**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 COMPUTER SCIENCE AND ENGINEERING**


# 19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING

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

## 1.6 Algorithmic problem solving:

### 6. Methods of Specifying an Algorithm:

- Three ways to specify an algorithm
  - Pseudocode
  - Flowchart
  - Programming language

## 1.6 Algorithmic problem solving:

### 6. Methods of Specifying an Algorithm:..

6.1 Pseudocode :

- ***Pseudocode*** is a mixture of a natural language and programming language-like constructs.
- Pseudocode is usually more precise than natural language, and its usage often yields more concise algorithm descriptions.

## 1.6 Algorithmic problem solving:

### 6. Methods of Specifying an Algorithm:..

### 6.2 Flowchart:

- In the earlier days of computing, the dominant vehicle for specifying algorithms was a ***flowchart.***
- ***A Flow chart*** is a method of expressing an algorithm by a collection of connected geometric shapes containing descriptions of the algorithm's steps.

## 1.6 Algorithmic problem solving:

### 6. Methods of Specifying an Algorithm:..

### 6.3 Programming language:

- A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output.
- Programming languages generally consist of ***instructions for a computer***.
- Programming languages can be used to create programs that implement specific algorithms.
- Eg : C, C++, COBAL, JAVA, Python ... Etc

## 1.6 Algorithmic problem solving:

### 7. Proving an Algorithm's correctness:

- Once the algorithm has been specified, then its ***correctness*** must be proved.
- An algorithm must yield a required result for every legitimate input in a finite amount of time.
- For some algorithm, a proof of correctness is quite easy; for others, it can be quite complex.

## 1.6 Algorithmic problem solving:

### 7. Proving an Algorithm's correctness:..

- A common technique for proving correctness is to ***use mathematical induction*** because an algorithm's iterations provide a natural sequence of steps needed for such proofs.

- The notion of correctness for approximation algorithm is less straightforward than it is for exact algorithms.

## 1.6 Algorithmic problem solving:

### 8. Analyzing an Algorithm:

- Our algorithms need to possess several qualities. After correctness, the most important one is efficiency.

- There are two kind of algorithm efficiency: i) **Time efficiency** ii) **Space efficiency**

- Time efficiency: Indicates **how fast the algorithm runs**.

## 1.6 Algorithmic problem solving:

### 8. Analyzing an Algorithm:..

- Space efficiency: indicates **how much extra memory** the algorithm needs.

- Another desirable characteristic's of an algorithm are ***simplicity and generality.***

- If you are not satisfied with the algorithm's ***efficiency, simplicity, or generality,*** you must return to the drawing board and redesign the algorithm.

## 1.6 Algorithmic problem solving:

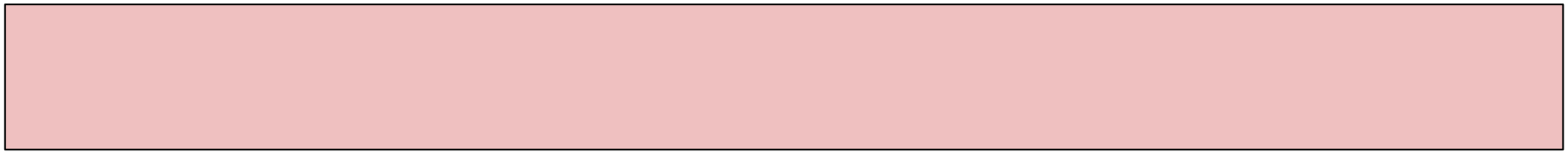
### 9. Coding an Algorithm:

- Most algorithms are destined to be ultimately implemented as computer programs.

- The coding / implementation of an algorithm is done by a suitable programming language like C, C++, JAVA

- It is very essential to write an optimized code (efficient code) to reduce the burden of compiler.

## 1.6 Algorithmic problem solving:



- As a rule a good algorithm is a result of repeated effort and rework.

- Even if you have been fortunate enough to get an algorithmic idea that seems perfect, you should still try to see whether it can be improved.

## 1.6 Algorithmic problem solving:

- An important issue of algorithmic problem solving is the question of whether or not every problem can be solved by an algorithm.

- Fortunately, a vast majority of problems in practical computing can be solved by an algorithm.

**Summary:**

- An algorithm is a sequence of non ambiguous instructions for solving a problem in a finite amount of time.
- An input to an algorithm specifies an instance of the problem the algorithm solves.
- Algorithm can be specified in a natural language or a pseudocode; they can also be implemented as computer programs.

**Summary:**

- Algorithm design techniques are ***general approaches to solving problems algorithmically***, applicable to a verity of problems from different areas of computing.
- The same problem can often be solved by several algorithms.
- Algorithms operate on **data**. This makes the issue of data structuring critical for efficient algorithmic problem solving.


THANK YOU