1. Introduction to Python and its features

Ans. Python is a high level, interpreted programming language.

Benefits: -

Versatile: Suitable for web development, data analysis, machine learning, automation, and more.

Large community: A vast ecosystem of libraries and frameworks.

Cross platform: Runs on various operating systems

1. History and evolution of python

Python was founded by Guido Van Russom in 1991 and managed by Python software development

1. Advantages of python over other programming languages
   * Readability and Simplicity
   * Clean Syntax
   * Reduced Code
   * Versatility and Wide Applications
   * Extensive Libraries and Frameworks
   * Cross-Platform Compatibility
   * Interpreted Language
   * Object-Oriented Programming
   * Modules
2. Writing and executing my first python program

Num=int(input(“Enter a number: ))

Print(num)

1. Understanding PEP-8 guidelines

PEP 8 is the Python Enhancement Proposal that provides guidelines and best practices for writing Python code.

It covers topics like naming conventions, code layout, and documentation.

1. Indentation, comments, and naming conventions in Python.

Indentation: -

Indentation in Python refers to the use of whitespace (spaces or tabs) at the beginning of a line of code to define code blocks.

Comments: -

Comments are used to explain the purpose of specific code sections, making your code more readable and understandable for both yourself and others.

Types of Comments in Python:

Single-line Comments

Multi-line comments

Naming conventions: -

Python has a set of naming conventions, known as **PEP 8**, to ensure consistency and readability in your code.

1. Writing readable and maintainable code.

A = 10

B = 20

Print(A+B)

1. Understanding data types: integers, floats, strings, lists, tuples, dictionaries, sets.

**Integers (int):** Whole numbers without a decimal point.

**Float:** Numbers with a decimal point.

**String:** A sequence of characters enclosed in single or double quotes.

**Lists:** Ordered collections of items, mutable (can be changed).

**Tuples:** Ordered collections of items, immutable (cannot be changed).

**Dictionaries:** Unordered collections of key-value pairs

**Sets:** Unordered collections of unique items.

1. Python variables and memory allocation.

Variables stores data values and reference to memory location dynamically allocated during run-time

1. Python operators: arithmetic, comparison, logical, bitwise.

**Arithmetic Operators:**

**Addition (+):** Adds two numbers.

**Subtraction (-):** Subtracts one number from another.

**Multiplication (\*):** Multiplies two numbers.

**Division (/):** Divides one number by another.

**Modulo (%):** Returns the remainder of a division.

**Comparison Operators:**

**Equal to (==):** Checks if two values are equal.

**Not Equal to (!=):** Checks if two values are not equal.

**Greater Than (>):** Checks if one value is greater than another.

**Less Than (<):** Checks if one value is less than another.

**Greater Than or Equal To (>=):** Checks if one value is greater than or equal to another.

**Less Than or Equal To (<=):** Checks if one value is less than or equal to another.

**3. Logical Operators:**

**AND (and):** Returns True if both operands are True.

**OR (or):** Returns True if at least one operand is True.

**4. Bitwise Operators:**

**Bitwise AND (&):** Performs bitwise AND on each bit of the operands.

**Bitwise OR (|):** Performs bitwise OR on each bit of the operands.

**Bitwise XOR (^):** Performs bitwise XOR on each bit of the operands.

**Left Shift (<<):** Shifts bits to the left by a specified number of positions.

**Right Shift (>>):** Shifts bits to the right by a specified number of positions.

1. Introduction to conditional statements: if, else, elif.

**If: -** The if statement is used to execute a block of code if a certain condition is true.

**Else:** - The else statement is used to execute a block of code if the condition in the if statement is false.

**Elif:** - The elif statement allows you to check multiple conditions sequentially. It's a combination of else and if.

1. Nested if-else conditions.

Nested if-else conditions allow you to create more complex decision-making structures. This involves placing if-else statements within other if-else statements.

1. Introduction to for and while loops

For - A for loop is used to iterate over a sequence of values, such as a list, tuple, string, or range of numbers.

While - A while loop repeatedly executes a block of code as long as a given condition is true.

1. How loops work in Python.

Loops are essential tools in programming that allow you to execute a block of code repeatedly. Python offers two primary types of loops: for loops and while loops.

1. Using loops with collections (lists, tuples, etc.).

fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

1. Understanding the role of break, continue, and pass in Python loops.

**Break:** Terminates the loop immediately.

**Continue:** Skips the current iteration and moves to the next one.

**Pass:** Does nothing. It's a placeholder to avoid syntax errors.

1. Understanding how to access and manipulate strings

**Accessing characters:**

**Indexing:** Use square brackets [] to access individual characters.

**Slicing:** Extract a substring using [start:end]

**String manipulation:**

**len():** Returns the length of the string**.**

**upper():** Converts all characters to uppercase.

**lower():** Converts all characters to lowercase.

**join():** Joins elements of a list or tuple into a string.

**find():** Returns the index of the first occurrence of a substring.

**count():** Counts the number of occurrences of a substring.

1. Basic operations: concatenation, repetition, string methods

**Concatenation:** Joining of 2 strings using + operator

**Repetition:** To repeat the string as many times you want

**String methods:**

**len():** Returns the length of the string**.**

**upper():** Converts all characters to uppercase.

**lower():** Converts all characters to lowercase.

**join():** Joins elements of a list or tuple into a string.

**find():** Returns the index of the first occurrence of a substring.

**count():** Counts the number of occurrences of a substring.

1. String slicing.

String slicing is a powerful technique in Python that allows you to extract specific portions of a string. It's done using the slicing operator [:]