Name: Paramprakash Makwana

Module 13) Python Fundamentals

1.1) Introduction to Python and its Features (simple, high-level, interpreted language).

Ans:- Python is a simple, high-level, interpreted language known for easy syntax and readability. It is widely used for beginners and professionals.

1.2) History and evolution of Python.

Ans:- Python was created by Guido van Rossum in 1991. It has evolved with regular updates and strong community support.

1.3) Advantages of using Python over other programming languages.

Ans:- Python is easy to learn, has large libraries, and supports multiple domains like web, data science, and AI.

1.4) Installing Python and setting up the development environment (Anaconda, PyCharm, or VS Code).

Ans:- Python can be installed from python.org and used with IDEs like Anaconda, PyCharm, or VS Code for development.

1.5) Writing and executing your first Python program.

Ans:- A basic Python program like print("Hello World") is written and executed to verify installation.

1.6) Understanding Python’s PEP 8 guidelines.

Ans:- PEP 8 provides coding style guidelines to make Python code readable and consistent.

1.7) Indentation, comments, and naming conventions in Python.

Ans:- Python uses indentation to define blocks, comments for explanation, and clear naming conventions for variables.

1.8) Writing readable and maintainable code.

Ans:- Readable and maintainable code is achieved by proper formatting, meaningful names, and comments.

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

Ans:- Python supports data types like integers, floats, strings, lists, tuples, dictionaries, and sets.

1.10) Python variables and memory allocation.

Ans:- Variables in Python store references to objects, and memory is managed automatically.

1.11) Python operators: arithmetic, comparison, logical, bitwise.

Ans:- Python operators include arithmetic, comparison, logical, and bitwise operators for calculations and conditions.

1.12) Introduction to conditional statements: if, else, elif.

Ans:- Conditional statements like if, else, and elif are used to make decisions in a program.

1.13) Nested if-else conditions.

Ans:- Nested if-else allows one condition to be checked inside another condition.

1.14) Introduction to for and while loops.

Ans:- for and while loops are used to repeat a block of code multiple times.

1.15) How loops work in Python.

Ans:- Loops work by repeatedly executing code until a condition becomes false.

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

Ans:- Loops can iterate over collections like lists, tuples, strings, and dictionaries.

1.17) Understanding how generators work in Python.

Ans:- Generators produce values one at a time using the yield keyword, saving memory.

1.18) Difference between yield and return.

Ans:- return sends back a value and ends a function, while yield returns values one by one.

1.19) Understanding iterators and creating custom iterators.

Ans:- Iterators allow traversal through elements, and custom iterators are created using __iter__() and __next__().

1.20) Defining and calling functions in Python.

Ans:- Functions are defined using def and are called to perform specific tasks.

1.21) Function arguments (positional, keyword, default).

Ans:- Python supports positional, keyword, and default arguments in functions.

1.22) Scope of variables in Python.

Ans:- Variable scope defines where a variable can be accessed, such as local or global scope.

1.23) Built-in methods for strings, lists, etc.

Ans:- Python provides built-in methods for strings, lists, and other data types to manipulate data easily.

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

Ans:- break exits a loop, continue skips an iteration, and pass does nothing as a placeholder.

1.25) Understanding how to access and manipulate strings.

Ans:- Strings are accessed using indexing and manipulated using various built-in methods.

1.26) Basic operations: concatenation, repetition, string methods (upper(), lower(), etc.).

Ans:- String operations include concatenation, repetition, and methods like upper() and lower().

1.27) String slicing.

Ans:- String slicing extracts a part of a string using start, stop, and step values.

1.28) How functional programming works in Python.

Ans:- Functional programming in Python uses functions as first-class objects and avoids changing state.

1.29) Using map(), reduce(), and filter() functions for processing data.

Ans:- map(), filter(), and reduce() are used to process data efficiently in a functional style.

1.30) Introduction to closures and decorators.

Ans:- Closures remember values from their enclosing scope, and decorators modify function behavior without changing code.