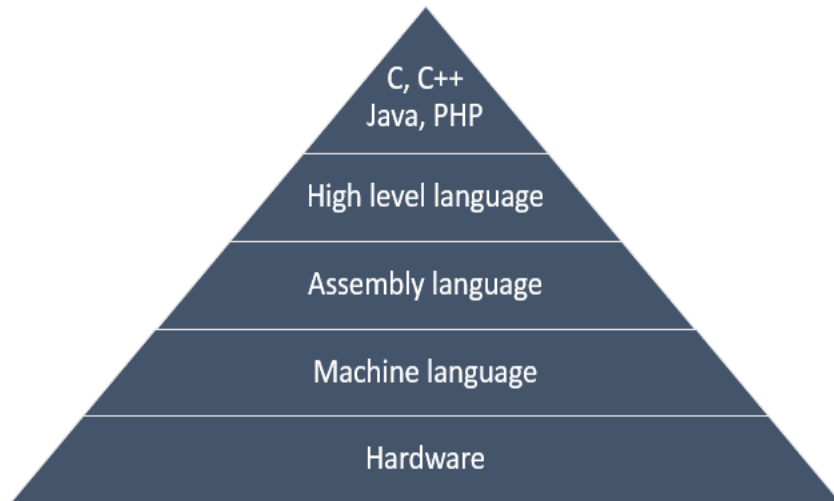


Fig 9.3 Hardware and computer Languages

Competency Level 9.6:

Explores integrated development environment (IDE) to identify their basic features

Learning Outcomes:

- Identifies the basic features of IDE
- Practices the instructions to
 - Open and save files
 - Compile, execute programs
- Uses the debugging facilities in IDE

Integrated development environment (IDE)

Integrated Development Environment is essential to handle the source code. Most IDE's consists code editor, compiler, debugger etc.

As mentioned above, once the source code is written in a high-level language, it is compiled to an object code ready for execution.

Let's see what happens in the process and what programs are responsible for each of these stages.

Editing using Editors – This is the first phase of the process. The algorithm has to be converted to statements according to the syntax (grammar) of the language, and these statements have to be typed for further processing. This process is called Editing and the software that helps to do this work is called Editors. There are various types of Editors providing different features. For example, three main functions of an Editor are creating a new file, saving a file, Opening a saved file etc. Basically, an Editor is word processing software with minimal features.

Debugging/ Compiling using Compilers – After the program editing is over, a source file is created. As mentioned above, this source file has to be converted to an object file by compiling. In the process of compilation, another process called debugging is taking place. This is the process that detects the syntax errors in the source file. Debugging can be done before compiling, but most of the time, debugging and compilation are happening simultaneously. Debugging program is called a Debugger, and the compilation program is called a Compiler.

Nowadays, there are comprehensive programs which are known as Integrated Development Environments (IDE) where editing, debugging/ compiling can be done very convenient way using one comprehensive program containing an Editor, a Debugger and a Compiler. They are very user-friendly and work through user-friendly interfaces.

Microsoft Visual Studio is such an IDE, Eclipse with PyDev plugin is an IDE for development in Python.

There are many online IDEs to develop Python programs.

Imperative programming language to encode algorithms:

In the following sections, programming language Python is used to demonstrate the program structures and syntax. These structures and syntax are almost similar in all the languages. Once the logical programming techniques are mastered, anybody can switch to any other language with little self-work.

- **Structure of a program**

A program has three main parts, namely, Input, Processing, and Output.

As we know, Input/ Output are handled using variable names. Logically, before using a variable, it has to be defined. So, we define the program variables, before the Input section. As mentioned earlier, a program uses predefined functions like print(), writeln() etc. These are stored in a library, and these library functions have to be loaded at the beginning.

Accordingly, a structure of a program looks as follows:

```
Start
    Importing essential libraries
    Declaration of variables/ constants
    Input data
    Processing
    Output results
End
```

- **Comment**

In a program, it is necessary to identify and understand the functions of different lines in an ordinary language understandable to humans. As we know, the program contains the language understandable to the computer, so to bridge the gap, we write comments at the beginning or end of commands. These are not executed by the compiler and the only use of Comments is to understand the function of the statement or the command. Comments are essential to understand the function of the commands/ program by people other than the programmer and even by the programmer at a later time.

In Python, we could write comments as below.

```
age=26
if (age>=18):
    print ("You are eligible to vote"); # To print the eligibility
```

Competency Level 9.7: Uses an imperative programming language to encode algorithms

Learning Outcomes:

- Identifies the structure of a program
- Uses comments to identify the usage of code for future reference
- Uses constants and variables in a program appropriately
- Identifies the primitive data types of a given program language
- Identifies and uses operators in a program
- Identifies precedence of operators
- Writes programs with the facilities of input from keyboard and output to standard devices

Python program

Python is a popular programming language, which was created in 1991 by Guido van Rossum.

What can Python do?

Python can be used:

- Alongside software to create workflows.
- To connect to database systems. It can also read and modify files.
- To handle big data and perform complex mathematics.
- For rapid prototyping, or for production-ready software development.
- On a server to create web applications.

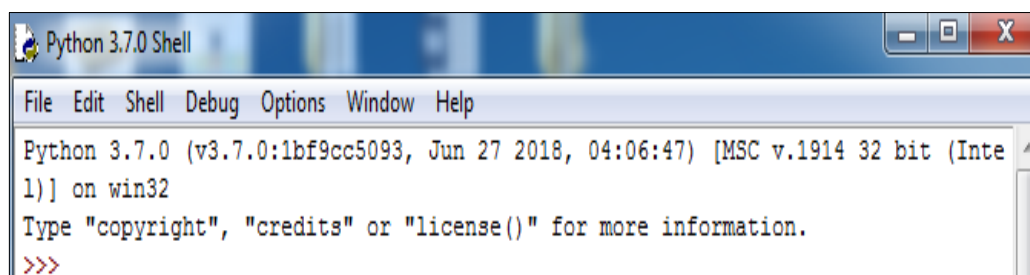
Features of Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has a syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Python Installation

If you find that you do not have python installed on your computer, you can freely download and install from the following website: <https://www.python.org/downloads/>

The IDLE of Python (Python 3.7):



Save the python file

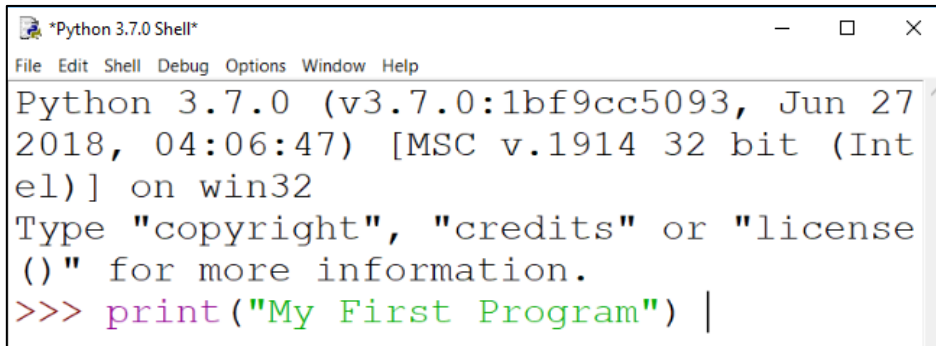
Python file should be saved with .py extension.

e.g:- "hello.py"

Let's develop our first Python program in the python shell and python editor.

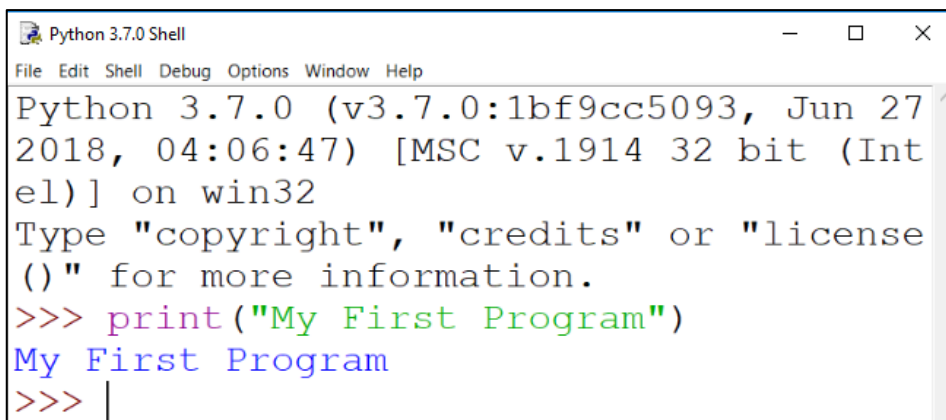
Write `print("My First Program")` in python shell and enter

Source



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("My First Program") |
```

Output



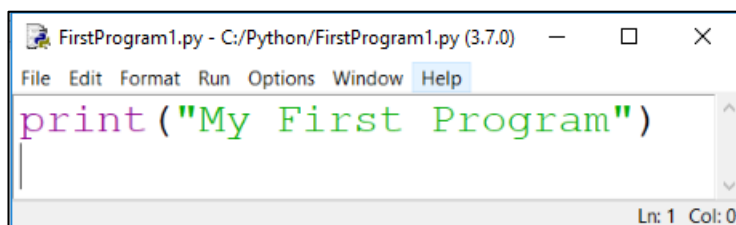
```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("My First Program")
My First Program
>>> |
```

Write `print("My First Program")` in python editor

Save as FirstProgram1.py

Open the Python editor

File → New

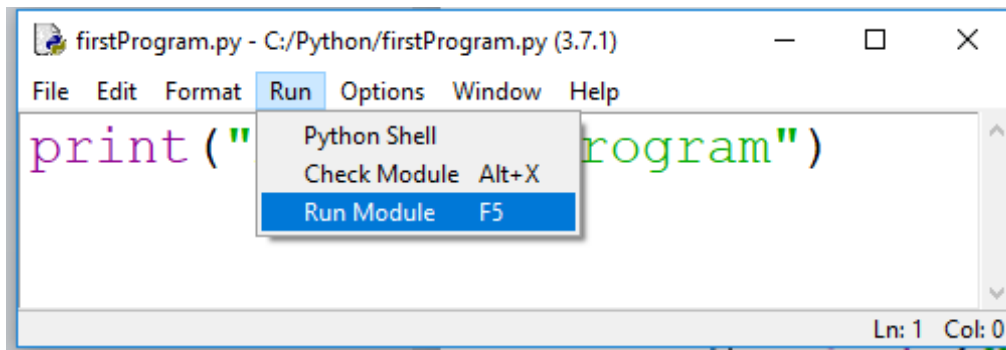


```
print("My First Program")
```

Ln: 1 Col: 0

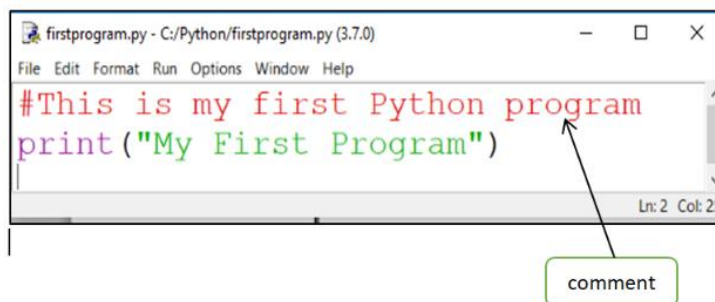
Run the program

Run → Run Module or press F5 key



Comment in Python

In Python, comments start with a # symbol, and Python will render the rest of the line as a comment:



Python Identifier

An identifier is a name that identifies an object (variable, function, class etc). Different languages use different rules for naming objects. In Python the following rules must be observed in naming objects.

- An Identifier name in python must start with a letter or underscore character
- After the first letter, the remainings of an identifier name can contain alpha-numeric characters and underscores and numbers (A-z, 0-9, and _)
- There is no limit to the maximum length of an identifier name
- Python keywords can't be used
- Identifier names are case-sensitive (age, Age, and AGE are three different variables)

Variable

A variable is the symbolic name assign for a place in the computer's memory where one can store data. Once a variable is created by a program, that variable can be used to store different values of the same data type at different times during program execution

Creating Variables in Python

Unlike other programming languages, Python has no command for declaring a variable. At the moment of creating a variable, the value should be assigned.

Example 1:

Code Output

```
File Edit Format Run Op
a=10
b="Saman"
print(a)
print(b)
```

```
>>>
10
Saman
>>>
```

Example 2:

Code Output

```
Python 2.7.8: ex1.py - D:/Python/ex1.py
File Edit Format Run Options Windows Help
a=6 # a is of type int
a="Kamal" # a is now a type str
print(a)
```

```
>>>
Kamal
>>>
```

Concatenation in python

In combine operation, Python uses the "+" operator

Example 3:

Code Output

```
File Edit Format Run Options Windows Help
x = "a programming language"
print("Python is " + x)
```

```
>>>
Python is a programming language
>>>
```

Standard Data types in python

Numbers

- Integral
 - Integer
 - Boolean
- Real
- Complex

Sequences

- Immutable sequences
 - Strings
 - Tuples
 - Bytes
 -
- Mutable sequences
 - Lists
 - Byte arrays

Set types

- Sets
- Frozen sets

Mappings

- Dictionaries

Variables of numeric types are created when you assign a value to them:

- `x = 1` `# int`
- `y = 2.8` `# float`
- `z = 1j` `# complex`

Specify a data type to a variable

There are situations to specify a type to a variable in a program. This can be done with the casting of the data type. Python uses classes to define data types.

- `int()` - constructs an integer number from an integer literal, a float literal or a string literal.
- `float()` - constructs a float number from an integer literal, a float literal or a string literal.
- `str()` - constructs a string from a wide variety of data types, including strings, integer literals and float literals.

Example 4:

Integer:

Code	Output
<pre>ex3.py.py - C:/Python33/python program/ex3.py.py File Edit Format Run Options Windows Hel a = int(5) b = int(4.6) c = int("2") print("a = ", a) print("b = ", b) print("c = ", c)</pre>	<pre>>>> a = 5 b = 4 c = 2 >>></pre>

Floats:

Code	Output
<pre>File Edit Format Run Options Windows Help a = float(2) b = float(3.8) c = float("4") d = float("5.2") print('a =', a) print('b =', b) print('c =', c) print('d =', d)</pre>	<pre>>>> a = 2.0 b = 3.8 c = 4.0 d = 5.2 >>></pre>

Example 6:

Strings:

Code	Output
<pre>File Edit Format Run Options Windows Help a = str("NIE") b = str(4) c = str(8.0) print("a =", a) print("b =", b) print("c =", c)</pre>	<pre>>>> a = NIE b = 4 c = 8.0 >>></pre>

String Literals

String literals in Python are surrounded by either single quotation mark or double quotation mark.

'hello' is same as "hello"

There are many string functions available in Python to handle strings literals.

Example 7:

Code

Output

```
File Edit Format Run Options Windows Help
a = "Welcome to, Ice Age!" World!"
print(a[1])
print(a[3:5])
print(a.strip())
print(len(a))
print(a.lower())
print(a.upper())
print(a.replace("e", "a"))
print(a.split(" ",))

>>>
e
co
Welcome to, Ice Age!" World!
28
welcome to, ice age!" world!
WELCOME TO, ICE AGE!" WORLD!
Walcoma to, Ica Aga!" World!
['Welcome to, Ice Age!" World!']
>>>
```

User Input:

Python allows the user to input data through command line input.

Example 8:

The following example asks for the user's name, by using the **input()** method, the program prints the name on the screen:

Code

Output

```
File Edit Format Run Options Windows Help
print("Enter your name :")
a = input()
print("Hello, " + a)

>>>
Enter your name :
kamal
Hello, kamal
>>>
```

Operators in Python

Operators are used to perform operations on variables and values.

Python operators:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Bitwise operators

Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Example9:

Code

```
File Edit Format Run Options Windows Help
M1 = 7
M2 = 8
print("M1+M2 =" , M1+M2)
```

Output

```
>>>
M1+M2 = 15
>>>
```

Example 10:

CodeOutput

```
File Edit Format Run Optio
a = 6
b = "Ravi"
print(a+b)
```

```
>>>
Traceback (most recent call last):
  File "C:\Python33\python program\Pyhon programme\ex11.py", line 3,
in <module>
    print(a+b)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>>
```

Note: Integer and string cannot be calculated. Try calculations using different data types.

Assignment Operators

Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	c = 5	c = 5
+=	m += 3	m = m + 3

Comparison Operators:

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	a == b
!=	Not equal	a != b
>	Greater than	a > y
<	Less than	b < y
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b

Logical Operators:

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	$a < 5$ and $b < 10$
or	Returns True if one of the statements is true	$a < 5$ or $b < 4$
not	Reverse the result, returns False if the result is true	not($a < 5$ and $b < 10$)

Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

Operator	Name	Description	Example Assume a=60, b=13
&	AND	Sets each bit to 1 if both bits are 1	$(a \& b) = 12$ (means 0000 1100)
	OR	Sets each bit to 1 if one of two bits is 1	$(a b) = 61$ (means 0011 1101)
~	NOT	Inverts all the bits	$(\sim a) = -61$ (means 1100 0011 in 2's complement)

Operators Precedence

The following table shows the precedence from the higher-order to the lower.

Operator	Description
**	Exponentiation (raise to the power)
~	Complement
+, -	unary plus and minus (e.g:- x++,x--)
*, /, %, //	Multiply, divide, modulo and floor division
+, -	Addition and subtraction
>>, <<	Right and left bitwise shift
&	Bitwise 'AND'
^	Bitwise 'XOR' (exclusive 'OR')
	Bitwise 'OR'
<=, <>, >=	Comparison operators
<>, ==, !=	Equality operators
=, %=, /=, //, -=, +=, *=, **=	Assignment operators
is, is not	Identity operators
in, not in	Membership operators
not	Logical operator
and	Logical operator
or	Logical operator

Competency Level 9.8: Uses control structures in developing programs

Learning Outcomes:

- Briefly describes control structures
- Lists and briefly describes the types of control structures
- Uses control structures appropriately in programming
- Applies nested control structures in programs

Control Structures

The flow of Control:

The flow of control in a program is implemented with three basic types of control structures:

- **Sequential:** Code statements are executed sequentially. (One line after another)
- **Selection:** Used to specify one or more conditions to be tested by the program.
- Python programming language provides the following decision-making statements:
 - if
 - if-else
 - if-elif-else
- **Repetition:** Used to execute a statement or group of statements multiple times. In Python, the following loops are used for repetition:
 - while
 - for

Indentation

To define a scope of a statement in the program, Python programming language uses indentation. Other programming languages often use curly-brackets for this purpose.

```
if b>a:
    print ("b greater than a")
```

Indentation

Example 11:

If statement:

Code

```
x = int(input("Enter your ICT marks :"))
if x>=75:
    print("You are qualified for scholarship")
```

Output

```
>>>
Enter your ICT marks :85
You are qualified for scholarship
>>>
```

Python language provides multi-way conditional decision statements.

- **If-else Statement**
- **if, elif, else**

Example 12: If-else Statement

Code

```
File Edit Format Run Options Windows Help
x = int(input("Enter your ICT marks :"))
if x >= 75:
    print("You are qualified for scholarship")
else:
    print("You are not qualified")
```

Output

```
>>>
Enter your ICT marks :58
You are not qualified
>>>
```

Example 13: if, elif, else

Code

```
File Edit Format Run Options Windows Help
a = float(input("Enter your z-score : "))
if a>=2.65:
    print("you are eligible to study computer science")
elif a>=2.40:
    print("you are eligible to study information systems")
else:
    print("you are eligible to study ICT")
```

Output

```
>>>
Enter your z-score : 2.32
you are eligible to study ICT
>>>
```

Loops in Python

- while statement
- for statement

While loops:

```
while condition :
    block
```

Example 14:

The following program prints numbers from 1 to 5.

Code

```
File Edit Format Run Options Windows Help
i = 1 # initialization
while i < 6: # condition
    print(i) # display
    i += 1 # increment
>>>
1
2
3
4
5
>>>
```

Output

Python for loop:

A for loop is used to repeat a block of statements for a known number of times.

Syntax

for *variable name* in structure:
 suit

Example 15:

Code Output

```
File Edit Format Run Options Windows Help
animals = ["cat", "dog", "fish"]
for a in animals:
    print(a)
```

```
>>>
cat
dog
fish
>>>
```

Variable Sui Structure

The break statement

With the break statement we can stop the loop even if the while condition is true:

Example 16:

Exit the loop when i is 3:

CodeOutput

```
% ex17.py - C:\Python33\python program\Python program
File Edit Format Run Options Windows Help
x = 1
while x < 6:
    print(x, end=' ')
    if (x == 3):
        break
    x += 1
```

```
>>>
1 2 3
>>>
```

With the break statement we can stop the loop before it has looped through all the items:

Example 17:

Code

Output

```
File Edit Format Run Options Windows Help
animals = ["cat", "dog", "fish"]
for i in animals:
    if i == "dog":
        break
    print(i)
```

```
>>>
cat
>>>
```

The Continue Statement

The current iteration stops with the continue statement and continues with the next:

Example 18:

Continue to the next iteration

if i is 3:

Code Output

```
File Edit Format Run Options
a = 0
while a < 6:
  a += 1
  if a == 3:
    continue
  print(a)
>>>
1
2
4
5
6
>>>
```

Example 19:

Code

Output

```
File Edit Format Run Options Windows Help
animals = ["cat", "dog", "fish"]
for i in animals:
  if i == "dog":
    continue
  print(i)
>>>
cat
fish
>>>
```

Competency Level 9.9: Uses sub-programs in programming

Learning Outcomes:

- Briefly describes the functions
- Lists and briefly describes the types of functions
- Identifies the structure of a function
- Compares local and global variables
- Identifies the behavior of a variable in terms of life time
- Identifies the need of return values and writes functions to obtain the appropriate return value
- Writes functions using relevant parameters and arguments
- Uses user defined functions

Function

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. So the function can return data as a result.

Creating Functions in Python

In Python a function is defined using the **def** keyword:

Example:

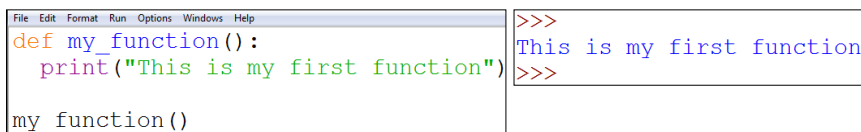
```
def my_function():  
    print("This is my first function")
```

Calling a Function

To call a function, use the function name followed by parenthesis:

Example 20:

CodeOutput



The screenshot shows a Python IDE window with a menu bar (File, Edit, Format, Run, Options, Windows, Help). The code editor contains the following code:

```
def my_function():  
    print("This is my first function")  
  
my_function()
```


To the right of the code editor, the output of the execution is shown in a separate window:

```
>>>  
This is my first function  
>>>
```

Parameters

Information can be passed to functions as a parameter.

Parameters are specified after the function name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

The following example has a function with **one parameter (fname)**. When the function is called, we pass along a first name, which is used inside the function to print the full name:

Example 21:

Code

```
File Edit Format Run Options Windows Help
def family_name():
    fname=input("Enter Your Name :")
    print(fname + " De Silva")
```

Output

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.3.1 (v3.3.1:d9893d13c628, Apr 6 2013, 20:25:12) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
>>> ===== RESTART =====
>>>
>>> family_name()
Enter Your Name :Kamal
Kamal De Silva
>>>
```

Default Parameter Value

The following example shows how to use a default parameter value.

If we call the function without parameter, it uses the default value:

Example 22:

Code

```
File Edit Format Run Options Windows Help
def add(a,b):
    c=a+b
    print("a+b= ",c)
```

Output

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.3.1 (v3.3.1:d9893d13c628, Apr 6 2013, 20:25:12) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
>>> add(3,8)
a+b= 11
>>>
```

Return Values

To let a function return a value, use the return statement:

Example 23:

Code Output

```
File Edit Format Run Options Windows Help
def return_val(a):
    return a * 2

print(return_val(4))
print(return_val(2))
print(return_val(6))
>>>
8
4
12
>>>
```

Scope of Variables

All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.

The scope of a variable determines the portion of the program where you can access a particular identifier.

There are two basic scopes of variables in Python:

- Global variables
- Local variables

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas the global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope.

Example 24:

Code

```
File Edit Format Run Options Window Help
r=float(input("Enter the radius value in cm "))
pi=3.14
def circle_area():
    area=pi*r*r
    print("Area of the circle for radius", r, " : ", area, "square cm")
def circle_circumference():
    circum=2*pi*r
    print("circumference of the circle for radius ", r, " cm : ", circum, "cm")
```

Output

```
===== RESTART: C:\Users\NIE\Desktop\glo.py =====
Enter the radius value in cm 4
>>> circle_area()
Area of the circle for radius 4.0 : 50.24 square cm
>>> circle_circumference()
circumference of the circle for radius 4.0 cm : 25.12 cm
>>> |
```

Competency Level 9.10: Uses data structures in programs

Learning Outcomes:

- Briefly explains the use of data structures
- Uses relevant data structures in programming

Data structures:

- Strings
- Lists
- Tuples
- Dictionaries

List

A list is a collection, which is ordered and changeable. In Python, lists are written with square brackets.

Example 25:

Create List

Code

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
print(mylist)
```

Output

```
>>>
['cat', 'dog', 'fish']
>>>
```

Example 26:

Print the second item of the list:

Code

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
print(mylist[1])
```

Output

```
>>>
dog
>>>
```

Example 27:

Change the second item as an ant

Code

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
mylist[1] = "ant"

print(mylist)
```

Output

```
>>>
['cat', 'ant', 'fish']
>>>
```

Example 28:

Print all items in the list, one by one:

Code

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
for b in mylist:
    print(b)
```

Output

```
>>>
cat
dog
fish
>>>
```

Example 29:

Check if "cat" is present in the list

Code

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
if "cat" in mylist:
    print("Yes, 'cat' is in my list")
```

Output

```
>>>
Yes, 'cat' is in my list
>>>
```

Example 30:

Print the total number of items in the list:

Code

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
print(len(mylist))
```

Output

```
>>>
3
>>>
```

Add an item to the list

The append() method uses to append an item in a list

Example 31:

Appending ant to the list

Code

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
mylist.append("ant")
print(mylist)
```

Output

```
>>>
['cat', 'dog', 'fish', 'ant']
>>>
```

Inserting items in the middle of a list

Example 32:

Insert ant in the second position in the list

Code

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
mylist.insert(1, "ant")
print(mylist)
```

Output

```
>>>
['cat', 'ant', 'dog', 'fish']
>>>
```

Removing Items from a list

The remove() method removes the specific item:

Example 33:

Removing the dog from the list

Code

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
mylist.remove("dog")
print(mylist)
```

Output

```
>>>
['cat', 'fish']
>>>
```


Removing the specified item from the list

The `del` keyword removes the specified indexed item:

Example 34:

Removing **zero-indexed** item from the list, mylist

Code

Output

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
del mylist[0]
print(mylist)
>>>
['dog', 'fish']
>>>
```

Clearing the list

The `clear()` method empties the list:

Example 35:

To clear all the items from mylist

Code

Output

```
File Edit Format Run Options Windows Help
mylist = ["cat", "dog", "fish"]
mylist.clear()
print(mylist)
>>>
[]
>>>
```

Python has a set of built-in methods that you can use on lists.

Method	Description
<code>append()</code>	Adds an element at the end of the list
<code>clear()</code>	Removes all the elements from the list
<code>copy()</code>	Returns a copy of the list
<code>count()</code>	Returns the number of elements with the specified value
<code>extend()</code>	Add the elements of a list (or any iterable), to the end of the current list
<code>index()</code>	Returns the index of the first element with the specified value
<code>insert()</code>	Adds an element at the specified position
<code>pop()</code>	Removes the element at the specified position
<code>remove()</code>	Removes the item with the specified value
<code>reverse()</code>	Reverses the order of the list
<code>sort()</code>	Sorts the list

Note: Some methods already discussed above

Tuples

A tuple is a collection which is ordered and **unchangeable (immutable)**. In Python, tuples are written with round brackets.

Example 36:

Create a Tuple

Code

```
File Edit Format Run Options Windows Help
cityname = ("Kandy", "Jaffna", "Colombo")
print(cityname)
```

Output

```
>>>
('Kandy', 'Jaffna', 'Colombo')
>>>
```

Example 37:

Return the item in position 1:

Code

```
File Edit Format Run Options Windows Help
cityname = ("Kandy", "Jaffna", "Colombo")
print(cityname[1])
```

Output

```
>>>
Jaffna
>>>
```

Example 38:

As discussed earlier values in a tuple cannot be changed

Code

```
File Edit Format Run Options Windows Help
cityname = ("Kandy", "Jaffna", "Colombo")

cityname[1] = "Galle"
# the value is still the same:
print(cityname)
```

Output

```
>>>
Traceback (most recent call last):
  File "C:\Python33\python program\Pyhon programme\ex39.py",
    line 3, in <module>
    cityname[1] = "Galle"
TypeError: 'tuple' object does not support item assignment
>>>
```

Example 39:

Iterate items and print the values:

Code

```
File Edit Format Run Options Windows Help
subjectTuple = ("Maths", "Science", "ICT")
for x in subjectTuple:
    print(x)
```

Output

```
>>>
Maths
Science
ICT
>>>
```

Example 40:

Checking the availability of an item in a tuple

Check if "ICT" is present in the tuple:

Code

```
File Edit Format Run Options Windows Help
subjectTuple = ("Maths", "Science", "ICT")
if "ICT" in subjectTuple:
    print("Yes, 'ICT' is in the fruits tuple")
```

Output

```
>>>
Yes, 'ICT' is in the fruits tuple
>>>
```

Example 41:

Print the number of items in the tuple:

Code

Output

```
File Edit Format Run Options Windows Help >>>
subjectTuple = ("Maths", "Science", "ICT") 3
print(len(subjectTuple)) >>>
```

Tuples:

Tuples are immutable, therefore, cannot remove items, but can delete the tuple.

Built-in methods in tuples

Method	Description
<u>count()</u>	Returns the number of times a specified value occurs in a tuple
<u>index()</u>	Searches the tuple for a specified value and returns the position of where it was found

Dictionaries

A dictionary is a collection that is unordered, changeable and indexed. In Python, dictionaries are written with curly brackets, and they have keys and values.

Example 42:

Create and print a dictionary

Code

```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019
}
print(Stu_details)
```

Output

```
{'Name': 'Saman', 'country': 'Sri Lanka', 'Year': 2019}
>>> |
```

Get the value of a key.

get() method can be also used to get the value of the key.

Example 43:

Get the value of the "Country" key

CodeOutput

```
File Edit Format Run Options Windows Help >>>
Stu_details = { Sri Lanka
    "Name": "Saman", Sri Lanka
    "Country": "Sri Lanka",
    "year": 2019 >>>
}
x = Stu_details["Country"]
print(x)
x = Stu_details.get("Country")
print(x)
```

Changing the values of the keys:

Example 44:

Change the "year" value from 2019 to 2018:

Code

```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019
}

Stu_details["year"] = 2018

print(Stu_details)
```

Output

```
{'Name': 'Saman', 'Country': 'Sri Lanka', 'year': 2018}
>>> |
```

Printing the key names.

Example 45:

Printing key names from the Dictionary

Code Output

```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019
}
for a in Stu_details:
    print(a)
```

```
Name
Country
year
>>> |
```

Printing the key values

Example 46:

Printing all key values from a dictionary

Code

```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019
}
for a in Stu_details :
    print(Stu_details[a])
```

Output

```
>>>
=====
Saman
Sri Lanka
2019
>>>
```

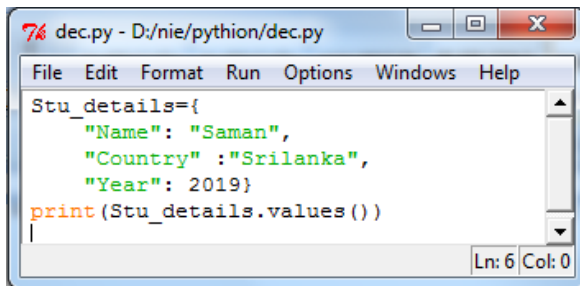
You can also use the **values()** function to return the values of a dictionary.

Loop through both keys and values, by using the **items()** function:

Example 47:

Checking the existing items in a dictionary using the item() function

Code



```
7% dec.py - D:/nie/pythion/dec.py
File Edit Format Run Options Windows Help
Stu_details={
    "Name": "Saman",
    "Country": "Srilanka",
    "Year": 2019}
print(Stu_details.values())
Ln: 6 Col: 0
```

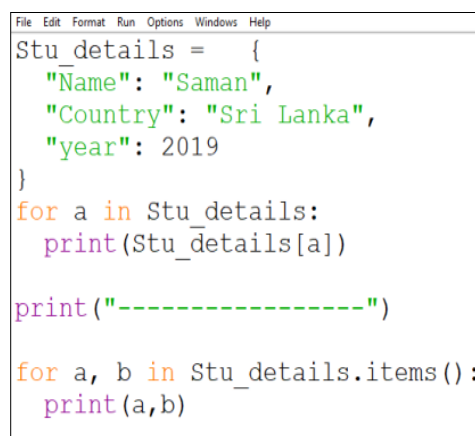
Output

```
dict_values(['Saman', 'Sri Lanka', 2019])
>>> |
```

Example 48:

Checking whether an existing item in the dictionary

Input



```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019
}
for a in Stu_details:
    print(Stu_details[a])
print("-----")
for a, b in Stu_details.items():
    print(a,b)
```

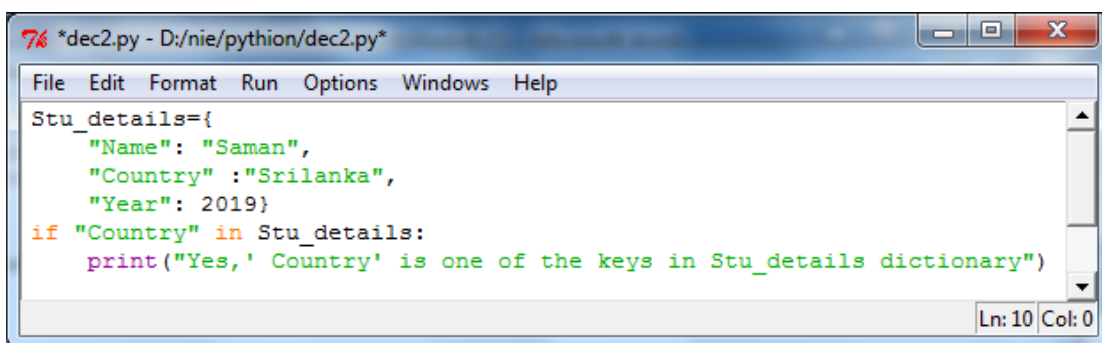
Output

```
>>>
===== RESTART: H:
Saman
Sri Lanka
2019
-----
Name Saman
Country Sri Lanka
year 2019
>>>
```

Example 49:

Check if "Country" is present in the dictionary

Code



```
7% *dec2.py - D:/nie/pythion/dec2.py*
File Edit Format Run Options Windows Help
Stu_details={
    "Name": "Saman",
    "Country": "Srilanka",
    "Year": 2019}
if "Country" in Stu_details:
    print("Yes, 'Country' is one of the keys in Stu_details dictionary")
Ln: 10 Col: 0
```

Output

```
Yes, 'Country' is one of the keys in the dictionary1 dictionary
>>> |
```

Example 50:

Print the number of items in the dictionary

Code Output

```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019 }
print(len(Stu_details))
```

```
>>>
=====
3
>>>
```

Adding an item to the dictionary, with a new index key and assign a value.

Example 51:

To the school index key, add value 'ABC' in the Stu_details dictionary

Code

```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019 }

Stu_details["School"] = "ABC"

print(Stu_details)
```

Output

```
{'Name': 'Saman', 'Country': 'Sri Lanka', 'year': 2019, 'School': 'ABC'}
>>>
```

Removing Items

There are several methods to remove items from a dictionary:

The **pop()** method removes the item with the specified key name

Example 52:

Removing the country from the dictionary

Code

```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019 }

Stu_details.pop("Country")

print(Stu_details)
```

Output

```
{'Name': 'Saman', 'year': 2019}
>>>
```

The **popitem()** method removes the last inserted item.

Example 53:

Removing the last inserted item

Code

```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019 }

Stu_details.popitem()
print(Stu_details)
```

Output

```
{'Name': 'Saman', 'Country': 'Sri Lanka'}
>>>
```

The **del** keyword removes the item with the specified key name.

Example 54:

Remove country from dictionary

Code

```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019 }

del Stu_details ["Country"]

print(Stu_details)
```

Output

```
{'Name': 'Saman', 'year': 2019}
>>>
```

The **clear()** keyword empties the dictionary.

Example 55:

Clear the dictionary

Code

```
File Edit Format Run Options Windows Help
Stu_details = {
    "Name": "Saman",
    "Country": "Sri Lanka",
    "year": 2019 }

Stu_details.clear()

print(Stu_details)
```

Output

```
{ }
>>>
```

Dictionary Methods

Python has a set of built-in methods that you can use on dictionaries.

Method	Description
<code>clear()</code>	Removes all the elements from the dictionary
<code>fromkeys()</code>	Returns a dictionary with the specified keys and values
<code>get()</code>	Returns the value of the specified key
<code>items()</code>	Returns a list containing the tuple for each key-value pair
<code>keys()</code>	Returns a list containing the dictionary's keys
<code>pop()</code>	Removes the element with the specified key
<code>popitem()</code>	Removes the last inserted key-value pair
<code>update()</code>	Updates the dictionary with the specified key-value pairs
<code>values()</code>	Returns a list of all the values in the dictionary

Competency Level 9.11: Handles files and databases in programs

Learning Outcomes:

- Uses basic file operations (open, close, read write and append)

Python File Handling

- Python has several functions for creating, reading, updating, and deleting files
- Uses basic file operations (open, close, read-write and append)
- The key function for working with files in Python is the open() function

The open() function takes two parameters; **filename** and **mode**.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

In addition, you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

Syntax

To open a file for reading it is enough to specify the name of the file:

```
f = open("n1.txt")
```

The code above is the same as:

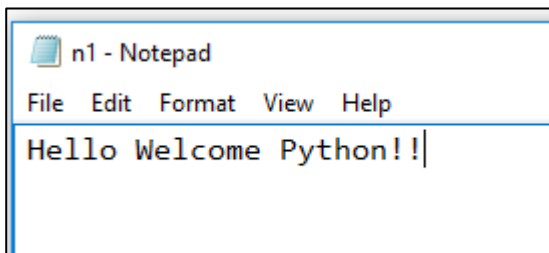
```
f = open("n1.txt", "rt")
```

Because "r" for read, and "t" for text are the default values, you do not need to specify them.

Note: Make sure the file exists, or else you will get an error.

Open a File on the Server

Assume we have the following file, located in the same folder as Python:



To open the file, use the built-in **open()** function.

The **open()** function returns a file object which has a **read()** method for reading the content of the file:

Example 56:

Code

```
f = open("n1.txt", "r")
print(f.read())
```

Output

```
===== RESTART: D:/
Hello Welcome Python!!
>>> |
```

Read-Only Parts of the File

By default, the **read()** method returns the whole text, but you can also specify how many characters you want to return:

Example 57:Return the 5 first characters of the file:

Code

```
f = open("n1.txt", "r")
print(f.read(5))
```

Output

```
>>>
RESTART
Hello
>>> |
```

Read Lines

You can return one line by using the **readline()** method:

Example 58:

Read one line of the file

Code

```
f = open("n1.txt", "r")
print(f.readline())
```

Output

```
===== RESTART:
Hello Welcome Python!!
>>> |
```

Example 59:

Code

```
f = open("n1.txt", "r")
for x in f:
    print(x)
```

Output

```
===== RESTART:
Hello Welcome Python!!
>>> |
```

Write to an existing file

To write to an existing file, you must add a parameter to the **open()** function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

1. Open the file "n1.txt" and append content to the file:

```
f = open("n1.txt", "a")
```

```
f.write("Now the file has one more line!")
```

2. Open the file "n1.txt" and overwrite the content:

```
f = open("n1.txt", "w")
```

```
f.write("Woops! I have deleted the content!")
```

note: the "w" method will overwrite the entire file

Create a new file

To create a new file in Python, use the **open()** method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exists

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

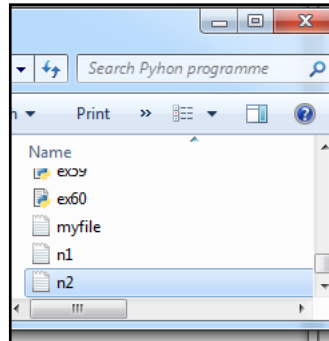
Example 60:

Create a file called "n2.txt":

Code

```
f = open("n2.txt", "x")
```

Output



Delete a file

To delete a file, you must import the OS module, and run its `os.remove()` function:

Remove the file "n1.txt":

```
import os
os.remove("n1.txt")
```

Check if file exists and then delete

To avoid getting an error, you check if the file exists before you try to delete it:

Check if file exists and then delete the file:

```
import os
if os.path.exists("n1.txt"):
    os.remove("n1.txt")
else:
    print("The file does not exist")
```

Delete an empty folder

To delete an entire folder, use the `os.rmdir()` method:

Remove the folder "myfolder":

```
import os
os.rmdir("myfolder")
```

Note: You can only remove **empty folders**

Competency Level 9.12: Manages data in databases

Learning Outcomes:

- Embeds SQL statements in programming languages to retrieve, add, modify and delete data

Python MySQL

Python can be used in database applications. One of the most popular databases is MySQL.

MySQL Database

Download the following free software to experiment with the code examples of MySQL:

MySQL database:	https://www.mysql.com/downloads/
WAMP server:	https://sourceforge.net/projects/wampserver/
XAMPP server:	https://www.apachefriends.org/index.html

Install MySQL Driver

Python needs MySQL driver to access the MySQL database; therefore "MySQL Connector" driver is used. PIP can be used to install "MySQL Connector". PIP is a package manager for Python packages or modules. PIP is most likely installed in the Python environment (When installing a Python, select "Customize installation" instead of "Install Now". Then PIP will be installed automatically).

Navigate the command line to the location of Python's script directory, and type the following to check whether PIP is installed.

```
C:\Users\your name\AppData\Local\Programs\Python\Python37-32\Scripts> pip --version
```

If PIP is not installed, download and install it from this page:

<https://pypi.org/project/pip/>

Navigate the command line to the location of Python's script directory, and type the following to download and install "MySQL Connector":

```
C:\Users\your name\AppData\Local\Programs\Python\Python37-32\Scripts>pip install MySQL-connector
```

Test MySQL Connector

To test if the installation was successful, or if you already have "MySQL Connector" installed, create a Python page with the following content. If the below code was executed with no errors, "MySQL Connector" is installed and ready to be used.

```
import mysql.connector
```

Create Connection

Use the username and password of MySQL database to create a connection

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    passwd="yourpassword"
)
print(mydb)
```

Normally default username is "root" and no password

MySQL Create Database

Example 61:

Create a database named "school"

```
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    passwd="yourpassword"
)
mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE school")
```

MySQL-DeleteDatabase

Example 62:

Delete school database

```
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    passwd="yourpassword"
)
mycursor = mydb.cursor()
mycursor.execute("DROP DATABASE school")
```

MySQL-Create Table

Example 63:

Create student table with: **regNo, name, address and contactNo** fields
Define the name of the database when creating the connection

```
import mysql.connector
|
mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    passwd="yourpassword",
    database="school"
)
mycursor = mydb.cursor()
mycursor.execute("CREATE TABLE student (
regNo VARCHAR(10) NOT NULL,
name VARCHAR(150) NOT NULL,
address VARCHAR(250),
contactNo VARCHAR(10), PRIMARY KEY(regNo))")
|
```

MySQL-Modify Table

Example 64:

Add a new field called "dob" to the student table

```
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    passwd="yourpassword",
    database="school"
)
mycursor = mydb.cursor()
```

MySQL- Drop Table

Example 65:

Delete student table

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    passwd="yourpassword",
    database="school"
)
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE student")
|
```

MySQL-Insert data into the Table

To fill a table in MySQL, "INSERT INTO" statement is used *mydb.commit()* is required to make the changes, otherwise no changes are made the table.

Example 66:

Insert a record in the "student" table

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    passwd="yourpassword",
    database="school"
)
mycursor = mydb.cursor()
sql = "INSERT INTO student (regNo, name, address,contactNo, dob) VALUES (%s, %s, %s, %s, %s)"
val = ("r001","Ravi", "Colombo 5", "0715623410","2000-10-21")
mycursor.execute(sql, val)
mydb.commit()
print(mycursor.rowcount, "record inserted.")
```

Output:

1 record inserted.

('r001', 'Ravi', Colombo 5, '0715874510', '2000-10-21'),

Example 67:

Insert multiple rows

To insert multiple rows into a table, use the **executemany()** method. The second parameter of the **executemany()** method is a list of tuples, containing the data that want to be inserted.

```
sql = "INSERT INTO student (regNo, name, address,contactNo, dob) VALUES (%s, %s, %s, %s, %s)"
val = [
    ('r002','Mala', 'Anurathapura', '0715874510', '2001-06-14'),
    ('r003','Geetha', 'Kandy', '0775857410', '2001-02-12'),
    ('r004','Kumara', 'Vavuniya', '0710055210', '2000-08-13')
]
mycursor.executemany(sql, val)
mydb.commit()
print(mycursor.rowcount, "was inserted.")
```

Output:

3 records were inserted.

MySQL Select Data

1. Select all records
2. Select specific columns
3. Select with filter
4. Sort the result

To select from a table in MySQL, use the "SELECT" statement. The **fetchall()** method fetches all rows from the last executed statement. **fetchone()** method fetches only one row.

MySQL Select all records

Example 68:

Select all records from the "student" table, and display the result

```
sql = "INSERT INTO student (regNo, name, address,contactNo, dob) VALUES (%s, %s, %s, %s, %s)"
val = [
    ('r002','Mala', 'Anurathapura', '0715874510', '2001-06-14'),
    ('r003','Geetha', 'Kandy', '0775857410', '2001-02-12'),
    ('r004','Kumara', 'Vavuniya', '0710055210', '2000-08-13')
]
mycursor.executemany(sql, val)
mydb.commit()
print(mycursor.rowcount, "was inserted.")
```

Output:

```
('r001', 'Ravi', ', Colombo 5', '0715874510', '2000-10-21')
('r002','Mala', 'Anurathapura', '0715874510', '2001-06-14')
('r003','Geetha', 'Kandy', '0775857410', '2001-02-12')
('r004','Kumara', 'Vavuniya', '0710055210', '2000-08-13')
```

MySQL Select specific columns

Example 69:

Select only the regNo, name and address columns:

```
mycursor.execute("SELECT regNo, name, address FROM student")
myresult = mycursor.fetchall()
for x in myresult:
    print(x)
```

Output

```
('r001', 'Ravi', 'Colombo
5')
('r002','Mala',
'Anurathapura')
```

Select with filter

When selecting records from a table, selection can be filtered by using the "WHERE" statement

Example 70:

Select record(s) where the birthday is "2001-06-14"

```
mycursor.execute("SELECT * FROM student WHERE dob='2001-06-14 '")
```

Output

```
('r002','Mala', 'Anurathapura', '0715874510', '2001-06-14')
```

Example 71:

Select records where the contactNo starts with "071"

```
mycmymcursor.execute("SELECT name, contactNo FROM student WHERE contactNo LIKE '071%'")
```

Output

```
('r004','Kumara', 'Vavuniya', '0710055210', '2000-08-13')
```


MySQL Sort the result

Use the ORDER BY statement to sort the result in ascending or descending order
The ORDER BY keyword sorts the result ascending by default. To sort the result in descending order, use the DESC keyword

Example 72:

Sort the result alphabetically by name

```
mycursor.execute("SELECT * FROM student ORDER BY name ")
```

Output

```
('r003','Geetha', 'Kandy', '0775857410', '2001-02-12')  
(r004,'Kumara', 'Vavuniya', '0710055210', '2000-08-13')  
(r002,'Mala', 'Anurathapura', '0715874510', '2001-06-14')  
(r001', 'Ravi', ', Colombo 5', '0715874510', '2000-10-21')
```

Example 73:

Display birthdays in descending order

```
mycursor.execute("SELECT name, dob FROM student ORDER BY dob DESC ")
```

Output

```
('r002','Mala', 'Anurathapura', '0715874510', '2001-06-14')  
(r003','Geetha', 'Kandy', '0775857410', '2001-02-12')  
(r001', 'Ravi', ', Colombo 5', '0715874510', '2000-10-21')  
(r004','Kumara', 'Vavuniya', '0710055210', '2000-08-13')
```

MySQL Update Table

Existing records can be updated in a table by using the "UPDATE" statement

Example 74:

Overwrite the name column from "Ravi" to "Sami".

```
import mysql.connector  
  
mydb = mysql.connector.connect(  
    host="localhost",  
    user="yourusername",  
    passwd="yourpassword",  
    database="school"  
)  
mycursor = mydb.cursor()  
sql = "UPDATE student SET name='Sami' WHERE name='Ravi' "  
mycursor.execute(sql)  
mydb.commit()  
print(mycursor.rowcount, "record(s) affected")
```

Output

```
1 record(s) affected
```

MySQL Delete

Records can be deleted from an existing table by using the "DELETE FROM" statement.

Example 75:

Delete any record where the address is "Kandy"

```
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    passwd="yourpassword",
    database="school"
)
mycursor = mydb.cursor()
sql = "DELETE FROM student WHERE address='Kandy' "
mycursor.execute(sql)
mydb.commit()
print(mycursor.rowcount, "record(s) deleted")
```

Output

```
1 record(s) deleted
```

Competency Level 9.13: Searches and sorts data

Learning Outcomes:

- Uses sequential searching technique appropriately
- Implements bubble sort technique appropriately

MySQL Searching Techniques

Searching is the algorithmic process of finding a particular item in a collection of items. A search typically answers either *True* or *False* as to whether the item is present.

In Python, there is a very easy way to ask whether an item is in a list of items. We use the *in* operator.

```
>>> 15 in [3, 5, 2, 4, 1]
False
>>> 3 in [3, 5, 2, 4, 1]
True
```

MySQL Sequential Search

When data items are stored in a collection such as a list, they have a linear or sequential relationship. Each data item is stored in a position relative to the others. In Python lists, these relative positions are the index values of the individual items. Since these index values are ordered, it is possible to visit them in sequence. In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data structure.

Sorting Techniques:

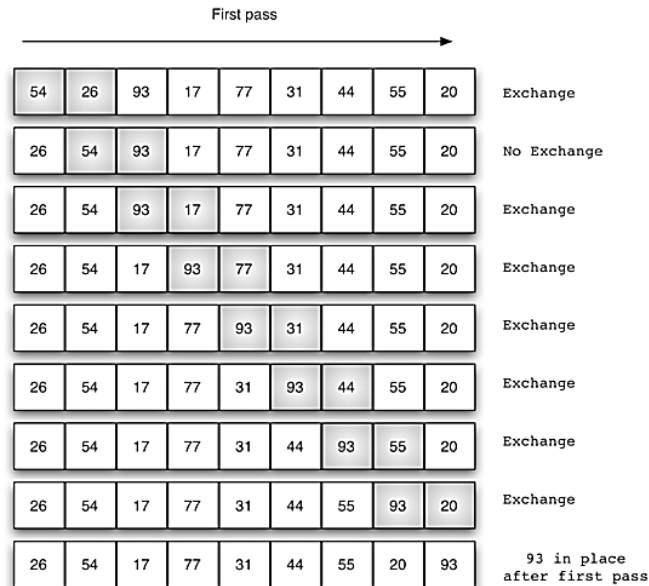
Sorting refers to arranging data in a particular format. It can be ascending or descending order. Sorting algorithm specifies the way to arrange data in a particular order.

Bubble sort:

The bubble sort makes multiple passes through a list. It compares adjacent items and exchanges those that are out of order. Each pass through the list places the next largest value in its proper place. In essence, each item “bubbles” up to the location where it belongs.

Sorting list in ascending order:

- compare 1st and 2nd elements
- if 1st is larger than the 2nd, swap
- compare 2nd and 3rd, and swap if necessary
- continue until compare the last two elements
- largest element is now at the last element in the array
- repeat starting from the beginning until no swaps are needed (i.e.the list is sorted)
- each time you travel the array, elements bubbling up the largest element to the end of the array



At the start of the second pass, the largest value is now in place. There are $n-1$ items left to sort, meaning that there will be $n-2$ pairs (n is a number of elements in the list). Since each pass places the next largest value in place, the total number of passes necessary will be $n-1$. After completing the $n-1$ passes, the smallest item must be in the correct position with no further processing required.

Bubble Sort as a Python Function

```
def bubble_sort(L):
    swapped = True # set flag to True to repeat sorting
    while swapped:
        swapped = False
        for i in range(len(L) - 1):
            if L[i] > L[i + 1]:
                # Swap the elements
                L[i], L[i + 1] = L[i + 1], L[i]
                # Set the flag to True so we'll loop again
                swapped = True
```

Review Questions

1.
 - Develop a Python program to accept an amount of money, to be paid by a customer in rupees, entered from the keyboard. If the amount is greater than or equal to Rs. 2,000 rupees, a 15% discount is given to the customer. Then display the final amount that the customer has to pay.
 - Modify the above program to display the message “No discount...” if the amount is less than 2,000.
2. Write a program to compute the sum of all even numbers up to 100, inclusive.
3. Develop a Python program to compute the sum of integers between two numbers given as input.
4. Develop a program to display the integers from 100 to 200.
5. Can the loop body of a for loop never get executed?
6. Develop a program to display the following symbol pattern using a nested loop:

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
```

The World Wide Web

Competency 10: Develops websites incorporating multi-media technologies (using HTML 5)

Competency Level 10.1: Explores the need for web

Learning Outcomes:

- Describes www
- Analyses the systematic arrangements of contents and structure of a web

The World Wide Web (WWW) invented by Tim Berners Lee in 1989 at the European Centre for Nuclear Research, as a major application on the Internet. It was the first application on the Internet that caught the attention of the public and it dramatically changed how people interact with the network of computers. Essentially, the World Wide Web (WWW) is an application that runs on top of the Internet, which is in turn made up of a lot of interconnected computers. According to Wikipedia, World Wide Web is an information system where documents and other web resources are identified by Uniform Resource Locators (URLs), which may be interlinked by hypertext, and are accessible over the Internet. The basic element in WWW is the web page that a user views when he or she accesses it. Some of the computers on the Internet, called web servers, are designed to serve the web pages. These web servers are always on computers and are ready to serve the web pages all the time. The other computers request web pages from the web servers and are called the client computers. The WWW operates on-demand, the users request web pages from web servers whenever they want and receive them from web servers. The WWW has many wonderful features that people like a lot; it provides a platform for any individual who has access to a computer and the Internet to make information available on the web at very less cost. Formatted Text, multimedia, graphics, forms, and applets are part of WWW and are accessible by the client computers.



WWW also serves as a platform for many innovative applications like search engines such as Google, other popular Google applications such as e-mail, Social media such as Facebook, multimedia streaming platforms such as YouTube, online photo sharing applications such as Flickr, etc.

A Web page can consist of a number of web objects such as a simple text file, for example, an HTML file, an image file such as a jpeg file, a Java applet, an audio file or a video clip. The text in a web page called hypertext means that it is the text, which contains links to other text. Any word in a hypertext document can be specified as a pointer to a different hypertext document where more information pertaining to that word can be found. A web page stored in a server computer can be accessed by issuing the address of the server, which houses the resource or object, called a Uniform Resource Locator (URL), or a hyperlink. Each URL has three components: the first part is the protocol, which tells the browser how to communicate with a website's server, in order to send and retrieve information, the second part is the domain name, which is an identifier to a specific site, and the third part is the path, which directs the browser to a specific page on the website. Typically a web page will consist of a base HTML file and

several other objects that are embedded into the base HTML file. Each embedded object will be referred to using its URL. If a given web page has n number of other objects referred in it, then we have a total of (n+1) objects on the web page including the base HTML file.

Web pages requested by clients can be viewed using a special application called web browsers. Present-day web browsers are intelligent applications and they provide many features to the users. Google Chrome, Mozilla Firefox, Internet Explorer and Apple's Safari are examples of popular web browsers. Web pages are written in HyperText Markup Language (HTML). The markup facility of the HTML enables the web browsers to render plain text, combined with additional tags, as easily readable attractive text on the display of computers. HTML Tags are symbols that indicate a visual element for the page, such as a heading, that governs the size, color, and other properties of the element. Each of these web browser applications have their strengths and weaknesses, and can render the HTML code slightly differently. It would be better that when HTML code is written it should be tested on each of this web browser application software for completeness.



Figure 3.1 some of the well-known web browsers

Competency Level 10.2: Analyses user requirements (multimedia contents)

Learning Outcomes:

- Creates effective and appropriate information layout of a website
 - Identifies the web pages of a website
 - Identifies the contents of a web page
- Identifies navigation structure

Analyses user requirements (multimedia contents)

Objectives of a web site should be defined well to build a great web site. It helps the designer to identify the requirements of the users, to define them and to satisfy the user requirements. A set of well defined such objectives and their usability solutions can increase the user experience and will keep the users regularly visit the web sites.

Some possible questions about the objectives of a website are:

1. Why do you need a new website?
2. What outcomes do you want to achieve?
3. Do you have an existing website? Are there any problems with it?
4. How do you plan to improve the user experience? E.g. Speed, content, clarity, etc
5. What do you try to boost through the new website?
6. Are you going to inform about new opportunities available in your organization?
7. Are you planning to expand your business?

You have to define some goals of having a website that helps you to achieve your objectives. For example, increasing the number of inquiries to the website or increasing online sales can be defined as goals for an online shopping website.

You may also want to identify the key audiences that you need to appeal to in order to reach your objectives. The set of your key audiences may include the potential customers of your business and your employees. Once you have identified the prospective audiences, it is better to list the priorities of these audiences and doing so will help you in planning the design layout and the navigation design of your website.

Requirements Gathering for a Website

User requirements of a web site describe the needs of users, the functionalities that the web site should do, and the goals of a web site. The designer of the website should define, document and describe the user requirements of the web site. These requirements would change over time and the designers should keep the website updated based on the changing requirements. Some of the possible requirements that may be collected before commencing the design are listed below:

- Content – Determining the content of the web site that needs to be updated on the web site is an essential part of the design.
- Layout of the design and the navigation pattern – Determining what would appear in the layout of the website, and how the user would be able to navigate from the home page to other pages.
- Usability – identifying the accessibility requirements of the web site.

- Security – identifying the security requirements of the web site.
- Loading times – determining the loading speed of a website depending on the kind of service it provides.
- Legal – identifying the legal implications that the web site must adhere to.

Competency Level 10.3: Identifies appropriate HTML tags to design a single web page

Learning Outcomes:

- Analyses the arrangement of contents of a web page
- Analyses the organization of contents in a web page
- Creates a simple web page

Identifies appropriate HTML tags to design a single web page

HTML Tags are symbols that indicate a visual element for the page, such as a heading, that governs the size, color, and other properties of the element. When a web browser loads a web page, it renders the content of the web page on the computer’s display according to the included HTML markup and content.

A number of tools are available to create HTML pages and for learning purposes, the notepad application available in desktop computer is sufficient. Notepad is a lightweight application that enables the typing of HTML pages without much trouble. Typed web pages could be saved using the extension .html. There exist multiple sophisticated tools that enable HTML typing and there even other online resources available to type in HTML.

The structure of an HTML page

An HTML page contains a series of tags; the larger tag is the <html> tag that contains two structural elements <head> and <body>. The essential code of an HTML5 looks like as follows:

Code snippet:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
  </body>

</html>
```

Output:

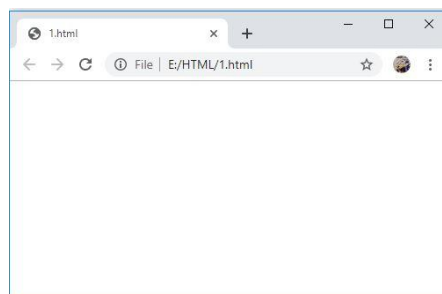


Figure 3.3.1

Figure 3.3.1 First example of html code to create an empty webpage.

The above document does not produce any output since it does not have any content to display. The <!DOCTYPE HTML> tells the browser what type of document is following the file. When the browser knows the type of the document, it can present it to the viewer of the document in a much faster manner.

The primary tag of a web page is the `<html>` tag. All the content that is presented to the browser for rendering should be included within the `<html>` and `</html>` pair.

Inside the `<html>` and `</html>` tags are two main components, the `<head>` and the `<body>`. The head section contains information about the current document, often referred to as metadata. This metadata may include the title of the document, keywords and descriptions that describe the page, author details, and copyright statements among other information. The `<title>` and `</title>` pair within the head are used to inform the title of the webpage which is displayed at the topmost bar of the browser.

The `<body>` and `</body>` contains all the content of the web page that can be rendered through the browser. Within the body pair, the code that can be used to style the web page is written. By including these two major sections of a web page, the structure of a web page will look like:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Title of the document which appears at the title bar of the browser
  </title>
</head>
  <body>
    The content of the document appears here
  </body>
</html>
```

Output:

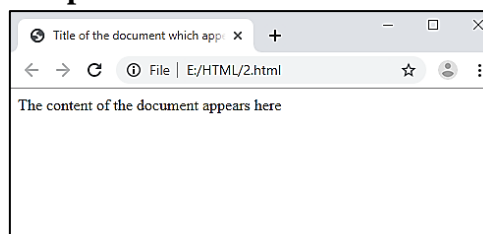


Figure 3.3.2

The `<body>` tag is capable of accepting numerous attributes, including the ID attribute, for example, `<body id="home">`. A page with a distinctive ID in the `<body>` tag can be targeted for specific styling using CSS. We will study more about the other possible attribute when we study about the CSS.

Typing in an HTML document

To type in an HTML document you can use a simple editor such as notepad, and to view the web page that you have created you may use a web browser such as Mozilla Firefox, Google Chrome or Apples Safari. You can type in the code as shown above in the text editor and then you can save the HTML document with a name using the `.html` extension, for example, `html_example1.html`. Now you may notice that the saved document has an icon of the browser that you use on your computer. The saved web document can now be viewed using a web browser, by double-clicking on the icon of the document and the output will look like:

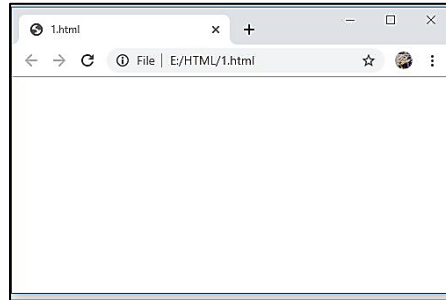


Figure 3.3.3

Background colour in HTML:

A web page can be assigned a background colour of choice by adding an extra attribute to the `<body>` tag.

E.g. `<body bgcolor="#E6E6FA">` can be used to assign a background colour to the html document. Syntax of it is `<body bgcolor="color_name|hex_number|rgb_number">`, where `color_name` can specify a particular colour such as “Blue”, `hex_number` can specify the hexa decimal code of a colour, and `rgb_number` can specify the RGB code.

Note: This facility for setting background colour is available in previous versions of HTML and, in HTML 5 these will come under CSS. We will cover them when we study CSS later in this chapter.

Heading tags in HTML:

Headings are used to separate and introduce major divisions within a web page. The `<h1>` to `<h6>` tags in HTML are used to define the six levels of headings on a web page according to their importance. `<h1>` defines the most important heading whereas the `<h6>` defines the least important heading.

Formatting text in HTML:

The `<p> .. </p>` pair defines paragraph in HTML and the text within the `<p> .. </p>` pair are displayed in the plain text format. In HTML, the tag `
` is used to insert a single line break. It can be used to break sentences into a number of separate lines. It is referred to as an empty tag since it does not have an ending tag. The ` .. ` tag pair is used to **bold** the text that is written within the pair. The `<u> .. </u>` tag pair is used to underline a given sentence written within the pair. The `<i> .. </i>` tag pair is used to define the part of a text in *italics*. The `.. ` pair is used to specify the font to be used in the text. It also allows specifying the font size and colour of the text. In between paragraphs of text, a `
` tag can be inserted to include a single line break in the text. The `
` tag is an empty tag and it does not have an end tag.

Note: At times developers of web pages need to include comments in their web pages for maintenance purposes. These comments help the developers to debug the code in case of errors in rendering the web pages. Comments within HTML code can be included using the `<!--` and `-->` tag pair.

Let us see an example of the above formatting tags in an html document:

Example Code:

```
<!DOCTYPE html>
<html>
  <head>
    <title> My new Webpage </title>
  </head>
  <body>
    <h1> Welcome to My New Webpage </h1>.
    <p> This is a paragraph of text.
    </p>
    <p>This is normal text - <b> and this is bold text </b></p>
    <p>This is normal text - <u> and this is underlined text </u></p>
    <p>He named his car <i>The lightning</i>, because it was very fast.</p>
    <font size="3" color="red"> This is some text! </font>
    <font size="2" color="blue"> This is some text! </font>
    <font face="verdana" color="green"> This is some text! </font>
    <!-- This is a comment -->
  </body>
</html>
```

When the above code is viewed using a web browser the output will look like:



Figure 3.3.4

Fig 3.3.4 html code to display and the corresponding output

Note: There are many other types of text formatting tags available in HTML. Students are encouraged to find them and try them by including them in the above code snippet by doing a self-search on the web. A good source for many of the HTML tags and attributes is w3school.com

Lists in HTML:

Lists are a common text element on the Web, often used to break up the page and highlight key points. When the list items do not need to be in any particular order, an unordered list is used.

Ordered Lists:

Ordered lists are mainly used when the numbering of the items in the list is important. Ordered lists can be created using the `..` tag pair. There are many numbering styles available in the ordered list.

Example:

```
<ol>
  <li> First item </li>
  <li> Second item </li>
  <li> Third item </li>
  <li> Fourth item </li>
</ol>
```

Note: The outer wrapping tag `` can have a number of attributes added to it. The *start* attribute in the `` tag can be used to specify a number to start the ordered list to begin numbering than the usual number “1”. For example, `<ol start = “100”>` will enable the list to begin with a number 100. In this case the first item would be numbered as 100, second would be numbered with 101, and third would be 102, and so on.

Nested ordered lists

Ordered lists can be nested therefore they can multiple levels of numbering within a listing. For Example, let us create a nested ordered list as follows:

```
<ol>
  <li>Installation
  </ol>
    <li>Computer set up</li>
    <li>Monitor set up
      <ol>
        <li>Model XYZ</li>
        <li>Model ABC</li>
      </ol>
    </li>
  </ol>
  <li>Maintenance</li>
  <li>Use</li>
</ol>
```

When viewed using a browser, the above code snippet will generate an output as follows:

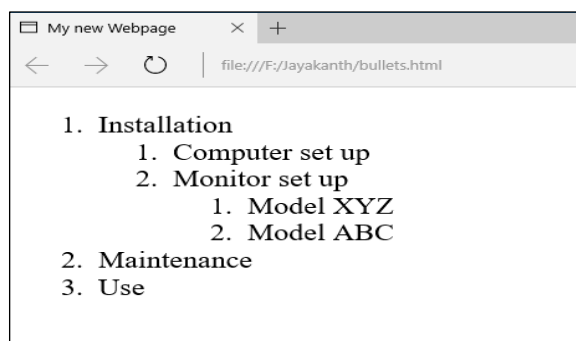


Figure 3.3.5

The above output shows the different levels with the restarting of numbers. You can change the numbering from the standard Decimal numbers to different styles using CSS. We will see it later when we study the Cascading Style Sheets.

Note: Unordered lists and ordered lists can be combined in any sequence by nesting one within the other.

Unordered List:

In HTML, an unordered list is composed of two tags: `` and ``. The `` tag is the outermost structure that declares the unordered list. Within the `` tag, a series of `` tags creates the items in the list. Here's a short example of the HTML code for an unordered list:

(write about the bullets in lists and there are many different styles to include)

```
<ul>
  <li>Tomatoes</li>
  <li>Onion</li>
  <li>Garlic</li>
</ul>
```

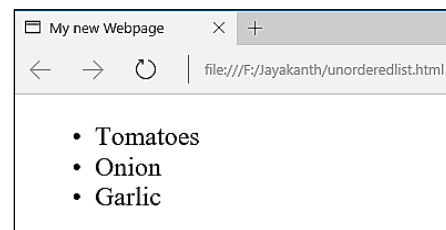


Figure 3.3.6

Note: The `` tag can contain any amount of text, from a single word to multiple lines of text. Bullets in a list can be a standard unordered list gives equal weight to all the bulleted items, one after another. In some situations, it is desirable to depict multiple levels of content with sub-items. HTML provides the capacity to incorporate any level of sub-items desired by nesting `` tags.

Like unordered lists, the numbered variety uses two key elements: an outer wrapping tag and a separate tag for each list item. The only difference is that the outer tag is not `` but ``. Here's a brief example:

```
<ol>
  <li>Pull mask from overhead bin</li>
  <li>Place mask over face</li>
  <li>Pull strings tight</li>
</ol>
```

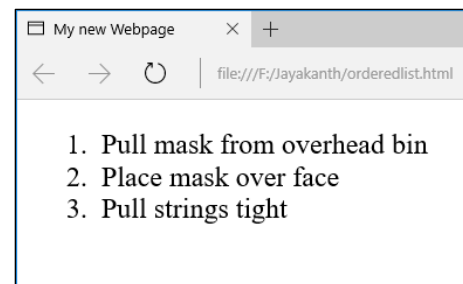


Figure 3.3.7

Table

In html you can create a table using the three different tags: `<table>`: This is the outermost element that contains the other two tags and all content. `<tr>`: The `<tr>` tag defines the table row and holds the final element. `<td>`: The `<td>` tag stands for table data; the complete `<td>` tag is also known as a table cell. Any content that is displayed in the table is placed between the opening and closing `<td>` tag pair. For example,

```

<table>
  <tr>
    <td>First name</td>
    <td>Last name</td>
    <td>Extension</td>
  </tr>
  <tr>
    <td>Pat</td>
    <td>Peterson</td>
    <td>x394</td>
  </tr>
</table>

```

It will create a table that has three columns and two rows.

When viewed using a browser, the above code snippet will generate an output as follows:

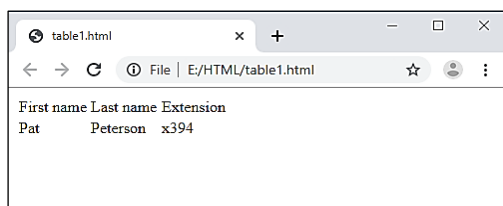


Figure 3.3.8

The above declaration does not have column header and a column header can be easily added by having a `<th>` tag instead of a `<td>` tag. For example,

```

<table><tr>
  <th>First name</th>
  <th>Last name</th>
  <th>Extension</th>
</tr>
<tr>
  <td>Pat</td>
  <td>Peterson</td>
  <td>x394</td>
</tr>
</table>

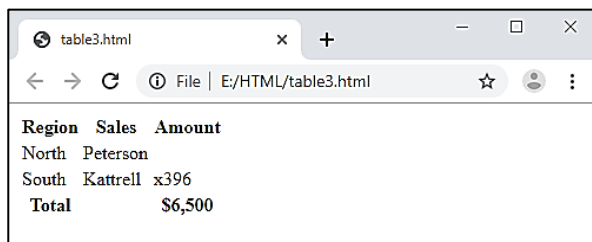
```



Figure 3.3.9

In this case we have explicitly said the table headers and the table data. Browsers will usually display the header of the table in bold letter. HTML has more tags that enable the creation of more structured tables with header, body and footer regions. For example,

```
<table>
  <thead>
    <tr>
      <th>Region</th>
      <th>Sales</th>
      <th>Amount</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <th>Total</th>
      <th>&nbsp;</th>
      <th>$6,500</th>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>North</td>
      <td>Peterson</td>
      <td></td>
    </tr>
    <tr>
      <td>South</td>
      <td>Kattrell</td>
      <td>x396</td>
    </tr>
  </tbody>
</table>
```



Region	Sales	Amount
North	Peterson	
South	Kattrell	x396
Total		\$6,500

Figure 3.3.10

The colspan and rowspan attributes both take numeric values to define how many columns or rows will be spanned, respectively. For example, if I wanted to create a table that had two headers, each of which spanned two of the four columns, my code would look like this:

```

<table border =2>
  <tr>
    <th colspan="2">Atlantic Division</th>
    <th colspan="2">Pacific Division</th>
  </tr>
  <tr>
    <td>New York</td>
    <td>Boston</td>
    <td>San Francisco</td>
    <td>Los Angeles</td>
  </tr>
</table>

```

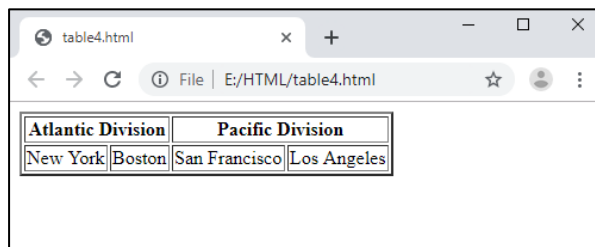


Figure 3.3.11

When rendered in the browser, the headers are centered over the spanned columns as shown in Figure 3.3.11. To better show the spanning and centering, I added a CSS rule to give the table a width of 300 pixels as well as another to create the outlining borders. Implementing the rowspan attribute requires a different table configuration than what is needed for colspan:

```

<table border=2>
  <tr>
    <th rowspan=2>Atlantic Division</th>
    <td>New York</td>
  </tr>
  <tr>
    <td>Boston</td>
  </tr>
  <tr>
    <th rowspan=2>Pacific Division</th>
    <td>San Francisco</td>
  </tr>
  <tr>
    <td>Los Angeles</td>
  </tr>
</table>

```

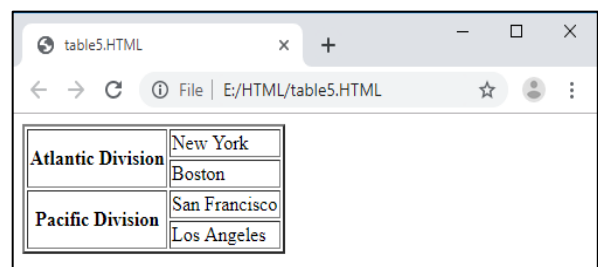


Figure 3.3.12

Competency Level 10.4: Uses HTML to create linked web pages

Learning Outcomes:

- Explains hypertext markup language
- Identifies the standards of HTML
- Saves the source document with suitable extensions
- Designs the web page by inserting appropriate multimedia objects according to user requirements
- Organizes data using lists and tables in the web page
- Links pages and multimedia objects to the web page

HTML to create linked web pages

We have looked at the basic structure of a web page that can be written using HTML, and a few other tags that can be included to create a web page. We have also seen that a designed web page can be viewed using a web browser to view the content of the web page. Now let us see how we create a set of linked web pages and their properties.

Html is the language of the web. As we know HTML stands for a hypertext markup language. In writing web pages, we have hypertext which implies the text is not normal text but is hypertext that has a link which when clicked by a mouse pointer can take the user to a new web page or to another portion of the text in the same web page. The markup language component allows us to mark up our programs with additional tags and other attributes to further enhance the text. Therefore, they could be more readable on the displays. HTML5 is the latest version of HTML and it has many advanced features such as the ability to play built-in videos and animations without the need to download additional add on, than the previous versions.

A web page can contain text and audiovisual content on it and can be presented to the user as a web page. Having just one web page will not serve the purpose for many enterprises and there need to be a number of interconnected web pages designed to present the content to the viewers. Since enterprises have much more information that needs to be classified and presented differently, multiple web pages that present the information in such a way are needed to be developed and linked together. When a number of web pages are developed and linked together it is called a web site. A website for an enterprise contains a number of interconnected web pages that contain hyperlinks, therefore pages could be navigated from one to another. The main web page of a web site is called the home page and it is assumed to be the starting page of a web site from which the user begins navigating the web site. The home page also facilitates the navigation to other web pages of the web site by providing navigational links in it.

Linked pages of a website

Hyperlinks

In an HTML document, there will be a number of links that have a connection to a content that appears on the same web page, or a connection that takes you to a new webpage. Such links in an HTML document are called hyperlinks. Web pages can be linked to each other therefore the web pages can be navigated back and forth easily and the user experience can be improved.

To jump from one page to another, the <a> tag, the anchor tag can be used. The text or image enclosed by the <a> tag anchors one side of the link to the current page, and the href attribute (short for hypertext reference) specifies the other side, the destination. For example, consider,

```
<a href="home.html">Home</a>
```

In this example, when the user clicks the word Home, the browser would jump to the home.html page.

When pages within the same site are linked, document relative URLs can be used. For example, when the page that is being linked is within the same folder, they can be linked as follows:

```
<a href="home.html">Home</a>
```

```
<a href="services.html">Services</a>
```

```
<a href="products.html">Products</a>
```

When another site on the web needs to be linked, an absolute URL must be specified. For example,

```
<a href="http://www.nie.lk/index.htm">National Institute of Education</a>
```

It will take the user to the home page of the National Institute of Education.

Competency Level 10.5: Uses Style sheet to change the appearance of web pages

Learning Outcomes:

- Briefly explains style sheet and its usage
- Uses the comments and correct syntax in CSS
- Uses appropriate selectors to select elements in CSS
- Inserts CSS in HTML web pages to improve the appearance
- Applies various CSS formatting in HTML web pages to improve the appearance

Style sheets to change the appearance of web pages

Cascading Style Sheets (CSS) provide the look and feel for the HTML content. CSS enables the designer to change how the text, images, and links appear on a web page quickly and easily. Using the CSS, the designer has the freedom of styling the web page independent of the content using a presentation layer where you can make global formatting options. The cascading part of the CSS tells how the elements on a web page are affected by the CSS rules. With CSS, web page styles are made up of one or more style rules.

A CSS rule has three components: the selector, the properties and their values. For example, the following is an HTML rule,

```
h1{color : red;}
```

Where h1 is the selector, color is the property and red is the value assigned for color. Multiple properties and values can be assigned for a single selector, enclosed within curly braces and separated by semicolons. For example, the following CSS rule says that the heading selector h1 has three properties and their corresponding values:

```
h1{  
    color : blue;  
    margin : 0 ;  
    padding : 5px;  
}
```

CSS Placement

CSS rules can be integrated into an HTML page in a number of ways: as an external style sheet, embedded within the HTML page itself, and inline as an attribute within the tag.

External style sheets

These are used to provide a consistent look-and-feel to any number of related pages, up to and including an entire website. An external style sheet is connected to an HTML page in one of two ways: either with a <link> tag or with a @import directive within a <style> tag. For example, say you wanted to include the CSS rules written in a file called main.css. The <link> syntax would look like this:

```
<link href="styles/main.css" type="text/CSS" rel="stylesheet" />
```

The href attribute provides the path to the external style sheet, and type specifies the kind of document the browser can expect. The relationship of the HTML page to the linked file is defined by the rel attribute; the two possible values are stylesheet and alternate stylesheet. If you wanted to use the @import syntax, you would write code like this:

```
<style> @import { URL("styles/main.css"); } </style>
```

Notice that @import is actually a CSS rule with the single URL property, written somewhat differently from standard CSS declarations. When used with an HTML page, the @import rule must be within a <style> tag.

External style sheets have the tremendous advantage of being able to affect multiple HTML pages simultaneously. Change any CSS rule, save the external style sheet, publish it, and immediately the modification can be seen by any site visitor to any of the associated pages. You can see why external style sheets are widely used by web designers across the industry.

Internal or embedded styles

CSS rules can also be included in the <head> section of an HTML page. CSS rules are embedded through use of the <style> tag, like this:

```
<style type="text/css"> body {margin: 0; padding: 0; background-color: white ;} h1, h2, h3, h4 {color: red; margin: 0; padding: 5px ;} </style>
```

Inline

An inline style is applied by using the style attribute within an HTML tag. For example, if you want to colour a <h1> tag red with in an inline style, your code would look this:

```
<h1 style="color: red;"> Important Message Ahead</h1>
```

CSS selectors

In CSS there are four basic types of selectors:

- Tags: An HTML tag can serve as a CSS selector. The use of HTML tags as a CSS selector is very straightforward. When an HTML tag, such as <p>, is defined as a selector with CSS, all <p> tags are immediately affected unless another CSS style overrules it.
- IDs: Typically ID selectors are intended to be used once per HTML page. An ID can be defined by using a leading hash (#) symbol and can then be applied in an HTML tag using ID attribute. For example, consider the following ID definition and its application.

```
#header {width: 960px;}  
<div id="header">
```

- Classes: A class is another custom selector, which can be used as many times as needed on a web page. To define a class selector a "." symbol is used at the beginning.

```
.legalNotice { font-size: small; }
```

To apply the class selector to an HTML tag, use the class attribute:

```
<div class="legalNotice">
```

The names of classes and IDs must begin with a letter and not contain any white spaces or other special characters. Similarly, classes and IDs are case-sensitive. In other words, .firstParagraph is not the same as FirstParagraph.

- Compound: Tags, IDs, and classes can be combined to create a compound selector, which pinpoints a particular section of the page.

Validating CSS Rules

It is essential to ensure that the page that has been designed is error-free and the CSS is syntax error-free. CSS validators can be used for this purpose, which ensures that the CSS is accurate

and does not have any unsupported selectors, properties, or values. The W3C hosts a CSS validation service and it is hosted at <http://jigsaw.w3.org/css-validator>

Styling Text with CSS

The font family property lists a series of fonts that can be used, therefore even if one font is not available on the computer on which the browser is running the other font will be taken automatically by the browser. For example, the declaration,

```
font-family: Arial, Helvetica, sans-serif;
```

It makes the browser to first look for the font Arial, if it is not available, look for Helvetica and if that too is not available look for sans-serif.

The size of the font for text on the Web is determined by the font-size property. You can use a named, relative measurement such as large or small like this:

```
h1 { font-size: x-large; }
```

Colours for text can be set by the CSS colour property by using

```
h1, h2 {color: maroon ;}
```

Whereas, colour names, or hexadecimal colour values or RGB or RGBA could also be used

Competency Level 10.6: Uses an authoring tool to create web pages

Learning Outcomes:

- Briefly explains web authoring tools
- Creates web pages using a web authoring tool

An authoring tool to create web pages

Any software or collection of software components that authors can be used to create or modify web content, to use by other people is called an authoring tool. Web authoring is the practice of creating web documents using modern web authoring software and tools. Web authoring software is a type of desktop publishing tool that allows users to navigate the tricky environment of HTML and web coding by offering a different kind of graphical user interface.

Examples of web authoring tools include:

- web page authoring tools (e.g., WYSIWYG HTML editors)
- software for directly editing source code or markup
- software for converting to web content technologies (e.g., "Save as HTML" features in office suites)
- integrated development environments (e.g., for web application development)
- software that generates web content on the basis of templates, scripts, command-line input or "wizard"-type processes
- software for generating/managing entire web sites (e.g., content management systems, courseware tools, content aggregators)
- email clients that send messages in web content technologies
- multimedia authoring tools
- debugging tools for web content
- software for creating mobile web applications
- scripting libraries
- web application frameworks, IDEs, and SDKs

Competency Level 10.7: Creates dynamic web pages using PHP and MySQL

Learning Outcomes:

- Defines dynamic web pages
- Creates data source and enters data
- Creates PHP code to save/retrieve data to and from MySQL
- Develop simple web based information systems

Dynamic web pages using PHP and MySQL

Web sites can be classified as static web sites or dynamic websites. In static websites, the HTML and CSS are used to design the web pages and the content is pretty much static in its nature. In dynamic web pages, the HTML, CSS, PHP along with some database management system, preferably MySQL, are used therefore the website provides more interactive content to the user. Dynamic web pages are more interactive web pages where a user requests something from the web page and then obtains it as a result of his query. We have already seen the properties of HTML and CSS. PHP is a server language that carries out data manipulations at the server-side and sends the result data in the code to be displayed on the web site. MySQL is a database management system that stores data and helps in manipulating and resending it on the web pages using PHP.

First example with PHP

```
<body>
  <?php
    echo 'My First PHP Script Example';
  ?>
</body>
```

The above code will simply print the text 'My First PHP Script Example' into your web page. This code is similar to the HTML we are familiar with and the only difference is that the code now has some PHP commands. The extra commands included making this as a PHP program and we need to store this file with a ".php" extension. This will indicate the server that the file contains PHP code and therefore the server should pass it to the PHP interpreter for processing. The PHP program is responsible for passing back a clean file suitable for display in a web browser. At its very simplest, a PHP program will output the only HTML.

To trigger the PHP commands, *<?php* tag is used. The entire PHP code is placed right after this tag and it is ended with a closing *?>* tag. Each statement written within this tag pair is called a PHP command and each command should end with a ";".

Structure of PHP programs

Comments:

In PHP, a single line comment is indicated using *//*, and a multi-line comment is enclosed within a pair of */** and **/*.

Variables:

In PHP variables are used to store intermediate values and all variables in PHP start with a \$ sign followed by the name of the variable. A valid variable name starts with an alphabet (A-Z, a-z) or underscores (_), followed by any number of alphabets, numbers or underscores.

Example_1:

```
<?php
$abc = "Welcome"; //a valid variable declaration
$Abc = "W3resource.com"; // a valid variable declaration
$9xyz = "Hello world"; // an invalid variable declaration since it starts with a number
$_xyz = "Hello world"; // a valid variable declaration, it starts with an underscore
$_9xyz = "Hello world"; // a valid variable declaration
?>
```

Note: PHP variables are case sensitive therefore \$abc and \$Abc will be treated differently. In the above variable declarations, the quotation marks indicate that the “welcome” is a string of characters. Strings must be enclosed either within a single quotation or double quotation marks. The values of variables are displayed by typing the command

```
echo $abc;
```

The value stored in a variable can also be assigned to another variable,

```
$new_abc = $abc;
```

Example_2:

```
<?php // test1.php
$username = "Fred Smith";
echo $username;
echo "<br>";
$current_user = $username;
echo $current_user;
?>
```

Other types of variables could also be declared in php like:

```
$count = 17.5;
```

Operators in PHP

Arithmetic Operators

Operator	Description	Example
+	Addition	$\$i + 1$
-	Subtraction	$\$i - 1$
*	Multiplication	$\$i * 10$
/	Division	$\$i / 5$
%	Modulus (remainder of a division)	$\$i \% 5$
++	Increment	$++i$
--	Decrement	$--i$

Assignment Operators

Operator	Description	Equivalent to
=	$\$i = 10$	$\$i = 10$
+=	$\$i += 10$	$\$i = \$i + 10$
-=	$\$i -= 5$	$\$i = \$i - 5$
*=	$\$i *= 3$	$\$i = \$i * 3$
/=	$\$i /= 5$	$\$i = \$i / 5$
.=	$\$i .= \j	$\$i = \$i . \$j$
%=	$\$i \% = 5$	$\$i = \$i \% 5$

Comparison Operators

Operator	Description	Example
==	Is <i>equal</i> to	<code>\$i == 5</code>
!=	Is <i>not equal</i> to	<code>\$i != 10</code>
>	Is <i>greater than</i>	<code>\$i > 10</code>
<	Is <i>less than</i>	<code>\$i < 10</code>
>=	Is <i>greater than or equal to</i>	<code>\$i >= 5</code>
<=	Is <i>less than or equal to</i>	<code>\$i <= 5</code>

Logical Operators

Operator	Description	Example
&&	And	<code>\$j == 3 && \$k == 2</code>
and	Low-precedence <i>and</i>	<code>\$j == 3 and \$k == 2</code>
	Or	<code>\$j < 5 \$j > 10</code>
or	Low-precedence <i>or</i>	<code>\$j < 5 or \$j > 10</code>
!	Not	<code>!(\$j == \$k)</code>
xor	Exclusive or	<code>\$j xor \$k</code>

Arrays in PHP

Arrays in PHP are a collection of key/value pairs. This means that it maps values to keys. Array keys (or indexes) may be either integers or string whereas values can be any type. An array can be declared using the **array()** language construct. There are three different types of arrays in PHP, indexed arrays, associative arrays, and multidimensional arrays.

Indexed Array

An *indexed array* named `fruits` with three elements can be created as follows:

```
$fruits = array("Banana", "Apple", "Orange");  
Or  
$fruits[0] = "Banana";  
$fruits[1] = "Apple";  
$fruits[2] = "Orange";
```

- **Associative array**

An *associative array* is an array that uses named keys that you assign to them. An associative array can be created as follows:

```
$variable_array_name = array( key1=> value1, key2=> value2, key3=> value3, ..... );
```

For example, consider the following declaration:

```
$fruits = array("fruit1" => "Banana", "fruit2" => "Apple", "fruit3" => "Orange");  
or  
$fruits['fruit1'] = "Banana";  
$fruits['fruit2'] = "Apple";  
$fruits['fruit3'] = "Orange";
```

Example- Looping through array:

```
<?php
    $fruits = array("Banana", "Apple", "Orange");
    $arrlength = count($cars); // retrieves the length of the array
    for($x = 0; $x < $arrlength; $x++)
    {
        echo $cars[$x];
        echo "<br>";
    }
?>
```

Example- Associative array:

```
<?php
    $fruits = array( "fruit1" => "Banana", "fruit2" => "Apple", "fruit3" => "Orange");
    foreach($fruits as $x => $x_value)
    {
        echo "Key=" . $x . ", Value=" . $x_value;
        echo "<br>";
    }
?>
```

PHP has built-in functions that enable the sorting of arrays in ascending or descending order using alphabetical or numerical order.

The following list shows available sort functions in PHP:

- `sort()` - sorts arrays in ascending order
- `rsort()` - sorts arrays in descending order
- `asort()` - sorts associative arrays in ascending order, according to the value
- `ksort()` - sorts associative arrays in ascending order, according to the key
- `arsort()` - sorts associative arrays in descending order, according to the value
- `krsort()` - sorts associative arrays in descending order, according to the key

For example to sort an array in descending order,

```
<?php
    $fruits = array("Banana", "Apple", "Orange");
    sort($fruits);
?>
```

This would sort the array fruits, in the ascending alphabetical order in fruit names.

The above declaration implies that the fruit1 and fruit2 are the keys to the array \$fruits and have the values of "Banana" and "Apple" respectively.

Control Structures in PHP

▪ Conditional Statements in PHP

PHP provides the following different structures of conditional statements:

▪ The *if* statement:

The if statement has the following syntax and executes some code when the given condition is true:

```
if (condition)
{
    code to be executed if condition is true;
}
```

For example:

```
<?php
if ($age < "60") {
    echo "You are about to retire!";
}
?>
```

▪ The *if ... else* statement:

The if ... else statement has the following syntax and executes some code when the given condition is true and some other code when the given condition is false.

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

For example:

```
<?php
$age1 = 60;
$age2 = 55;
if ($age1 > age2) {
    echo "Age 1 is greater than Age 2";
} else {
    echo "Age 2 is greater than Age 1";
}
?>
```

- **The *if...elseif...else* Statement**

The *if...elseif...else* statement executes different codes for more than two conditions. The syntax of *if... elseif...else* is as follows:

```
if (condition)
{
    code to be executed if this condition is true;
}
elseif (condition)
{
    code to be executed if this condition is true;
}
else
{
    code to be executed if all conditions are false;
}
```

For example:

```
<?php
    $Marks = 60;
    if ($age1 >=75) {
        echo "Your Grade is A";
    }
    elseif($age1>=65){
        echo "Your Grade is B ";
    }
    elseif($age1 >=55){
        echo " Your Grade is C ";
    }
    elseif($age1 >=45){
        echo " Your Grade is S ";
    }
    else{
        echo "Sit the exam again";
    }
?>
```

▪ The Switch Statement

The switch statement is also used as a conditional statement when there are multiple actions that need to be taken based on multiple conditions. The switch statement in PHP has the following syntax:

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

In the above syntax of the switch statement, based on the evaluated value of the variable *n*, it is compared with the values of each case, and when there is a match, the code attached will be executed. The *break* command is included at the end of each block of code that is attached to each case to prevent the next block of code is executed automatically. The *default* (case) statement is used at the end to indicate the block of code that needs to be executed if there is no match in the cases.

For example:

```
<?php
    $favcolour = "red";
    switch ($favcolour) {
        case "red":
            echo "Your favorite colour is red!";
            break;
        case "blue":
            echo "Your favorite colour is blue!";
            break;
        case "green":
            echo "Your favorite colour is green!";
            break;
        default:
            echo "Your favorite colour is neither red, blue, nor green!";
    }
?>
```

Repetition

▪ While Loop

While loop is a looping structure available in PHP which first checks for a condition and if the condition is evaluated true, then shall execute the set of commands within the loop until the condition becomes false.

An example for a while loop in php is given below:

```
<?php
    $count = 1;
    while ($count <= 12)
    {
        echo "$count times 12 is " . $count * 12 . "<br>";
        ++$count;
    }
?>
```

The above loop will execute 12 times and as soon as the value of count becomes 13, the loop will exit.

▪ Do ... While loop

Do while loop is a slight variation of a while loop where the looping structure allows the execution of the set code at least once, and then allows execution of the code based on the condition after that.

The following example illustrates the structure of a do while loop:

```
<?php
    $count = 1;
    do
    echo "$count times 12 is " . $count * 12 . "<br>";
    while (++$count <= 12);
?>
```

▪ For Loop

A for loop statement takes three parameters namely, an initialization expression, a condition expression, and a modification expression. The syntax of the for loop will be as follows:

```
for (expr1 ; expr2 ; expr3)
{
    php command 1;
    php command 2;
    .....
    .....
}
```


For example:

```
<?php
    for ($count = 1 ; $count <= 12 ; ++$count)
    {
        echo "$count times 12 is " . $count * 12;
        echo "<br>";
    }
?>
```

Functions

Functions in PHP are used to separate portions (or blocks) of PHP code from the main part of the program. These portions (or blocks) of code can perform certain tasks and therefore can be called anywhere in the program as many times as needed. Writing functions reduce the repetition of the same code in the program and it makes it much easier to modify the code only in one place rather than at multiple locations. A function will be executed when it is called in the main program.

Syntax of declaring a function is as follows:

```
function functionName() {
    code to be executed ;
}
```

To create a function, declare it in the manner shown below:

```
<?php
    function writeWelcomeMessage()
        //declaration of the function begins here
    {
        echo "Hello World !";
    }
    writeWelcomeMessage();
        // this will call the function to be executed here
?>
```

Functions also accept arguments as part of their declaration to receive information. Arguments are specified after the function name, inside the parentheses and multiple arguments are separated by commas. For example, in the following code, we define a function that accepts two arguments and then we call the function with different values passed to it.

```
<?php
    function printNameAndYear($fname, $year)
    {
        echo "Mr./Ms. $fname was born in $year <br>";
    }
    // now calling of the function with different values takes place
    familyName("Rajah", "1975");
    familyName("Leela", "1978");
    familyName("Anuraj", "1983");
?>
```

Functions could also return values as a result of the processing within the function. The “*return*” statement is used at the end of the function and can return a single value. For example, the following code returns the sum of two values passed as arguments to it.

```
<?php
function sum($x, $y)
{
    $z = $x + $y;
    return $z;
}
echo"5 + 10 = ". sum(5,10) ."<br>";
echo"7 + 13 = ". sum(7,13) ."<br>";
echo"2 + 4 = ". sum(2,4);
?>
```

Databases:

A *database* is a structured collection of records or data stored in a computer system and organized in such a way that it can be quickly searched and information can be rapidly retrieved. MySQL is probably the most popular database management system for web servers, and the combination of PHP and MySQL works well on any operating system. The *SQL* in MySQL stands for *Structured Query Language*. A database query is a question or a request sent to the database to obtain particular information or a record. A MySQL database contains one or more *tables*, each of which contains *records* or *rows*. Within these rows are various *columns* or *fields* that contain the data itself.

For example, consider the following table:

Author	Title	Type	Year
Mark Twain	The Adventures of Tom Sawyer	Fiction	1876
Jane Austen	Pride and Prejudice	Fiction	1811
Charles Darwin	The Origin of Species	Nonfiction	1856
Charles Dickens	The Old Curiosity Shop	Fiction	1841
William Shakespeare	Romeo and Juliet	Play	1594

The above table contains details about classic novels and we name it as *classics* and it is in the database named *publications*. There are a number of columns with the topmost columns containing the headings and each of the columns is called a *field* in the table. The column headings are called the *field names*. Under each column heading, there are data for each column. Each row of the table contains a complete record of classic novels.

Fundamentals of MySQL

To create a database named *publications* in MySQL issue the following command:

```
CREATE DATABASE publications;
```

The above command will create a database named *publications* and is now ready for use. Now you may issue the command **USE *publications***; to enable it to work on. As now you have the database ready for use, you can now start creating tables. The syntax for creating tables in MySQL is as follows:

```
CREATE TABLE table_name(
    column1 datatype,
    column2 datatype,
    column3 datatype,
    .....);
```

Suppose we want to create the table named *classics*. The following command will create the table:

```
CREATE TABLE classics (
    author VARCHAR(128),
    title VARCHAR(128),
    type VARCHAR(16),
    year CHAR(4)
);
```

Issuing the command `DESCRIBE classics`; is used to ensure that the table has been created without any issues. It also helps us to recall the field names and their data types in a given table.

For example, issuing the command,

mysql>DESCRIBE classics; will return the following information:

Field	Type	Null	Key	Default	Extra
author	varchar(128)	YES		NULL	
title	varchar(128)	YES		NULL	
type	varchar(16)	YES		NULL	
year	char(4)	YES		NULL	

Let's look at each of the headings in detail:

Field: The name of each field or column within a table.

Type: The type of data being stored in the field.

Null: Whether a field is allowed to contain a value of NULL.

Key: MySQL supports *keys* or *indexes*, which are quick ways to look up and search for data. The Key heading shows what type of key (if any) has been applied.

Default: The default value that will be assigned to the field if no value is specified when a new row is created.

Extra: Additional information, such as whether a field is set to auto-increment.

Data types in MySQL are used to notify the type of data that each field would contain.

MySQL supports the following data types and data ranges:

Data type	Bytes Used	Example
CHAR(<i>n</i>)	Exactly <i>n</i> (<=255)	CHAR(5) "Hello" uses 5 bytes CHAR(57) "Goodbye" uses 57 bytes
VARCHAR(<i>n</i>)	Up to <i>n</i> (<= 65535)	VARCHAR(7) "Morning" uses 7 bytes VARCHAR(100) "Night" uses 5 bytes
BINARY(<i>n</i>)or BYTE(<i>n</i>)	Exactly <i>n</i> (<= 255)	As CHAR but contains binary data
VARBINARY(<i>n</i>)	Up to <i>n</i> (<= 65535)	As VARCHAR but contains binary data

Numerical Data types:

Data type	Bytes used	Minimum value		Maximum value	
		Signed	Unsigned	Signed	Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8.38e+6	0	8.38e+6	1.67e+7
INT / INTEGER	4	-2.15e+9	0	2.15e+9	4.29e+9
BIGINT	8	-9.22e+18	0	9.22e+18	1.84e+19
FLOAT	4	-3.40e+38	<i>n/a</i>	3.4e+38	<i>n/a</i>
DOUBLE / REAL	8	-1.80e+308	<i>n/a</i>	1.80e+308	<i>n/a</i>

Adding Data to a Table

To add data to a table, *insert* command is used. For example, the following two lines of commands can be used to insert data into the *classics* table.

```
INSERT INTO classics(author, title, type, year)
VALUES('Mark Twain','The Adventures of Tom Sawyer','Fiction','1876');
INSERT INTO classics(author, title, type, year)
VALUES('Jane Austen','Pride and Prejudice','Fiction','1811');
```

The following SQL commands could be issued to alter the table properties:

To **rename** the name of a table:-

```
ALTER TABLE <older name> RENAME <new name>;
```

Example:

```
ALTER TABLE classics RENAME pre1900;
```

To **change the data type** of a column in a table:-

```
ALTER TABLE <table name> MODIFY <field name><new data type>;
```

Example:

```
ALTER TABLE classics MODIFY year SMALLINT;
```

To **add a new column** to a table:-

```
ALTER TABLE <table name> ADD <new column name><new data type>;
```

Example:

```
ALTER TABLE classics ADD pages SMALLINT UNSIGNED;
```

To **remove a column** from a table:-

```
ALTER TABLE <table name>DROP <column to be dropped>;
```

Example:

```
ALTER TABLE classics DROP pages;
```

Indexes in Databases

Indexes in database tables help to search through the tables in a faster manner. Indexes can be added to a table whenever it is created or at a later stage whenever needed. Indexes are actually a column of data or a combination of more than one column of data that can be used to search through the table. With a judgment that requires the database designer to predict whether he/she will be searching any of the data in that column, the database designer has to decide which column would be used as an index. For example, in the table, *classics*, that we have already created, an index can be created based on the *author* or based on the *title columns* as follows:

```
ALTER TABLE classics ADD INDEX(author(20));  
ALTER TABLE classics ADD INDEX(title(20));
```

The above commands will create indexes on both the *author* and *title* columns, limiting each index to only the first 20 characters. An alternative of using ALTER TABLE to add an index is, to use the CREATE INDEX command.

For example, the following two commands are equivalent:

```
ALTER TABLE classics ADD INDEX(author(20));
```

or

```
CREATE INDEX author ON classics (author(20));
```

Primary Keys

Primary keys enable us to search a database table using a single unique key to access a row of a table. Having a unique key to access rows of a table are essential when we want to combine data from multiple tables. The following SQL statement will create the table *classics* with a primary key "ISBN":

```
CREATE TABLE classics (  
    author VARCHAR(128),  
    title VARCHAR(128),  
    category VARCHAR(16),  
    year SMALLINT,  
    isbn CHAR(13),  
    INDEX(author(20)),  
    INDEX(title(20)),  
    INDEX(category(4)),  
    INDEX(year),  
    PRIMARY KEY (isbn));
```

Querying SQL Databases

The first command to learn is the **SELECT** which allows us to extract data from tables. The syntax of using the select command:

```
SELECT <column_name>
FROM <table_name>;
```

The following example illustrates how to select the *author* and *title* column data from the *classics* table:

```
SELECT author, title
FROM classics;
```

WHERE keyword: The *where* keyword enables you to narrow down queries by returning only those *where* a certain expression is true. The syntax of using where keyword in an SQL statement is as follows:

```
SELECT <column_name>
FROM <table_name>
WHERE <conditional_expression>;
```

For example, to retrieve the rows that match the string “Mark Twain” for author name is written as follows:

```
SELECT author, title
FROM classics
WHERE author="Mark Twain";
```

The **ORDER BY** statement sorts returned results by one or more columns in ascending or descending order. For example, the following query returns author name and title from the classics table sorted in the author name order.

```
SELECT author, title
FROM classics
ORDER BY author;
```

The following query returns the author and titles returned whereas the results are sorted in the descending order of the title.

```
SELECT author, title
FROM classics
ORDER BY title DESC;
```

In a similar fashion to ORDER BY, you can group results returned from queries using **GROUP BY**, which is good for retrieving information about a group of data.

For example, if you want to know how many publications there are of each category in the *classics* table, you can issue the following query:

```
SELECT category, COUNT(author)
FROM classics
GROUP BY category;
```

The above query will return the following output:

Category	COUNT(author)
<i>Classic Fiction</i>	3
<i>Non-Fiction</i>	1
<i>Play</i>	1

Databases often will have multiple tables containing related data and joining them using a single select statement to display columns that match a given condition on the common columns. For example, suppose we have two tables, *classic* (which contains the details of books as earlier) and *customers* (which has details about customers who have purchased books including a column containing the *ISBN* of the book purchased). In this case, the *ISBN* of the book is a common column to both the tables.

```
SELECT name, author, title from customers, classics
WHERE customers.isbn = classics.isbn;
```

That's a brief introduction to MySQL and we will see more MySQL in action when we combine MySQL with PHP.

PHP Forms

The main way that website users interact with PHP and MySQL is through the use of HTML forms. Handling forms is a multipart process. First, a form is created, into which a user can enter the required details. This data is then sent to the webserver, where it is interpreted, often with some error checking. If the PHP code identifies one or more fields that require reentering, the form may be redisplayed with an error message. When the code is satisfied with the accuracy of the input, it takes some action that usually involves the database, such as entering details about a purchase.

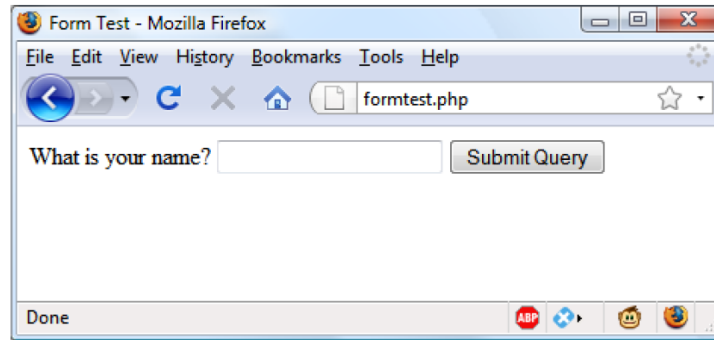
To build a form, you must have at least the following elements:

- An opening `<form>` and closing `</form>` tag
- A submission type specifying either a Get or Post method
- One or more input fields
- The destination URL to which the form data is to be submitted

For example, consider the following php code:

```
<html>
  <head>
    <title>Form Test</title>
  </head>
  <body>
    <form method="post" action="formtest.php">
      What is your name?
      <input type="text" name="name">
      <input type="submit" value="Submit Query">
    </form>
  </body>
</html>
```

The result of opening the formtest.php in a browser would look like:

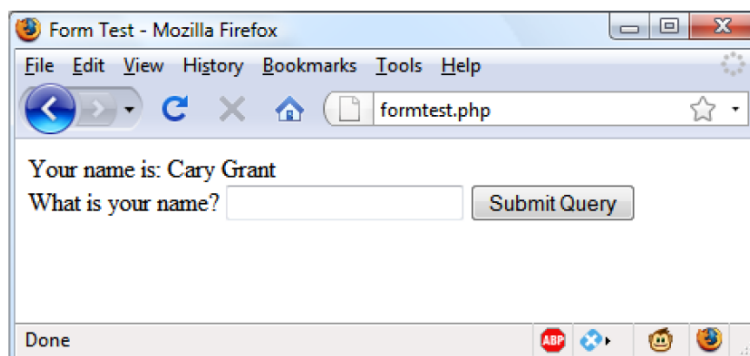


When you enter your name as the input data and (press) submit, absolutely nothing will happen other than the form being redisplayed.

We will add some more PHP code to process the data submitted by the form.

```
<?php //formtest.php
    if (isset($_POST['name'])) {
        $name = $_POST['name'];
        echo "Your name is ".$name; }
    else {
        echo "(Not entered)";
    }
?>
<html>
    <head>
    <title>Form Test</title>
    </head>
    <body>
    <form method="post" action="formtest.php">
    What is your name?
    <input type="text" name="name">
    <input type="submit" value= "Submit Query">
    </form>
    </body>
</html>
```

The *if-else* statement at the beginning checks whether the value for the field *name* has been submitted. Then within the body of the code, the value stored in the *name* is printed. The output of the above code shall look like:



Input Types:

HTML forms are very versatile and allow you to submit a wide range of input types, from text boxes and text areas to checkboxes, radio buttons, and more.

Text boxes:

Text boxes accept a wide range of alphanumeric, text and other characters in a single line box. The general format of a text box input is as follows:

```
<input type="text" name="name" size="size" maxlength="length" value="value">
```

The *size* attribute specifies the width of the box (in characters of the current font) as it should appear on the screen, and *maxlength* specifies the maximum number of characters that a user is allowed to enter into the field.

Text Areas:

When you need to accept input of more than a short line of text, use a text area. This is similar to a text box. But, as it allows multiple lines, it has some different attributes. Its general format looks like this:

```
<textarea name="name" cols="width" rows="height" wrap="type">  
</textarea>
```

Here if you have default text to display, you must put it before the closing `</textarea>`, and it will be displayed and be editable by the user:

For example:

```
<textarea name="name" cols="width" rows="height" wrap="type">  
    This is some default text.  
</textarea>
```

Checkboxes:

When you want to offer a number of different options to a user, from which he can select one or more items, checkboxes are the best option. Here is the syntax of checkboxes:

```
<input type="checkbox" name="name" value="value" checked="checked">
```

If you include the `checked` attribute, the box is already checked when it is displayed. The string you assign to the attribute should be either a double quote or the value `"checked"`, or there should be no value assigned. If you do not include the attribute, the box is shown unchecked.

Example of creating an unchecked box:

```
I Agree <input type="checkbox" name="agree">
```

If the user does not check the box, no value will be submitted. But if he does, a value of `"on"` will be submitted for the field name `agree`. If you prefer to have your own value submitted instead of the word *on* (such as the number 1), you could use the following syntax:

```
I Agree <input type="checkbox" name="agree" value="1">
```

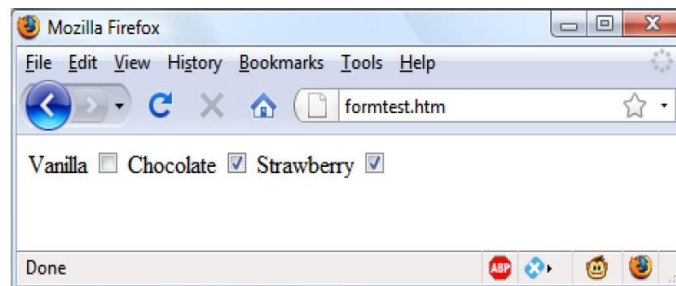
If you want to have the checkbox already checked as the default value then you may use as follows:

```
Subscribe? <input type="checkbox" name="news" checked="checked">
```

If you want to allow groups of items to be selected at once, the same name should be assigned for all.

This example allows its users to select his favourite ice creams:

```
Vanilla <input type="checkbox" name="ice" value="Vanilla">  
Chocolate <input type="checkbox" name="ice" value="Chocolate">  
Strawberry <input type="checkbox" name="ice" value="Strawberry">
```



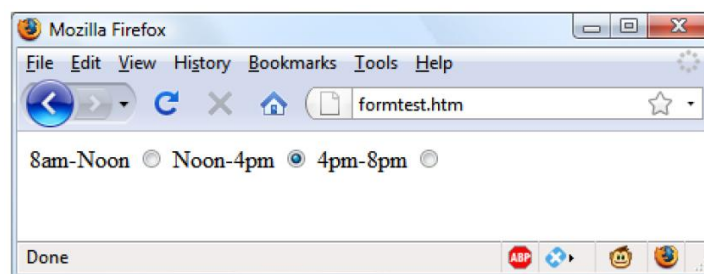
Radio buttons:

Radio buttons are used when you want only a single value to be returned from a selection of two or more options. All the buttons in a group must use the same name and because only a single value is returned, you do not have to pass an array.

For example, the following example illustrates if your website offers a choice of delivery times for items purchased from your store,

```
8am-Noon<input type="radio" name="time" value="1">  
Noon-4pm<input type="radio" name="time" value="2" checked="checked">  
4pm-8pm<input type="radio" name="time" value="3">
```

The output will look like:



Select

The <select> tag lets you create a drop-down list of options, offering either single or multiple selections. It conforms to the following syntax:

```
<select name="name" size="size" multiple="multiple">
```

The attribute size is the number of lines to display. Clicking on the display causes a list to drop down, showing all the options.

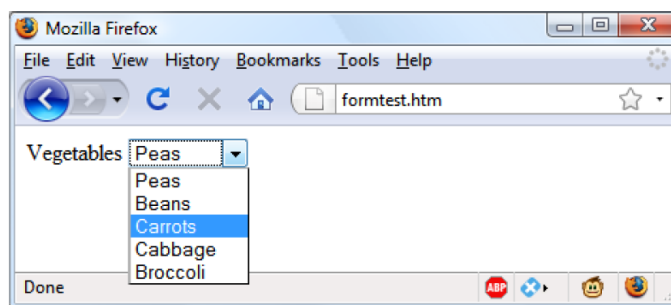
The following example illustrates the `<select>` tag:

Vegetables

```
<select name="veg" size="1">
  <option value="Peas">Peas</option>
  <option value="Beans">Beans</option>
  <option value="Carrots">Carrots</option>
  <option value="Cabbage">Cabbage</option>
  <option value="Broccoli">Broccoli</option>
</select>
```

This example offers five choices, with the first one, *Peas*, preselected (due to it being the first item). The diagram below shows the output where the list has been clicked to drop it down, and the option *Carrots* has been highlighted. If you want to have a different default option offered first (such as *Beans*), use the `<selected>` tag, like this:

```
<option selected="selected" value="Beans">Beans</option>
```



Labels:

You can provide even better user experience by utilizing the `<label>` tag. With it, you can surround a form element, making it selectable by clicking any visible part contained between the opening and closing `<label>` tags. For example, going back to the example of choosing a delivery time, you could allow the user to click the radio button itself *and* the associated text, like this:

```
<label>8am-Noon<input type="radio" name="time" value="1"></label>
```

The text will not be underlined like a hyperlink when you do this. But, as the mouse passes over, it will change to an arrow instead of a text cursor, indicating that the whole item is clickable.

Submit button:

To match the type of form being submitted, you can change the text of the submit button to anything you like by using the value attribute, like this:

```
<input type="submit" value="Search">
```

You can also replace the standard text button with a graphic image of your choice, using HTML such as this:

```
<input type="image" name="submit" src="image.gif">
```

The method attribute:

The method attribute specifies how to send form data. The form-data can be sent as URL variables (with `method="get"`) or as HTTP post transaction (with `method="post"`).

Both GET and POST create an array (e.g. `array(key => value, key2 => value2, key3 => value3, ...)`). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as \$_GET and \$_POST. \$_GET is an array of variables passed to the current script via the URL parameters. \$_POST is an array of variables passed to the current script via the HTTP POST method. Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send. The `<fieldset>` tag in forms is used to group related items together.

Saving form data into a database:

When data is submitted to a form using the input fields by the user pressing the submit button, the data is sent to the form quoted in the input form. Then the PHP file is written for **'insert'**ing connects to the MySQL database server, retrieves forms fields using the PHP \$_REQUEST variables and finally execute the insert query to add the records.

Here is an example code of HTML file to read the values for the variables using the input types and then a sample code for insert.php file to write the data back to the MySQL database.

```
</head>
<body>
  <form action="insert.php" method="post">
    <p>
      <label for="firstName">First Name:</label>
      <input type="text" name="first_name" id="firstName">
    </p>
    <p>
      <label for="lastName">Last Name:</label>
      <input type="text" name="last_name" id="lastName">
    </p>
    <p>
      <label for="emailAddress">Email Address:</label>
      <input type="text" name="email" id="emailAddress">
    </p>
    <input type="submit" value="Submit">
  </form>
</body>
```

Now the insert.php file to write back the data:

```
<?php // it is assumed that you have the necessary credentials to connect to mysql
$link = mysqli_connect("localhost", "root", "passwd", "demo");
// Check connection
if($link === false){ die("ERROR: Could not connect. " . mysqli_connect_error()); }
$first_name = mysqli_real_escape_string($link, $_REQUEST['first_name']);
$last_name = mysqli_real_escape_string($link, $_REQUEST['last_name']);
$email = mysqli_real_escape_string($link, $_REQUEST['email']);
// Attempt insert query execution
$sql = "INSERT INTO persons (first_name, last_name, email) VALUES ('$first_name',
    '$last_name', '$email')";
if(mysqli_query($link, $sql))
    { echo "Records added successfully."; }
else
    {echo "ERROR: Could not able to execute $sql. " . mysqli_error($link); }
// Close connection
mysqli_close($link); ?>
```

**To connect to the Database using PHP there are number of methods available.
In this section we discuss the following methods.**

- MySQLi object-oriented method
- MySQLi procedural method

Connect to a Database Using MySQLi (object-oriented) method

```
<html>
<body>
    <h1>Create DB </h1>
    <?php
        $servername = "localhost";
        $username = "root";
        $password = "";
        // Create connection
        $conn = new mysqli($servername, $username, $password);
        // Check connection
        if ($conn->connect_error) {die("Connection failed: " . $conn->connect_error);
        } else{
            echo "Connected successfully";
        }
    ?>
```

Connect to a Database Using MySQLi procedural method

```
$conn = mysqli_connect($servername, $username, $password);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error()); }
$sql = "CREATE DATABASE StudentDB2";

if (mysqli_query($conn, $sql)) {

    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}
mysqli_close($conn);
```

Database Operations

The following operations can be performed on a database using php.

- **Inserting Data to a table**
- **Display Data from a table**
- **Updating Data**
- **Delete Data from a table**

Inserting Data to a table

Suppose that we are trying to insert some new data (a complete record) in to the student information table named StInfo, which contains Student identity card number, Initial, Surname and Distance from home as the data of a student. The field names of the data are StNIC, Init, Surname and HDistance respectively:

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "StudentDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO StInfo (StNIC, Init, Surname, HDistance)
VALUES ('200212312512', 'M.', 'Silva',4)";
if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>

$sql = "INSERT INTO StInfo (StNIC, Init, Surname, HDistance)
VALUES ('200212312512', 'M.', 'Silva',4)";
if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
```

```

} else {
    echo "Error. Data could not be entered: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>

```

Display Data from a table

Data that is stored in the database tables can be viewed using php. For this purpose, data needs to be fetched from the database table using SELECT statement and then can be displayed:

```

<html>
<body>
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "StudentDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error); }
$sql = "SELECT * FROM StInfo";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "NIC: " . $row["StNIC"]. " - Name: " . $row["Init"]. " " . $row["Surname"]. "
Home Distance: ".$row["HDistance"]. "<br/>"; }
    } else {
        echo "0 results";
    }
}
$conn->close();
?></body></html>

```

In the above code, each row of data is fetched from the database table and is then copied in to the variable **\$row**. The values stored in the row are then get printed.

Updating Data

Data that already exists in the table can be updated with the new information in php using the UPDATE statement. In order to update the existing data, that data needs to be located first and then can be updated with new information. Typically the Primary Key is used to locate the data in the table and then the data in the retrieved row gets updated:

```
<html><body><?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "StudentDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "UPDATE StInfo SET Surname='Amarasinghe' WHERE
StNIC='200242134212'";
if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}
$conn->close();
?>
</body>
</html>
```

In the above example we search and locate a student with the StNIC value of 200242134212 and then update the surname of the student with new data.

Delete Data from a table

We often want to remove the data from tables permanently and DELETE statement allows us to carryout this operation in php. To delete data from tables, we first need to locate the row containing the data and then have to remove the data from the table:

```
<html><body><?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "StudentDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
//We display all the data from the table here
$sql = "SELECT * FROM StInfo";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    // output data of each row
```



```

        while($row = $result->fetch_assoc()) {
            echo "<tr><td>" . $row["StNIC"]. "</td><td>" . $row["Init"]. "</td><td>" .
            $row["Surname"]."</td><td>".$row["HDistance"]. "</td></tr>"; }
        } else { echo "0 results"; }
    $conn->close();
    echo "</table>";
?>

$StNIC = $_POST['st_Nic'];
$sql = DELETE FROM StInfo WHERE StNIC = $StNIC ;
if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}
$conn->close();
?>
</body>
</html>

```

In the above example student table is searched to locate a row of data with give student NIC number and then the data gets deleted.

Competency Level 10.8: Publishes and maintains web sites

Learning Outcomes:

- Publishes the developed website locally
- Identifies free web hosting sites from the Internet
- Publishes the developed website through a free web hosting site
- Investigates the factors affecting performance of website

Publishes and maintains websites

Publishing a website

The developer may choose to publish the website locally and it involves many things from the user's side. It involves that your organization has a good Internet connection, has obtained a static IP address from your ISP, ensuring that your router has been properly set therefore it can forward traffic to port 80, configure Windows Firewall to allow your webserver to communicate on port 80, you can set up Apache webserver on your machine, and then can test the webserver from your computer. Once everything works fine you may replace the default home page of the server with the home page you have designed.

Publishing a website can also be done by the professional web site hosting companies. Godaddy is a popular hosting provider and there are many such hosting companies available in the world. Getting help from such companies relieves you from the hassle of maintaining web servers, ensuring uninterrupted connectivity and keeping the website secure.

Maintaining a website

Websites must be maintained regularly if you want your web site to be a successful one. Regular maintenance of a web site can keep regular visitors of your web site happy because your web site gives them more updated fresh information and provides exciting news. In addition, maintaining your web site on a regular basis keeps all parts of your web site active and keeps the links and pages unbroken.

Maintenance of a website also involves regular scheduled backup of the website. Therefore, it ensures that you have the latest copy of your website as a backup and it prevents you from losing your website. Maintaining your website keeps your website more secure since it prevents your website from hacking. Visitors do not prefer to visit websites that are hacked. Software that is used to create a website is often get updated with the latest release that fixes many of the bugs in the system and comes up with more security measures. When such a release is available always make sure that you also keep your software get updated. The final thing to consider is collecting your site statistics. This will help you to identify the number of potential visitors to your website and determine how popular your website is among the users.

Exercises to workout in PHP form handling

1. Write a simple number - guessing game in PHP. The script should “think” of a random number between 1 and 100, then give the user five chances to guess the number. For each guess, the script should report whether the guessed number was too low, too high, or correct. (Hint: Use rand (1, 100) to generate a random number between 1 and 100.)
2. Create a php script that displays a form allowing the user to select one of three Amazon stores - amazon.com, amazon.ca, and amazon.co.uk — and then jumps to the relevant store based on the user’ s choice.
3. Design a php script to display the health suggestions for user based on the Body Mass Index (BMI) value. The formula is $BMI = \text{kg}/\text{m}^2$ where kg is a person’s weight in kilograms and m^2 is their height in metres squared.

Under weight - < 18.5
Normal – 18.5 – 24.9
Overweight – 25 – 29.9
Obese – 30 – 34.9
Extremely Obese – 35<

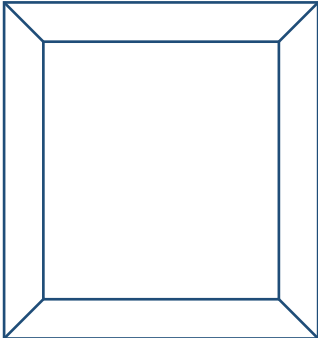
The measurement of height can be (cm, m, inch)

Weight:

Height:

Your Body Mass Index is BMI VALUE. This is considered SUGGESTION

4. Design a web page for kids' art gallery. This web page has two pages. In page one allows kids to upload their drawings, name of the kid, school and a small description about their drawing. Once they have submitted these information redirect to the second page which is needs to show that drawing with kid name, school name and the description as follows

	Name of the Kid School Name Description of the drawing
---	--

5. Create a script for displaying countries based on their population order. Ask user to enter the country name and the amount of population. Allow user to enter these details for five countries one by one. Once the count became five ask the user to select the displaying order. If they select ascending display the countries name in the descending order of their population else descending order.

Country Name:	<input type="text"/>
Population:	<input type="text"/>
	<input type="button" value="Next"/>

Display Mode:	<input type="button" value="▼"/>
	<input type="button" value="Display"/>
Countries' name in ascending/ descending order	

Competency 12: Explores applicability of ICT to business organizations and the competitive marketplace

Competency Level 12.1: Explores the role of ICT in the world of business

Learning Outcomes:

- Defines digital economy
- Lists and describes new business methods in digital economy
- Identifies the concepts behind pure brick, brick and click, and pure click organizations
- Describes the role of ICT in business functions of an organization

E-Commerce

ICT, Business and Competitiveness

Role of ICT in Modern Business

The relationship between Information and Communication Technology (ICT) and businesses is ever-growing. For few decades, ICT has been playing the role of the key enabler for successful businesses. In other words, ICT enables providing solutions to various business problems, especially such as inefficiencies in business processes, lack of productivity, challenging business competition, ineffective business models, increasing expectations of customers, inability to reach wider markets, etc. It all started with ICT being successfully deployed in computerizing the manual business processes of companies a few decades ago. Since then, the ICT kept on advancing every year and no public or private company nowadays can think of business success excluding ICT from the picture.

Digital Economy

The digital economy is an economy based on digital computing technologies¹. In other words, economic activities in a digital economy happen through digital mediums. Economies of most countries are now transformed into digital economies. Think about the way people read newspapers nowadays. Many people in Sri Lanka, especially the younger generation, do not buy printed newspapers. Instead, they read newspapers online. Not only that; they provide instant feedback on articles they read, share them among friends on social media and conduct discussions based on them. This scenario reflects a very important development which is a result of the advancement of ICT. That is, the advancement of ICT has created a platform for people to engage in economic activities in digital form. For example, people can search for products online, visually inspect their features in 2D and 3D form and purchase products online by making payments using digital or other virtual currencies. Another example of this is, a student can take up short-term freelancing jobs offered by companies in other countries such as translation jobs online, as part-time work to support their education. Thus, notably, the world's economy is more and more transforming to a digital economy with the advancement of ICT in which the production, distribution, and consumption of goods and services happening in digital form.

¹ This should not be confused with the subject you learn as Economics. Students should focus on the term 'Digital Economy'.

What advancement, in particular, has made this transformation? It is Internet technology. With Internet technology, especially the Web 2.0 technology, we can see a huge array of digital products such as product catalogs, TV and Radio programs, video content, music, books, magazines, newspaper articles, software and many more being produced, distributed and consumed worldwide. The internet technology has also given rise to the use of virtual currencies such as cryptocurrencies² making alternative digital forms of payments available for merchants and consumers. The internet technology has further made it possible nowadays to connect people with people but also people with things as well as things with things! For example, home appliances such as washing machines and refrigerators come with embedded microcontrollers and networking capabilities so that they could be connected via the Internet with human users as well as other appliances (and systems) alike. These capabilities made possible by the advancing Internet technology have reduced the geographical, social and cultural distances between firms and people around the world, making the entire world one global village. This is what we call globalization.

ICT and Business Transformation

ICT, as the key enabler of modern business success, transforms the way companies operate. With more and more people browsing the Internet, especially using mobile and handheld devices, and with more and more people use social networks, organizations have equal challenges and opportunities in the market. When people are exposed to more and more information through the Internet about products and services, their expectations as customers elevate rapidly. On the other hand, due to globalization, business competition among firms too tends to grow as international market players could enter into markets avoiding geographical barriers with low-cost substitute products as well as products with world-class technologies. However, wiser organizations transform themselves strategically using ICT effectively to explore more and more opportunities in the digital economy while facing the challenges arisen by the same.

The key ICT-based tool for business transformation is the *information system*. Organizations use information systems to strategically transform themselves to get a competitive advantage. That means information systems help organizations to implement a business strategy to operate better than its competitors in the market. An ICT-based information system uses computing technologies to collect, store and disseminate business data and, process them to get useful information and make them available for key stakeholders of the firm (such as managers, suppliers, and customers) to make decisions. Particularly, information systems help organizations to transform themselves into *digital firms* in which the key business processes are accomplished through digital networks, key business relationships are digitally enabled and mediated and the key business assets are digitally managed³. For example, Information systems help placing and handling orders as well as maintaining relationships with suppliers electronically letting them to view internal information such as production schedules and getting the information transferred via electronic data interchange (EDI).

ICT-enabled digital firms can transform themselves with new business models as alternatives for traditional business models. A *business model* is a particular way of doing business. *Pure brick* organizations, which are usually selling physical products offline are typically following a traditional business model where the customers visit the store, talk to sales staff, place orders at sales counters, make cash payments (unless corporate customers to whom some credit period

² Read about bitcoins and other similar cryptocurrencies being used in online business

³ Read Chapter 01 of Management Information Systems by Laudon and Laudon (12E or later) for more details

will be given), produce the invoice to the warehouse, collect items and leave. However, with the advancement of ICT, organizations can become *pure click* organizations as well as *brick and click* organizations and come up with innovative business models which strategically help them to get a competitive advantage. For example, most modern insurance companies and banks have become brick and click organizations with innovative business models that make use of the Internet. That means, such organizations have introduced an online business component in addition to the traditional offline business component. For example, in the case of motor insurance claiming, apart from the traditional model of motor insurance claiming some insurance firms offer to click to claim facility where the driver can contact the call center over the phone, proceeding the details of an accident, uploading photos of the necessary documents as well as the details damages through a mobile app and leave the scene after a verification process⁴.

Some other good examples of new business models are reverse auctions, group purchasing, and e-marketplaces. In reverse auctions, traditional roles of the buyer and seller are reversed. Instead of buyers who compete for goods by making bids, in reverse auctions sellers compete to sell the product by placing bids in response to a product or service request placed by a single buyer⁵. Group purchasing, on the other hand, helps making use of economies of scale. A seller can offer goods at a price lower than the set price and still make a good profit if the quantity sold is larger. Group purchasing makes use of this opportunity by getting orders from several individuals, aggregating them into a single order (as one order from a group of buyers) and placing it with a seller who is willing to fulfill that order. E-market places are virtual platforms (online platforms) on which buyers and sellers can meet. In such e-marketplaces, sellers can open virtual stores and buyers can search for goods and make purchases online⁶. Notably, these three business models are successfully being used by pure click organizations which entirely exist on the Internet. Such organizations do not have physical premises and they do not engage in offline business as well. With well-set communication, payment and distribution infrastructure in place (thankful to ICT), pure click organizations ensure that the right item is delivered to the right customer at right time in the right quality.

ICT Applications in Business Functions

ICT applications in business are diverse. On one hand, it could be in diverse industries including health, education, agriculture, banking and hospitality. On the other hand, it could be in business functions such as accounting, human resource management, production, sales, and marketing. Other than that, there are common ICT applications in supply chain management (SCM), customer relationship management (CRM) and business communication. In any case, however, ICT application involves building an information system which helps converting data to useful information and thereby helps managing a particular business process efficiently and effectively. Furthermore, advancing the Internet of things (IoT) and mobile computing technologies as well as the pervasive use of smartphones enables smart systems which supports the foregoing of many industries. For example, smart classrooms, smart health systems, and smart agricultural systems extensively make use of such technologies to fulfill various objectives in respective industries.

⁴ Read about Click2Claim of fairfirst Insurance, Sri Lanka.

⁵ Watch the video <https://www.youtube.com/watch?v=vdqPHgGKgjU> for more details.

⁶ Visit www.ebay.com and see how an e-marketplace work. Note that eBay is also an example for a forward auction.

Use of ICT in business functions as information systems could be broadly classified into two categories; transaction processing and decision support. Transactions are elementary activities being recorded in organizations. For example, a sale taking place at a supermarket is a transaction. Similarly, a new delivery of stocks, recruitment of a new employee, a cash withdrawal at an ATM by a customer, a book borrowed at a library and a new item manufactured at a factory are examples for transactions. Every business function needs information systems to record such transactions for future use. Typically, transaction processing systems help keeping track of such transactions in business functions. On the other hand, managers need information systems to help them to make decisions. For example, at the end of the day, some managers may need to know how much raw material stock is remaining to decide whether to make a raw material order or not. Another group of managers may want to forecast sales in the year to come by simulating sales patterns according to different market conditions. The most senior managers would need to know the overall health of the organizations to make strategies for a future couple of years. Organizations use management information systems (MIS), decision support systems (DSS) and executive support systems (ESS) for these purposes.

ICT enabled information systems are being used in every business function with unique objectives and characteristics.

- **Accounting:** According to the business dictionary⁷, accounting is the practice and body of knowledge concerned primarily with methods for recording transactions, keeping financial records, performing internal audits, reporting and analyzing financial information to the management and advising on taxation matters. Thus, it is the systematic process of dealing with incomes, expenditures, assets, and liabilities of an organization. For example, recording transactions pertaining to incomes and expenditures of a given period helps calculating the overall profit or loss during that period. Accounting information systems help performing various accounting activities with the help of ICT⁸. For example, handling accounts payable is an activity of the Accounts Department. It will handle payments to be made to the suppliers for items they supplied to the company such as raw materials. There, the system helps to keep track of purchase orders made to suppliers, inventory records on goods actually received (according to purchase orders) and the invoices received from suppliers pertaining to the purchase orders they completed. The system also helps to cross-check the purchase order details, inventory records and the details on the invoices and verify before authorizing payments. This could have been a tedious task if it was done manually but the ICT-based information systems make it accurate, efficient and fast.
- **Human Resources Management (HRM):** Human resource management function deals with managing people of the organization. Activities of the HRM function include recruitment, transfer, training, evaluation, maintaining attendance, etc. HRM function utilizes ICT-based information systems for managing these activities. In fact, in modern human resource information systems (HRIS), there exist modules, which mean sub-systems, to deal with those activities separately, For example, the recruitment module will help to manage the recruitment process by keeping (transaction) data related to managing CVs, screening for interviews, interview results, recruitments, etc. Apart from these, some organizations provide self-service systems to employees to access their data related to salary, deductions, leaves, and attendance. This is typically done through a web portal in which the employees login to the system through a web interface. Once logged into the portal, employees can see the services made available to them through the system.

⁷ Read www.businessdictionary.com/definition/accounting.html for more details.

⁸ Accounting information systems heavily rely on data from other functions. See <https://www.slideshare.net/wiweck/accounting-information-system-18527651> for details.

- **Production:** According to Laudon and Laudon, production management is concerned with the planning and control of all those activities that transform inputs into outputs by adding some value to inputs. For example, in a soft drinks production plant, inputs such as sugar, flavors, water, etc. will be converted into soft drinks and come out in containers (bottles) as the output. In such a plant, the managers have to plan what to produce, when, in what quantities and where to distribute. If they produce the wrong product, that will be rejected by the consumers. If they produce more, excess stock will be remaining in the warehouses. If they produce less, there will be shortages in the market. This will make the consumers unhappy. Thus, there exist standard activities in production management such as product design, development, production planning, scheduling, etc. All these activities require data and, once again ICT-based information systems play a critical role in assisting managers to manage the production.
- **Sales and Marketing:** No business can survive without winning the hearts of the customers. All companies try to make the right product available to customers through the right channel at the right price. Therefore, all successful firms attempt to understand their customers better and try to keep them satisfied and loyal to the company. ICT provides means for companies to understand their customers by segmenting them based on purchase data, knowing what items are usually bought together and also by predicting who will buy what, when and where. Furthermore, Web 2.0 technology, as well as the resulting social media platforms, has enabled the companies to listen to their customers in the form of ratings, feedbacks, status updates, blogs, comments, tags, etc. Analysis of these different types of data gives rich market insights to firms to compete better in the market.
- **Supply Chain Management:** Supply chain links a firm with its external partners such as raw material suppliers, logistic companies and distributors. However, if the activities of these partners are not synchronized with the production activities of the firm, the end products cannot be successfully offered to the market. ICT-based supply chain management systems⁹ play a critical role here by automating the flow of information across organizational boundaries and thereby linking external actors in the supply chain with the firm. For example, supplier organizations can view production schedules of the firm (by connecting to the production management information system through a web interface) and accordingly adjust their production schedule, so that an uninterrupted raw material supply is ensured.
- **Business Communication:** ICT has changed the way communication happens in organizations. E_mail had been a popular mode of business communication for a couple of decades. With the development of broadband technologies, communication platforms such as Skype and Zoom became popular for video conferencing between employees and managers in geographically dispersed locations. Moreover, the broadband Internet, as well as the pervasive use of mobile devices, has given rise to more mobile-based communication platforms such as Viber and WhatsApp. These mobile apps help taking voice and video, sharing documents, pictures, videos and other files, as well as instant messaging. Due to these changes in business communication enabled by ICT, it has even made it possible for employees to work from anywhere (teleworking, telecommuting) which is getting increasingly popular especially among people like freelancers and working mothers.

Moreover, emerging technologies such as the Internet of Things (IoT), Big Data and Cloud Computing together with social computing has paved way for smarter systems to be increasingly

⁹ Search and read about the term inter-organizational systems on Google.

applied in the context of business management and communication. For example, paradigms such as Industry 4.0¹⁰ focuses on creating cyber-physical systems of automation and data exchange in manufacturing environments with the help of sensors, Internet-based communication links, and big data analytics. Smart systems are getting popular in industries such as agriculture, healthcare, education, logistics as well as hospitality. For example, smart classrooms facilitate value-added learning for the learners whereas tour guide systems supported by augmented reality help tourists to have better experiences worldwide¹¹.

¹⁰ Read the article on https://www.baslerweb.com/en/vision-campus/markets-and-applications/image-processing-industry-4-0/?gclid=EAlalQobChMI9sDOv8S62gIV1iMrCh0SgwgqEAAYAiAAEgKUIvD_BwE

¹¹ Read the article on <https://thinkdigital.travel/wp-content/uploads/2013/04/10-AR-Best-Practices-in-Tourism.pdf>

Competency Level 12.2 Analyses the relationship between ICT and business operations

Learning Outcomes:

- Distinguishes the e-commerce and e-business
- Investigates the scope of e-commerce and e-business
- Lists and briefly describes the types of e-commerce transactions
- Identifies the advantages and disadvantages of e-commerce

Electronic Business

Electronic business (E-business) could be simply explained as doing business electronically. This means that the business processes are conducted with the help of a computer network based on Internet technology. It helps data and information to be shared quickly within and across business functions, as well as external parties, and thereby the business processes to be conducted effectively and efficiently. As most early security concerns pertaining to the use of the Internet have now been addressed, we could see major organizations re-thinking their businesses in terms of the Internet and conducting e-business to buy parts and supplies from other companies, collaborate on sales promotions, and conduct joint research¹².

E-Business Operation

Most students get confused with the two terms of *electronic commerce (e-commerce)* and *electronic business*. Compared to e-commerce, e-business is a broader concept. E-commerce is actually a part of e-business. Broadly speaking e-business facilitates business transactions to be conducted electronically. Think about the order fulfillment process of a company which happens electronically. It is actually a cross-functional process which involves the sales, production, finance, and warehouse (and distribution) functions of the organization. Therefore, there exists an information system module (i.e. an information system component) for the business activities of each function. However, as shown in *figure 4.2.1* these modules are integrated as a single unit by connecting them to a single database through internet technology.

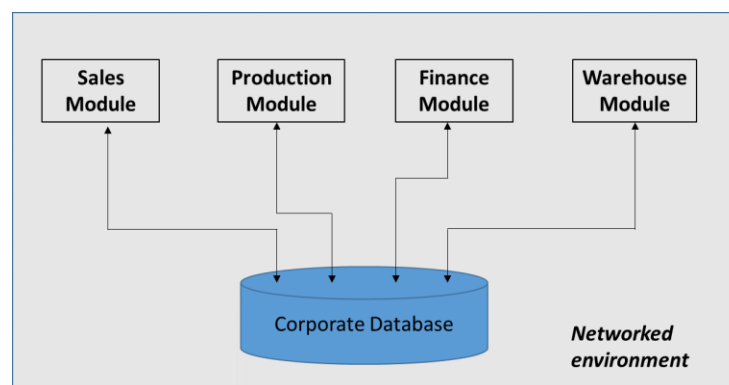


Figure 4.2.1: Integrated modules in an e-business environment

With this architecture, details of transactions of one function become accessible to the other functions and even for external parties¹³. For example, a purchase order generated through the sales module will be saved in the database. The production module, as well as the finance module, will simultaneously get an alert of the new purchase order added into the system. Based on that, the production staff will view the corresponding items to be produced and update

¹² Read the article <https://searchcio.techtarget.com/definition/e-business>

¹³ There involves security measures to avoid information getting into unnecessary hands.

production schedules accordingly through the production module. At the same time, the finance staff will generate the invoice according to the requested items in the purchase order as they too can view the items through the finance module. Once the production is completed the status of the purchase order will be updated through the production module as completed and the warehouse staff will accordingly be alerted. The warehouse staff will then arrange the shipment once the payment is done according to the invoice generated earlier. Notably in this scenario, the electronic form of transactions enables better coordination among business functions and reduces the chances of errors and delays taking place in the process. Furthermore, access to the central corporate database could be provided to external parties such as customers, suppliers, and distributors through appropriate modules and interfaces. For example, when you access your bank account through an e-banking system and do online transactions, you actually perform e-business transactions¹⁴. Practically, organizations adopt e-business capabilities through enterprise resource planning (ERP) systems which has a broader set of standard modules for various functions of a typical organization¹⁵.

Electronic Commerce

Electronic commerce (e-commerce) is simply a part of electronic business. E-commerce performs buying and selling activities of products and services online. For example, an e-commerce transaction takes place when you buy a book from Amazon over the Internet. Similarly, when you transfer money from your account to someone else's account using your bank's e-banking service, an e-commerce transaction takes place. The difference, in this case, is the transaction involves a service, not a product. E-commerce could be seen as a system of various components and actors that act and interact to enable online transactions to happen to exchange goods and services. Those factors include online marketplaces, virtual storefronts, information brokers, content providers, portals, online service providers and virtual communities.

- **Online marketplace:** Online marketplace is an e-commerce site which acts as a platform for multiple large and small-scale sellers to offer their products and services to a global pool of customers. For example, individual merchants can open virtual stores on Amazon's platform (Amazon's online marketplace) and sell their products and services. Rover.com is another interesting online marketplace on which a range of products and services related to pet care is being offered¹⁶.
- **Virtual storefronts:** Virtual storefronts make use of digital technologies to display products being offered on an electronic screen. Tesco is a famous company for opening virtual storefronts in places like airports and subway stations. Figure 4.2.2¹⁷ shows a virtual storefront opened by Tesco at an airport and a customer adding items on display to a shopping cart using a mobile app by scanning the QR code associated with each item. Through the app, the customer can confirm her order and Tesco delivers the goods to the intended destination. On the other hand, all online shops available in online market places are examples of virtual storefronts.

¹⁴ E-banking is actually e-commerce but it is a part of e-business.

¹⁵ Read the article <http://www.netsuite.com/portal/resource/articles/erp/what-is-erp.shtml>

¹⁶ Read the article <https://hackernoon.com/top-5-most-popular-online-marketplaces-how-to-join-the-champions-league-a313dbdfd338>

¹⁷ Figure source 4.2.2: <http://vonbismark.com/wp-content/uploads/2012/08/tesco-trials-interactive-virtual-store-gatwick-0.jpg>



Figure4.2.2: A customer purchasing from a virtual storefront

- **Information brokers:** Information brokers are the sites that help consumers to filter the required information from a myriad of publicly available information sources. For example, an information broker site will help you to find the site that sells the book you look for to purchase at the lowest price¹⁸. This helps consumers to save their valuable time searching for information and doing comparisons based on multiple criteria manually. Information brokering sites utilize agent technology for the brokering process.
- **Content providers:** Content providers are the sites that offer digital content such as news, music, photos, videos and games online. One of the best examples for a content provider is Apple Inc.'s iTunes Store which revolutionized the music industry of the world. Content providers make money by charging for downloads, providing display space for advertisements or subscription fees¹⁹.
- **Portals:** Portals provide the entry point to e-commerce by aggregating data from a large number of providers and letting customers search for goods and services²⁰. Most modern portals not only provide the searching facility but also provide the facilities for personalizing the content based on user profile make recommendations on products and services and deliver personalized advertisements²¹. The search engine www.yahoo.com is a very good example of an online portal through which the users can reach numerous categories of products and services.
- **Online service providers:** Online service providers provide various services online such as news, search engine, banking, health care, entertainment, social networking, etc. Facebook, for example, provides social networking services to worldwide users online. Google, on the other hand, provides various online services including search engine, email, file storage, and social networking.
- **Virtual communities:** Virtual communities are formed by individuals connected through online social networks sharing similar interests, opinions or feelings¹. For example, the fan

¹⁸ Read the example on http://www.cs.toronto.edu/km/xib/document/broker_tutorial/definition.html

¹⁹ Read the article <http://ecommercetoyouu.blogspot.com/2015/04/content-provider.html>

²⁰ Read the article <https://www.ecommercetimes.com/story/61955.html> for examples

²¹ Read <http://smallbusiness.chron.com/portal-business-model-3869.html>

pages and groups on Facebook are virtual communities which comprise of individuals who have similar interests. Virtual communities play an important role in e-commerce as potential pools of customers as well as an effective means of marketing. Virtual communities can also provide resources to firms online in the form of opinions, insights, expertise as well as creative skills.

E-commerce can be classified into several classes such as B2B, B2C, C2B, C2C, B2E, and G2C. This classification is done based on the two parties involved.

- **B2B:** Business to Business (B2B) e-commerce happens when a business organizations purchase goods or services from another business organization online. For example, large manufacturing organizations buy rawmaterials and accessories required for production from suppliers online²².
- **B2C:** Business to Consumer (B2C) is the most common form of e-commerce, which means the online buying and selling transactions happening between businesses and their end-customers. Individuals purchasing electronic items online or customers using e-banking services are examples of B2C e-commerce.
- **C2B:** Consumer to Business (C2B) involves consumers offering services (or any kind of resource) to businesses online. For example, some consumers write blog posts on behalf of business organizations after consuming their products which attracts more customers to the firm. Reverse auctions where the consumers name the price for a product or a service is another example for C2B e-commerce²³.
- **C2C:** Consumer to Consumer (C2C) e-commerce is a type of commerce that happens between two online consumers where one sells a product or a service to the other on a dedicated virtual platform. The popular online auction site eBay.com is a renowned example for such a virtual platform dedicated for C2C e-commerce.
- **B2E:** Business to Employee (B2E) e-commerce happens when business firms offer online services to their employees. The objective of B2E is to attract and retain high-quality employees to the organization by offering them various incentives such as flexible working hours, training opportunities and otherbenefits. Businesses typically use online portals to enable B2E e-commerce²⁴. Employees can log into the portal using their usernames and passwords and consume the services offered to them through the portal.
- **G2C:** Government to Citizen (G2C) means governments offering various services to their citizens through the Internet. For example, the Office of the Provincial Commissioner of Motor Traffic (Western Province) in Sri Lanka offers the online facility of obtaining revenue licenses for motor vehicles to the citizens²⁵. This type of e-commerce has become popular with e-government initiatives around the world.

As mobile devices get proliferated, more and more people browse the Internet using mobile and other handheld devices. As such, firms have now gone beyond traditional e-commerce and have focused on mobile-based e-commerce transactions which are commonly known as m-commerce. Mobile-friendly web sites, as well as mobile based information services such as geo-advertising and geo-information services, are getting widespread making m-commerce the next generation electronic commerce.

²² Read <https://www.linkedin.com/pulse/worlds-12-b2b-websites-currently-holds-top-ranks-saqib-ilyas/>

²³ Read <https://www.businessnewsdaily.com/5001-what-is-c2b.html>

²⁴ Read <https://searchcio.techtarget.com/definition/B2E>

²⁵ <https://www.erevenuelicense.motortraffic.wv.gov.lk/erl/view/logout.action>

ICT has also contributed largely to creating more secure and alternative payment mechanisms for business transactions. Especially in electronic commerce, several alternative payment mechanisms are preferred over traditional cash-based payment methods such as cash on delivery (COD). One of the key ICT-based developments in payment mechanisms is the payment gateways which enable payments using credit/debit cards. A payment gateway is a software service that securely handles the credit card information exchange and verification in order to complete an electronic commerce transaction. Once the customer authenticated a credit card payment for an online purchase he/she wishes to proceed, the merchant's webserver securely collects the credit card details (in encrypted format) from the customer's browser and sends the details of the payment to its payment gateway. The payment gateway directs the details to the payment processor of the merchant's bank through which the details will be sent to the relevant card association (Visa, MasterCard, etc.) The card association directs the details to the card-issuing bank which responds with an accept/reject message after a verification process. The processor forwards the response to the payment gateway to be interpreted at the merchant's website. If accepted, the transaction gets completed and the customer gets a confirmation message, whereas if rejected, the transaction gets aborted delivering a failure message to the customer. The merchant submits all approved payments to the processor of his/her bank to process as a batch to get money transferred to the designated account²⁶.

One issue with online payments with payment gateways is the customer has to submit credit card details to every merchant (or rather at every merchant's web site) that they shop with. This increases the number of chances of getting your sensitive data being recorded by one of those merchants. There are third party systems such as PayPal that reduces this risk by mediating the payment process and allow sending and receiving money more securely. In the case of PayPal, both buyers and sellers can register with PayPal using a valid email address and link any number of bank accounts and credit cards to that. Buyers can then shop securely with sellers who allow payment with PayPal (usually those who have the 'Checkout with PayPal' option). A buyer who makes a payment through PayPal can choose the bank account or the credit card he/she wishes to use to make the payment. PayPal does not disclose any sensitive information of the buyer to the seller but mediates the payment authentication process and ensures that the seller receives money to the designated bank account in his/her PayPal account²⁷.

Virtual currencies, in particular, the cryptocurrencies are also getting popular as alternative payment mechanisms in online payments. A cryptocurrency is a digital equivalent to cash which could be used to exchange for goods and services. Cryptocurrencies use cryptography to control the generation of additional units as well as to secure transactions. Among the popular cryptocurrencies, there are Bitcoin, Litecoin, Ethereum, and Zcash. The cryptocurrencies work based on the technology called 'blockchain' and individuals can buy/sell cryptocurrencies through third-party websites (and their mobile apps) using credit or debit cards. However, some countries have fully banned or restricted the usage of cryptocurrencies. The central bank of Sri Lanka is also closely paying attention to the cryptocurrency-related activities happening in the country²⁸.

Apart from security, privacy is another challenging issue in e-commerce. E-commerce sites keep track of personal information of their visitors. Moreover, they monitor what customers are doing online by tracking what sites they visit, what products and services they search as well as what products and services do they buy. Using this information, they make profiles of customers and

²⁶ Read the article on <https://www.hostmerchantservices.com/articles/payment-gateway-articles/how-payment-gateways-work/>

²⁷ Watch the video on <https://www.youtube.com/watch?v=7aFqCqqaBZQ&t=230s>

²⁸ <https://www.cbsl.gov.lk/en/news/public-awareness-on-virtual-currencies-in-sri-lanka>

accordingly conduct targeted promotions. However, customers may not always be aware to what extent have they been monitored and there is a growing concern among customers about their privacy getting violated. Hence, Government agencies work on insisting e-commerce sites to have privacy policies to ensure that the customers' privacy is protected at their sites²⁹.

ICT in the Forefront

ICT is widely used by companies to connect with their customers and promote their products and services. The Internet and other electronic media provide a range of technologies and platforms to communicate a strong message to the customers regarding a firm's offering.

²⁹ Read the article <https://cyber.harvard.edu/olds/ecommerce/privacytext.html>

Competency Level 12.3 Analyses the ICT in terms of generating and delivering an improved products and services to consumers

Learning Outcomes:

- Defines e- marketing
- Identifies the role of ICT in e-marketing
- Investigates the usage of databases in marketing activities
to improve the product and services according to the requirements of the customers
- Identifies the ways of gaining competitive advantages using ICT

E_Marketing

Electronic marketing is also known as web marketing, Internet marketing, as well as online marketing. It is simply making use of electronic media, especially the Internet for marketing activities³⁰. Such activities typically include promotions, public relations as well as after-sales support. For example, many customers make purchases based on the emails they receive containing product catalogs and special offers. Social media that are being increasingly used by firms to stay connected with customers, share information regarding new offerings and promotions as well as to hear from customers any suggestions, complaints, as well as comments are some of the other examples. Social media, on the other hand, acts as the digital equivalent to word-of-mouth marketing as the customers help to spread the message about the firm's offerings by sharing, tagging, liking and commenting. Online advertisements are also very effective to spread the message about a particular product or a service. The advantages of e-marketing include the ability of a firm to adapt based on the customers' feedbacks, the ability to make an instant impact, the ability to directly approach specific market niches (based on profiling), the ability to analyze the responses in real-time, etc.³¹

Mobile Marketing

Mobile marketing is a revolutionary form of e-marketing. As we know, many people now use smartphones and other mobile handheld devices to browse the Internet. This shift from desktops to mobile devices has created new opportunities for marketers to use multiple channels to reach a customer through their mobile handheld devices. These channels may include email, SMS, MMS as well as the very popular social media such as Facebook, Viber, WhatsApp, etc. Technology is developing rapidly to create innovative ways to personalize the marketing messages sent to users as much as possible. According to the latest edition of the textbook - Management Information Systems by Laudon and Laudon – the proliferation of mobile devices introduces a new dimension to marketers to recognize their targets, which is the location. They introduce three notable location-based services the mobile marketers can use namely geosocial services, Geoadvertising and geoinformation services. The geosocial services help users to know where people with similar interests as him or her meet. Facebook is a popular social networking site that offers this service. Geoadvertising services help connecting local merchants with nearby customers. For example, the local restaurants can offer lunchtime promotions to the crowd within a specific distance expecting to attract a percentage of them as customers. Geoinformation services provide information about a subject, such as an exhibit at a museum, on a geographical map for the people passing nearby.

³⁰ Note: marketing is not only promotion or advertising

³¹ Read http://www.iaapa.org/docs/handout-archive---ops/Mon_KHAN_E-MARKETING.pdf

M-commerce is an important component of mobile marketing in which electronic commerce takes place through mobile devices. In other words, the customers today prefer to shop online for products and services through their mobile devices which makes firms to consider the mobile as an important channel to transact with customers. For example, most banks, insurance firms, and telco companies offer mobile apps for customers to receive their services through mobile devices. More and more firms offer mobile-friendly e-commerce sites expecting better engagement with the customers. There also exist mobile advertising platforms such as Apple's iAd and Google's AdMob.

Database Marketing

When the customers' online activities increase through both web and mobile interfaces, more and more databases become existing which contain information about current and prospective customers. These databases not only contain customers' personal information but also their web browsing patterns as well as location details. Database marketing involves making use of these databases to directly contact existing or potential customers regarding firms' offerings. Database marketing is a direct marketing technique and it could be well supported with intelligent data mining techniques. For example, mining of web access patterns of individual customers can help to create unique profiles of customers based on their interests and other characteristics. This helps to make more personalized offers to customers which are more likely to be accepted than a bulk offer sent to all customers irrespective of their likes and dislikes.

Review questions

1. Explain what is digital transformation
2. Discuss the digital transformation happened in various business fields using the modern taxi apps and the websites like Airbnb and Amazon
3. Explain how information technology becomes important during global emergency situations such as pandemics
4. Differentiate electronic business and electronic commerce
5. Write down different online payment methods
6. Explain the functionality of an online payment gateway using a suitable diagram
7. Explain two new examples for each of the e-commerce types described above
8. Explain how the Internet of Things, Big Data and Cloud Computing technologies could be used to create a Smart System
9. Explain how social media could be effectively used for businesses
10. Explain why the smart mobile phone has become an important tool in business
11. Briefly describe the functionality of an information system developed for human resource management
12. Discuss whether the famous website 'YouTube' an example for a content provider

Competency 13: Explores new trends and future directions of ICT

Competency Level 13.1 Explores new trends and future directions in computing

Learning Outcomes:

- Describes intelligent and emotional computing.
- Explains artificial intelligences
- Appreciates man- machine coexistences

New Trends and Future Directions of ICT

Nowadays ICT is changing our lives more than ever before. The speed and convenience of many of our day today activities rely on ICT. This heavy adoption of ICT by both individuals and markets has inspired scientists and engineers in ICT to actively do research on new information and communication technologies as well as their applications in various domains. The Gartner hype cycle³²– 2019 presented in Figure 5.1 depicts the state of adoption, maturity and social application of many emerging technologies that are related to ICT as of 2020. The hype cycle, which is a very useful tool to recognize top emerging technologies in the world, has five key phases that shows;

1. The inception or the trigger of a technology
2. Peak with inflated expectations
3. Trough of disillusionment with failed implementations and promises
4. The slope of enlightenment with success stories
5. Plateau of productivity with mainstream adoption

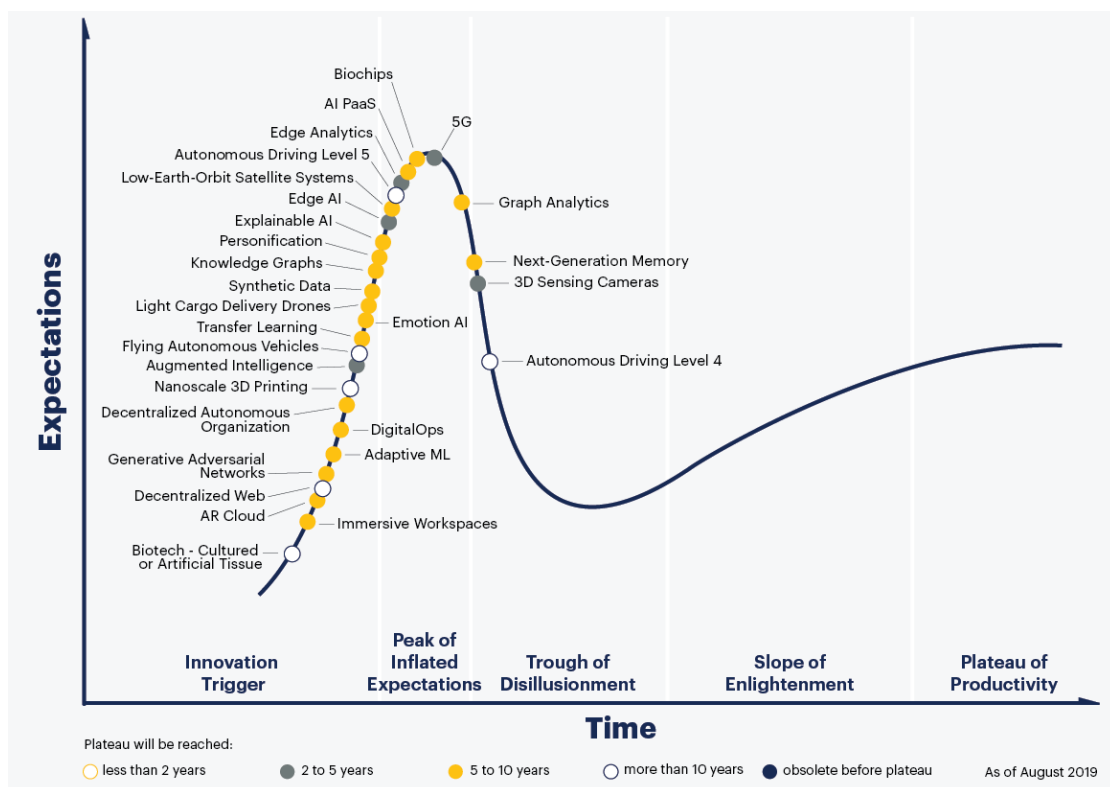


Figure 5.1: Gartner hype cycle – 2019

Source: <https://towardsdatascience.com/gartner-2019-hype-cycle-for-emerging-technologies-whats-in-it-for-ai-leaders-3d54ad6ffc53>

³² Read https://en.wikipedia.org/wiki/hype_cycle

The notable emerging and trending information and communication technologies mentioned in the hype cycles of 2018 and 2019 include Artificial Intelligence, Blockchain, IoT platforms, Quantum Computing and smart robots. Moreover, they mention about many other technologies that involve artificial intelligence as the core technology such as deep neural networks, autonomous driving, knowledge graphs, flying autonomous vehicles, brain-computer interface, etc. Thus, artificial intelligence, which leads to the creation of intelligent and emotional machines that help mankind in various endeavors, could be considered as one of the key emerging information and communication technologies.

Intelligent and emotional machines

Evidence on human interest about intelligent and emotional machines could be found even in the ancient Greek mythology, which contain characters such as Talos and Caucasian Eagle sent to earth by Zeus for human protection. Even the modern fictions such as ‘2001: A Space Odyssey’ by Sir Arthur C Clarke as well as the famous 80’s tele series ‘Knight Rider’ had machines that exhibit not only intelligent but also some emotional characteristics. For example, KITT, the intelligent car in Knight Rider, engages in several conversations with its master Michael Knight that shows some emotions. On the other hand, humans, throughout history, have attempted to build self-operating machines as well as machines with control mechanisms designed to automatically follow a predetermined sequence of operations, or respond to predetermined instructions. Those machines are known as automatons. However, the growth of such attempts to build truly intelligent machines had been very slow until the discipline of artificial intelligence was formally begun in the mid-20th century. Figure 5.2 contains the famous Maillardet's automaton found in Franklin Institute's museum.



Figure 5.2: The famous Maillardet's automaton in Franklin Institute's museum
Source: <https://www.fi.edu/history-resources/automaton>

Artificial intelligence

The term artificial intelligence could be considered as coined during the 1956 summer workshop held at the Dartmouth College in Hanover, New Hampshire. There, a group of researchers interested in thinking machines did an extended brainstorming session to elicit innovative ideas on creating such machines. Since then, various definitions of artificial intelligence have been proposed by scientists. According to one such definition, AI (artificial intelligence) is the simulation of human intelligence processes by machines, especially computer systems³³. The human intelligence processes include the ability to perceive the environment, synthesis perceived

³³<https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence>

information, learn and remember facts and apply knowledge to reason about events and solve problems. In general, the intelligence demonstrated by machines so far is considered only as imitating human (or natural) intelligence. This could be well explained by the two hypotheses presented by John Searle in 1980. According to Searle, artificial intelligence has two forms; strong form and the weak form.

- Artificial intelligence, strong form: an AI system can think and have a mind
- Artificial intelligence, weak form: an AI system can only pretend it thinks and has a mind

Thus, in order to create a truly intelligent and emotional machine, one must focus on the strong form of artificial intelligence. The researchers in the field of Artificial General Intelligence (AGI) are working on the creation of such machines (or computer systems) which are capable of working in multiple fields. Such a machine, for example, would learn to drive a car (one field) as well as defuse a landmine (another field). Moreover, machines with a strong form of AI demonstrate emotions since they have a mind. For example, such machines may learn to appreciate (truly) an object in terms of its beauty! There is another extension of the strong form of artificial intelligence called Artificial Super Intelligence, which is based on the idea that a system with the capabilities of an AGI, in the absence of the physical limitations of humans, would learn and improve far beyond the human level as a super system.

However, humans have yet been able to develop intelligent machines only with the weak form of artificial intelligence. That means, such machines demonstrate intelligence similar to humans (or even better than humans) when performing some narrow tasks such as playing a board game or predicting weather. However, their capabilities are limited only to that narrow task and their ability to improve their capabilities on other tasks is minimal. Therefore, such a form of artificial intelligence is also called artificial narrow intelligence. IBM Deep Blue that defeated the world champion Garry Kasparov in the game of Chess (1997), IBM Watson that defeated human players in a gameshow called Jeopardy (2011), and the algorithm called AlphaGo that defeated the human champion in the game of Go (2016) could be considered as machines with weak intelligence.

There are several techniques of artificial narrow intelligence developed by the artificial intelligence researchers to use when building intelligent systems and some notable techniques among them are mentioned in Table 5.1.

Man-machine coexistence

One of the key fears of mankind when it comes to the creation of intelligent machines is that they will one day take the place of humans. Thus, the research in artificial intelligence as well as ICT in general emphasis the need for man-machine coexistence in which humans and machines work together to create unprecedented breakthroughs in the workforce³⁴. For example, the Honda Research's humanoid robot Asimo was developed to work with humans in workplaces corporatively³⁵. Asimo has also demonstrated its ability to work with multiple other Asimos, which is an example for the concept of machine-machine coexistence.

³⁴ Read <https://www.forbes.com/sites/forbestechcouncil/2017/07/06/man-machine-and-multiplicity-how-ai-and-humans-can-coexist-harmoniously/#524d497317df>

³⁵ Watch video <https://www.youtube.com/watch?v=JIRPICfnmhw>

Technique	Description
Search techniques	Searching for a goal state in state space. For example, finding the winning state in a board game with a human player
Expert systems	Rule-based systems that store knowledge as If-Then rules and advice humans to solve problems with reasoning using the stored knowledge.
Natural Language Processing	Algorithms that recognize and even understand human languages
Speech recognition	Algorithms that recognize words in speech and understand short voice sentences
Computer vision and scene recognition	Algorithms that interpret and understand visual contents in scenes captured by cameras
Machine learning	A set of techniques that can learn hidden patterns in data. For example, machine learning algorithms can be used to predict whether a new customer will default a bank loan or not by exploring the past loan data of all customers of the bank
Neural networks	A notable technique in the domain of machine learning, which is based on a large network of information processing units called artificial neurons which are organized into two or more layers. A significant development in this area could be seen with the emergence of deep neural networks
Genetic algorithms	A class of optimization algorithms based on the biological process of evolution. There, similar to the survival of the fittest rule in biological species, current good (fit) solutions are combined to generate better solutions until a significantly optimal solution is obtained.
Fuzzy logic systems	A technique used to control systems smoothly through fuzzy rules based on linguistic statements. For example, if the room temperature is 'hot', turn the fan 'fast' is a fuzzy rule. Here 'temperature' and 'fan speed' are fuzzy variables that take fuzzy values like 'hot' and 'fast' rather than binary values 'true' or 'false'

Table 5.1: Intelligent Techniques

Competency Level 13.2 Explores the fundamentals and applications of agent technology

Number of Periods: 04

Learning Outcomes:

- Briefly describes software agents and their characteristics
- Briefly describes multi-agent systems and their characteristics
- Identifies the applications of agent systems

Software agents

In computer systems, a software agent is a piece of software that works on behalf of a human user or another computer application. Cortana, in Windows could be considered as an example for a software agent. It is a piece of software inside the Windows operating system that acts as a virtual assistant. It provides personalized services to the human user such as searching for apps and files, recognizing voice commands, giving reminders, etc. Software agents in most cases reflect (weak) intelligence by their abilities such as recognizing voices, interpreting the text in natural language, predicting based on past data, and reasoning based on observations. Software agents could be:

- **Autonomous:** can take decisions by themselves without the interference of the human user
- **Proactive:** can initiate actions on its environment
- **Reactive:** can react to events in the environment
- **Cooperative:** can cooperate with other agents and humans
- **Able to learn:** can learn by observation, experience, and data
- **Social ability:** can interact and communicate with others complying to the social rules and norms

These features actually help to differentiate a software agent from an object in software. A good classification of software agents is depicted in Figure 5.3.

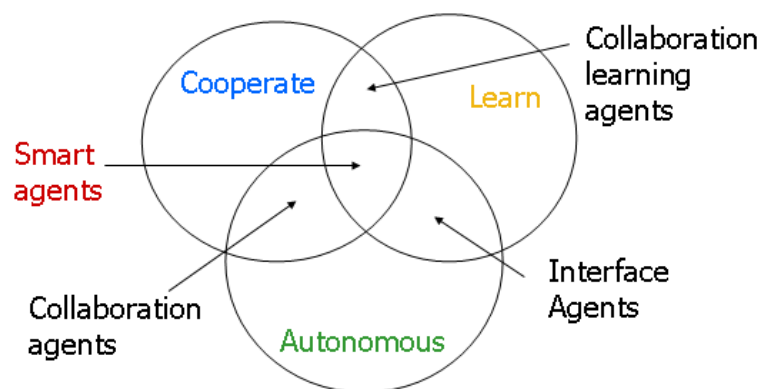


Figure 5.3: a classification of software agents based on Nwana’s primary attribute dimension
Source: https://en.wikipedia.org/wiki/Software_agent#/media/File:Nwana_Category_of_Software_Agents.gif

Multi Agent Systems (MAS)

Software agents can interact with other software agents to execute a common task. In most cases, a single software agent is not capable of solving complex problems. Thus, the applications aiming at solving complex problems require multiple software agents working as a single unit, which is called a Multi-Agent System or MAS. In other words, a multi-agent system (MAS) is a

loosely coupled network of software agents who interact to solve problems that cannot be solved by individual problem solvers³⁶. In such a multi-agent system, the agents are³⁷;

- **autonomous:** meaning that they can make independent decisions by being self-aware
- **having only a local view:** meaning that they only know how to execute their part and cannot see beyond their scope
- **acting in a decentralized manner:** meaning that there is no agent to control the system

Figure 5.4 depicts a general functionality of a multi-agent system in which a user gets some information from multiple sources through an information broker. Each source is assigned to an information agent who submits the relevant information to the broker who filters and provides the appropriate information to the user. The user accesses the system through an interface agent.

Applications of Agent systems

Agent systems are found in a range of applications. As mentioned before Cortana in Windows as well as Siri on Apple iOS systems are good examples for software agents that are practically implemented. Such virtual assistants could be seen in many computer systems to make the tasks of the human users easier. Agents also assist humans in banking, e-learning, e-commerce, booking, and in various other similar systems. For example, information searching in electronic commerce applications involves a multi-agent system similar to the architecture depicted in Figure 5.4. There, the users submit their search criteria for products to an interface agent. The interface agent then connects with an information broker agent who filters information received through agents at various merchants' websites in response to the users' search criteria to provide the relevant information to the users. Online booking systems too adopt such architecture to filter information on behalf of the users, connect to respective merchants and do the booking on behalf of the human user.

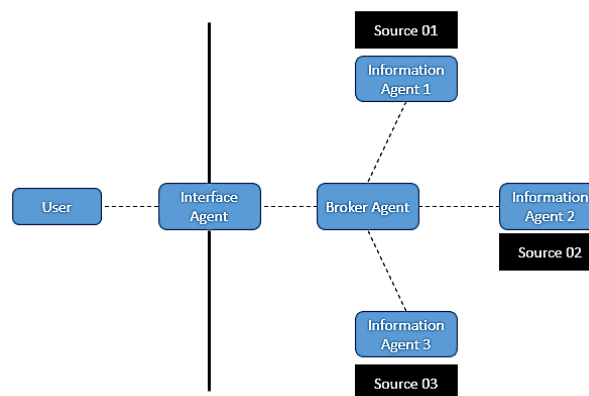


Figure 5.4: Architecture of a multi-agent system

³⁶<https://www.cs.cmu.edu/~softagents/multi.html>

³⁷https://en.wikipedia.org/wiki/Multi-agent_system

Competency Level 13.3 Analyzes the existing models of computing and proposes new models

Learning Outcomes:

- Predicts the technologies beyond von-Neumann computers

Beyond Von-Neumann computer

The von Neumann architecture is the basis for the modern computer. As depicted in Figure 5.5, it provides a simpler architecture to design computer systems. However, as the demand for computational resources rapidly grows, the von Neumann computer has now been pushed to its extreme limits of capabilities. According to Moore’s law, the processing speed of microprocessors doubles once every 18 months. However, there are physical limitations for this trend to continue further such as managing the excessive heat generated in microprocessors when they are functioning with more and more transistors embedded into them to increase the processing speed. Therefore, the researchers have already started to look for alternative architectures that can meet the future demand for computational resources conveniently.

Quantum computing

Quantum computing is an alternative approach proposed to overcome the limitations of the modern-day microprocessors. In the traditional approach, information is represented using a binary system which has only two states (0 and 1). However, quantum computers which are based on quantum physics, can represent information using multiple states. Therefore, quantum computers would be exponentially faster than the traditional computers based on the von Neumann architecture³⁸.

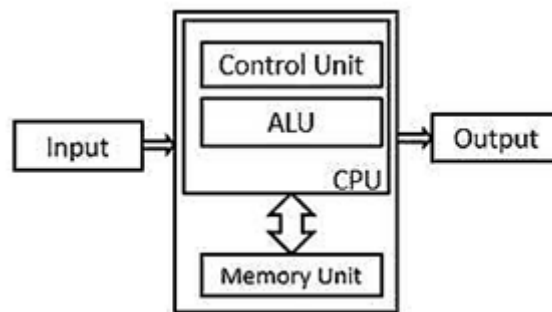


Figure 5.5: von Neumann architecture

Source: <http://www.polytechnichub.com/wp-content/uploads/2017/04/Von-Neumann-architecture.jpg>

Nature-inspired computing

Researchers are inspired by nature when developing new computing techniques, especially intelligent computing techniques. They observe natural phenomena and try to understand how such phenomena could be helpful to address complex natural problems. Thereby the researchers become able to develop new techniques that mimic such natural processes to solve complex problems that mankind faces nowadays. Such techniques are generally called nature-inspired computing techniques or natural computing techniques.

In nature-inspired computing, for example, the researchers observe how biological species, like ant colonies, beehives, and flocks of birds, react to stimuli, process information and make decisions. For example, certain flocks of birds collectively display altruistic behavior by leaving

³⁸<https://whatis.techtarget.com/definition/quantum-computing>

a healthy bird to remain with an injured bird on the ground³⁹. Such observations have given rise to groundbreaking computing techniques such as neural networks, swarm intelligence, evolutionary computation and artificial immune systems. This branch of computer science is closely related to biology-inspired computing and artificial intelligence⁴⁰.

Table 5.2 contains some additional nature-inspired computing techniques other than genetic algorithms and neural networks that have been introduced before.

Technique	Description
Swarm intelligence	An artificial intelligence technique based on the intelligent collective behaviors observable in the entities comprising of swarms of creatures such as ant colonies, beehives and bird-flocks. For example, bees are observed to have optimally selected (i.e. collective intelligence) the location for their hive in most of the time. Swarm intelligence enables to model complex and adaptive systems such as social systems as collections of agents who interact with each other based on some simple rules. The emergent patterns resulting from those individual agents' interactions are used to understand and control the particular system. Examples of swarm intelligence include developing optimal routing strategies, simulating crowd movements, building teams, etc.
Membrane computing	A branch of natural computing that abstracts computing models from the architecture and the functionality of living cells, as well as from the organization of cells in tissues, organs (brain included) or other higher-order structures such as colonies of cells (e.g., of bacteria) ⁴¹ .
Intelligent communication systems and protocols	Observing phenomena such as the patterns of spreading epidemics to develop new communication systems and protocols, which are faster and robust

Table 5.2: Nature inspired computing techniques

Apart from the above techniques, creating sensor networks by observing the sensory organs, creating artificial immune systems by observing the biological immune systems and making learning classifier systems based on human cognition are examples for nature or biology-inspired computing⁴².

³⁹<https://www.computersciencedegreehub.com/faq/what-is-nature-inspired-computing/>

⁴⁰https://en.wikipedia.org/wiki/Bio-inspired_computing#Areas_of_research

⁴¹http://www.scholarpedia.org/article/Membrane_Computing#Applications

⁴²https://en.wikipedia.org/wiki/Bio-inspired_computing#Areas_of_research

Review Questions

1. Discuss the intelligent features of the machine called KITT in the tele series 'Knight Rider'
2. Present your views on the limitations of intelligence of the chess playing machine called IBM Deep Blue
3. Explain the functionality of intelligent agents in online auction, e-learning and intrusion detection systems using the common characteristics of intelligent agents
4. Explain the reasons for the current high interest about machine learning and neural networks
5. Discuss how intelligent agent technology could be used when creating information systems for the following
 - a. Managing natural disasters such as flood and tsunami
 - b. E-commerce
 - c. Banking activities
6. Explain why interface agents are required when creating agent systems
7. Discuss how agent and multi agent technologies could have been used in the robot called Asimo
8. Explain why we need an architecture that has gone beyond the Von-Neumann architecture
9. Suppose an expert system has been created to assist a professional who provides education-counselling service to students.
 - a. How should the functionality of such a system be?
 - b. Is it possible to consider this system as an example for an agent system? Justify your answer
10. Explain with examples why it is important to study natural phenomena in order to create intelligent machines

References:

- Abazi, B. *Learn Magento CMS and E-commerce for beginners. CreateSpace Independent Publishing Platform* (March 30, 2017).
- Agarwal, A. *Emerging Technology Trends - Frequently Asked Questions. Independently published* (April 13, 2018).
- Anthony Lee, D. B. *Bootstrapping E-commerce. Reid & Wright Publishing, LLC; 2 edition* (June 1, 2018).
- Boehm, A., & Ruvalcaba, Z. *Murach's HTML5 and CSS3, 4th Edition 4th ed. Edition. Mike Murach & Associates; 4th ed. edition* (March 2, 2018).
- Ghosh, M. *Web Design Basic, A Beginner's Guide to HTML5 & CSS3. Independently published* (December 17, 2018).
- Greengard, S. *The Internet of Things. The MIT Press* (March 20, 2015).
- lexander Barkalov, L. T. *Foundations of Embedded Systems. Springer; 1st ed. 2019 edition* (February 5, 2019).
- McMahon, J. *E-commerce A Beginners Guide to e-commerce. CreateSpace Independent Publishing Platform* (January 27, 2017).
- Meyer, J. *ESSENTIAL GUIDE TO HTML5: USING GAMES TO LEARN HTML5 AND JAVASCRIPT. Apress; 2nd ed. edition* (November 6, 2018).
- Pfister, C. *Getting Started with the Internet of Things. Maker Media, Inc; 1 edition* (May 17, 2011).
- Rhynes, J. P. *HTML5 and CSS3, The Basics. Introduction for Beginners. CreateSpace Independent Publishing Platform* (April 16, 2018).
- <https://www.w3schools.com>.
- <https://www.w3schools.com/php/>.
- <https://www.w3schools.com>.
- <http://ecommercetoyouu.blogspot.com/2015/04/content-provider.html>.
- <http://smallbusiness.chron.com/portal-business-model-3869.html>.
- <http://vonbismark.com/wp-content/uploads/2012/08/tesco-trials-interactive-virtual-store-gatwick-0.jpg>.
- http://www.cs.toronto.edu/km/xib/document/broker_tutorial/definition.html.
- http://www.iaapa.org/docs/handout-archive---ops/Mon_KHAN_E-MARKETING.pdf.
- <http://www.netsuite.com/portal/resource/articles/erp/what-is-erp.shtml>.
- <http://www.polytechnichub.com/wp-content/uploads/2017/04/Von-Neumann-architecture.jpg>.
- http://www.scholarpedia.org/article/Membrane_Computing#Applications.
- <https://cyber.harvard.edu/olds/ecommerce/privacypolicy.html>.
- <https://hackernoon.com/top-5-most-popular-online-marketplaces-how-to-join-the-champions-league-a313dbdfd338>.
- <https://searchcio.techtarget.com/definition/B2E>.
- <https://searchcio.techtarget.com/definition/e-business>.

<https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence>.

<https://study.com/academy/lesson/virtual-communities-definition-types-examples.html>.

<https://thinkdigital.travel/wp-content/uploads/2013/04/10-AR-Best-Practices-in-Tourism.pdf>.

<https://whatis.techtarget.com/definition/quantum-computing>.

<https://www.arduino.cc/en/main/Software>.

<https://www.arduino.cc/en/Main/Software>.

<https://www.baslerweb.com/en/vision-campus/markets-and-applications/image-processing-industry-4> .

<https://www.businessnewsdaily.com/5001-what-is-c2b.html>. (n.d.).

<https://www.cbsl.gov.lk/en/news/public-awareness-on-virtual-currencies-in-sri-lanka>.

<https://www.computersciencedegreehub.com/faq/what-is-nature-inspired-computing/>.

<https://www.cs.cmu.edu/~softagents/multi.html>.

<https://www.ecommercetimes.com/story/61955.html>.

<https://www.erevenue.license.motortraffic.wv.gov.lk/erl/view/logout.action>.

<https://www.fi.edu/history-resources/automaton>.

<https://www.forbes.com/sites/forbestechcouncil/2017/07/06/man-machine-and-multiplicity-how-ai-and-humans-can-coexist-harmoniously/#524d497317df>.

<https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/>.

<https://www.hostmerchantservices.com/articles/payment-gateway-articles/how-payment-gateways-work/>.

<https://www.linkedin.com/pulse/worlds-12-b2b-websites-currently-holds-top-ranks-saqib-ilyas/>. (n.d.).

<https://www.makerspaces.com/arduino-uno-tutorial-beginners/>.

<https://www.plcademy.com/arduino-tutorial-for-beginners-chapter-1/>.

<https://www.slideshare.net/wiweck/accounting-information-system-18527651>. (n.d.). Retrieved from <https://www.slideshare.net>

<https://www.w3schools.com>.

<https://www.w3schools.com/php/>.

https://www.w3schools.com/python/python_exercises.asp.

https://www.w3schools.com/python/python_exercises.asp.

<https://www.w3schools.com/sql/>.

<https://www.youtube.com/watch?v=7aFqCqqaBZQ&t=230s>.

<https://www.youtube.com/watch?v=JIRPICfnmhw>.

<https://www.youtube.com/watch?v=vdqPHgGKgjU>.

MySQL database at <https://www.mysql.com/downloads/>.

Retrieved from <https://www.w3schools.com/php/>.

WAMP server at <https://sourceforge.net/projects/wampserver/> .

www.businessdictionary.com/definition/accounting.html.

www.ebay.com.

XAMPP server at <https://www.apachefriends.org/index.h>.

Glossary

English-Sinhala-Tamil Glossary			
No	English	Sinhala	Tamil
1.	abstract model	විදුක්ත ආකෘතිය	கருத்தியல் மாதிரி
2.	acceptance testing	ප්‍රතිග්‍රහණ පරීක්ෂාව	ஏற்புச் சோதனை
3.	access privilege	ප්‍රවේශවීමේ වරප්‍රසාදය	அணுகல் உரிமை
4.	agile model	සුවලස ආකෘතිය	சுறுசுறுப்பு மாதிரி
5.	alternate key	විකල්ප යතුර	மாற்றுச் சாவி
6.	American Standard Code for Information Interchange(ASCII)	තොරතුරු හුවමාරුව සඳහා වූ ඇමරිකානු සම්මත කේතය	தகவல் இடைமாற்றுக்கான அமெரிக்க நியம விதிக்கோவை
7.	amplitude	විස්තාරය	வீச்சம்
8.	amplitude modulation	විස්තාර මූර්ජනාව	வீச்சப் பண்பேற்றம்
9.	analog	ප්‍රතිසම	ஒப்புமை
10.	anchor	රැඳවුම	நிலை நிறுத்தி
11.	application layer	අනුප්‍රයෝග ස්ථරය	பிரயோக அடுக்கு
12.	architecture	නිර්මිතය	கட்டமைப்பு
13.	arithmetic and logical unit (ALU)	අංක ගණිත හා තාර්කික ඒකකය	எண்கணித மற்றும் தர்க்க அலகு
14.	array	අරාව	அணி
15.	artificial intelligence	කෘතීම බුද්ධිය	செயற்கை நுண்ணறிவு
16.	Affective computing	බුද්ධිමත් සහවිච්චවේගී පරිගණනය	நுண்ணறிவு உணர்திறன்மிக்க கணித்தல்
17.	associative law	සංසටන න්‍යාය	கூட்டு விதி
18.	attenuation	වැනැරීම/හායනය	நொய்மை
19.	attribute	උපලක්ෂණය /ගුණය/ උපලක්ෂණය	பண்புகள்
20.	authoring tool	සම්පාදන මෙවලම	படைப்பாக்கக் கருவி
21.	Automated Teller Machine (ATM)	ස්වයංකෘතමුදල් ගනුදෙනු යන්ත්‍රය	தானியங்கிப் பணம் கையாள் இயந்திரம்
22.	autonomous	ස්වයංපාලක/ ස්වතන්ත්‍ර/ස්වායත්ත	சுயாதீன
23.	axiom	ස්වසිද්ධිය/ප්‍රත්‍යක්ෂය	வெளிப்படை உண்மை
24.	backups	උපස්ථ	காப்பெடுத்தல்
25.	bandwidth	කලාප පළල/බඳස් පළල	பட்டை அகலம்
26.	batch processing	කාණ්ඩ සැකසුම	தொகுதி முறைவழியாக்கம்
27.	big data	මහාදත්ත	பெரிய தரவு
28.	binary	ද්වීමය	துவிதம், இருமம்
29.	binary coded decimal (BCD)	ද්වීමය කේතීක දශමය	இருமக் குறிமுறை தசமம்
30.	bio-inspired computing	ජෛව ප්‍රේරිත පරිගණනය/ ජෛව අනුප්‍රේරිත පරිගණනය	உயிரியல் உள்ளீர்ப்புக் கணிப்பு
31.	bit coin	බිටු කාසි	நுண்கடன் பணம் செலுத்தல்
32.	bitwise	බිටු අනුසාරිත	பிட் வாரி

33.	bitwise logical operation	பிட்டு அனுசாரணைக்கான மேலதரம்	பிட் வாரி தர்க்கச் செயற்பாடு
34.	black box testing	காணாமல்பட்ட பரிசீலனை	கறுப்புப்பெட்டிச் சோதிப்பு
35.	blogging	வெளியிடுதல்	வலைப்பதிவிடல்
36.	boot-up	ஆரம்பித்தல்	தொடங்குதல்
37.	broadcasting	பரப்பல்	தொலைபரப்பல்
38.	browsing	பார்வை	மேலோடல்
39.	bubble sort	புப்பெரிய பரிசீலனை	குமிழி வகைப்படுத்தல்
40.	built-in	உட்பொதிக்கப்பட்ட பகுதி	உட்பொதிந்த
41.	business process re-engineering (BPR)	புதுப்பிக்கப்பட்ட புதுப்பிக்கப்பட்ட	வணிக செயல்முறை மீள்கட்டமைப்பு
42.	candidate key	பரிசீலனை	பிரதிநிதித்துவச் சாவி
43.	cardinality	பரிசீலனை	எண்ணளவை
44.	cathode ray tube (CRT)	காணாமல்பட்ட பரிசீலனை	கதோட்டுக் கதிர் குழாய்
45.	central processing unit (CPU)	மையகம்	மத்திய செயற்பாட்டு அலகு
46.	characteristics	பண்புகள்	சிறப்பியல்புகள்
47.	check box	பரிசீலனை	சரிபார்ப்புப் பெட்டி
48.	client-server model	பரிசீலனை-பரிசீலனை பரிசீலனை	சேவைப் பயனர் மாதிரி
49.	clock	பரிசீலனை	கடிகாரம்
50.	cloud computing	பரிசீலனை	மேகக் கணிமை
51.	coaxial cable	பரிசீலனை	ஓரச்சு வடம்
52.	code editor	பரிசீலனை	குறிமுறை தொகுப்பி
53.	comment	பரிசீலனை	விளக்கக் குறிப்பு
54.	commutative law	பரிசீலனை	பரிமாற்று விதி
55.	compact disc	பரிசீலனை	ஒளியியல் வட்டு
56.	compatibility	பரிசீலனை	பொருந்துதல்
57.	compiler	பரிசீலனை	தொகுப்பான்
58.	component	பரிசீலனை	கூறு
59.	composite key	பரிசீலனை	கூட்டுச் சாவி
60.	constant	பரிசீலனை	மாறிலி
61.	content management system (CMS)	பரிசீலனை	உள்ளடக்க முகாமைத்துவ முறைமை
62.	context switching	பரிசீலனை	சந்தர்ப்ப நிலைமாற்றல்
63.	contiguous allocation	பரிசீலனை	அடுத்தடுத்தான ஒதுக்கீடு
64.	control structure	பரிசீலனை	கட்டுப்பாட்டுக் கட்டமைப்பு
65.	control unit (CU)	பரிசீலனை	கட்டுப்பாட்டலகு
66.	credit card	பரிசீலனை	கடன்ட்டை
67.	customization	பரிசீலனை	தனிப்பயனாக்கல்
68.	data	பரிசீலனை	தரவு
69.	data and control bus	பரிசீலனை	தரவும் கட்டுப்பாட்டுப் பாட்டையும்
70.	database management system (DBMS)	பரிசீலனை	தரவுத்தள முகாமைத்துவ முறைமை
71.	data definition language (DDL)	பரிசீலனை	தரவு வரையறை மொழி

72.	data dictionary	දත්ත ශබ්දකෝෂය	தரவு அகராதி
73.	data flow diagram	දත්ත ගැලීම් සටහන	தரவு பாய்ச்சல் வரைபடம்
74.	data flow model (DFM)	දත්ත ගැලීම් ආකෘතිය	தரவு பாய்ச்சல் மாதிரி
75.	data link layer	දත්ත සබැඳි ස්ථරය	தரவு இணைப்பு அடுக்கு
76.	data manipulating language (DML)	දත්ත හැසුරුම් බස	தரவு கையாளல் மொழி
77.	data migration	දත්ත පරිවහනය	தரவு பெயர்ச்சி
78.	debugging	හිදොස් කිරීම	வழு நீக்கல்
79.	decision support system (DSS)	කීරණ සහාය පද්ධති	தீர்மான உதவு முறைமை
80.	declarative	ප්‍රකාශනමය	அறிவிப்பு
81.	default values	පෙරනිම් අගය	இயல்புநிலை மதிப்பு
82.	defragmentation	ප්‍රතිබන්ධනය	துணிக்கை நீக்கல்
83.	demodulation	විමුර්ජනය	பண்பிறக்கம்
84.	device	උපාංගය / උපකරණය	சாதனம்
85.	device driver	උපාංග ධාවක මෘදුකාංග	சாதனச் செலுத்தி
86.	digital	අංකිත	இலக்க முறை
87.	digital camera	අංකිත කැමරාව	இலக்கமுறைப் படக்கருவி
88.	digital economy	අංකිත ආර්ථිකය	இலக்கமுறைப் பொருளாதாரம்
89.	digitizer	සංඛ්‍යාංකකය	இலக்கமாக்கி
90.	direct implementation	සෘජුස්ථාපනය	நேரடி அமுலாக்கம்
91.	disk formatting	තැටි/ඩිස්ක හැඩසවි ගැන්වීම	வட்டு வடிவமைப்பு
92.	distortion	විකෘතිය	திரிபு
93.	distributive law	විභාජනනිය	பங்கீட்டு விதி
94.	document flow diagram	ලේඛන ගැලීම් සටහන	ஆவணப் பாய்ச்சல் வரைபடம்
95.	domain	වසම	ஆள்களம்
96.	domain name server (DNS)	වසම් නාමසේවාදායකය	ஆள்களப் பெயர் சேவையகம்
97.	domain name system (DNS)	වසම් නාම පද්ධතිය	ஆள்களப் பெயர் முறைமை
98.	dynamic host configuration protocol (DHCP)	ගතික ධාරක පාලන නියමාවලිය	மாறும் விருந்தோம்பி உள்ளமைவு நெறிமுறை
99.	dynamic web page	ගතික වෙබ් පිටු	இயக்குநிலை வலைப்பக்கம்
100.	e-commerce	විද්‍යුත් වාණිජ්‍යය	மின் வர்த்தகம்
101.	economical feasibility	ආර්ථිකශක්‍යතාව	பொருளாதாரச் சாத்தியப்பாடு
102.	elementary process description(EPD)	මූලික ක්‍රියාවලි විස්තරය	அடிப்படைச் செய்முறை விபரிப்பு
103.	e-market place	ඉ-වෙළඳ පොළ	இலத்திரனியல் சந்தை இடம்
104.	encryption	ගුප්ත කේතනය	மறைகுறியாக்கம்
105.	enterprise resource planning system (ERPS)	ව්‍යවසාය සම්පත් සැලසුම් පද්ධතිය	நிறுவன மூலவள திட்டமிடல் முறைமை
106.	entity	භූතාර්ථය/අභිභූතත්වය/සත්තාව	நிலைபொருள்

107.	entity identifier	ஆதாரி/அதிசூதனீயம் கடிகீயம்	நிலைபொருள் அடையாளங்காட்டி
108.	entity relationship(ER) diagram	ஆதாரி கமீகனீயம் ரூபகமகை	நிலைபொருள் உறவுமுறை அட்டவணை
109.	executable	கூயாதீமக ககூ கககீ	இயக்கத்தகூ
110.	executive support system (ESS)	பீடியக கககை ககீயம்	நிறைவேற்று உதவு முறைமை
111.	expert system	பீகீககூ ககீயம்	நிபுணத்துவ முறைமை
112.	extended binary coded decimal interchange code (EBCDIC)	பீகீககை ககீயம் கீயம் ககூ	நீடித்த துவகித குறிமுறை தகம இடமார்த்தக் குறி
113.	extended entity relationship (ER) diagram	பீகீககைஆதாரி கமீகனீயம் ரூபகமகை	விரிவாக்கப்பட்ட நிலைபொருள் உறவுமுறை அட்டவணை
114.	feasibility study	கககை அடியககை	சாத்தியப்பாடு கற்ககை
115.	feedback loop	ககீககீககூ ககூ	பின்னூட்டல் வகையம்
116.	fetch-execute cycle	அகககூ-கூயககககூ ககூ	தருவிப்பு நிறைவேற்றுக் ககூ
117.	fiber optic	கககை ககீ	இழை ஒளியியல்
118.	file	ககூ	ககூ
119.	file hierarchy	ககூககூககூ	ககூ படிநிலை
120.	firewall	ககீ ககூ	தீக்கவர்
121.	normal form	ககூ ககூ அககூ	இயல்பாக்கல் வடிவம்
122.	fixed internal hard disk	அககூ அகககீயம் ககூ ககூ	நிலையான உள்ளக வந்தட்டு
123.	flash memory	ககூ/ கீககீ கககை	பளிச்சீட்டு நினைவகம்
124.	flash memory card	ககூ/ கீககீ கககை ககூ	பளிச்சீட்டு நினைவக அட்டை
125.	flat file system	பீக ககூ ககீயம்	சமதகக் ககூ முறைமை
126.	flip-flop	ககூ-ககூ	ககூ-ககூ
127.	float	ககூ/ககூ	ககூ
128.	floppy disk	ககூ ககூ	ககூ வட்டு
129.	flow chart	ககூ ககூ	பாய்ச்சுற் ககூ
130.	folder	ககூ ககூ	ககூமுறை
131.	foreign key	அகககூ ககூ	அந்நியச்சாவி
132.	formatting	ககூ ககூ	வடிவமைத்தல்
133.	frame	ககூ	சட்டகம்
134.	frequency modulation	ககூ ககூ	அதிர்வெண் பண்பேற்றல்
135.	full adder	ககூககூ	முழுமைக் ககூ
136.	function	ககூ / ககூ	சார்பு
137.	functional dependency	ககூககூ ககூ	செயல் சார்புநிலை
138.	functional requirement	ககூககூ அகககூ	செயல்படு தேவை
139.	quantum computing	கீககூ ககூ	ககூ ககூ அடிப்படை
140.	gateway	ககூ ககூ / ககூ ககூ	ககூ
141.	genetic algorithm	ககூ ககூ	ககூ ககூ

142.	geographical information system(GIS)	ஜியோகிராஃபிக் டேட்டா கால்டர் சட்டீடீகீய/தீகீகீய டேட்டா கால்டர் சட்டீடீகீய	புவியியல் தகவல் முறைமை
143.	graph plotter	புரீகீகீர் ட்ரூபுள்கர்ஸூய	படவரையி
144.	graphic tablet	பீகூகபீலகய	வரைவியல் விவரமாக்கி
145.	gridcomputing	சூலகபீரீகூகய	கோட்டுச்சட்டகக் கணிமை
146.	guided media	கீயலூ டூடீய	வழிபடுத்தப்பட்ட ஊடகம்
147.	half adder	ஈரீடாகலகய	அரை கூட்டி
148.	hand trace	கீகீகூகூரீகீய	கைச் சுவடுகள்
149.	hard disk	டூபீ கரீய / டூபீ டீகீகய	வந்தட்டு
150.	hardware	டூபீய	வன்பொருள்
151.	hexadecimal	ஈபீ டூகூய	பதினறுமம்
152.	hierarchical model	டூரூபீ ஈகூகீய	படிநிலை மாதிரி
153.	host	ஈகீகாரகய	விருந்தோம்பி
154.	hub	கூகீய	குவியன்
155.	human operator	தீகீகீகூகூகரூவீ	மனித இயக்குபவர்
156.	hybrid approach	டூலூகூகீ பூவீகய	கலப்பு அணுகல்
157.	hyperlink	ஈபீகூகீகீகீய	மீ இணைப்பு
158.	Integrated circuits (IC)	ஈகூகூகீ பரீபீ	ஒருங்கிணைந்த சுற்று
159.	icon	கீரூகய	சிறு படம்
160.	identity	ஈரீகூகூய	அடையாளம்
161.	image	ரூகய	படிமம்
162.	imperative	பீடாகூகீக	கட்டளை
163.	incremental	பீரீடாகூகீக	ஏறுமான, அதிகரிப்பு
164.	indexed allocation	ஈகூகூகீ பீகூகய	கூட்டி ஒதுக்கீடு
165.	information	கூகூகூ	தகவல்
166.	inkjet printer	கீகீய பீகூகீ டூகய	மைத்-தாரை அச்சுப்பொறி
167.	instant messaging	கீகீகீ பகீபூபீ கரீகீ	உடனடிச் செய்தியிடல்
168.	integrated development environment (IDE)	ஈகூகூகீ கூபீரீகீ பரீகரய	ஒருங்கிணைந்த விருத்தி சூழல்
169.	integration test	ஈகூகூகீ பரீகீகூய	ஒருங்கிணைந்த சோதிப்பு
170.	intelligent and emotional computing	டூகீகீகீ கீய பீகீகீகீ பரீகூகய	நுண்ணறிவும் உணர்திறனுமிக்க கணித்தல்
171.	interface	ஈகூகூ டூகூகூ	இடைமுகம்
172.	internet service provider(ISP)	ஈகீகீசூல கீகீய ஈபயகீகூ	இணையச் சேவை வழங்குனர்
173.	interpreter	ஈரீபீகூகூகய	மொழிமாற்றி
174.	interrupt	ஈகூகூபீகூ	இடையூறு
175.	intranet	ஈகீகீ:சூலய/ ஈகீகீசூல	அகவிணையம்
176.	internet of things (IoT)	ஈபீபூபீயஈகீகீசூலய/ ஈபீபூபீய ஈகீகீசூலய/ ஈகீகீசூல ஈய	பொருட்களின் இணையம்
177.	iteration	பூகீகீகூகய	மீள் செயல்
178.	karnaugh map	கூகூ கீகீகீக	கானோ வரைபடம்
179.	knowledge management system(KMS)	டூகூகீ கலூகூகூகூ சட்டீடீகீய	அறிவு முகாமைத்துவ முறைமை
180.	large scale integration	பீகூல பரீகூகூகீ ஈகூகூகய	பாரிய அளவு ஒருங்கிணைப்பு

	(LSI)		
181.	latency	சமூல/ஒத்தவை	மறைநிலை
182.	least significant	அடிமலவேசைசி	சிறும மதிப்பு
183.	legend	விசைநர் சாடெ	குறி விளக்கம்
184.	life cycle of data	உதீந சீவந வநுத	தரவு வாழ்க்கை வட்டம்
185.	light emitting diode (LED) display	஁ரலேக விலேவகஉதேவீ சக்டேரககத	ஒளிகாலும் இருவாயித் திரை / ஒளி உமிழும இரு முனையம்
186.	linked allocation	சகடே விசாசதத	இணைப்பு ஒதுக்கீடு
187.	linker	சக்டீரகத	இணைப்பி
188.	liquid crystal display (LCD)	உவிசீவிவக சக்டேரககத	திரவப்பளிங்குக் கணினித் திரை
189.	list	லுசீசீவ	பட்டியல்
190.	liveware	சீலா஁	உயிர் பொருள்
191.	local publishing	சீலாநீத சுகிஉடகீரீம	உள்ளக வெளியீடு
192.	local area network (LAN)	சீலாநீத சுகே஁ சாலத	இடத்தூரி வலையமைப்பு
193.	logic gate	லார்கீக உீலாரத	தர்க்கப் படலை
194.	Logical Data Modeling (LDM)	லார்கீக உதீந ஁கககீகரதத	தர்க்கத் தரவு மாதிரியுருவாக்கல்
195.	logical data structure	லார்கீக உதீந வசுத	தர்க்கத் தரவுக் கட்டமைப்பு
196.	logical design tools	லார்கீக சலுசுதீ ஡ேவலுதீ	தர்க்க வடிவமைப்புக் கருவி
197.	looping	லூசதத	வளைய வரல்
198.	machine code	ததீநு கீதத	இயந்திரக் குறியீடு
199.	machine-machine coexistence	ததீநு-ததீநு சதசவலுதீ	இயந்திர- இயந்திர ஒருங்கிருத்தல்
200.	magnetic ink character reader (MICR)	வூதீ஁கீந கீதீந ஁நுலகூநு கீதவதத	காந்த மை ஁முத்துரு வாசிப்பான்
201.	magnetic stripe reader	வூதீ஁க கீரூ கீதவதத	காந்தப்பட்டி வாசிப்பான்
202.	magnetic tape	வூதீ஁க சவித	காந்த நாடா
203.	malware	஁நீதீவி ஡ாஉகா஁	தீம்பொருள்
204.	management information system (MIS)	கலுதீகாகரதத தாரதூரூ சஉடீ஁தீத	முகாமைத்துவ தகவல் முறைமை
205.	man-machine coexistence	தீதீசீ-ததீநு சதசவலுதீ	மனிதன் - இயந்திரம் ஒருங்கிருத்தல்
206.	media access control (MAC)	஡ா஁த சுவே஁ சாலக	஁஁டக அணுகல் கட்டுப்பாடு
207.	memory management unit (MMU)	஡தக கலுதீகாகரதத சீககத	நினைவக முகாமைத்துவ அலகு
208.	meshtopology	஁டேசீவிலகத	கண்ணி இடத்தியல்
209.	microprocessor	கீசூஉ சகசதத	நுண்செயலி
210.	microwave	கீசூஉ தர஁	நுண்ணலை
211.	mini disk	கூ஁ா தரவித	சிறு வட்டு
212.	mobile computing	ச஁஁஡ சரி஁ததத	செல்லிடக் கணிமை
213.	mobile marketing	ச஁஁஡ ஁லேவிகரதத	செல்லிடச் சந்தைப்படுத்தல்
214.	modularization	஡ா஁தீசூலகரதத	கூறு நிலையாக்கம்
215.	modulation	஡ூர்சதத	பண்பேற்றம்
216.	most significant	விசீ஡லவேசைசி	அதியுயர் மதிப்பு
217.	mother board	஡வூ சூவிர்வி	தாய்ப்பலகை

218.	multi agent systems	பெரும்பான்மை அமைப்புகள்	பல்முக்கவர் முறைமை
219.	multi user-multi task	பெரும்பான்மை -பெரும்பான்மை	பற்பயனர்-பற்பணி
220.	multi-core processors	பெரும்பான்மை செயலிகள்	பல்கரு செயலி
221.	multimedia objects	பெரும்பான்மை பொருள்	பல்லாடக பொருள்
222.	multiplexer	பெரும்பான்மை	பல்சேர்ப்பி
223.	multiplexing	பெரும்பான்மை	பல்சேர்ப்பு
224.	multiprocessing	பெரும்பான்மை	பன்முறைவழியாக்கி
225.	multitasking	பெரும்பான்மை	பற்பணி
226.	multi-threading	பெரும்பான்மை	பல் செயல்கூறு
227.	nature inspired computing	பெரும்பான்மை	இயற்கை உள்ளீர்ப்புக் கணிப்பு
228.	nested loop	பெரும்பான்மை	நீடித்த வளையம்
229.	network addresses translating (NAT)	பெரும்பான்மை	வலையமைப்பு முகவரி பெயர்ப்பு
230.	network architecture	பெரும்பான்மை	வலையமைப்புக் கட்டமைப்பு
231.	network layer	பெரும்பான்மை	வலையமைப்பு அடுக்கு
232.	network model	பெரும்பான்மை	வலையமைப்பு மாதிரி
233.	neural network	பெரும்பான்மை	நரம்பியல் வலையமைப்பு
234.	non-functional requirement	பெரும்பான்மை	செயல்சாராத தேவைகள்
235.	normalization	பெரும்பான்மை	இயல்பாக்கல்
236.	null	பெரும்பான்மை	வெற்று
237.	objectcode	பெரும்பான்மை	பொருள் குறி
238.	object oriented	பெரும்பான்மை	பொருள் நோக்குடைய
239.	object- relational model	பெரும்பான்மை	பொருள் உறவுநிலை மாதிரி
240.	octal	பெரும்பான்மை	எண்மம்
241.	office automation system (OAS)	பெரும்பான்மை	அலுவலகத் தன்னியக்க முறைமை
242.	offline	பெரும்பான்மை	தொடரறு நிலை
243.	one's compliment	பெரும்பான்மை	ஒன்றின் நிரப்பி
244.	online	பெரும்பான்மை	தொடரறா நிலை
245.	open source	பெரும்பான்மை	திறந்த மூலம்
246.	operational feasibility	பெரும்பான்மை	செயற்பாட்டுச் சாத்தியப்பாடு
247.	operator category	பெரும்பான்மை	செயலி வகை
248.	operator precedence	பெரும்பான்மை	செயலி முன்னுரிமை
249.	optical character reader (OCR)	பெரும்பான்மை	ஒளியியல் எழுத்துரு வாசிப்பான்
250.	optical mark reader (OMR)	பெரும்பான்மை	காந்த மை எழுத்துரு வாசிப்பான்
251.	output	பெரும்பான்மை	வெளியீடு
252.	packet switching	பெரும்பான்மை	பொதி மடைமாற்றல்
253.	paging	பெரும்பான்மை	பக்கமிடல்
254.	paradigm	பெரும்பான்மை	கோட்பாட்டுச் சட்டகம்

		புதிலாசை/புதிர்சை	
255.	parallelimplementation	ஈலாநீநர்ஈலாசை	சலாந்நர அலுலாக்கல்
256.	parameter passing	சலாநீநி ஈலீல	பரலாநாக் கடத்தல்
257.	parity	ஈலலால	சலநிலை
258.	password	லுரசடட	கடவுச்சலால்
259.	payment gateway	லலுலீ லாஈலீ டீலார்ச	பணக் கலாடுப்பலவு நுலுலாலயில்
260.	periodic refreshing	ஈலலீந சலலீலகர்லாசை	காலலுலுலு புதுப்பித்தல்
261.	peripheraldevice	சலீலநீந ஈலாலல / ஈசஈலல	புறச் சாலநலல்
262.	phablet	லலலீலல	பெப்பலட்
263.	phased implementation	ஈலலீஈலாசை / சிலலர் ஈலாலலகலீல	கட்ட அலுலாக்கல்
264.	phase modulation	கலா லுலீசை	நிலை பண்பேற்றல்
265.	phishing	நலுலல	வழிப்பறித்தல்
266.	physical layer	லலலீலகலீல	பெலாதீக அலுக்கு
267.	physical memory	லலலீல லநகல	பெலாதீக நிலைலகல்
268.	pilot implementation	நிலாலக ஈலாசை / நிலாலக ஈலாலலகலீல	லுன்னலலடி அலுலாக்கல்
269.	piracy	லலலர்லல/ லுலீலல	கலவு
270.	pirated software	லலலர்/லுலீலீல லலலலல	நிலுட்டு ஡ென்பலலுள்
271.	plagiarism	லுலீலர்லல லலலீல	கலுத்துத் திலுட்டு
272.	point to point connection	ஈலு லலலல ஈலீலலலலல	லுன்றுடலுன்னு லுலைப்பு
273.	pointing device	டலலீலுலீ ஈலாலல	சுட்டி சாலநலல்
274.	port	கலலலல	லாலயில், துலு
275.	portable external hard disk	சலலல/ஈலலலல லலல லலல லலல லலல	காலலத்தகு புற லலலலட்டு
276.	portal	டீலார்ச/ ஈலுலலடீலார்ச	லலைலாலசல்
277.	Point of sale (POS) machine	லீலுலுலீலலல ஈலீல	லிற்பலை லு லுலந்நில
278.	postulate	ஈசகலீலல	லுலுலு
279.	power supply	லீலு ஈசலலல/சல ஈசலலல	லில் லுலுங்கி
280.	presence check	நலலல சலலீலல	லுலுத்தல் சலலபார்த்தல்
281.	presentation layer	ஈலலீலல/லுலீலலலலலலல	லுலலலலல அலுக்கு
282.	primary key	லுலீலல/லுலீ ஈலு	லுதலலல சாலல
283.	primitive data type	லுலீலல டலல லலல	புலலீகத் தலல லகை
284.	privacy	லலலீலலலலல	அந்நலங்கல்
285.	private key	லலலீலலல ஈலு	பிலரத்திலலகச் சாலல
286.	process	ஈலாலலல/ஈலாலலல/ ஈலலல	சலலல. ஈ. ஡ுலுலலலலலல
287.	process control block(PCB)	ஈலாலலலலலலலலல	சலலல கட்டுப்பாட்டுத் தலகுதல
288.	process management	ஈலாலல ஈலலலலலலல	சலலல ஡ுலலலலலலல
289.	process states	ஈலாலல லலலல	சலலல நிலை
290.	process transition	ஈலாலலலலலலல	சலலல நிலைலலல
291.	product commercialization	லீலலலல லலலலலலலல	தலலலலல லலலலலலலலலலலல
292.	product of sum (POS)	லலலலலலல லலலல	கூட்டுத்தலலலலலலலலலல

293.	program translator	கருவிகளில் பரிவர்த்தனை	செய்நிரல் மொழிபெயர்ப்பான்
294.	proprietary	கீழ்க்கண்ட சக்தி	தனியுரிமை
295.	protocol	கிடைசுவர்ப்பு	நடப்பொழுங்கு
296.	prototyping	மூலக்கணிதக் கருவிகள்	மூலவகை மாதிரி
297.	proxy server	கிடைசுவர்ப்பு கருவிகள்	பதிலாளர் சேவையகம்
298.	pseudo code	விதிகளில் கருவிகள்	போலிக்குறி
299.	public switch telephone network (PSTN)	பொது கம்பி வளங்களைப் பிணைக்கான சேவை	பொது ஆளியிடப்பட்ட தொலைபேசி வலையமைப்பு
300.	public key	பொது கருவிகள்	பொதுச் சாவி
301.	pulse code modulation	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	துடிப்புக்குறி பண்பேற்றம்
302.	pulse width modulation	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	துடிப்பு அகலப் பண்பேற்றம்
303.	radio button	பிணைக்கான கருவிகள்	ரேடியோ பொத்தான்
304.	random access memory (RAM)	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	தற்போக்கு அணுகல் நினைவகம்
305.	range check	பொது கருவிகள்	வீச்சு சரிபார்த்தல்
306.	rapid application development (RAD)	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	துரித பிரயோக விருத்தி
307.	read only memory (ROM)	பொது கருவிகள்	வாசிப்பு மட்டும் நினைவகம்
308.	real time	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	நிகழ்நேரம்
309.	record	பொது கருவிகள்	பதிவு
310.	redo	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	மீளச் செய்
311.	redundancy	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	மிகைமை
312.	reference model	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	வலையமைப்பின் கட்டமைப்பு
313.	refreshing	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	புத்துயிர்ப்பித்தல்
314.	register memory	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	பதிவகம்
315.	relational	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	தொடர்பு, உறவுநிலை
316.	relational model	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	உறவுநிலை மாதிரி
317.	relational database	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	உறவுநிலை தரவுத்தளம்
318.	relational instance	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	தொடர்பு முறை எடுத்துக்காட்டு
319.	relational schema	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	தொடர்பு முறைத் திட்டம்
320.	relationship	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	தொடர்புமுறை
321.	remote	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	தொலை, தூர
322.	render	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	வழங்கு
323.	repeater	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	மீளி, மீட்டி
324.	repetition	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	மீள் செயல்
325.	reset button	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	மீளமைப்புப் பொத்தான்
326.	retrieve	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	மீள்பெறு
327.	return value	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	திரும்பல் பெறுமானம்
328.	reverse auction	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	எதிர்மாற்று ஏலம்
329.	ringtopology	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	வளைய இடத்தியல்
330.	router	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	வழிப்படுத்தி, வழிச்செலுத்தி
331.	routing	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	வழிச்செலுத்தல்
332.	scanner	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	நுணுகு நோக்கி
333.	scheduler	கம்பி வளங்களைப் பிணைக்கான மூலக்கணிதக் கருவிகள்	ஒழுங்குபடுத்தி

334.	scope of variable	வீவிலு பருகய	மாறி செயற்பரப்பு
335.	query	வீலுசுல	வினவல்
336.	selection	வீலு	தெரிவு
337.	selector	லுருகய	தேர்வி, தேர்ந்தெடுப்பி
338.	sensor	கலுவீடுகய	உணரி
339.	sequence	அனுனுலுய	தொடர்
340.	sequential circuit	அனுனுலுலு பர்புபுய	தொடர்சு சுற்று
341.	sequential search	அனுனுலுலு கெலுலு	வரிசைமுறைத் தேடல்
342.	server	கீலுலுலுயகய / அனுனுலுலுயகய	சேவையகம்
343.	session layer	கலுலு கீலுருய	அமர்வு அடுக்கு
344.	sharable pool	னுலுலுலுலு டுபுசுய	பகிரதகு பொது இடம்
345.	sign-magnitude	லுலுலுலுலுலு டுலுலுலுலு / கலுலுலுலுலு பர்புலுலுலுலு / அலுலுலு பர்புலுலுலுலு	குறியுடைய வீசுசுளவு
346.	single user-multi task	லுலு பர்புலுலுலு-லுலு கார்புய	தனிப்பயனர்-பற்புணி
347.	single user-single task	லுலு பர்புலுலுலு-லுலு கார்புய	தனிப்பயனர்-தனிப்புணி
348.	smart card	சுலுலுலு கார்புலுலு	சூட்டிகை அட்டை
349.	smart phone	சுலுலுலு டுலுலுலுலு	சூட்டிகைத் தொலைபேசி
350.	smart system	சுலுலுலு பட்டலுலுலு	சூட்டிகை முறைமை
351.	social networking	கலுலு டுலுலுலுலு	சமுலு வலையமைப்பாக்கல்
352.	software	லுலுலுலு	லுலுலுலுலு
353.	software agent	லுலுலுலுலு காரலு	லுலுலுலுலு முகவர்
354.	sort	வீலு	வரிசைப்படுத்து
355.	source	டுலுலு	முலம்
356.	spiral model	கலுலுலு அலுலுலுலு	சுருளி மாதிரி
357.	spooling	லுலுலு	சுற்றுதல்
358.	Startopology	லுலுலு கலுலுலுலு	வினலுலு இடத்தியல்
359.	stepwise refinement	லுலுலுலுலுலு டுலுலுலுலு	படிமுறை நீக்கல்
360.	storage	அலுலுலுலு	சேமிப்பு
361.	storage allocation	அலுலுலுலு லுலுலுலு	சேமிப்பு லுலுலுலு
362.	stored program concept	அலுலுலு லுலுலுலுலுலுலுலு	சேமிக்கப்பட்ட செய்நிரல் ஂண்க்கரு
363.	structure	லுலுலுலு	கட்டமைப்பு
364.	structure chart	லுலுலு கலுலுலு	கட்டமைப்பு வரைபு
365.	structured	லுலுலுலுலு	கட்டமைப்புடைய
366.	structured query language (SQL)	லுலுலுலுலுலுலுலுலு	கட்டமைப்பு வினவல் மொழி
367.	submit button	லுலுலு லுலுலுலு	சமர்ப்பித்தல் பொத்தான்
368.	subnet mask	லுலுலுலு அலுலுலுலு	உபவலை மறைமுலம்
369.	sub-netting	லுலுலு-லுலுலுலு	உபவலையமைப்பு
370.	sub-program	லுலுலு-லுலுலுலுலு	துணைசு செய்நிரல்
371.	sum of products (SOP)	லுலுலுலுலுலு லுலுலுலு	பெருக்கலுலுலுலு கலுலுலுலுலு
372.	supply chain management	கலுலுலுலு டுலுலு கலுலுலுலுலுலு	விநியுலுலு சலுலுலுலுலுலுலு முகலுலுலுலுலுலு
373.	swapping	டுலுலுலுலுலு	இடமாற்றல்
374.	switch	கலுலுலு	ஆளி

375.	syntax	காரக ரீதி	தொடரியல்
376.	system development life cycle (SDLC)	புதிதானது கட்டுப்பாட்டு சுழற்சி	முறைமை விருத்தி வாழ்க்கை வட்டம்
377.	table	வகுப்பு	அட்டவணை
378.	table check constraint	வகுப்பு சரிசெய்தல் கட்டுப்பாடு	அட்டவணை சரிபார்த்தல் கட்டுப்பாடு
379.	tag	குறிப்பு	ஓட்டு
380.	Technical feasibility	தொழில்நுட்ப வாய்ப்பு	தொழில்நுட்பச் சாத்தியக் கற்கை
381.	telecommuting	தூரத்தில் கட்டுப்பாடு / தூர கட்டுப்பாடு	தொலைச்செயல்
382.	testing strategy	சரிசெய்தல் திட்டமிடல்	பரிசீலிப்புத் திட்டம்
383.	text and font	உரை மற்றும் எழுத்துரு	வாசகமும் எழுத்துருவும்
384.	text formatting	உரை வடிவமைப்பு	வாசக வடிவமைப்பு
385.	text input	உரை உள்ளீடு	வாசக உள்ளீடு
386.	normal form	சுருக்கமான வடிவம்	இயல்பான வடிவம்
387.	thumbnail	சுருக்கமான படிமம்	குறுப்படம்
388.	time division modulation (TDM)	கால வகுப்பு மீட்டிங்	நேரப் பிரிப்பு பண்பாக்கம்
389.	time sharing	கால பகிர்வு	நேரப்பகிர்வு
390.	timing	கால அளவு	நேரக்கணிப்பு
391.	top down design	மேலிருந்து கீழான வடிவமைப்பு	மேலிருந்து கீழான வடிவமைப்பு
392.	touch pad	தொடுப்புத் தட்டை / தொடுப்புத் தட்டை	தொடு அட்டை
393.	touch screen	தொடுப்புத் தட்டை	தொடுதிரை
394.	transaction processing system(TPS)	தொடர்பு கட்டுப்பாட்டு அமைப்பு	பரிமாற்றச் செயலாக்க முறைமை
395.	transitive dependency	கட்டுப்பாட்டுப் பரிமாற்றம்	மாறும் சார்பு நிலை
396.	transport layer	தொடர்பு அடுக்கு	போக்குவரத்து அடுக்கு
397.	transport protocol	தொடர்பு கட்டுப்பாட்டு அமைப்பு	போக்குவரத்து நடப்பொழுங்கு
398.	tuple	குறியீடு/சேர்வை	பதிவு: நிரை
399.	twisted pair	திரிசுருட்டி	முறுக்கிய சோடி
400.	two's compliment	இரட்டை அகலம்	இரட்டை நிரப்பி
401.	type check	வகை சரிசெய்தல்	வகை சரிபார்த்தல்
402.	constraint	கட்டுப்பாட்டு வகை	கட்டுப்பாடு வகை
403.	ubiquitous computing	எங்குமே கிடைக்கக்கூடிய கணினி	எங்கும் வியாபித்த கணினிமை
404.	undo	திரும்பி எடுக்க	செயல்தவிர்
405.	unguided media	நிர்ணயமின்றி மூலம்	வழிபடுத்தப்படாத ஊடகம்
406.	uni-casting	ஒருவருக்கு	தனிப்பரப்பல்
407.	unicode	ஒருவருக்கு/ ஒருவருக்கு	ஒற்றைக்குறி முறை
408.	unique constraint	ஒருவருக்கு கட்டுப்பாடு	தனித்துவக் கட்டுப்பாடு
409.	unit testing	ஒருவருக்கு சோதனை	அலகுச் சோதனை
410.	universal	எல்லாவுக்கும்	பொது
411.	updating	தொடர்பு கட்டுப்பாட்டு அமைப்பு	தற்காலப்படுத்தல்
412.	user	பயனர்	பயனர்
413.	user defined	பயனர் வரையறை	பயனர் வரையறை
414.	validation	சரிசெய்தல்	செல்லுபடியாக்கல்

415.	variable	விவரம்	மாறி
416.	very large scale integration (VLSI)	ஓர் மிகப் பெரிய அளவிலான ஒருங்கிணைப்பு	மிகப் பெரிய அளவிலான ஒருங்கிணைப்பு
417.	video graphic adapter (VGA)	வீடியோ கிராபிக் அடாப்டர்	காணொளி வரையி பொருத்தி
418.	virtual community	அதன் சூழல்	மெய்நிகர் சமூகம்
419.	virtual memory	அதன் நினைவு	மெய்நிகர் நினைவுகம்
420.	virtual storefront	அதன் வெளியேற்ற முகப்பு	மெய்நிகர் கடைமுகப்பு
421.	waterfallmodel	ஊதாநீர் அடிப்படையில்	நீர் வீழ்ச்சி மாதிரி
422.	wave length	அலை அளவு	அலை நீளம்
423.	web portal	வெப்பேஜ்	வலை வாசல்
424.	web server	வெப்சைவர்கள்	இணைய சேவையகம்
425.	web service provider	வெப்சைவ் வழங்கிகள்	இணைய சேவை வழங்குனர்
426.	white box testing	கம்ப்யூட்டர் சோதனை	வெண்பெட்டிச் சோதிப்பு
427.	world wide web (WWW)	உலகளாவிய வலை	உலகளாவிய வலை
428.	uniform resource locator (URL)	ஒரே மாதிரி அடையாளம்	சீர்மை வள இருப்பிடங்காட்டி
429.	uniform resource identifier(URI)	ஒரே மாதிரி அடையாளம்	சீர்மை வள அடையாளங்காட்டி