**Python Fundamentals**

(Module-13)

**Assignment**



## SUBMITTED BY:

PRIYA PAREKH

## SUBMITTED TO:

CHINMAYEE MAM

BACKEND DEVELOPMENT TOPS TECHNOLOGIES

1. Introduction to Python Theory

**Theory:**

**Question-1:** **Introduction to Python and its Features.**

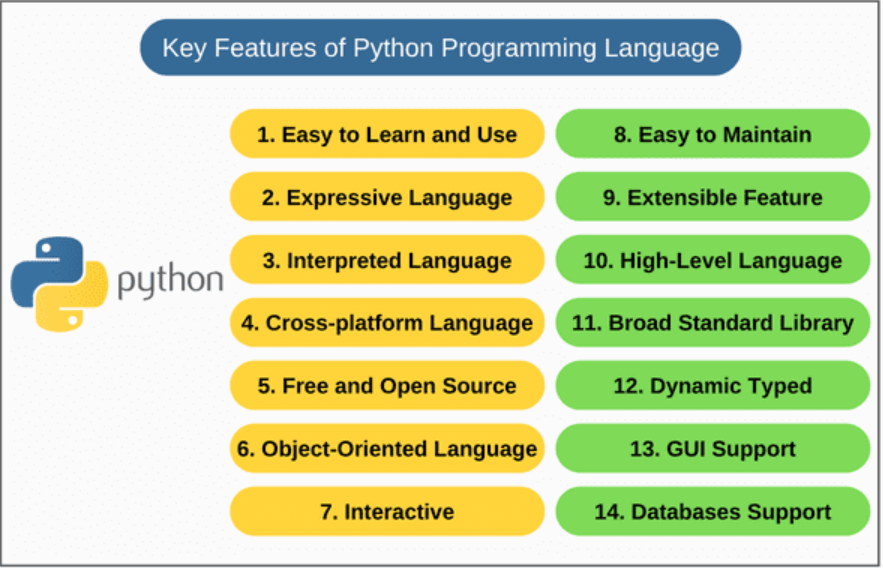
**Answer:**

Introduction:-

Python is a popular programming language. Guido van Rossum created it.

Python is a general-purpose, dynamically typed, high-level, compiled and interpreted, and purely object-oriented programming language that supports procedural, object-oriented, and functional programming.

Features of Python:-



**Question 2:** **History and evolution of Python.**

**Answer:**

**History:**

**Guido van Rossum created Python**. In the late 1980s, Guido van Rossum, a Dutch programmer, began working on Python while at the Centrum Wiskunde & Informatica (CWI) in the Netherlands. He wanted to create a successor to the **ABC programming language** that would be easy to read and efficient.

**In February 1991, the first public version of Python, version 0.9.0, was released.** This marked the official birth of **Python as an open-source project**. The language was named after the British comedy series "**Monty Python's Flying Circus**".

**Evolution:**

Python development has gone through several stages. **In January 1994, Python 1.0 was released as a usable and stable programming language.** This version included many of the features still present in Python today.

**From the 1990s to the 2000s**, Python gained popularity for its simplicity, readability, and versatility. **In October 2000, Python 2.0 was released**. Python 2.0 introduced list comprehensions, garbage collection, and support for Unicode.

In December 2008, Python 3.0 was released. Python 3.0 introduced several backward-incompatible changes to improve code readability and maintainability.

Throughout the 2010s, Python's popularity increased, particularly in fields like data science, machine learning, and web development. Its rich ecosystem of libraries and frameworks made it a favourite among developers.

**Question 3: Advantages of using Python over other programming languages.**

**Answer:**

Python is a widely used programming language that is utilized extensively in various disciplines such as web development, data analysis, artificial intelligence, and scientific computing. It has a number of benefits that make it a suitable option for different types of projects.

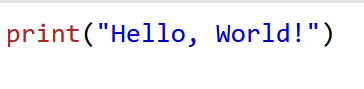
For a variety of reasons, a sizable community of programmers favours Python. Python is suggested as a programming language for many industries.

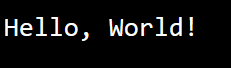
1. **Ease of Learning**: Simple and readable syntax, making it beginner-friendly.
2. **Versatility**: Used in web development, data science, AI/ML, automation, and more.
3. **Rich Libraries**: Extensive libraries like Numpy, Pandas, Tensor Flow, etc.
4. **Cross-Platform**: Write once, run anywhere (Windows, Mac, Linux).
5. **Community Support**: Large, active community for troubleshooting and resources.
6. **Rapid Development**: Faster prototyping due to concise code and built-in features.
7. **Integration**: Easy to integrate with other languages (e.g., C/C++, Java).
8. **Automation**: Great for scripting and automating repetitive tasks.

It’s a go-to choice for developers in many domains due to its flexibility and ecosystem.

In conclusion, Python is a versatile, high-performance, and easy-to-learn programming language that is widely used in a variety of fields. Its simplicity of usage and learning, versatility, performance, strong support for data analysis and scientific computing, large and active community, flexibility in deployment, support for object-oriented and functional programming, strong support for testing and debugging, and support for integration with other languages and systems make it a popular choice for developers and organizations.

**Question 4: Writing and executing your first Python program**

****

****

1. **Programming Style**

**Theory:**

**• Understanding Python’s PEP 8 guidelines.**

**• Indentation, comments, and naming conventions in Python.**

**• Writing readable and maintainable code.**

**Answer:**

**PEP 8** is the official style guide for writing Python code. It provides conventions to improve code readability and maintainability.

PEP Guidelines:

* Indentation: 4 spaces (no tabs).
* Line length: 79 characters.
* Blank lines to separate code blocks.
* Imports:
* One per line.
* Group: standard library, third-party, local imports.
* Spacing:
* Spaces around operators and after commas.
* No spaces inside parentheses/brackets.
* Comments: Clear, concise, meaningful.
* Avoid extraneous whitespaces.
* Consistent use of single or double quotes.
* Minimize nested blocks.
* Use if \_\_name\_\_ == "\_\_main\_\_".
* **Comments** explain the purpose of the code.
* **Variable naming** follows snake\_case.
* **Spacing** is consistent around operators.
* The code is simple and PEP 8 applicable.

3. Core Python Concepts

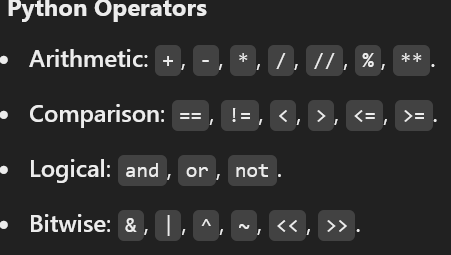
**Theory:**

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

**• Python variables and memory allocation.**

* **Variables**: Containers for storing data values, e.g., x = 10.
* **Dynamic Typing**: Python variables do not need explicit type declaration.
* **Memory Allocation**: Python uses references and garbage collection for memory management.

**• Python operators: arithmetic, comparison, logical, bitwise.**

****

*Practical Example 1: How does the Python code structure work?*

*Answer:*

* ***Variables and Constants****:*
* *These are used to store data values. Variables can hold data that may change, while constants store fixed values.*
* ***Expressions and Calculations****:*
* *Python allows performing calculations and operations using variables, constants, and operators.*
* ***Indentation****:*
* *Python uses indentation (usually 4 spaces) to define blocks of code, such as loops, conditionals, or nested logic. Proper indentation is mandatory.*
* ***Control Flow Statements****:*
* *Python includes control statements like if, else, while, and for to control the flow of the program.*
* ***Input and Output****:*
* *Python uses input() to take user input and print() to display output.*
* ***Code Execution****:*
* *Python programs execute line by line from top to bottom unless control statements alter the flow.*
* ***Comments****:*
* *Comments are used to explain code, starting with #. They are ignored by the Python interpreter.*

*Practical Example 2: How to create variables in Python?*

*Answer:*

*A variable is the name given to a memory location. A value-holding Python variable is also known as an identifier.*

* ***Assign a Value****:*
* *Use the = operator to assign a value to a variable. For example:*
  + *x = 5 creates a variable x and assigns it the value 5.*
* ***Dynamic Typing****:*
* *You don’t need to specify the data type of a variable. Python automatically determines the type based on the value assigned.*
* ***Naming Rules****:*
* *Variable names must start with a letter or an underscore (\_).*
* *They cannot start with a number or contain special characters.*
* *They are case-sensitive, so name and Name are different.*
* ***Reassignment****:*
* *Variables can be reassigned to new values, even of a different data type. For example:*
* *x = 5 (integer) can later be changed to x = "Hello" (string).*

*4. Conditional Statements*

**Theory:**

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

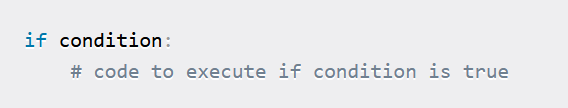
Answer:

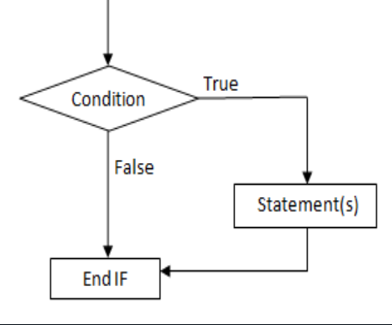
Conditional Statements are statements in Python that provide a choice for the control flow based on a condition

1. If

If the statement is only used to check the true part of the code.

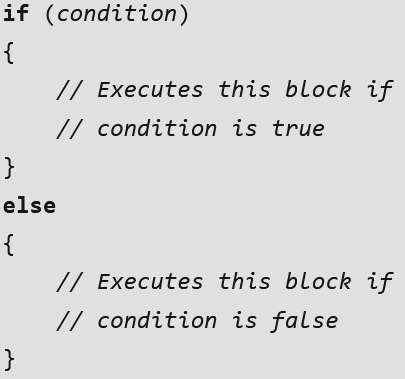
This is the simplest decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e. if a certain condition is true then a block of statements is executed otherwise not.

Syntax: 



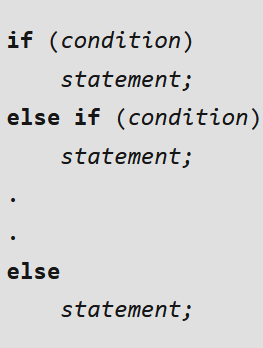
2)If…Else Statement

* In If…else statement if checks true part and else is used to check false part.
* You can’t give condition in else part.
* Else can’t be written without IF statement.



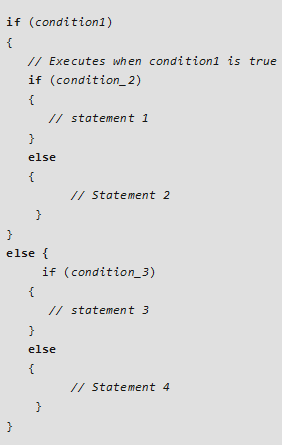
3) If…else if…else

It’s if else ladder. It’s used for multiple conditions but result will be one.



4) Nested If

* *Nested if statements mean an if statement inside another if statement.*
* *It's like placing one decision within another decision. You use this when you want to check for a new condition after a previous condition has already been found true.*

**

*Que. Introduction to For and While loop.*

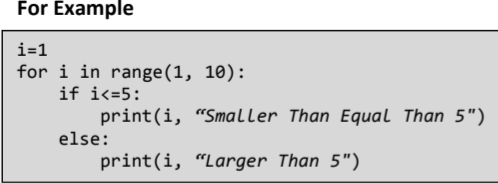
*Answer:*

*For Loop:*

*For loop has the ability to iterate over the items of any sequence, such as a list or a string.*

* *Syntax: for i in sequence:*

*Statements(s)*

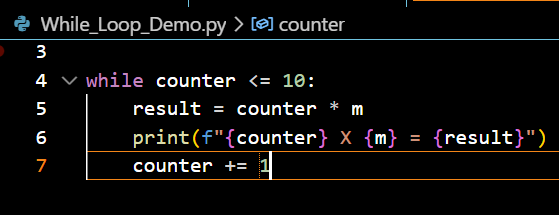
**

*While Loop:*

*A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.*

* *Syntax: while expression:*

*Statement*

**

*Que. How loops work in Python.*

*Answer:*

*A for loop iterates over a sequence (a list, a tuple, a dictionary, a set, or a string).*

*This is less like a keyword in another programming language and works more like an iterator.*

Loops in Python are used to repeat actions efficiently. The main types are For loops (counting through items) and While loops (based on conditions). Additionally, Nested Loops allow looping within loops for more complex tasks. While all the ways provide similar basic functionality, they differ in their syntax and condition-checking time.

*Que. Using loops with collections (lists, tuples, etc.).*

*Answer:*

You can loop through the tuple and list items by using for loop.

You can also loop through the tuple items by referring to their index number.

Use the range () and len () functions to create a suitable iterable.

6. Generators and Iterators

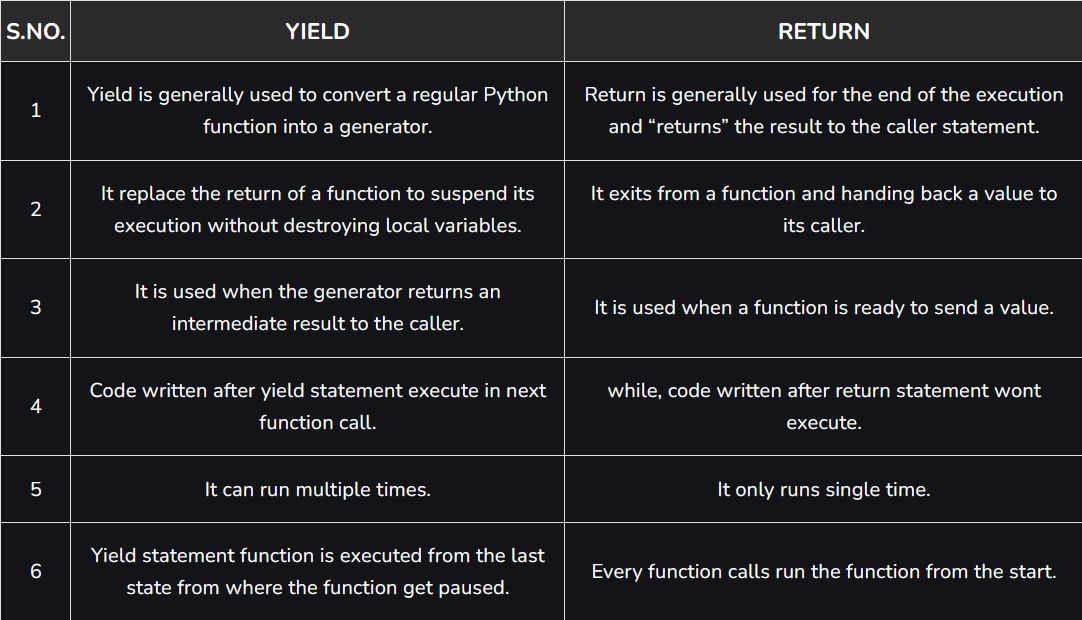
*Que. Understanding how generators work in Python.*

*Answer:*

Python generator functions are a powerful tool for creating iterators.

A generator function is a special type of function that returns an iterator object. Instead of using return to send back a single value, generator functions use yield to produce a series of results over time. This allows the function to generate values and pause its execution after each yield, maintaining its state between iterations.

*Que. Difference between yield and return.*

**

*Que. Understanding iterators and creating custom iterators.*

*Answer:*

An iterator in Python is an object that holds a sequence of values and provide sequential traversal through a collection of items such as lists, tuples and dictionaries. . The Python iterators object is initialized using the **iter()** method. It uses the **next()** method for iteration.

1. **\_\_iter\_\_():** \_\_iter\_\_() method initializes and returns the iterator object itself.
2. **\_\_next\_\_():** the \_\_next\_\_() method retrieves the next available item, throwing a StopIteration exception when no more items are available.

*7. Functions and Methods*

*Que. Defining and calling functions in Python.*

*Answer:*

Python is an object-oriented language and it uses functions to reduce the repetition of the code.

In Python, there is a reserved keyword "def" which we use to define a function in Python, and after "def" we give the name of the function which could be anything but we cannot use the reserved keyword of Python.

### Defining a Function

Before you can call a function, you need to define it. In Python, you define a function using the def keyword followed by the function name and parentheses.

### Calling a Function

To call a function, you simply use the function name followed by parentheses.

**

*Que. Function arguments (positional, keyword, default).*

*Answer:*

## ****Keyword-Only Arguments****

**Keyword-only** arguments mean whenever we pass the arguments (or value) by their parameter names at the time of calling the function in Python in which if you change the position of arguments then there will be no change in the output.

## ****Positional-Only Arguments****

Position-only arguments mean whenever we pass the arguments in the order we have defined function parameters in which if you change the argument position then you may get the unexpected output. We should use positional Arguments whenever we know the order of argument to be passed. So now, we will call the function by using the position-only arguments in two ways, and both cases, we will be getting different outputs from which one will be correct and another one will be incorrect.

### **Default Arguments:**

Python has a different way of representing syntax and default values for function arguments. Default values indicate that the function argument will take that value if no argument value is passed during the function call. The default value is assigned by using the assignment (=) operator of the form keyword\_name = value.

*Que. Scope of variables in Python.*

*Answer:*

In Python, variables are the containers for storing data values.

We do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it.

### Python Local variable

Local variables are those that are initialized within a function and are unique to that function. It cannot be accessed outside of the function. Let’s look at how to make a local variable.

### Python Global variables

Global variables are the ones that are defined and declared outside any function and are not specified to any function. They can be used by any part of the program.

Que. Built-in methods for strings, lists, etc.

Answer:

Built-in method for string:

Capitalize(), casefold(), center(), count(), endswith(), format(), index(), Isalnum(), Isalpha(), Isdigit(), Isupper(), islower(), replace(), title(), swapcase(), etc.

Built-in method for list:

Append(), copy(), clear()< count(), pop(), extend(), index(), insert(), remove(), sort().

*Built-in method for Tuple:*

*Index() and count().*

*Built-in method for Dictionary:*

*clear(), copy(), get(), items(), pop(), popitem(), setdefault(), update(), values), count(), index().*

*Built-in method for Set:*

*Add(), clear(), copy(), difference(), discard(), intersection(), isdijoint(), issuperset(), pop(), remove(),union(), update().*

* 1. *Control Statements (Break, Continue, Pass)*

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

*Answer:*

Loop control statements in Python are special statements that help control the execution of loops (for or while). They let you modify the default behavior of the loop, such as stopping it early, skipping an iteration, or doing nothing temporarily.

## Break Statement in Python

The break statement in Python is used to exit or “break” out of a loop (either a for or while loop) prematurely, before the loop has iterated through all its items or reached its condition. When the break statement is executed, the program immediately exits the loop, and the control moves to the next line of code after the loop.

## Continue Statement in Python

Python Continue statement is a loop control statement that forces to execute the next iteration of the loop while skipping the rest of the code inside the loop for the current iteration only, i.e. when the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped for the current iteration and the next iteration of the loop will begin.

## Pass Statement in Python

Pass statement in Python is a null operation or a placeholder. It is used when a statement is syntactically required but we don’t want to execute any code. It does nothing but allows us to maintain the structure of our program.

*Que. Understanding how to access and manipulate strings.*

*Answer:*

Strings are sequences of characters and are one of the most fundamental data types in programming.

### **1. Creating Strings**

Strings can be created using single ('), double (") or triple quotes (''' or """ for multi-line strings).

### **2. Accessing Strings**

Strings in Python are indexed, meaning each character has a position (starting from 0).

### **3. String Slicing**

You can extract substrings using slicing.

### **4. Modifying Strings**

Strings are **immutable**, meaning you cannot modify them directly, but you can create a new string.

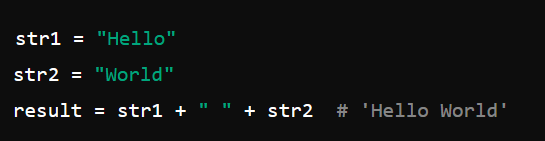
### **5. String Methods and Functions**

Python provides various built-in methods to work with strings.

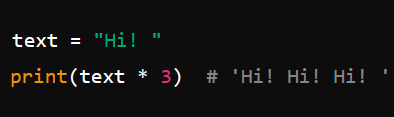
Que. Basic operations: concatenation, repetition, string methods (upper(), lower(), etc.).

Answer:

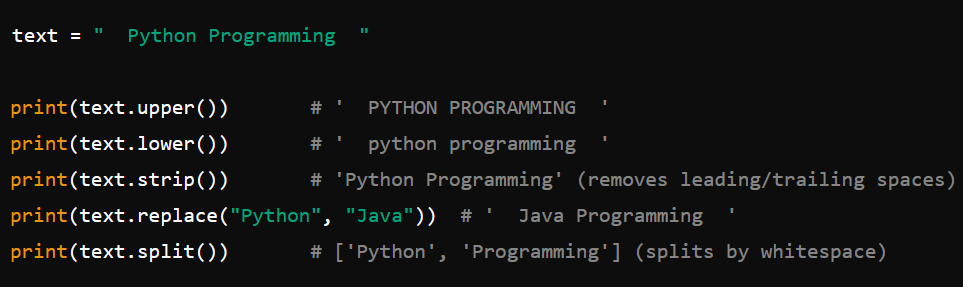
* 1. Concatenation



* 1. Repetition



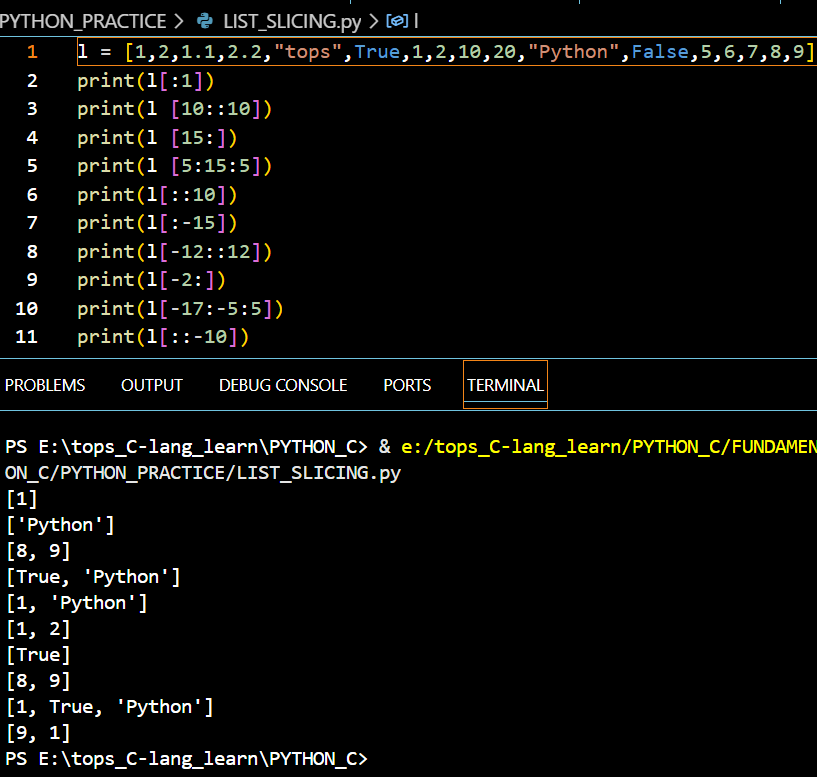
* 1. String Methods



Que. String slicing.

Answer:

String **slicing** allows you to extract a portion (substring) of a string using a specific range of indices.



*10.Advanced Python (map(), reduce(), filter(), Closures and Decorators)*

*Que. How functional programming works in Python.*

*Answer:*

### **1. First-Class Functions**

Functions in Python are treated as first-class citizens, meaning they can be:

* Assigned to variables
* Passed as arguments to other functions
* Returned from functions

### **2. Pure Functions**

A **pure function** always returns the same output for the same input and has **no side effects** (does not modify global state or external variables).

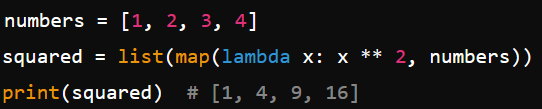
### **3. Higher-Order Functions (HOFs)**

Functions that **take other functions as arguments** or **return functions**.

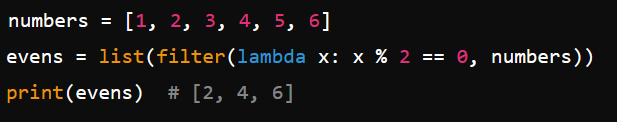
Que. Using map(), reduce(), and filter() functions for processing data.

Answer:

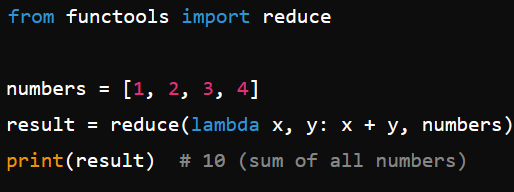
**map()** → Applies a function to all items in an iterable.



**filter()** → Filters elements based on a condition.



**reduce()** → Reduces an iterable to a single value (requires functools.reduce).

**

*Que. Introduction to closures and decorators.*

*Answer:*

### **1. Closures**

A **closure** is a function that remembers the values from its enclosing scope, even when the outer function has finished executing.

#### **How Closures Work:**

* A nested (inner) function is defined inside an outer function.
* The inner function **retains access** to the variables of the outer function **even after the outer function has returned**.

### **2. Decorators**

A **decorator** is a higher-order function that modifies the behavior of another function **without modifying its code**.

#### **How Decorators Work:**

* A decorator is a function that **takes another function as input**.
* It adds functionality to the function and **returns a new function**.