NEP NEW SYLLABUS

PROBLEM SOLVING AND PYTHON PROGRAMMING

Credits : 2

For F.Y.B.Sc. (Data Science - DS-101-T)

Semester - I

As Per New Syllabus, Effective from June 2025-26

Dr. Ms. Manisha Bharambe

M.Sc. (Comp. Sci), M.Phil, Ph.D (Comp. Sci.)
Associate Professor
Computer Science Department
MES Abasaheb Garware College, Pune 4

Price ₹ 250.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N7806

Problem Solving and Python Programming **ISBN 978-93-6993-607-6**

First Edition : October 2025

© : Authors

The text of this publication, or any part thereof, should not be reproduced or transmitted in any form or stored in any computer storage system or device for distribution including photocopy, recording, taping or information retrieval system or reproduced on any disc, tape, perforated media or other information storage device etc., without the written permission of Authors with whom the rights are reserved. Breach of this condition is liable for legal action.

Every effort has been made to avoid errors or omissions in this publication. In spite of this, errors may have crept in. Any mistake, error or discrepancy so noted and shall be brought to our notice shall be taken care of in the next edition. It is notified that neither the publisher nor the authors or seller shall be responsible for any damage or loss of action to any one, of any kind, in any manner, therefrom. The reader must cross check all the facts and contents with original Government notification or publications.

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off J.M. Road, Pune – 411005
Tel - (020) 25512336/37/39
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Survey No. 10/1A, Ghule Industrial Estate
Nanded Gaon Road
Nanded, Pune - 411041

DISTRIBUTION CENTRES

PUNE

Nirali Prakashan
(For orders outside Pune)
S. No. 28/27, Dhayari Narhe Road, Near Asian College
Pune 411041, Maharashtra
Mobile : 9049672225

Nirali Prakashan
(For orders within Pune)
119, Budhwar Peth, Jogeshwari Mandir Lane
Pune 411002, Maharashtra
Mobile : 9657703145, 9890997937
Email : niralilocal@pragationline.com

MUMBAI

Nirali Prakashan
Rasdhara Co-op. Hsg. Society Ltd., 'D' Wing Ground Floor, 385 S.V.P. Road
Girgaum, Mumbai 400004, Maharashtra
Mobile : 7045821020, Tel : (022) 2385 6339 / 2386 9976
Email : mumbai@niralibooks.com

DISTRIBUTION BRANCHES

DELHI

Nirali Prakashan
Room No. 2 Ground Floor
4575/15 Omkar Tower, Agarwal Road
Darya Ganj, New Delhi 110002
Mobile : 9818561840
Email : delhi@niralibooks.com

BENGALURU

Nirali Prakashan
Maitri Ground Floor, Jaya Apartments,
No. 99, 6th Cross, 6th Main,
Malleswaram, Bengaluru 560003
Karnataka; Mob : 9686821074
Email : bengaluru@niralibooks.com

NAGPUR

Nirali Prakashan
Above Maratha Mandir, Shop No. 3,
First Floor, Rani Jhanshi Square,
Sitabuldi Nagpur 440012 (MAH)
Tel : (0712) 254 7129
Email : nagpur@niralibooks.com

KOLHAPUR

Nirali Prakashan
438/2, Bhosale Plaza, Ground Floor
Khasbag, Opp. Balgopal Talim
Kolhapur 416 012, Maharashtra
Mob : 9850046155
Email : kolhapur@niralibooks.com

JALGAON

Nirali Prakashan
34, V. V. Golani Market, Navi Peth,
Jalgaon 425001, Maharashtra
Tel : (0257) 222 0395
Mob : 94234 91860
Email : jalgaon@niralibooks.com

SOLAPUR

Nirali Prakashan
R-158/2, Avanti Nagar, Near Golden
Gate, Pune Naka Chowk
Solapur 413001, Maharashtra
Mobile 9890918687
Email : solapur@niralibooks.com

For Order Enquiries: marketing@pragationline.com | www.niralibooks.com

Also find us on www.facebook.com/niralibooks

Preface ...

We take this opportunity to present this book entitled as **"Problem Solving & Python Programming"** to the students of First Semester - B.Sc. (Data Science). The object of this book is to present the subject matter in a most concise and simple manner. The book is written strictly according to the New Syllabus (NEP Pattern).

The book has its own unique features. It brings out the subject in a very simple and lucid manner for easy and comprehensive understanding of the basic concepts, its intricacies, procedures and practices. This book will help the readers to have a broader view on Data Structures. The language used in this book is easy and will help students to improve their vocabulary of Technical terms and understand the matter in a better and happier way.

We sincerely thank Shri. Dineshbhai Furia and Shri. Jignesh Furia of Nirali Prakashan, for the confidence reposed in us and giving us this opportunity to reach out to the students as well as teachers.

We have given our best inputs for this book. Any suggestions towards the improvement of this book and sincere comments are most welcome on niralipune@pragationline.com.

Authors

■■■

Syllabus ...

1. Introduction to Problem Solving [5 Hrs.]

1.1 What is Problem Solving?

1.2 Problem Solving Steps

1.3 Algorithms - Definition, Characteristics, Examples, Advantages and Limitations

1.4 Flowcharts - Definition, Notations, Examples, Advantages and Limitations, Comparison with Algorithms

1.5 Pseudo Codes - Notations, Examples, Advantages and Limitations

1.6 Introduction to Programming

1.7 Programming Languages as Tools, Programming Paradigms, Types of Languages

1.8 Converting Pseudo-code to Programs

2. Introduction to Python [10 Hrs.]

2.1 History, Feature of Python, Setting-up Path, Working with Python Interpreter, Basic Syntax, Variable and Data Types, Operators

2.2 Conditional Statements - If, If-Else, Nested if-else, Examples

2.3 Looping - For, While, Nested Loops, Examples

2.4 Control Statements - Break, Continue, Pass

2.5 String Manipulation - Accessing String, Basic Operations, String Slices, Function and Methods, Examples

2.6 Lists - Introduction, Accessing List, Operations, Working with Lists, Function and Methods

2.7 Tuple - Introduction, Accessing Tuples, Operations Working, Function and Methods, Examples

2.8 Dictionaries - Introduction, Accessing Values in Dictionaries, Working with Dictionaries, Properties, Function, Examples

2.9 Functions - Defining a Function, Calling a Function, Types of Function, Function Arguments, Anonymous Function, Global and Local Variable, Examples

3. Classes, Objects and Inheritance [8 Hrs.]

3.1 Classes and Objects

3.1.1 Classes as User Defined Data Type

3.1.2 Objects as Instances of Classes

3.1.3 Creating Class and Objects

3.1.4 Creating Objects by Passing Values

3.1.5 Variables and Methods in a Class

3.2 Inheritance
- 3.2.1 Single Inheritance
- 3.2.2 Multilevel
- 3.2.3 Multiple Inheritance
- 3.2.4 Hybrid Inheritance
- 3.2.5 Hierarchical Inheritance
- 3.2.6 IS-A Relationship and HAS-A Relationship

4. Modules and Packages [7 Hrs.]

4.1 Built-in Modules
- 4.1.1 Importing Modules in Python Program
- 4.1.2 Working with Random Modules
- 4.1.3 E.g. - Built-ins, Time, Date Time, Calendar, Sys, etc.

4.2 User Defined Functions
- 4.2.1 Structure of Python Modules

4.3 Packages
- 4.3.1 Predefined Packages
- 4.3.2 User Defined Packages

■■■

Contents ...

1. Introduction to Problem Solving	**1.1 – 1.40**
2. Introduction to Python	**2.1 – 2.84**
3. Classes, Objects and Inheritance	**3.1 – 3.30**
4. Modules and Packages	**4.1 – 4.32**

■■■

CHAPTER 1

Introduction to Problem Solving

Objectives ...

After learning this chapter you will be able:

- To learn Basic Concepts of Problem and Problem Solving
- To study Algorithms with Examples
- To understand Flowcharts with its Notations and Examples
- To learn Programming Languages with its Types
- To study Compilation Process with Linker, Loader, Interpreters etc.
- To learn Errors with its Types
- To understand Good Programming Practices

1.1 INTRODUCTION

- Problem solving is a central aspect of computer science, involving a deep understanding of both the nature of a problem and the steps required to solve it.
- A problem is defined as, a situation or a question that present uncertainty, perplexity or difficulty.
- Problem solving is a creative and innovative process for finding solutions to problems and it need to be followed systematically.
- Problem solving aspects mainly focuses on:
  1. **Problem definition** in which we must specify what must be done and not on how is to be done. In problem definition we write down the problem statement to solve the problem.
  2. In **problem solving** we must think over the possible solutions to the problem and list out the tasks required to move towards the solution of the problem.
  3. **Use of specific examples** is a useful technique when we are stuck during problem solving stage.
  4. **Similarities among the problems** is another aspect of problem solving refers similar problems that are solved in past. By finding out the similarities among the problems, we may get some hints and solutions to solve the current problem in hand.

5. In **working backwards from the solution**, we must try to make analysis backward i.e., from the solution to the starting point. Which help us to analyze the path from a problem to a solution.
6. There are a number of **General Problem Solving** strategies that are repeatedly used in computing science for problem solving. Some of the are explained below:
    - **Divide and Conquer** is the most widely strategy for problem solving, suitable for problems where the sub-problems are of the same type as the original problem. It partitions the problem into smaller sub-parts, finds solutions for the sub-parts, and then combines the solutions for the parts into a solution for the whole.
    - **Dynamic Programming** problem solving is useful when we can build-up the solution as a sequence of the intermediate steps.
    - **General Search, Back Tracking and Branch-and-Bound** strategies are variants of the basic dynamic programming strategy, based on guiding a computation in such a way that minimum amount of efforts is required in exploring the solutions.

1.2 PROBLEM SOLVING USING COMPUTERS

- Computer problem solving can be summed up in one word – it is demanding! Computer problem solving is an intricate process requiring much though, think, systematic planning, logical precision, persistence and attention to detail.
- A computer is a very powerful and versatile electronic machine capable of performing many different tasks, yet it has no intelligence or thinking power.
- A computer performs many tasks exactly in the same manner as it is told to do. Furthermore, we have to give step-by-step instructions to the computer for solving the problem.
- In order to solve a problem using a computer, we must first express the procedure used to solve the problem in terms of computer instructions.
- A computer program is just a collection of these instructions, that are necessary to solve a specific problem. The method that is used to solve the problem known as algorithm.
- A step-by-step procedure used to solve a problem is called as algorithm. Algorithms clearly outlines the procedure to solve a given problem.
- The step-by-step instructions/rules of an algorithm to solve a given problem are simple, easy to understand the solution of the problem.
- Computer program for any given problem is generally formulated through flowcharts. Flowchart is a diagrammatic/pictorial notation for representing algorithms
- Flowcharts or flow diagram shows the control flow of algorithms using diagrams in a visual/graphical manner.
- Pseudocode provides a simplified algorithm representation, enabling problems and solutions to be more easily understood.
- Problem solving is defined as, a creative and systematic process which gives solution of the given problem.

- To obtain the computer solution to a problem once we have the program we usually have to supply the program with **input** or data.
- The program then takes this input and **manipulates/processes** it according to its instructions and eventually produces an **output/result** which represents the computer solution to the problem.

1.3 PROBLEM SOLVING STEPS

- A problem-solving technique follows certain steps in finding the solution to the problem. Let us look the steps one by one:
  - **Step 1 : Identify the Problem:** Identifying the problem is the first step in solving the problem. Problem identification is very essential before solving any problem.
  - **Step 2 : Understand the Problem:** before solving any problem, it is important to understand it.
  - **Step 3 : Analyzing the Problem:** After understanding the problem to be solved, we look different ways of solving the problem. Here we search an appropriate solution to the problem under consideration. The result of this step is a broad/detailed overview of the sequence of operations that are to be carries out to solve the given problem.
  - **Step 4 : Developing the Solution:** Here, the overview of the sequence of operations that was the result of analysis step is expanded to form a detailed step by step solution to the problem under consideration.
  - **Step 5 : Evaluate the Solution:** When the solution is evaluated then check whether the solution is correct or not. Also check it satisfies the requirements or not.

1.4 ALGORITHMS

- A finite set of steps that must be followed to solve any problem is called an algorithm. An algorithm is a set of steps designed to solve a problem or accomplish a task.
- An algorithm is a step-by-step procedure or a set of rules designed to perform a specific task or solve a particular problem.

1.4.1 Definition of an Algorithm

- An algorithm can be defined as, a step-by-step procedure that provides solution to a given problem. **OR**
- An algorithm is a process/procedure consisting of a finite number of precisely defined steps for solving a problem.

1.4.2 Characteristics of an Algorithm

- Every algorithm should follow following characteristics or criteria:
  1. **Input:** An algorithm should accept zero or more inputs. Random number generation is an example of zero input.
  2. **Output:** An algorithm must produce at least one result/output.
  3. **Definiteness:** Each step is an algorithm must be clear and unambiguous. This helps the one who is following the steps to take a definite action.

4. **Finiteness:** An algorithm must halt. Hence, it must terminate after a finite number of steps.
5. **Effectiveness:** Each instruction of an algorithm must be feasible. It should be possible to carry out the instructions of the algorithm.

1.4.3 How to Write an Algorithm?

- There are no well-defined standards for writing algorithms. We write algorithms in a step-by-step manner, but it is not always the case.
- Algorithm writing is a process and is executed after the problem domain is well-defined. That is, we should know the problem domain, for which we are designing a solution.
- Before starting writing an algorithm the user must obtain information about the problem that needs to be solved.
- Following example shows design of an algorithm to add two numbers and display the result.

  Step 1 : Start
  Step 2 : Declare three integers x, y and z
  Step 3 : Define values of x and y
  Step 4 : Add values of x and y
  Step 5 : Store output of Step 4 to z
  Step 6 : Print z
  Step 7 : Stop
- Alternatively, the algorithm can be written as follows:

  Step 1 : Start Add
  Step 2 : Get values of x and y
  Step 3 : $z \leftarrow x + y$
  Step 4 : display z
  Step 5 : Stop

1.4.4 Examples for Algorithms

- Let us see some simple examples on algorithms for problem solving.

Example 1: To find the sum of two numbers.

Step 1 : Start
Step 2 : Read two numbers A and B
Step 3 : Calculate the sum of A and B and store it in SUM
Step 4 : Display the value of SUM
Step 5 : Stop

Example 2: To find area and perimeter of rectangle.

Step 1 : Start
Step 2 : Input Side Length and Breadth say L, B
Step 3 : Area = $L \times B$
Step 4 : Perimeter = $2 \times (L + B)$
Step 5 : Display Area, Perimeter
Step 6 : Stop

Example 3: To convert temperature from Fahrenheit to Celsius.

Step 1 : Start

Step 2 : Input temperature in Fahrenheit say F

Step 3 : C = 5.0/9.0 (F – 32)

Step 4 : Display Temperature in Celsius C

Step 5 : Stop

Example 4: To find the smallest of two numbers.

Step 1 : Start

Step 2 : Input two numbers say num1, num2

Step 3 : IF num1 < num2 then
print smallest is num1
else
print smallest is num2
endif

Step 4 : Stop

Example 5: To find the sum of first n natural numbers (1 + 2 + 3 + ….. + n).

Step 1 : Start

Step 2 : Accept the n term.

Step 3 : Let, i=1, SUM=0

Step 4 : Let, SUM = SUM + i

Step 5 : Let i = i + 1 (or Increment i by 1)

Step 6 : if (i <= n) then goto Step 4

Step 7 : Display SUM

Step 8 : Stop

1.4.5 Advantages and Disadvantages of Algorithms

- Algorithms are the building blocks of computer programs, enabling computers to process data make decisions and perform task/operations.

Advantages of Algorithms:

1. An algorithm is a step-wise representation of a solution to a given problem, which makes it easy and simple to understand.
2. Algorithms provide a way to solve problems efficiently, often with the least amount of resources.
3. Well-designed algorithms can solve complex problems quickly.
4. Algorithms provide a systematic approach to problem-solving, ensuring consistent and reliable results.
5. An algorithm is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge.
6. Every step in an algorithm has its own logical sequence so it is easy to debug.
7. An algorithm facilitates program development by acting as a design document or a blueprint of a given problem solution.

Disadvantages/Limitations of Algorithms:

1. Writing an algorithm takes a long time so it is time consuming.
2. In large algorithms, the flow of program control (branching and looping) becomes difficult to show and track.
3. Understanding complex logic through algorithms can be very difficult.

1.5 FLOWCHARTS

- A flowchart is a graphical/pictorial representation of the logic for solving a problem. A flowchart is a type of diagram (graphical or symbolic) that represents an algorithm or process.

1.5.1 Definition of Flowchart

- A flowchart is commonly used programming tool for problem solving. Flowchart is often considered as a blueprint of a design used for solving a specific problem.
- A flowchart is defined as, "a step-by-step diagrammatic representation of the logical steps to solve a given problem". **OR**
- Flowchart is "a visual or graphical representation of an algorithm".

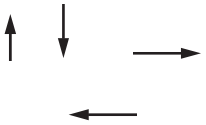
1.5.2 Symbols/Notations in Flowcharts

- A flowchart is a chart that graphically represent detailed series of steps (algorithm or control flow) needed/required for solve a problem.
- Flowcharts are constructed or designed by using special geometrical symbols. Each symbol is used to indicate specific operation.
- Flowchart symbols shows the pictorial representation of the sequence of operations to be performed to solve a problem.
- The symbols that we make use while drawing flowcharts as given below are as per conventions followed by International Standard Organization (ISO).

Table 1.1

Sr. No.	Name of Symbol	Symbol	Description
1.	Oval		**Terminal:** Used to represent start and end (stop) of flowchart.
2.	Parallelogram		**Input/Output:** Used for input and output operation.
3.	Rectangle		**Processing:** Used for arithmetic operations and data-manipulations.

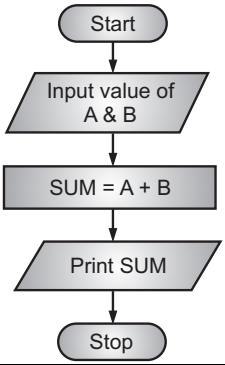
contd. …

4.	Dimond		**Decision:** Used to represent the operation in which there are two/three alternatives, true and false etc.
5.	Directed Lines		**Flow Lines:** Used to indicate direction the flow of logic by connecting symbols.
6.	Small Circle		**Connector:** Used to join the different parts of a flowchart.

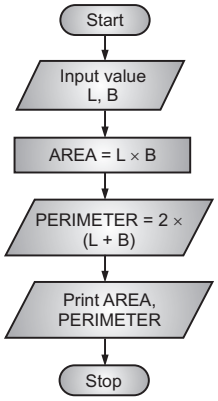
1.5.3 Examples for Flowcharts

- Let us see some simple examples of flowchart.

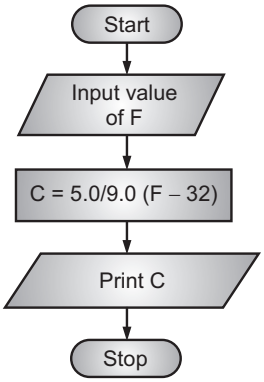
Example 6: To find the sum of two numbers.



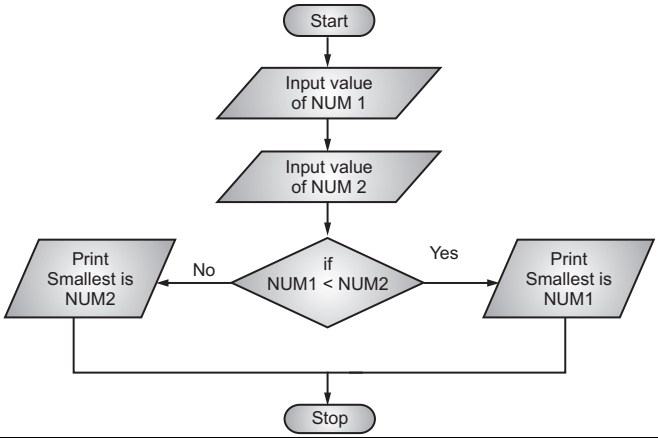
Example 7: To find area and perimeter of rectangle.



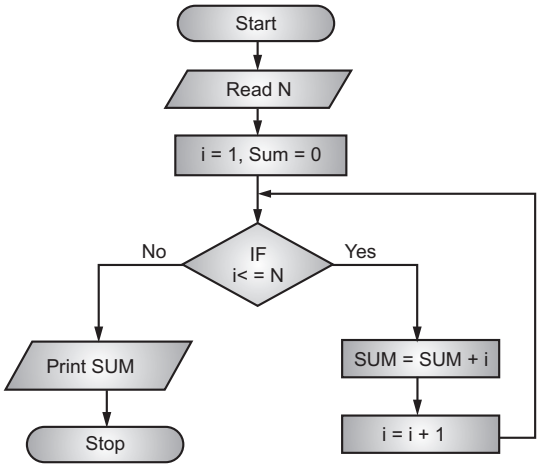
Example 8: To convert temperature from Fahrenheit to Celsius.



Example 9: To find the smallest of two numbers.



Example 10: To find the sum of first n natural numbers (1+2+3+.....+n).



1.5.4 Advantages and Disadvantages of Flowcharts

Advantages of Flowcharts:

1. **Better Communication:** A flowchart can be used as a better way of communication of the logic of a system and steps involve in the solution.
2. **Effective Analysis:** A flowchart of a problem can be used for effective analysis of the problem.
3. **Documentation of Program/System:** Program flowcharts are a vital part of a good program documentation. Program document is used for various purposes like knowing the components in the program, complexity of the program etc.
4. **Coding of the Program:** Any design of solution of a problem is finally converted into computer program. Writing code becomes easy by referring the flowchart of the solution.
5. **Systematic Debugging:** A flowchart is very helpful in detecting, locating and removing mistakes (bugs) in a program in a systematic manner, because programmers find it easier to follow the logic of the program in flowchart form.
6. **Efficient Program Maintenance:** Once, a program is developed and becomes operational it needs time to time maintenance. With help of flowchart maintenance become easier.
7. **Simplify the Logic:** As flowchart provides the pictorial representation of the steps; therefore, it simplifies the logic and subsequent steps.

Disadvantages of Flowcharts:

1. **Complex Logic:** Sometimes for large problems, the program logic is quite complicated. In that case, flow chart becomes complex and clumsy.
2. **Time Consuming:** Flowcharts are very time consuming and laborious to draw with proper symbols and spacing, especially for large complex programs.

3. **Alterations and Modifications:** If alterations are required in the flow chart, it may require redrawing of it completely.
4. **Reproduction:** As the flow chart symbols cannot be typed, reproduction of flow chart becomes a problem.
5. **No Standard:** There are no standards determining the amount of detail that should be included in the flowchart.

1.5.5 Difference between Algorithm and Flowchart

- Table 1.2 compares algorithm and flowcharts:

Table 1.2

Sr. No.	Algorithm	Flowchart
1.	Algorithm is method of representing the step-by-step logical procedure for solving a problem.	Flowchart is diagrammatic/pictorial representation of an algorithm.
2.	It contains simple English statements.	It is constructed using different types of symbols.
3.	It is difficult to show branching and looping.	It is easy to show branching and looping.
4.	As compared to flowchart, less time consuming and less laborious.	Flowcharts are very time consuming and laborious to draw with proper symbols.
5.	Modification in algorithm is easy.	Modification in flowchart may require redrawing.
6.	Debugging is difficult in complex problems.	Debugging is easy as comparative to algorithm for complex problems.

Examples of Algorithm and Flowchart:

Example 11: Algorithm and flowchart to find area and perimeter of circle.

R : Radius of Circle.

Area : Area of Circle.

Perimeter: Perimeter of Circle.

Algorithm:

Step 1 : Start

Step 2 : Input Radius of Circle say R

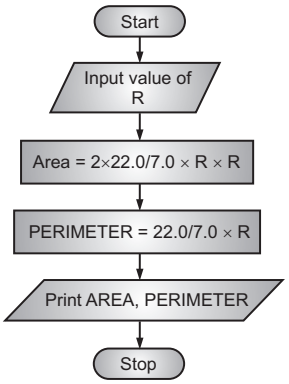
Step 3 : Area = 22.0/7.0 × R × R

Step 4 : Perimeter = 2 × 22.0/7.0 × R

Step 5 : Display Area, Perimeter

Step 6 : Stop

Flowchart:



Example 12: Algorithm and flowchart to find simple interest.

P : Principle Amount.

N : Time in Years.

R : % Annual Rate of Interest.

SI : Simple Interest.

Algorithm:

Step 1 : Start

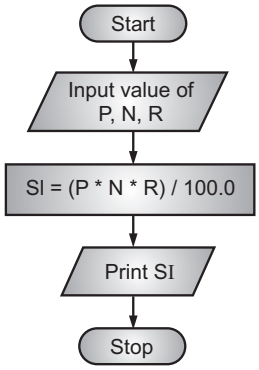
Step 2 : Input value of P, N, R

Step 3 : SI = (P × N × R)/100.0

Step 4 : Print SI

Step 5 : Stop

Flowchart:



Example 13: Algorithm and flowchart to find compound interest.

P : Principle Amount.

N : Time in Years.

R : % Annual Rate of Interest.

CI : Compound Interest.

Algorithm:

Step 1 : Start

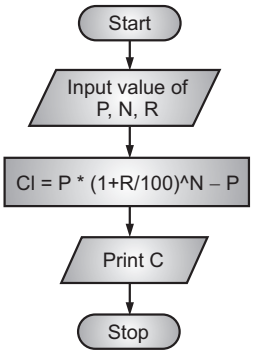
Step 2 : Input value of P, N, R, C

Step 3 : CI = P (1 + R/100)^{N} − P

Step 4 : Print CI

Step 5 : Stop

Flowchart:



Example 14: Algorithm and flowchart to swap two numbers without using temporary variable.

Algorithm:

Step 1 : Start

Step 2 : Input Two Numbers Say A, B

Step 3 : Display Before Swap Values A, B

Step 4 : A = A + B

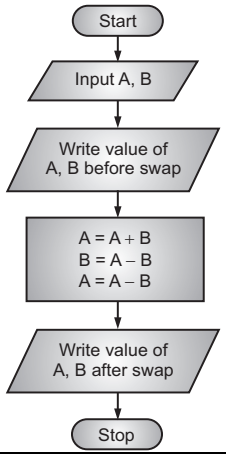
Step 5 : B = A − B

Step 6 : A = A − B

Step 7 : Display after Swap Values A, B

Step 8 : Stop

Flowchart:

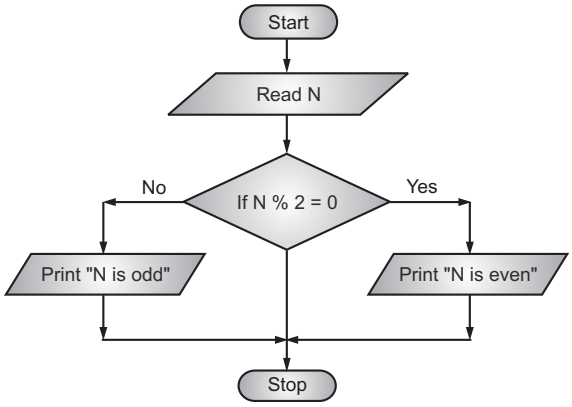


Example 15: Algorithm and flowchart to check given number is even or odd.

Algorithm:

Step 1 : Start
Step 2 : Read Number N
Step 3 : if N%2 = = 0 Then
Print "N is even"
Step 4 : else
Print "N is an odd"
Step 5 : Stop

Flowchart:

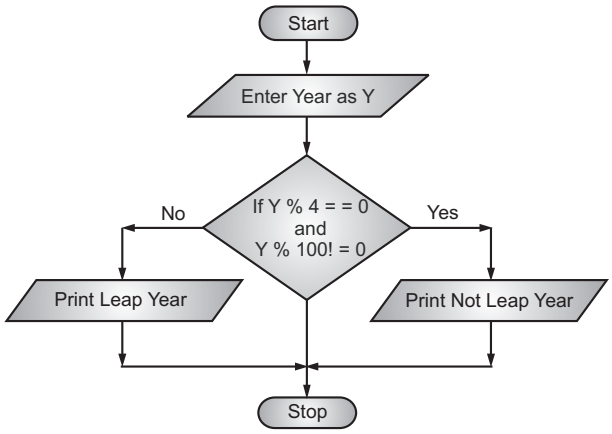


Example 16: Algorithm and flowchart to check year is leap year or not.

Algorithm:

Step 1 : Start
Step 2 : Input Year as Y
Step 3 : Check if (Y%4==0 AND Y%100 != 0 OR Y%400 == 0) Then goto Step 4 Else goto Step 5
Step 4 : Print "Leap Year" and goto Step 6
Step 5 : Print "Not Leap Year"
Step 6 : Stop

Flowchart:



Example 17: Algorithm and flowchart to find largest of three numbers.

Algorithm:

Step 1 : Start
Step 2 : Read variables a, b and c.
Step 3 : if a>b and a>c then
Print a is the largest number
else if b>c
Print b is the largest number
Else
Print c is the greatest number
Step 4 : Stop

Flowchart:

