Introduction to Programming

# What is programming?

***Programming is the process of designing and creating sets of instructions that a computer can execute to perform specific tasks or solve problems***. These instructions are written using programming languages, such as Python, JavaScript, Java, and many others.

Programmers write code by ***breaking down tasks into smaller, manageable steps*** that a computer can understand and execute. This involves ***defining algorithms, which are sequences*** of steps needed to accomplish a particular task. Programmers use various tools and environments to write, test, and debug their code to ensure it works correctly.

From smartphones to home appliances, programming is behind the functionality of devices we use daily. Software applications and user interfaces are products of programming, shaping how we communicate, work, and entertain ourselves.

# Fundamentals of Programming

Programming fundamentals are the essential concepts and principles that form the foundation of computer programming. Here are some key elements:

1. **Variables and Data Types:** Variables are containers for storing data. Data types specify the kind of data that variables can hold, such as integers, floating-point numbers, strings, boolean values, etc.
2. **Operators:** Operators are symbols used to perform operations on variables and values. Examples include arithmetic operators (+, -, \*, /), comparison operators (>, <, ==, !=), logical operators (AND, OR, NOT), etc.
3. **Control Structures:** These include conditional statements (if-else, switch-case) for decision-making and loops (for, while, do-while) for repetitive tasks.
4. **Functions/Methods:** Functions or methods are blocks of code that perform a specific task. They allow you to modularize code and make it reusable.
5. **Data Structures:** These are ways of organizing and storing data efficiently. Examples include arrays, lists, stacks, queues, trees, and graphs.
6. **Algorithms:** Algorithms are step-by-step procedures or formulas for solving problems. They are a sequence of well-defined instructions to perform a task or solve a problem.
7. **Input/Output:** Handling input from users or external sources and producing output is fundamental. This involves reading data from keyboards, files, or networks and displaying results to screens, files, or other devices.
8. **Error Handling:** Dealing with errors and exceptions that may occur during program execution is crucial. This involves using try-catch blocks or other mechanisms to handle unexpected situations gracefully.
9. **Debugging and Testing:** Debugging involves identifying and fixing errors or bugs in the code. Testing ensures that the program works as expected by running various test cases.
10. **Software Development Life Cycle (SDLC):** Understanding the different phases of software development, including requirements gathering, design, implementation, testing, deployment, and maintenance.

These fundamentals are applicable across various programming languages and form the basis for creating software applications. Mastering these concepts helps in becoming a proficient programmer and allows you to adapt to different languages and frameworks more easily.

# Practice questions

1. Write an algorithm in pseudo-code to get a name from the user and display a greeting to the user.

Solution –

**Algorithm GreetUser**

**Input:**

name (user's name)

**Output:**

greeting message

**Steps:**

START

1. Display "Please enter your name:"

2. Read name

3. Display "Hello, Good Morning " + name

STOP

2. Write a simple algorithm in pseudo-code that takes two numbers as input and displays their sum, difference, product, and division.

**Algorithm CalculateOperations**

**Input:**

num1, num2 (two numbers)

**Output:**

sum, difference, product, division (results)

**Steps:**

START

1. Read num1

2. Read num2

3. sum = num1 + num2

4. difference = num1 - num2

5. product = num1 \* num2

6. If num2 is not equal to 0, then:

division = num1 / num2

Else:

Display "Cannot divide by zero."

7. Display sum, difference, product, division

STOP

3. Write an algorithm in pseudo-code that takes two numbers as input and displays the larger of the two.

**Algorithm FindLargestNumber**

**Input:**

num1, num2 (two numbers)

**Output:**

largest (the larger number)

**Steps:**

START

1. Read num1

2. Read num2

3. If num1 > num2, then:

largest = num1

Else if num2 > num1, then:

largest = num2

Else:

Display "Both numbers are equal."

STOP

4. Display "The largest number is: " + largest

STOP

4. Write an algorithm in pseudo-code that takes a number as input and displays whether it's positive, negative, or zero.

**Algorithm CheckNumber**

**Input:**

num (a number)

**Output:**

message (indicating if the number is positive, negative, or zero)

**Steps:**

START

1. Read num

2. If num > 0, then:

Display "The number is positive."

Else if num < 0, then:

Display "The number is negative."

Else:

Display "The number is zero."

STOP

5. Write an algorithm in pseudo-code that takes three numbers as input from the user and calculates their average value.

**Algorithm CalculateAverage**

**Input:**

num1, num2, num3 (three numbers)

**Output:**

average (the average value of the three numbers)

**Steps:**

START

1. Read num1

2. Read num2

3. Read num3

4. average = (num1 + num2 + num3) / 3

5. Display "The average value of the three numbers is: " + average

STOP

6. Write an algorithm in pseudo-code that converts temperature from Celsius to Fahrenheit.

**Algorithm CelsiusToFahrenheit**

**Input:**

celsius (temperature in Celsius)

**Output:**

fahrenheit (temperature in Fahrenheit)

**Steps:**

START

1. Read celsius

2. fahrenheit = (celsius \* 9/5) + 32

3. Display "Temperature in Fahrenheit: " + fahrenheit

STOP

7. Write an algorithm in pseudo-code that calculates simple interest based on the principal amount, rate, and time.

**Algorithm CalculateSimpleInterest**

**Input:**

principalAmount (the principal amount)

rate (the rate of interest per year)

time (time period in years)

**Output:**

simpleInterest (the calculated simple interest)

**Steps:**

START

1. Read principalAmount

2. Read rate

3. Read time

4. simpleInterest = (principalAmount \* rate \* time) / 100

5. Display "Simple Interest: " + simpleInterest

STOP

8. Write an algorithm in pseudo-code that takes a number as input and displays the sum of its square and its cube.

**Algorithm SumOfSquareAndCube**

**Input:**

num (a number)

**Output:**

sumOfSquareAndCube (sum of the square and cube of the number)

**Steps:**

START

1. Read num

2. square = num \* num

3. cube = num \* num \* num

4. sumOfSquareAndCube = square + cube

5. Display "Sum of square and cube: " + sumOfSquareAndCube

STOP

9. Write an algorithm in pseudo-code that calculates the ticket price for a movie based on the age of the customer as per the following criteria –

| **Age** | **Price** |
| --- | --- |
| Children (age <= 12) | $5 |
| Adults (age > 12 and <= 65) | $10 |
| Seniors (age > 65) | $7 |

**Algorithm CalculateTicketPrice**

**Input:**

age (customer's age)

**Output:**

ticketPrice (price of the movie ticket)

**Steps:**

START

1. Read age

2. If age <= 12, then:

ticketPrice = $5

Else if age > 12 and age <= 65, then:

ticketPrice = $10

Else:

ticketPrice = $7

3. Display "The ticket price for the customer is: $" + ticketPrice

STOP

10. Write an algorithm that determines if a given year is a leap year or not. Leap years are divisible by 4, but if they are divisible by 100, they must also be divisible by 400.

**Algorithm CheckLeapYear**

**Input:**

year (a year)

**Output:**

isLeapYear (boolean value indicating if it's a leap year or not)

**Steps:**

START

1. Read year

2. If (year is divisible by 4) and (year is not divisible by 100), then:

isLeapYear = true

Else if (year is divisible by 100) and (year is divisible by 400), then:

isLeapYear = true

Else:

isLeapYear = false

3. Display "Is the year a leap year? " + isLeapYear

STOP

# Practice Questions DIY

1. Write an algorithm in pseudo-code to get an age of a person as an input and display whether the person is eligible to vote or not. (A person with age more than 18 is considered to be eligible to vote.)

2. Write an algorithm in pseudo-code to get an age of a person as an input and display whether the person is a child or teenager or an adult.

3. Write an algorithm in pseudo-code to get a number indicating the student’s score as an input and determine and display grade based on the following criteria –

| **Score** | **Grade** |
| --- | --- |
| 90 and above | A |
| 80-89 | B |
| 70-79 | C |
| 60-69 | D |
| Below 60 | F |

4. Write an algorithm in pseudo-code to get a character as input from the user and determine and display if its vowel or a consonant.

5. Write an algorithm in pseudo-code to convert temperature from Fahrenheit to Celsius.