



**SNS COLLEGE OF ENGINEERING**  
Kurumbapalayam (Po), Coimbatore – 641 107

**An Autonomous Institution**

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade  
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai



**DEPARTMENT OF CSE (IoT & CYBER SECURITY INCLUDING BLOCKCHAIN TECHNOLOGY)**



# **19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING**

❖ **A readable, dynamic, pleasant, flexible, fast and powerful language**

## Recap:

- A Computer is **an electronic machine** that can be programmed to accept data (*input*), and process it into useful information (*output*). Data is put into secondary storage (*storage*) for safekeeping or later use.
- The computer has evolved from a large-sized simple calculating machine to a smaller but much more powerful machine.
- Problem is a thing that requires logical thought and /or mathematics to solve.
- Problem solving is the **systematic approach** to define the problem and creating number of solutions.

## Recap:

- Computers are built to solve problems with algorithmic solutions, which are often difficult or very time consuming when input is large.
- **A computational problem** is a problem that a computer might be able to solve or a question that a computer may be able to answer.
- **Computational thinking** is an approach to problem-solving that involves using a set of practices and principles from computer science to formulate a solution that's executable by a computer.

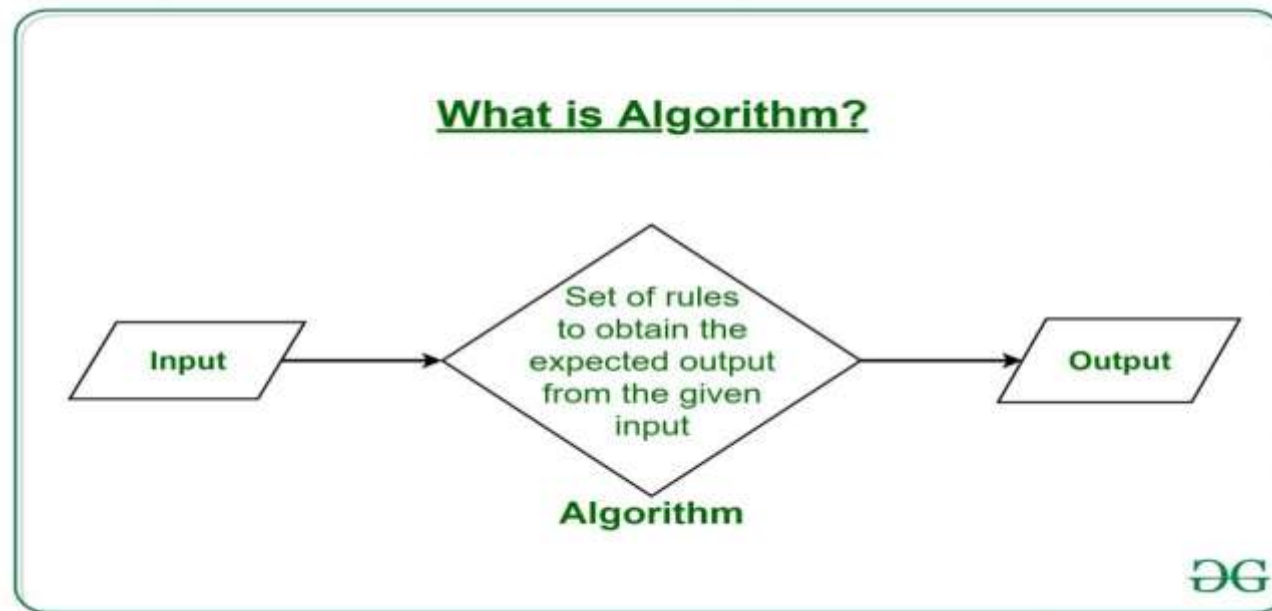
## 1.3 ALGORITHM:

- Algorithm is defined as a sequence of instructions that describe a method for solving a problem.
- Algorithm - *a sequence of clearly defined steps* that describe a process to follow a finite set of unambiguous instructions with clear start and end points.
- In other words algorithm is a step by step procedure for solving a problem.
- The term “algorithm” was derived from the name of Mohammed al Khowarizmi, a Persian mathematician in the ninth century.
- Al-Khowarizmi → Algorismus (in Latin) → Algorithm

## 1.3 ALGORITHM:

A recipe is a good **example** of an **algorithm** because, says what must be done, *step by step*.

It takes inputs (ingredients) and produces an output (the completed dish).



## **1.3 ALGORITHM:**

### **Defining algorithms:**

The definition of an algorithm is complex and involves several properties.

Those properties are:

- Collection of individual steps
- Definiteness
- Sequential

## 1.3 ALGORITHM:

### Defining algorithms:

#### 1) Collection of individual steps:

- An algorithm is a **collection of individual steps**.
- A recipe fits this analogy quite simply, filled as it is with steps like:
  - ‘pre-heat the oven to 180 degrees Celsius’
  - or
  - ‘add two tablespoons of sugar to the bowl’

## 1.3 ALGORITHM:

### Defining algorithms:

#### 2) Definiteness:

- Definiteness, meaning that **every step must be precisely defined.**
- Each step in an algorithm can have one and only one meaning, otherwise it is ambiguous.
- Similarly, chefs have come to the same conclusion, which is why they produce recipes using precise measurements instead of writing things like ‘some sugar’ or ‘cook it for a while’.



## 1.3 ALGORITHM:

### Defining algorithms:

#### 3) Sequential:

- Algorithms are also **sequential**.
- **The steps that make up the process must be carried out in the order specified.**
- Failing to do this means that the result of executing the algorithm is likely incorrect.

## 1.3 ALGORITHM:

### Defining algorithms:

#### 3) Sequential..

Think back to the analogy.

- Dicing an onion and frying an onion are different steps.
- Dicing an onion before you fry it has a different outcome than the reverse.
- Similarly, multiplying a number by 2 then adding 5 to it yields a different result from adding 5 first then doubling it.
- Like a recipe, you must respect the sequence when running through an algorithm for it to have any meaningful result.

## 1.3 ALGORITHM:

### Properties of Algorithms:

Every algorithm must have five essential properties:

- (1) **Inputs specified:** An algorithm must have zero or more inputs, We must specify the type of the data, the amount of data, and the form that the data will take.
- (2) **Outputs specified :** An algorithm has one or more outputs, which have a specified relation to the inputs.
- (3) **Definiteness:** Every detail of each step must be clearly specified.
- (4) **Effectiveness:** All operations to be performed must be sufficiently basic that they can be done exactly and in finite length.
- (5) **Finiteness:** An algorithm must always terminate after a finite number of steps.

## **1.3 ALGORITHM:**

### **Method for Developing an Algorithm:**

- (1) Define the problem: State the problem to be solved in clear and concise manner.
- (2) List the inputs and outputs
- (3) Describe the steps needed to convert input to output
- (4) Test the algorithm: Choose input data and verify that the algorithm works.

## 1.3 ALGORITHM:

### The Characteristics of a Good Algorithm :

- **Precision** – the steps are precisely stated (defined).
- **Uniqueness** – results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
- **Finiteness** – the algorithm stops after a finite number of instructions are executed.
- **Effectiveness** – algorithm should be most effective among many different ways to solve a problem.
- **Input** – the algorithm receives input.
- **Output** – the algorithm produces output.
- **Generality** – the algorithm applies to a set of inputs.

## **1.3 ALGORITHM:**

### **Real Life Example Procedure to cook Bread Toast :**

Step 1 : Grab a loaf of bread

Step 2 : Get a pan and place it on the stove let it heat

Step 3 : Pour some oil on the pan and wait for oil to be heated

Step 4 : Put a slice on the pan and roast until it become brown in shade

Step 5 : Turn the slice and roast until it become brown in shade

Step 6 : Get the toasted bread from the pan and serve it on a plate with anything or nothing.

## **1.3 ALGORITHM:**

*Example 1* : Algorithm for adding two numbers:

Step 1 : Get the 2 numbers from the user as input.

Step 2 : Perform addition of those 2 numbers.

Step 3 : Store the answer for display.

Step 4 : Display the stored value to the user.

## 1.4 Building blocks of algorithms (statements, state, control flow, functions):

An algorithm includes basic building blocks that are used to express any kind of the task to the computer.

Algorithms can be constructed from basic building blocks namely, **sequence**, **selection** and **iteration**.

1. Instructions/ Statements

2. State

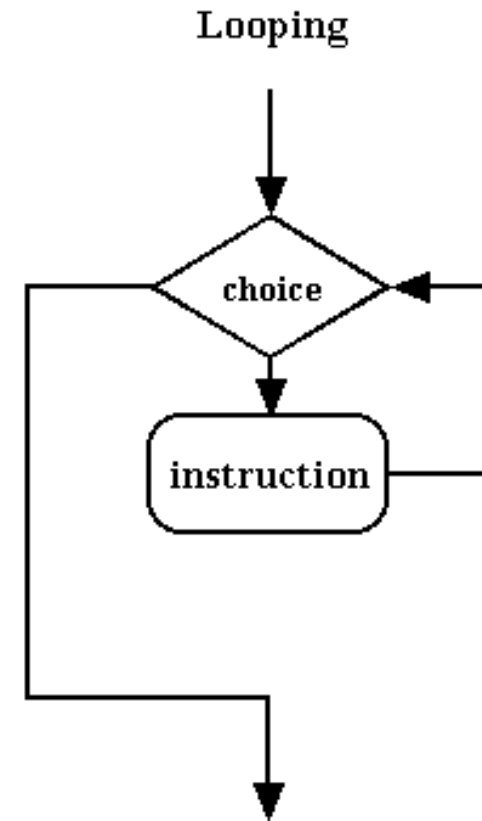
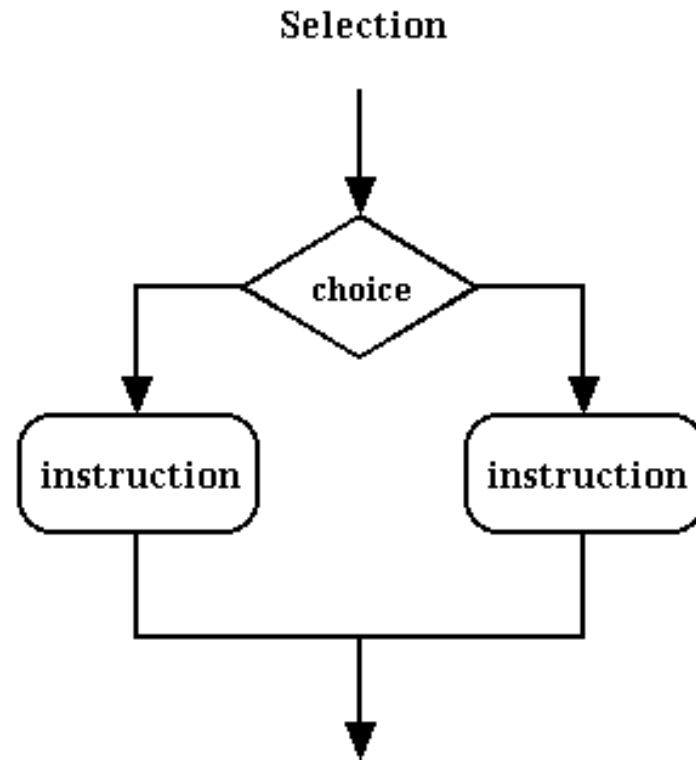
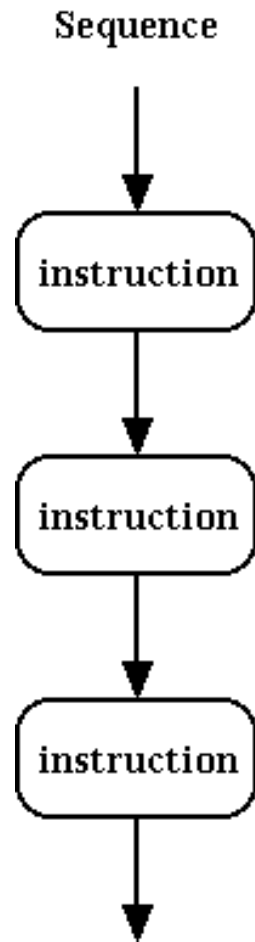
3. Control Flow

4. Functions



## 1.4 Building blocks of algorithms (statements, state, control flow, functions):

Algorithms can be constructed from basic building blocks namely, **sequence**, **selection** and **iteration**.



## 1.4 Building blocks of algorithms (statements, state, control flow, functions):

### Statements:

Statement is a single action in a computer. In a computer statements might include some of the following actions:

- input data-information given to the program
- process data-perform operation on a given input
- output data-processed result

### State:

Transition from one process to another process under specified condition with in a time is called *state*.

## 1.4 Building blocks of algorithms (statements, state, control flow, functions):

### Control flow:

The process of **executing the individual statements in a given order** is called control flow.

### The control can be executed in three ways:

1. sequence
2. selection
3. iteration

### Functions:

A section of computer code that performs a specific task.

## 1.4.1 Statements:

- A statement is the smallest standalone element of an imperative programming language that expresses some action to be carried out.
- There are two types of statement,
  - Simple Statement
  - Compound Statement

## 1.4.1 Statements:

- **Simple statements:** It is used to represent single action need to be done.

- **assertion:** `assert(ptr != NULL);`

Comparison

- **assignment:** `A:= A + 5`

Assigning a value 5 to A

- **goto:** `goto next;`

Sent the control to different block of same program

- **return:** `return 5;`

Return a value 5 after the execution of function

- **call:** `clearScreen()`

Calling the Function (`clearScreen`) which performs clearing previous outputs from the computer screen

## 1.4.1 Statements:

- **Compound statements:**

- It is a set of statements, that used to perform a sequence of operations repeatedly or condition based executions.

**block:** Set of statements

begin

integer NUMBER;

WRITE('Number? ');

READLN(NUMBER);

A:= A\*NUMBER

end

## 1.4.1 Statements:

### Compound statements:

#### *do-loop:*

Do

{

    computation(&i);

} while (i < 10);

Looping a set of statements repeatedly until some condition is satisfied. We can't predict when the condition becomes satisfiable. At least it will do the loop sequence once.

## 1.4.1 Statements:

### Compound statements:

#### *for-loop:*

for A:=1 to 10 do

    WRITELN(A)

end

Looping a set of statements repeatedly until some condition is satisfied. We can run the loop for certain iterations. Prediction of loop termination is possible.



## 1.4.1 Statements:

### Compound statements:

#### *if-statement:*

if  $A > 3$  then

    WRITELN(A)

else

    WRITELN("NOT YET");

end

Normally it contains two sets of statements. State or value is compared with a conditions if it is satisfied the “if” block will be executed otherwise else part will be executed.

## 1.4.1 Statements:

### Compound statements:

#### *switch-statement:*

```
switch (c)
```

```
{
```

```
case 'a': alert(); break;
```

```
case 'q': quit(); break;
```

```
}
```

It contains more than two blocks of statement each one has the conditions. When the program reaches a state with a value, first hit of matching conditions block will be executed. If nothing matches then default block of statements will be executed.

## 1.4.1 Statements:

### Compound statements:

#### *while-loop:*

```
while NOT EOF DO
```

```
    begin
```

```
        READLN
```

```
    end
```

- Looping a set of statements repeatedly until some condition is satisfied.
- We can't predict when the condition becomes satisfiable.
- This loop is entry controlled.
- Control will enter into the loop only if condition is satisfiable.

## 1.4.2 State:

- **State**: the current configuration of all information kept track of by a program at any one instant in time.
- As a computer progresses through an algorithm, just as you progress through a recipe, the state of things can change.
- Clearly sequencing the steps of an algorithm ensures that state always changes in the same way whenever the algorithm is executed.

## 1.4.2 State:

- In computer science, a program is described as stateful if it is designed to remember preceding events or user interactions; the remembered information is called the state of the system.
- If a program gets sufficient data processed then it moves to another state. A successful execution of program include the reaching the final state of the program.

## 1.4.2 State:

- At each instant in time, the environment in which the algorithm is being run exists in some particular state.
- But by the time the next step is executed, something might have changed. The environment really exists as a series of snapshots, one for each step of the algorithm.

## 1.4.2 State:

- The recipe analogy spells this out.
- At the start you might have butter, flour, milk, eggs and sugar. After each step, you take a photograph of the kitchen. The photos will show that, bit by bit, the state of the ingredients changes. Flour goes into a bowl; then the eggs join it; then the butter goes into the pan; and so on. There is no global view of the ingredients; just a series of snapshots.

## Summary:

- Algorithm is a **sequence of clearly defined steps** that describe a process to follow a finite set of unambiguous instructions with clear start and end points.
- The definition of an algorithm is complex and involves several properties such as , **i) Collection of individual steps, 2) Definiteness, 3) Sequential.**
- An algorithm includes basic building blocks that are used to express any kind of the task to the computer.
- Building blocks of an algorithms are: **1. Instructions/ Statements , 2. State ,3. Control Flow , 4. Functions.**



**THANK YOU**