COMPUTING FUNDAMENTALS USING PYTHON

Ms. Archana A
Assistant Professor

Department of Computer Applications

COMPUTING FUNDAMENTALS USING PYTHON

Introduction to Problem Solving

Archana A

Department of Computer Applications

Computing Fundamentals

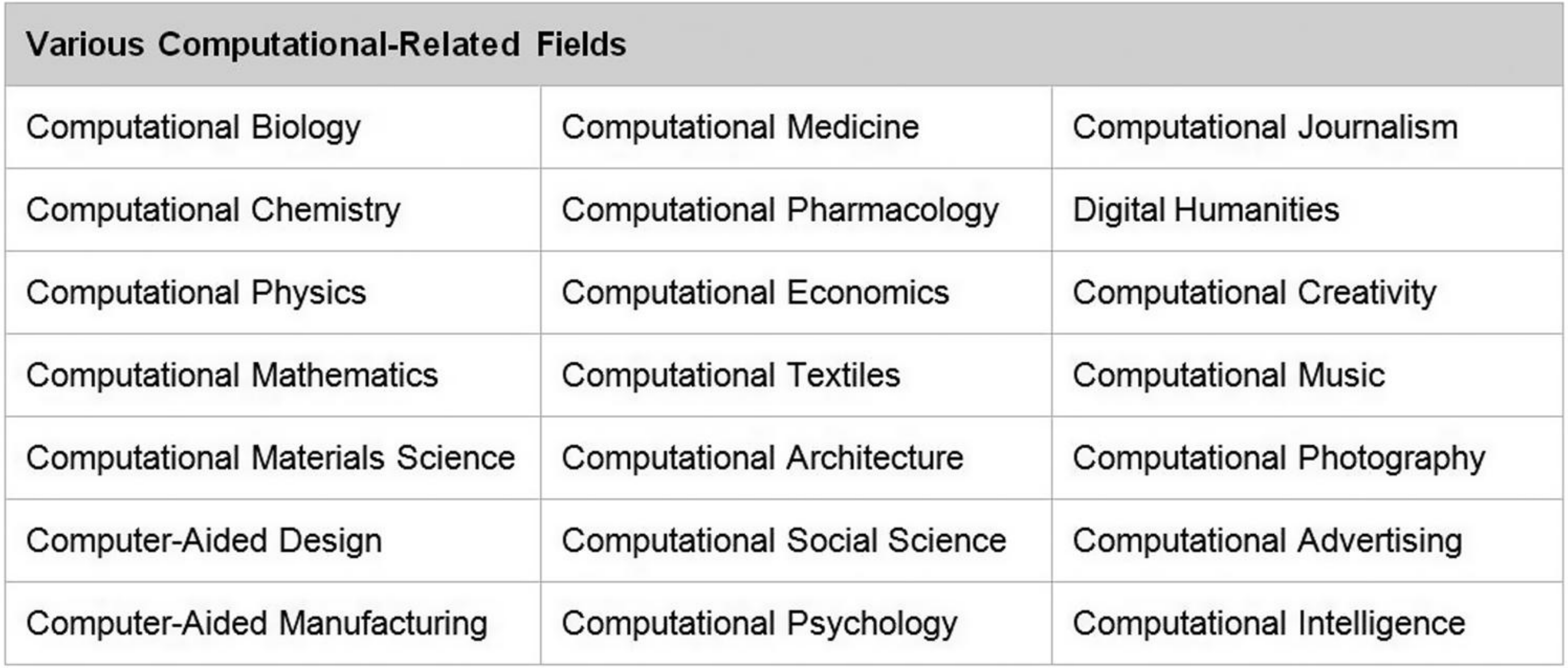
- Computing technology has changed, and is continuing to change the world. Essentially every aspect of life has been impacted by computing

- For example, Vast information resources, such as **Wikipedia**, now provide **easy, quick access to a breadth of knowledge** as never before.

- **Information sharing** via **Facebook** and **Twitter** has not only brought family and friends together in new ways.

Computing Fundamentals

- In the study of computer science, there is always changing technology. That is what makes the field of computer science so exciting.



Various Computational-Related Fields		
Computational Biology	Computational Medicine	Computational Journalism
Computational Chemistry	Computational Pharmacology	Digital Humanities
Computational Physics	Computational Economics	Computational Creativity
Computational Mathematics	Computational Textiles	Computational Music
Computational Materials Science	Computational Architecture	Computational Photography
Computer-Aided Design	Computational Social Science	Computational Advertising
Computer-Aided Manufacturing	Computational Psychology	Computational Intelligence

What is Computer Science ?

- It is about programming computers.
- Although programming is certainly a primary activity of computer science, programming languages and computers are only tools.
- Computer science is fundamentally about computational problem solving —that is, **solving problems by the use of computation.**

Computational Problem Solving

In order to solve the problem **two things that are needed** :

1. A **representation** that captures all the relevant aspects of the problem

2. **An algorithm** that solves the problem by use of the representation

Thus, computational problem solving finds a solution within a representation that translates into a solution for what is being represented

.

What is computational thinking?

- Computational thinking allows us to **take a complex problem**, **understand what the problem is** and **develop possible solutions**.

- We can then present these solutions in a way that a computer, a human, or both, can understand.

.

Computational Problem Solving

- Computational Thinking is the prerequisite skill for understanding the technologies of the future.

- It is a ***thought process***, rather than a specific body of knowledge about a device or language.

- Computational thinking is often associated with **computers** and **coding**

.

Core Components of Computational Thinking

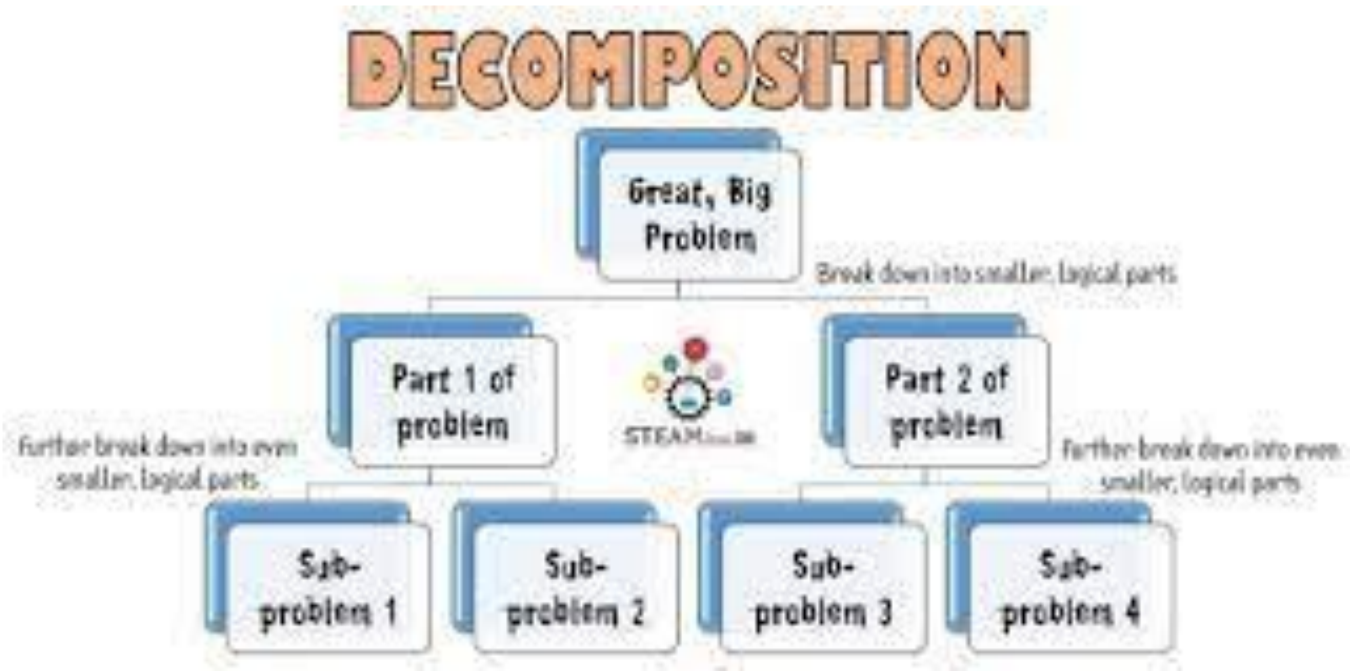
- There are **four** cornerstones of computational thinking. They are
  - **Decomposition,**
  - **Pattern recognition,**
  - **Abstraction, and**
  - **Algorithms.**

Core Components of Computational Thinking

- **Decomposition** : it is a process of breaking down complex problems into smaller, simpler problems.
- **Pattern recognition:** In this process we look for similar and common features, that is to make connections between similar problems and experience.
- **Abstraction** : focusing on the data that is relevant to main problem while ignoring unrelated or irrelevant details.
- **Algorithm:** this is the procedure of creating step by step solution to the given problem.

Decomposition:

- Decomposition is the first stage of computational thinking.
- Decomposition is to break down a complex problem or system into smaller, more manageable parts.


DECOMPOSITION
Great, Big Problem
Break down into smaller, logical parts
Part 1 of problem
Part 2 of problem
STEAM
Further break down into even smaller, logical parts
Further break down into even smaller, logical parts
Sub-problem 1
Sub-problem 2
Sub-problem 3
Sub-problem 4

Pattern Recognition:

- Moving along the problem-solving path, after you have decomposed the problem, it's time to analyze the data

- There are patterns in almost everything if you look closely. There are patterns in math, science, art, nature, literature. music, etc.

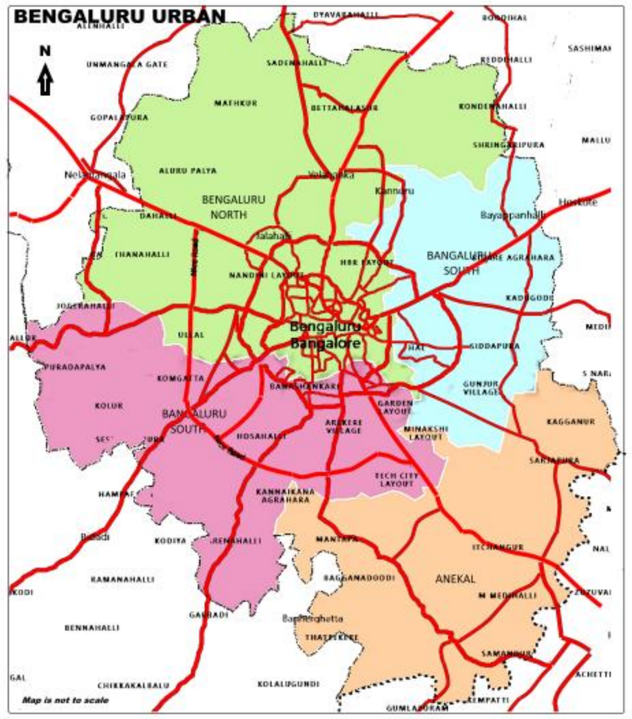
- If there is a problem, finding the pattern is often the most efficient way to solve it.

Abstraction:

- **A representation that leaves out detail of what is being represented is a form of abstraction.**
  - how you can hide the details that are not necessary when solving a problem.

- Abstraction may be the most complicated stage of computational thinking.

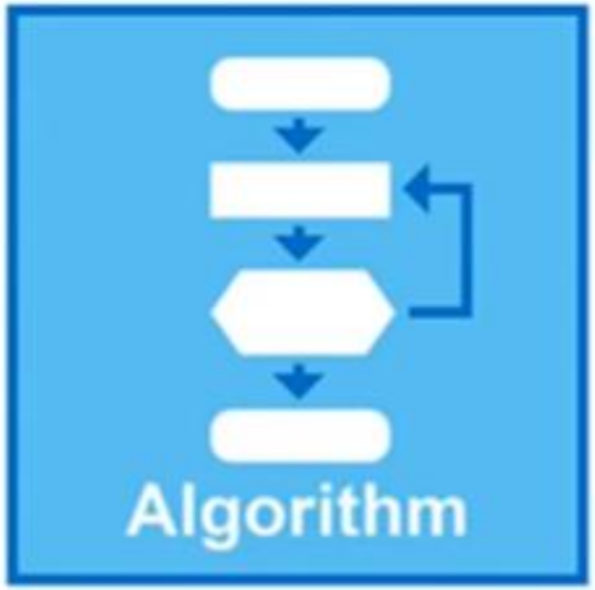
- A programmer hides all but the relevant data about an object in order to reduce complexity and increase efficiency



Algorithm:

- Algorithms are the final stage of computational thinking. Algorithms are **a way to develop a step-by-step solution to the problem**, or the rules to follow to solve the problem.

- **Identify specific similarities and differences among similar problems to work towards a solution**

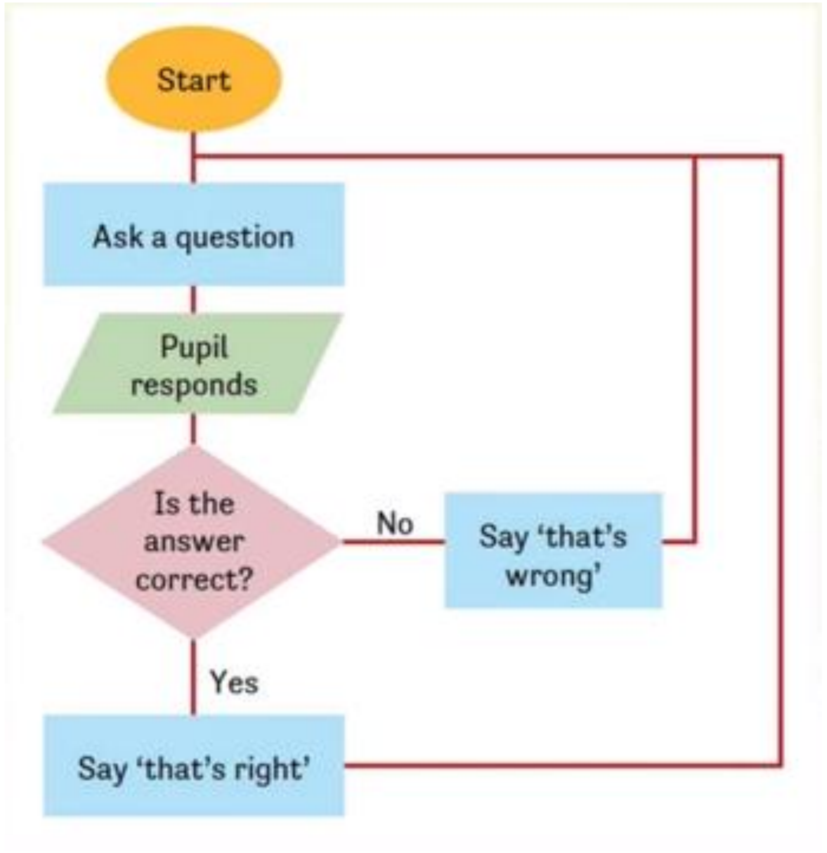


An Algorithm Development Process

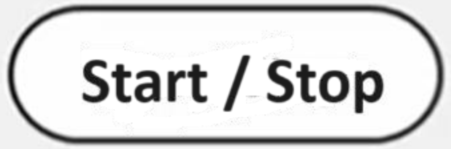
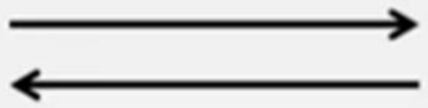
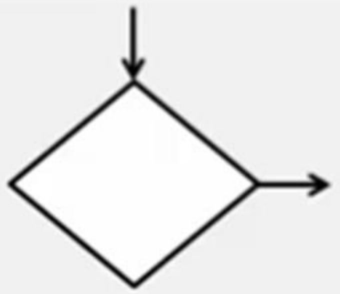
Here are the steps to write an algorithm

- Step 1: Obtain a description of the problem.
- Step 2: Analyze the problem.
- Step 3: Develop a high-level algorithm. ...
- Step 4: Refine the algorithm by adding more detail. ...
- Step 5: Review the algorithm

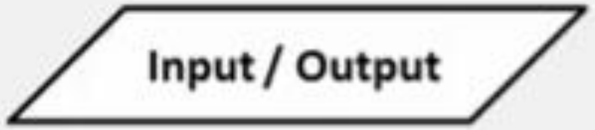
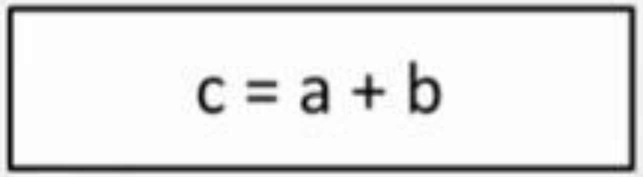
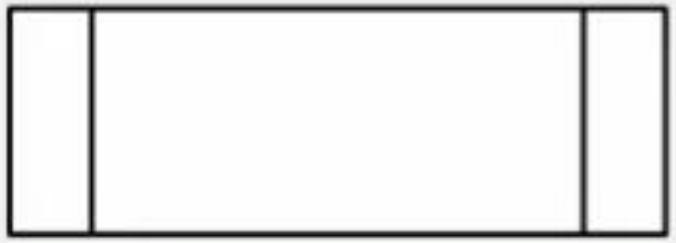
Algorithms can be best expressed using flow chart.



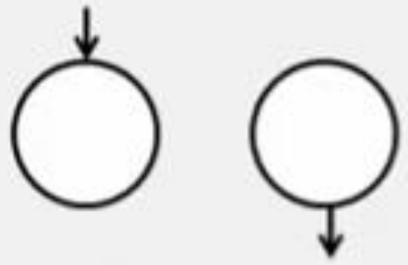
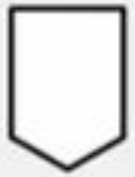
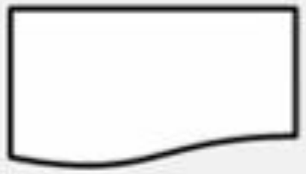
The components used to construct a flow chart are:

Name	Symbol
Terminal	 Start / Stop
Flow Line	
Decision	

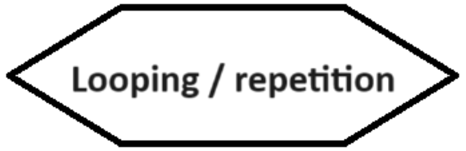
Components of Flow Chart:

Name	Symbol
Data	
Process	
Pre-define	

Computing Fundamentals

Name	Symbol
On-page Connector	
Off-page connector	
Document	

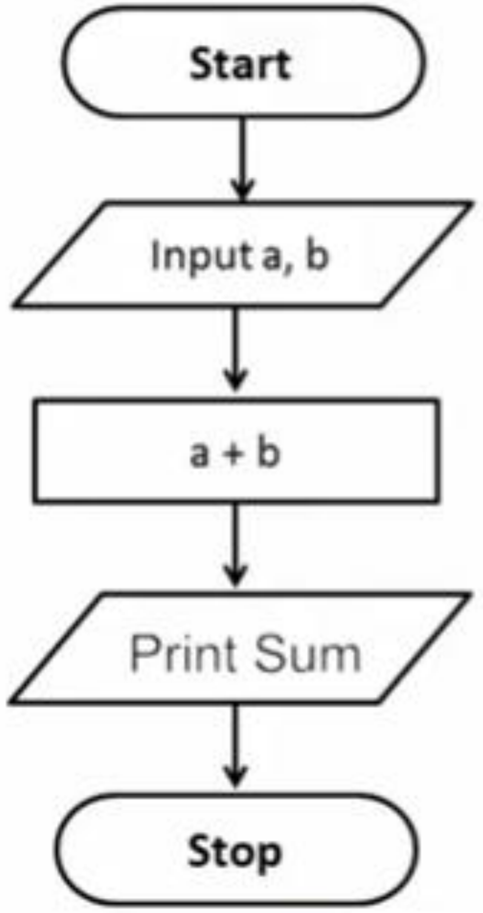
A group of instruction to be executed repeatedly



Algorithm:

```
1. Start
2. Read a, b
3. Sum ← a + b
4. Print Sum
5. End
```

Flowchart:



Activities on Algorithm and Flow chart

Algorithm Notations:

- An algorithm is written using following notations

1. Name – specifies the problem
2. Step number – each instruction is identified by a number called step number.
3. Comments – it describes the operation and is written within a pair of square brackets.
4. Beginning/Termination - it indicates the beginning/end of algorithm with a Start/Stop statement.

Activities on Algorithm and Flow chart

Write an algorithm to accept temperature in degree Celsius & convert it to Fahrenheit.

```
Name: To convert temperature in Celsius to Fahrenheit
Step1: Start
Step2: Read C
Step3: Calculate F = 9/5*C+32
Step4: [print F]
          Print ‘Temperature in F=‘, F
Step5: Stop
```

Activities on Algorithm and Flow chart

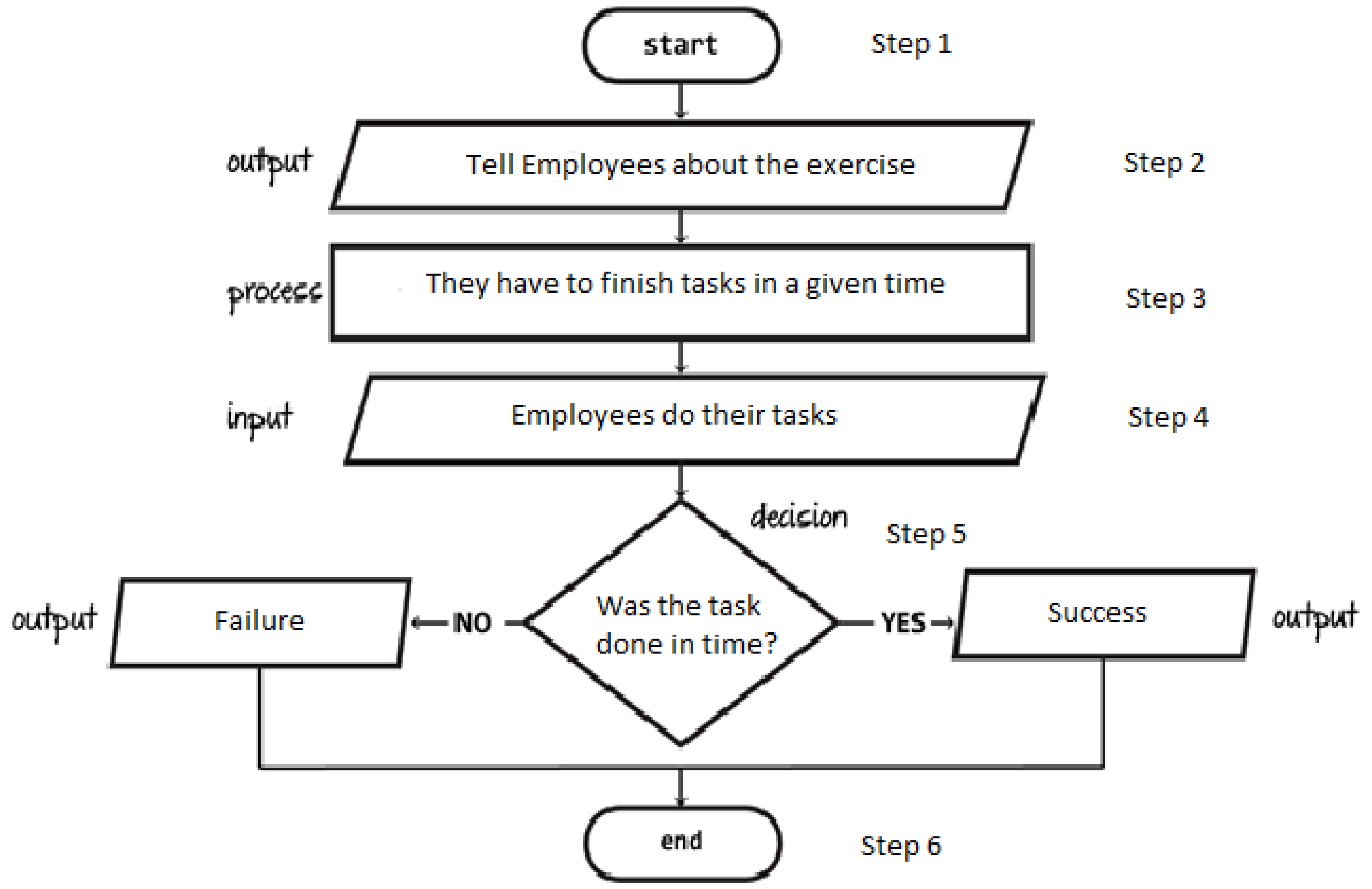
Flow Charts:

- It is a pictorial representation of step by step instruction of an algorithm.
- It specifies
  - the solution procedure,
  - the relevant computations,
  - points of decision and
  - other information required for the solution.

COMPUTING FUNDAMENTALS USING PYTHON

Activities on Algorithm and Flow chart

Draw a flow chart displaying employer assigning task to employees and check whether the task completed on time.



Activities on Algorithm

Try These: Algorithm with Flowchart

1. Write an algorithm to find simple interest by accepting principle, year, and rate of interest
2. Find the area and circumference of circle by accepting the radius
3. Accept a number and print it only if it is positive
4. Write an algorithm to find the largest of two number
5. Write an algorithm to find GCD of two numbers
6. To find the sum of first n natural numbers.

Activities on Flow chart

1. Order Processing System:

Scenario: A company receives and processes customer orders

2. Birthday party invitation:

Scenario: Inviting friends for birthday party.

3. Student Enrollment Process:

Scenario: Students enroll in courses at a university.

4. Online shopping:

Scenario: Search and buy products through online app.

5. Travel Booking System:

Scenario: A user books a trip online.

6. Hotel room Booking System:

Scenario: A user books a room online.

Activities on Flow chart

1. **Order Processing System:**

Scenario: A company receives and processes customer orders.

Step1: Start (Oval)

Step2: Receive Order (Parallelogram)

Step3: Verify Stock (Diamond)

Step4: Process Payment (Rectangle)

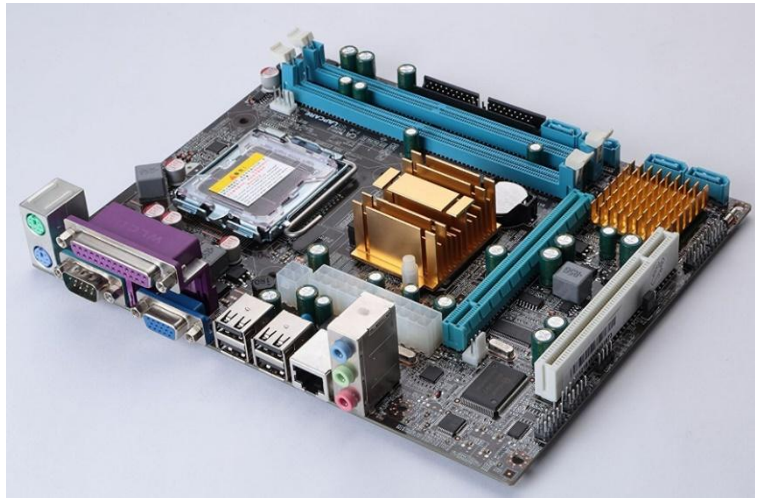
Step5: Pack and Ship (Rectangle)

Step6: Update Inventory (Rectangle)

Step7: End (Oval)

Computer Hardware

- Computer hardware comprises the physical part of a computer system.
- It includes the all-important components of the **central processing unit** (CPU) and **main memory**.
- It also includes **peripheral components** such as a keyboard, monitor, mouse, and printer.



Fundamental Hardware Components

Central processing unit (CPU) – **the “brain” of a computer system**. Interprets and executes instructions.

Main memory – **is where currently executing programs reside**. It is *volatile*, the contents are lost when the power is turned off.

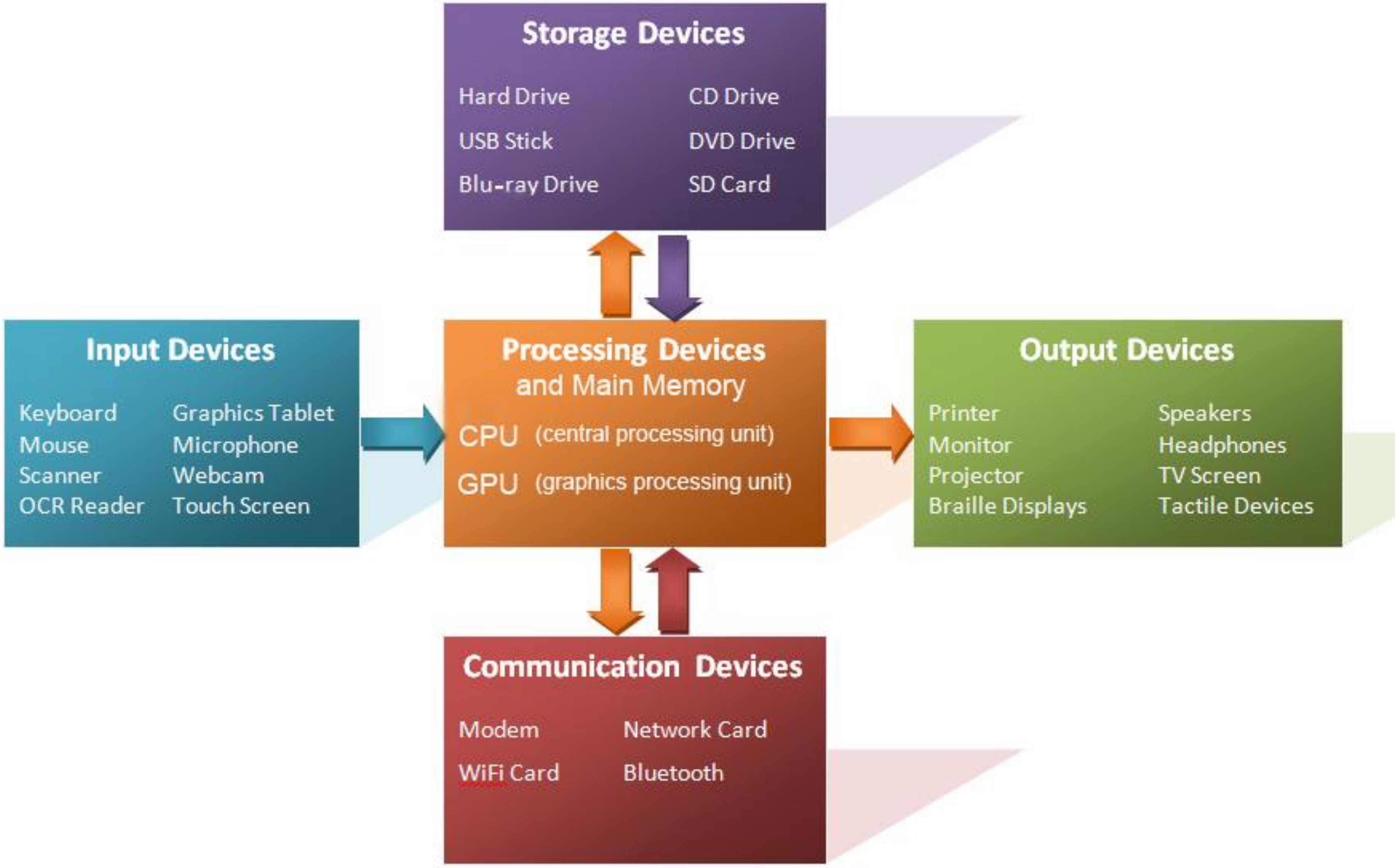
Secondary memory – **provides long-term storage of programs and data**. N*onvolatile*, the contents are retained when power is turned off.

- Can be magnetic (hard drive), optical (CD or DVD), or flash memory (USB drive).

Input/output devices – **mouse, keyboard, monitor, printer,** etc.

Buses – **transfer data between components within a computer system**. System bus (between CPU and main memory).

Computing Fundamentals – Fundamental Hardware Components



The Number System

- In everyday life we use decimal number system.
- A digital system is normally designed for **two state** operation and hence it uses the binary number system.
- In addition to binary system other systems like Octal and Hexadecimal system are also used.
- Digital computers and microprocessor based system uses hexadecimal and octal system

What is a Number System

A number system is a code that uses symbols to refer to number of items.

- **Binary Number System**
- **Decimal Number System**
- **Octal Number System**
- **Hexadecimal Number System**

Binary Number System:

- Base / radix 2.
- Only **two symbols** are used to represent numbers in this system and these are **0 and 1**. These are known as **bits**.

Decimal Number System:

- Base/radix 10
- 0, 1, 2, 3, 4, 5, 6 ,7, 8, 9 are the symbols used to represent numbers in base 10 system.

Octal Number System:

- Base/radix 8
- Eight different symbols are used to represent numbers.
- Symbols --- 0,1,2,3,4,5,6 and 7

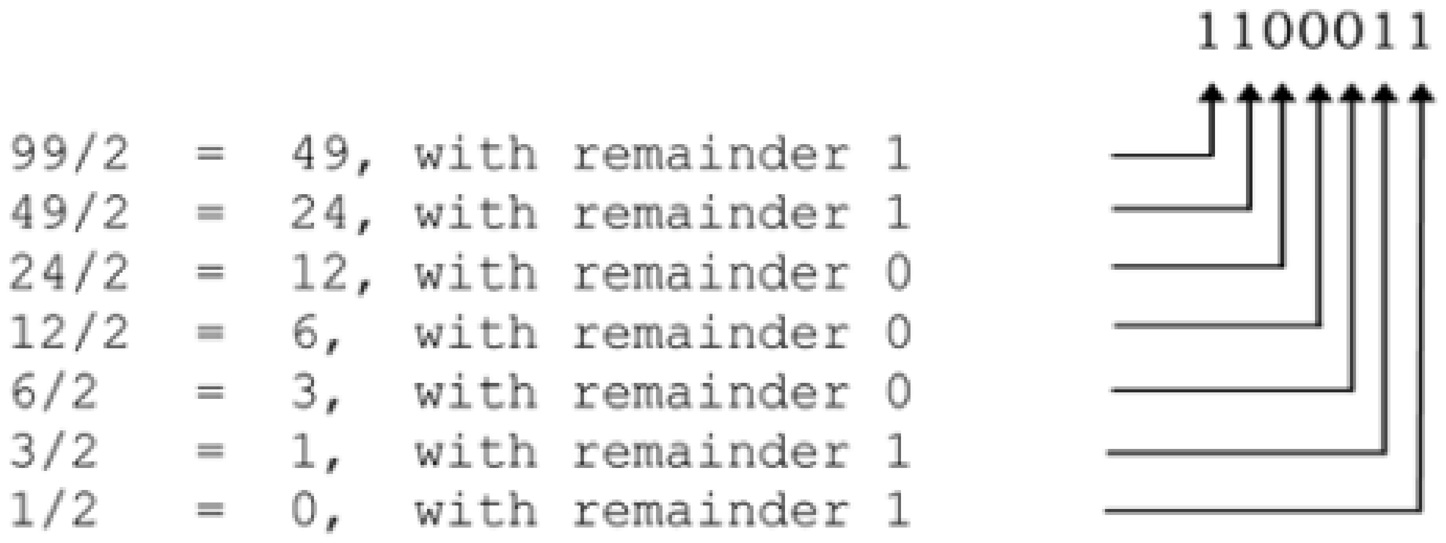
Hexadecimal Number System:

- Base/radix 16
- Sixteen different symbols are used to represent numbers.
- The digits 0 to 9 , A, B, C, D, E , F are used

Digital Computing - Number system Conversion

Decimal to Binary

- The algorithm for the conversion from base 10 to base 2 is to successively divide a number by two until the remainder becomes 0. The remainder of each division provides the next higher-order (binary) digit, as shown



```
99/2  =  49, with remainder 1
49/2  =  24, with remainder 1
24/2  =  12, with remainder 0
12/2  =  6,  with remainder 0
6/2   =  3,  with remainder 0
3/2   =  1,  with remainder 1
1/2   =  0,  with remainder 1
```

FIGURE 1-13 Converting from Base 10 to Base 2

Computing Fundamentals - Number system Conversion

Binary to Decimal

Example $(101101.10101)_2 = (\ \ )_{10}$

10 1101 . 10101 this can be written as,

$=1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} + 0x2^{-4} + 1x2^{-5}$

$= 32 + 8 + 4 + 1 + 0.5 + 0.125 + 0.03125$

$= 45 + 0.65625$

$= (45.65625)_{10}$

Octal to Decimal Conversion:

We use the same method as used in the binary case, except that we now have a radix of 8 instead of 2 (in binary).

Example--- $(1213)_8 = (\quad)_{10}$

$=1x8^3+2x8^2+1x8^1+3x8^\circ$

$= 512 + 128+8+3$

$=(651)_{10}$

Decimal to Octal Conversion:

- Conversion of decimal to octal can be performed by repeatedly **dividing the decimal number by 8** and using each reminder as a digit in the octal number being formed.

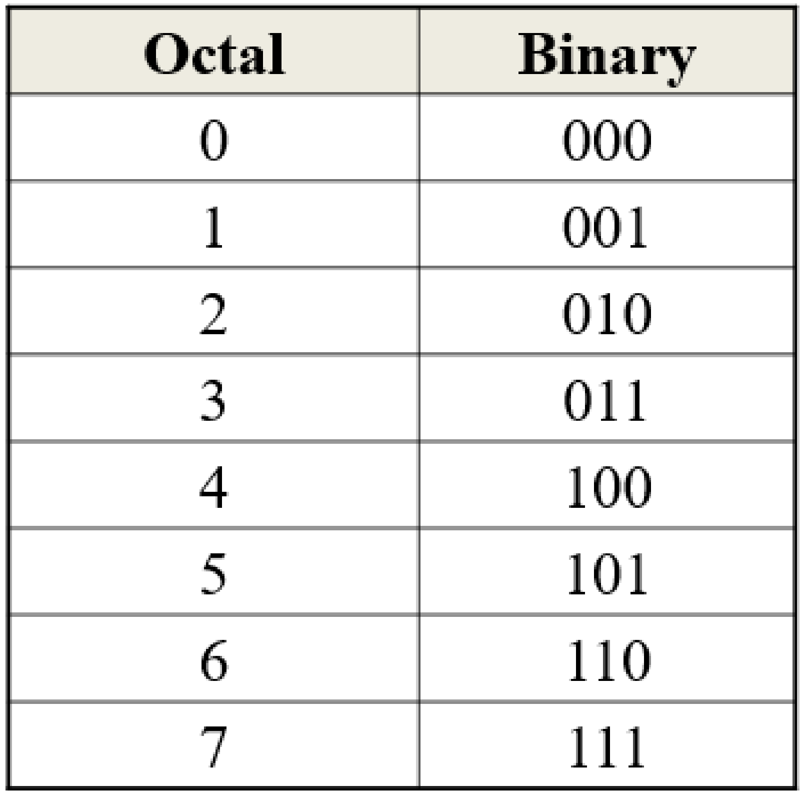
- For example to convert $(200)_{10}$ to an octal representation we divide as follows.

200 ÷ 8= 25 remainder is 0
25 ÷ 8= 3 remainder is 1
3 ÷ 8= 0 remainder is 3
Therefore $(200)_{10} = (310)_8$

Binary to Octal Conversion:

- There is a simple trick of converting a binary number to an octal number.
- Simply group the binary digits into groups of 3, Starting at the octal point and read each set of 3 binary digits according to the following table.

Binary to Octal Conversion:



Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Octal to Binary Conversion:

Consider for example:

$(67)_8=(110\ 111)_2$

$(760512.403)_8=(111110000101001010.100000011)_2$

Binary	Hexadecimal	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Decimal to Hexadecimal Conversion:

$(10)_{10} = (A)_{16}$

$125 \div 16 = 7 +$ remainder of D(13)

$7 \div 16 = 0 +$ remainder of 7

$(125)_{10} = (7D)_{16}$

Hexadecimal to Decimal Conversion:

This is straight forward but time consuming.

Example: $(BB)_{16}$ represents

$=B\text{x}16^1+B\text{x}16^\circ$

$=11\text{x}16+11\text{x}1$

$=176+11$

$=(187)_{10}$

Similarly $(AB6)_{16}=A\text{x}16^2 + B\text{x}16^1 + 6\text{x}16^\circ$

$=10\text{x}256 + 11\text{x}16 + 6$

$=2560 + 176 + 6$

$=(2742)_{10}$

Digital Computing - Number system Conversion

Binary to Hexadecimal Conversion:

- Break a binary number into groups of 4 bits and convert each group of 4 bits into hexa-decimal digit.

Examples

$(10111011)_2 = (BB)_{16}$

$(10010101)_2 = (95)_{16}$

$(11000111)_2 = (C7)_{16}$

Hexadecimal to Binary Conversion:

$(ABCD)_{16}=(1010101111001101)_2$

$(2309.7A)_{16}$

$=(0010001100001001.01111010)_2$

Hexadecimal to Octal Conversion:

$(ABC)_{16} = (101010111100)_2$
$= (5274)_8$

Note :There is no direct method for converting Hexadecimal to octal or Octal to Hexadecimal. Therefore first convert Hexadecimal/Octal to Binary then Binary To Hexadecimal/Octal.

Octal to Hexadecimal Binary Conversion:

$(234)_8 = (010011100)_2$

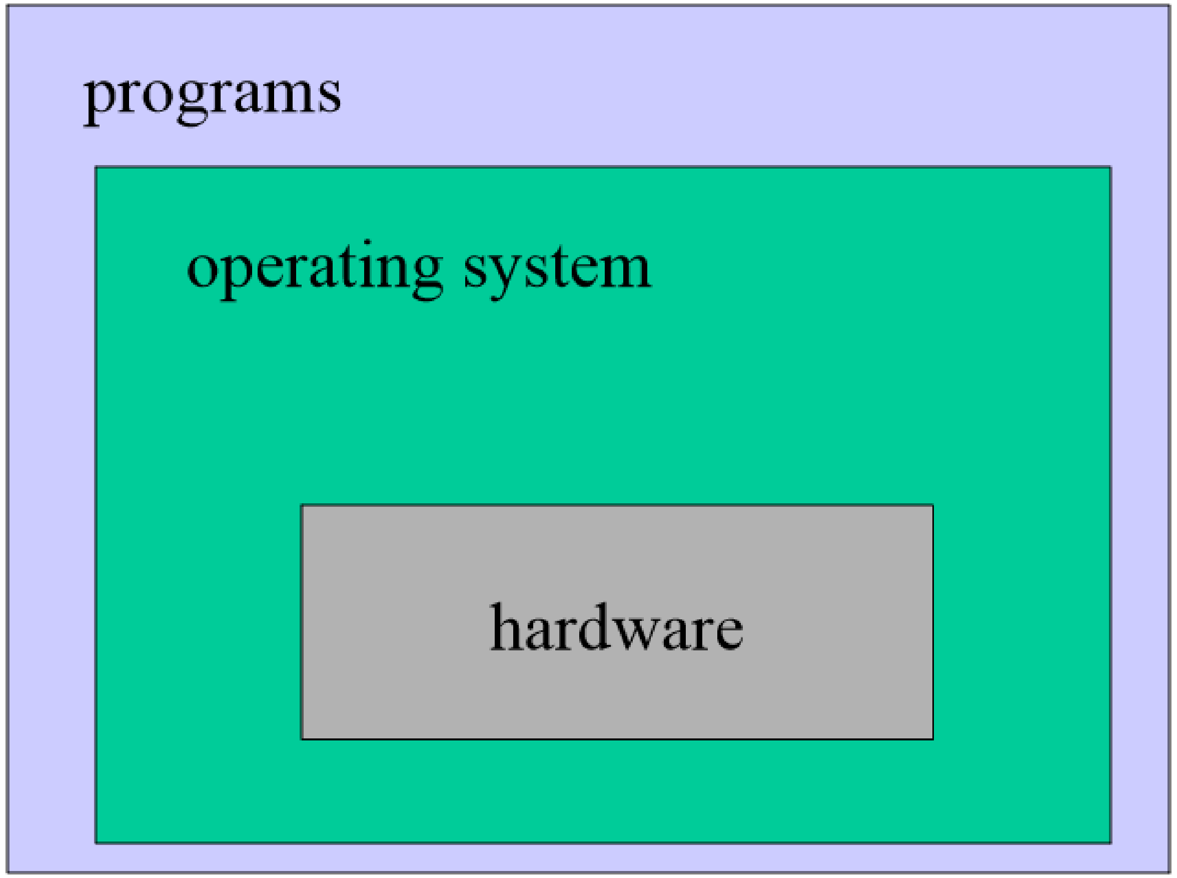
$= (09C)_{16}$

Operating System

An **operating system** is a **software** that manages and interacts with the hardware resources of a computer.

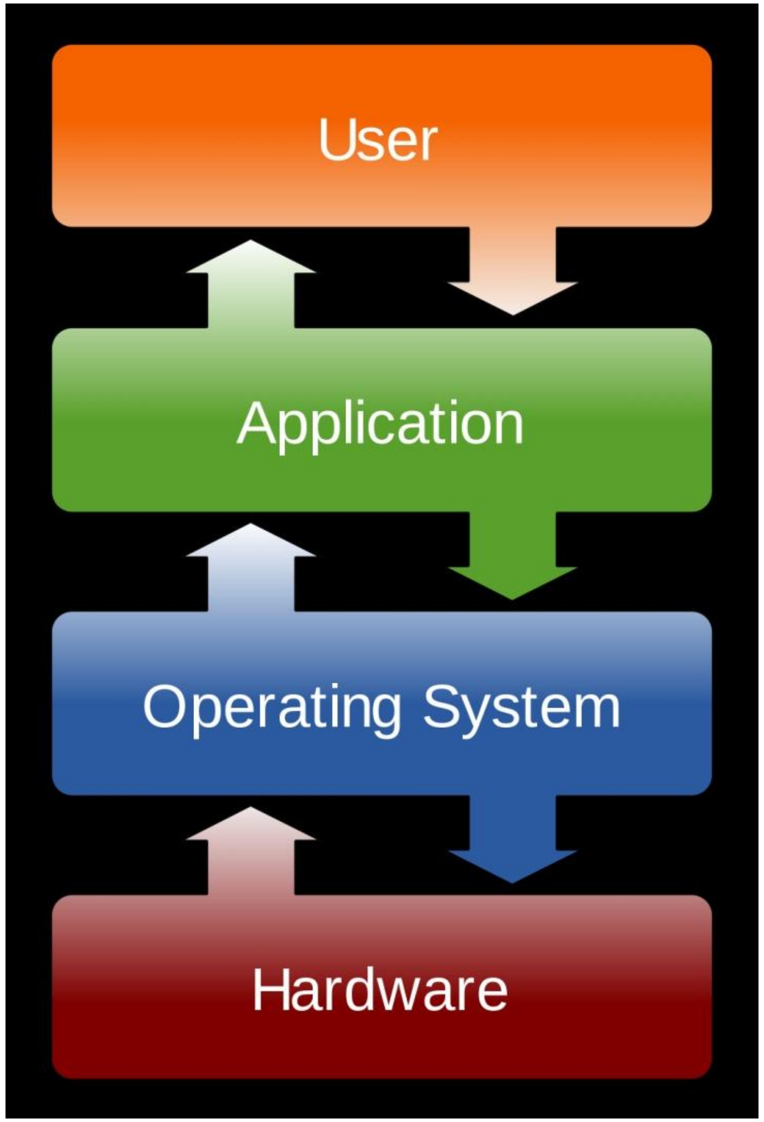
An operating system is intrinsic to the operation of a computer, it is referred to as **system software**.

Operating System Structure



Computing Fundamentals

- Operating system makes **computer hardware operational**. It is usually stored on the hard disk of a computer.
- It enables user and other programs to operate and communicate with the computer hardware.
- It is the master control program of a computer.



Computing Fundamentals

- An operating system acts as the "middle man" between the hardware and executing application programs

- For example, it **controls the allocation of memory** for the various programs that may be executing on a computer.

- Operating systems also provide a particular **user interface**. Thus, it is the operating system installed on a given computer that **determines the "look and feel" of the user interface** and how the user interacts with the system, and not the particular model computer.

Computer Software

- The first computer programs ever written were for a mechanical computer designed by **Charles Babbage** in the mid-1800s.
- The person who wrote these programs was a woman.
- **Ada Lovelace** who was a talented mathematician. Thus, she is referred to as **“the first computer programmer.”.**

What is computer software?

- **Computer software** is a set of program instructions, including related data and documentation, that can be executed by computer.

  This can be in the form of instructions on paper, or in digital form.

Computing Fundamentals

- Computer software consists of instructions that **control the operation of the computer**. Much software also includes **information for the computer to process**.

- There are four general kinds of software:

(1) Operating system software,
(2) Applications software,
(3) Programming languages,
(4) Utilities.

(1) Operating system software - reads and responds to user commands, and coordinates the flow of information among the different input and output devices.

- System software is the fundamental software that provides the basic functions and controls for a computer or other electronic device. In simple terms, system software can be thought of as the "**backbone**" or the foundational layer that allows a computer to function and operate

Computing Fundamentals

2. **Applications software** - made up of programs for all the specific uses of computers, including word processing, the management of financial documents, database management, and the processing of pictures and sounds.

3. **Programming Languages** – are used to create all other software whether it is Operating system or Application Software

4. **The Utility Software** is system software that helps to maintain the proper and smooth functioning of a Computer System

- Examples of utility programs are **antivirus software, backup software and disk tools**.

Syntax, Semantics and Program Translation

- Programming languages (called "**artificial languages**") are languages just as "**natural languages**" such as English and Mandarin (Chinese).

- ***Syntax* and *semantics*** are important concepts that apply to all languages.

What are Syntax and Semantics ?

- The **syntax** of a language is a **set of characters** and the acceptable sequences (arrangements) of those characters.

- English, for example, includes the letters of the alphabet, punctuation, and properly spelled words and properly punctuated sentences.

- The following is a syntactically correct sentence in English,
  "Hello there, how are you?"

- Where as if we write the same sentence
  "Hello there, hao are you?", is **not syntactically correct**,
  - The sequence of letters "hao" is not a word in the English language.

Semantics

The **semantics** of a language is the **meaning associated** with each syntactically correct sequence of characters.

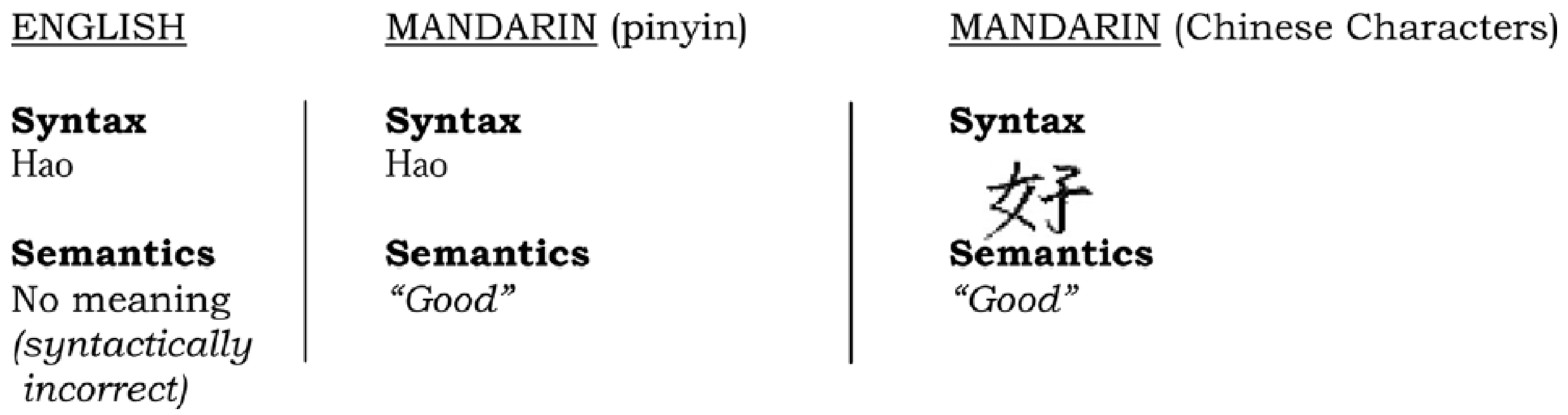
Consider the following sentence:

“Colorless green ideas sleep furiously.”

This sentence is syntactically correct, but has no meaning. Thus, it is *semantically incorrect*.

Every language has its own syntax and semantics.

In **Mandarin,** “Hao” is syntactically correct meaning “good.” (“Hao” is from a system called pinyin, which uses the Roman alphabet rather than Chinese characters for writing Mandarin.) Thus, every language has its own syntax and semantics



ENGLISH	MANDARIN (pinyin)	MANDARIN (Chinese Characters)
Syntax Hao	**Syntax** Hao	**Syntax** 好
Semantics No meaning *(syntactically incorrect)*	**Semantics** *“Good”*	**Semantics** *“Good”*

Computing Fundamentals

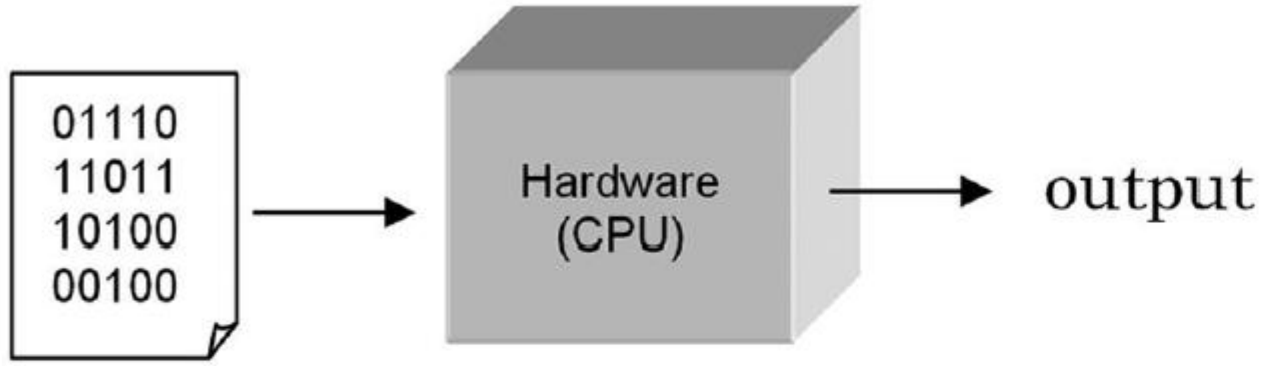
- The **Syntax** of a programming language is used to signify the **structure of programs without considering their meaning.**

- **Semantics** term in a programming language is used to figure out the relationship among the syntax and the model of computation.

  - Semantics emphasizes the interpretation of a program so that the programmer could understand it in an easy way or predict the outcome of program execution

Program Translation

- A central processing unit (CPU) is designed to **interpret and execute** a specific set of instructions represented in **binary form** (i.e., 1s and 0s) called **machine code**.
- Only programs in machine code can be executed by a CPU.



Computing Fundamentals

- Writing programs at this “low level” is tedious and error-prone. **Therefore, most programs are written in a “high-level” programming language such as Python**.

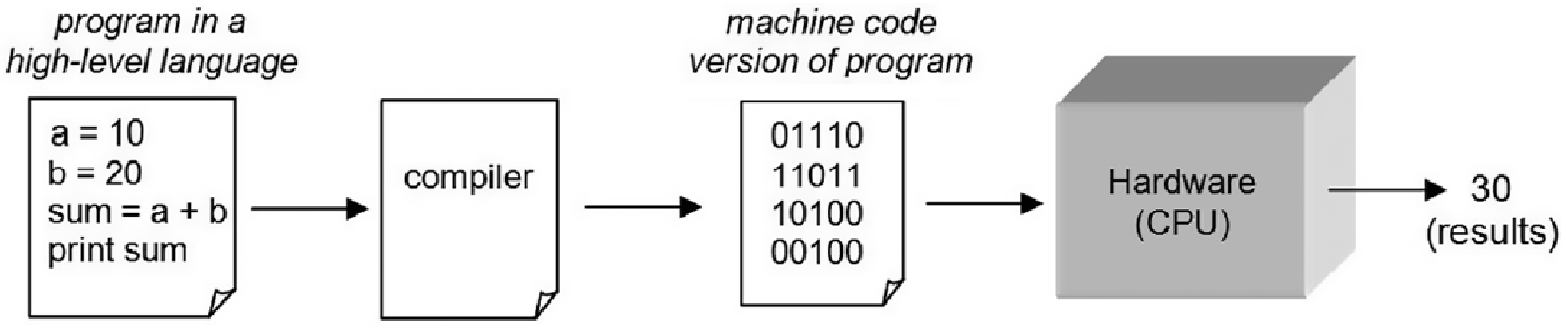
- Since the instructions of such programs are not in machine code that a CPU can execute, **a translator program must be used**.

There are two fundamental types of translators:

- **Compiler** translates programs into machine code to be executed by the CPU (C, C++, Java)
- **Interpreter** executes instructions in place of the CPU (Perl, Python etc.)

1. Compiler

A compiler takes the source code as a whole and translates it into object code all in one go. Once converted, the object code can be run unassisted at any time. This process is called **compilation**

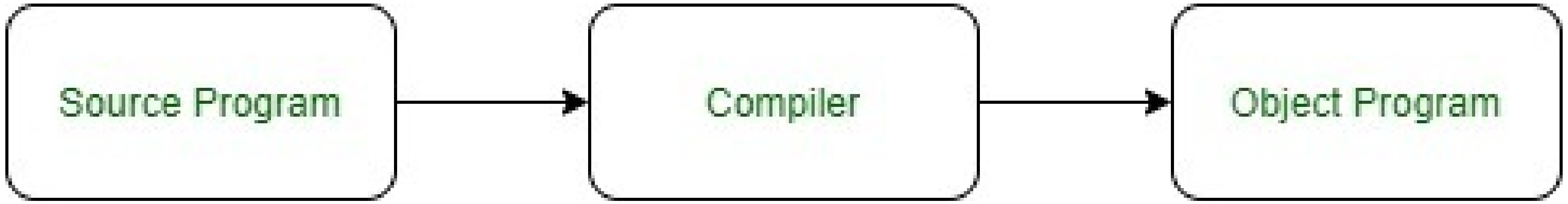


Compiled programs generally execute **faster** than interpreted programs.

Compiler:

- It is a translator which takes input i.e., High-Level Language, and produces an output of low-level language i.e. machine or assembly language.
- A compiler is more intelligent than an assembler it checks all kinds of limits, ranges, errors, etc.
- But its program run time is more and occupies a larger part of memory. It has slow speed because a compiler goes through the entire program and then translates the entire program into machine codes.

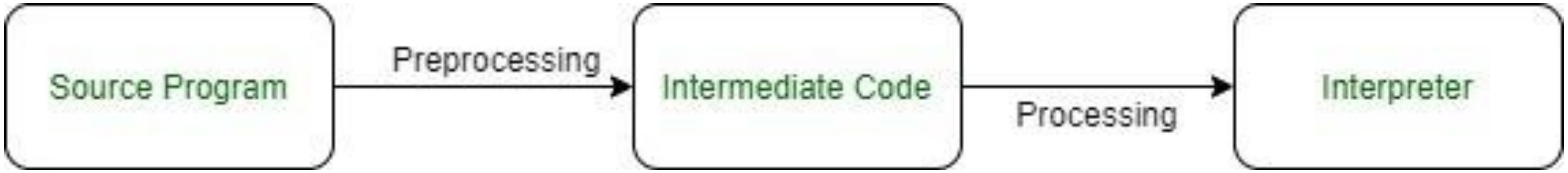
Compiler:



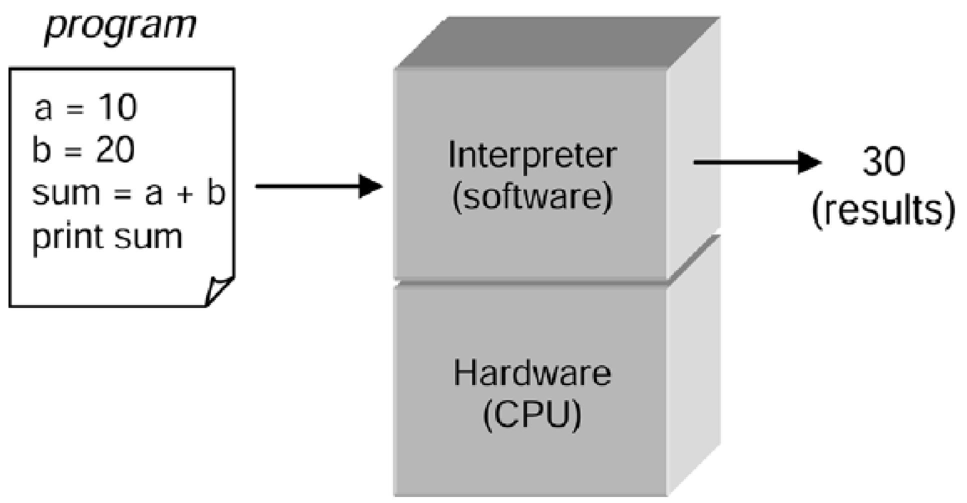
2. Interpreter

An interpreter is a program that translates a programming language into a comprehensible language. –

- It translates only one statement of the program at a time.
- Interpreters, more often than not are smaller than compilers.



Interpreter



An interpreter can immediately execute instructions as they are entered. This is referred to as **interactive mode**. This is a very useful feature for program development. Python, as we shall see, is executed by an interpreter.

Computing Fundamentals

	Compiler	Interpreter
1.	Compiler scans the whole program in one go.	Translates program one statement at a time.
2.	As it scans the code in one go, the errors (if any) are shown at the end together.	Considering it scans code one line at a time, errors are shown line by line.
3.	Main advantage of compilers is it's execution time.	Due to interpreters being slow in executing the object code, it is preferred less.
4.	It converts the source code into object code.	It does not convert source code into object code instead it scans it line by line
5	It does not require source code for later execution.	It requires source code for later execution.
Eg.	C, C++, C# etc.	Python, Ruby, Perl, SNOBOL, MATLAB, etc.

What are Syntax Errors and Semantic Errors?

- **Syntax errors are caused by invalid syntax**
- `for example, entering **prnt** instead of **print**.

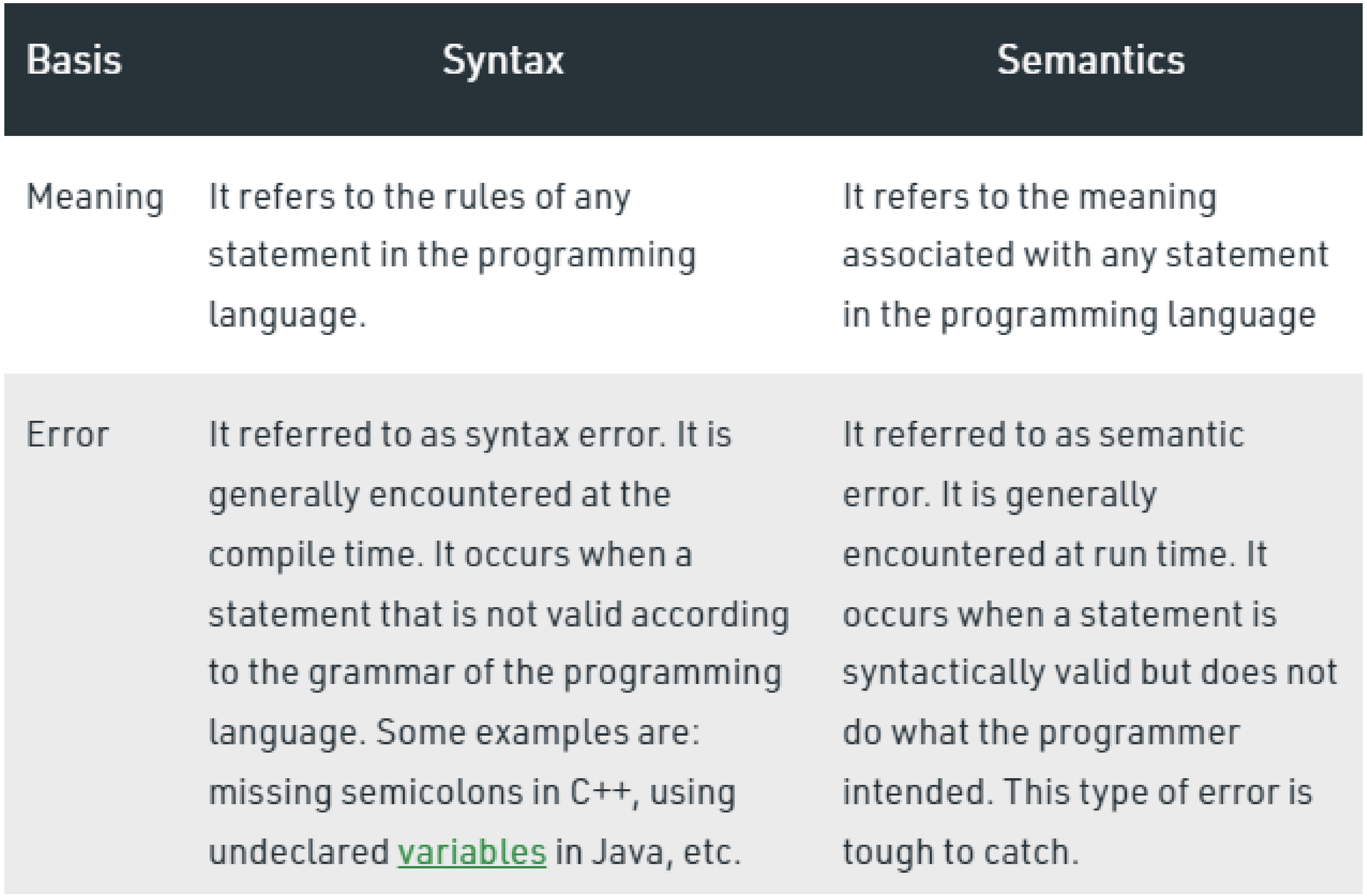
- Since a translator cannot understand instructions containing syntax errors, translators terminate when encountering such errors indicating where in the program the problem occurred.

- **In contrast**, **semantic errors** (generally called logic errors) **are errors in program logic**.

- Such errors cannot be automatically detected, since translators cannot understand the intent of a given computation.

COMPUTING FUNDAMENTALS USING PYTHON

Computing Fundamentals



Basis	Syntax	Semantics
Meaning	It refers to the rules of any statement in the programming language.	It refers to the meaning associated with any statement in the programming language
Error	It referred to as syntax error. It is generally encountered at the compile time. It occurs when a statement that is not valid according to the grammar of the programming language. Some examples are: missing semicolons in C++, using undeclared variables in Java, etc.	It referred to as semantic error. It is generally encountered at run time. It occurs when a statement is syntactically valid but does not do what the programmer intended. This type of error is tough to catch.

PES UNIVERSITY

THANK YOU

Ms. Archana A
Assistant Professor
Department of Computer Science
archana@pes.edu

+91 80 6666 3333 Extn 392