**GE3151-Problem Solving and Python Programming**

**PART-B**

**UNIT I - ALGORITHMIC PROBLEM SOLVING**

**1. What are the building blocks of an algorithm? Explain in detail.**

An **algorithm** is a step-by-step method for solving a problem in a finite number of steps. The building blocks of an algorithm help us construct logical solutions for problems.

**1. Input**

* The algorithm must have clearly defined inputs.
* Example: Two numbers to be added.

**2. Output**

* The algorithm must produce at least one output.
* Example: The sum of two numbers.

**3. Finiteness**

* The algorithm must terminate after a finite number of steps.

**4. Definiteness**

* Each step must be clearly and unambiguously defined.

**🔷 5. Effectiveness**

* The operations must be basic enough to be performed exactly and in a finite time.

**Algorithmic Control Structures (Core Building Blocks):**

**✅ 1. Sequential Structure**

* The default mode; steps are executed one after another.
* Example:

vbnet

Step 1: Start

Step 2: Input A, B

Step 3: Sum = A + B

Step 4: Print Sum

Step 5: Stop

**✅ 2. Selection (Decision-making)**

* Choose between alternatives using conditions.
* Examples: if, if-else, switch
* Example:

arduino

CopyEdit

if (marks >= 50) then

print "Pass"

else

print "Fail"

**✅ 3. Iteration (Loops)**

* Repeat a set of steps until a condition is met.
* Examples: while, for, do-while
* Example:

arduino

CopyEdit

for i = 1 to 5

print i

**✅ 4. Modularization (Functions/Procedures)**

* Reusable blocks of code for a specific task.

**2. Briefly describe iteration and recursion. Explain with algorithm.**

**Iteration**

* Repeating a block of code using loops until a condition is false.
* Uses constructs like for, while, or do-while.

✅ **Example: Factorial using Iteration**

text

CopyEdit

Algorithm: Factorial\_Iterative(n)

Step 1: Start

Step 2: Read n

Step 3: fact ← 1

Step 4: for i from 1 to n do

fact ← fact \* i

Step 5: Print fact

Step 6: Stop

**Recursion**

* A function that calls itself to solve smaller sub-problems.
* Needs a base case to avoid infinite calls.

✅ **Example: Factorial using Recursion**

text

CopyEdit

Algorithm: Factorial\_Recursive(n)

Step 1: if n == 0 or n == 1 then

return 1

Step 2: else

return n \* Factorial\_Recursive(n - 1)

**✅ Comparison**

| **Feature** | **Iteration** | **Recursion** |
| --- | --- | --- |
| Memory Usage | Less (no stack calls) | More (uses call stack) |
| Speed | Faster | Slower (due to function calls) |
| Simplicity | Easier for loops | Easier for complex tree problems |
| Use Case | Counting, Summing | Factorial, Fibonacci, Tree Traversals |

**3. Explain Algorithmic Problem Solving.**

**Algorithmic Problem Solving** is a process of solving problems using a series of logical steps defined by an algorithm.

**Steps in Algorithmic Problem Solving:**

**✅ 1. Understanding the Problem**

* Read the problem thoroughly.
* Identify inputs and expected outputs.

**✅ 2. Analyzing the Problem**

* Check if the problem is solvable.
* Determine constraints and boundaries.

**✅ 3. Developing an Algorithm**

* Use steps like flowcharts, pseudocode to express logic.

**✅ 4. Testing the Algorithm**

* Use sample inputs to check if expected outputs are obtained.

**✅ 5. Refining the Algorithm**

* Optimize the algorithm for time and space efficiency.

**Example Problem: Find the Largest of 3 Numbers**

**Step 1: Understand**  
Input: Three numbers (a, b, c)  
Output: Largest of the three

**Step 2: Design Algorithm**

text

CopyEdit

Step 1: Start

Step 2: Read a, b, c

Step 3: if a > b and a > c then

max ← a

else if b > c then

max ← b

else

max ← c

Step 4: Print max

Step 5: Stop

**Step 3: Refine**

* Can be converted to code, tested with various inputs.

**4. Write an algorithm and draw a flowchart to calculate 2⁴.**

**✅ Algorithm: Power\_Of\_Two**

text

CopyEdit

Step 1: Start

Step 2: Initialize base = 2, exponent = 4, result = 1

Step 3: Repeat exponent times:

result = result \* base

Step 4: Print result

Step 5: Stop

**✅ Flowchart Description**

**[Visual Representation]**

* Start
* Initialize base = 2, exponent = 4, result = 1
* Loop: multiply result by base, decrease counter
* End loop
* Output result
* Stop

If you'd like, I can generate the flowchart as an image for better understanding.

**5. Illustrate Algorithm, Pseudocode, and Flowchart with an example.**

**Example Task: Add two numbers**

**✅ 1. Algorithm**

text

CopyEdit

Step 1: Start

Step 2: Read A, B

Step 3: C ← A + B

Step 4: Print C

Step 5: Stop

**✅ 2. Pseudocode**

pseudocode

CopyEdit

BEGIN

INPUT A, B

C ← A + B

OUTPUT C

END

**✅ 3. Flowchart**

**[Flowchart Elements]**

* **Start** → Input A, B → Add A+B → Output result → **Stop**

Again, if you'd like, I can draw the flowchart and send it visually.

**UNIT II – DATA TYPES, EXPRESSIONS, STATEMENTS**

**1. What is the role of an interpreter? Give a detailed note on Python interpreter and interactive mode of operation.**

**✅ Role of an Interpreter:**

An **interpreter** translates high-level code into machine-understandable format **line by line**, executing each line immediately.

**Python Interpreter:**

* The **Python interpreter** processes Python code.
* It reads the code line-by-line, translates it into bytecode, and then executes it using Python's **PVM (Python Virtual Machine)**

**Modes of Python Execution**

| **Mode** | **Description** |
| --- | --- |
| **Interactive Mode** | Code is executed **line by line** in real-time. Useful for quick testing and debugging. |
| **Script Mode** | Used to write **entire programs** in .py files and run them all at once. |

**✅ Interactive Mode Example:**

bash

CopyEdit

>>> 2 + 3

5

>>> print("Hello")

Hello

* Use IDLE, Python shell, or terminal.
* Output shown immediately after input.

**✅ Script Mode Example:**

python

CopyEdit

# file: sample.py

a = 10

b = 20

print("Sum:", a + b)

Run using:

bash

CopyEdit

python sample.py

**Compiler vs Interpreter**

| **Feature** | **Compiler** | **Interpreter** |
| --- | --- | --- |
| Execution | Translates entire code at once | Translates code line-by-line |
| Speed | Faster execution | Slower due to line-by-line execution |
| Error Detection | Displays all errors after compilation | Stops at the first error |
| Example | C, C++ | Python, Ruby |

**Diagram: Compiler vs Interpreter Structure**

css

CopyEdit

High-Level Code → [Compiler] → Machine Code → Execution

High-Level Code → [Interpreter] → Line by Line Execution

**2. Illustrate values and different standard data types with relevant examples.**

**✅ Values:**

A **value** is a basic data element like 10, "hello", 3.14.

**✅ Standard Data Types in Python:**

| **Data Type** | **Example** | **Description** |
| --- | --- | --- |
| **int** | 10, -3 | Integer numbers |
| **float** | 3.14, 2.0 | Decimal numbers |
| **complex** | 2 + 3j | Complex numbers |
| **bool** | True, False | Boolean values |
| **str** | "Hello" | Text/Characters |
| **list** | [1, 2, 3] | Ordered, mutable sequence |
| **tuple** | (1, 2, 3) | Ordered, immutable sequence |
| **set** | {1, 2, 3} | Unordered, no duplicates |
| **dict** | {'a': 1} | Key-value pairs |

**3. Define variables. List down the rules for naming the variable with example.**

**✅ Definition:**

A **variable** is a name that refers to a memory location storing a value.

**✅ Example:**

python

x = 10

name = "Alice"

**🔷 Rules for Naming Variables:**

1. Must start with a letter (A–Z or a–z) or underscore \_
2. Can have letters, digits (0–9), and underscores
3. Cannot start with a digit
4. Case-sensitive (num ≠ Num)
5. Keywords (like for, while) cannot be used as variable names

**✅ Valid Names:**

python

CopyEdit

total, \_count, number2

**❌ Invalid Names:**

python

CopyEdit

2num, class, first-name # Starts with digit, keyword, invalid character

**4. List down the different types of operators and their function with suitable example.**

**✅ 1. Arithmetic Operators:**

python

CopyEdit

+ - \* / % // \*\*

python

CopyEdit

a = 10; b = 3

print(a + b) # 13

**✅ 2. Relational / Comparison Operators:**

python

CopyEdit

== != > < >= <=

python

CopyEdit

a = 5; b = 3

print(a > b) # True

**✅ 3. Assignment Operators:**

python

CopyEdit

= += -= \*= /= //=

python

CopyEdit

x = 10

x += 5 # x becomes 15

**✅ 4. Logical Operators:**

python

CopyEdit

and or not

python

CopyEdit

a = True; b = False

print(a and b) # False

**✅ 5. Bitwise Operators:**

python

CopyEdit

& | ^ ~ << >>

**✅ 6. Membership Operators:**

python

CopyEdit

in, not in

python

CopyEdit

print('a' in 'apple') # True

**✅ 7. Identity Operators:**

python

CopyEdit

is, is no

**5. What are the two modes of operation in Python? Analyze the differences between them.**

**✅ 1. Interactive Mode**

* Code is typed and executed line by line.
* Great for testing small code snippets.

**✅ 2. Script Mode**

* Code is written in .py file and executed as a whole.

**🔷 Differences Between Interactive and Script Mode:**

| **Feature** | **Interactive Mode** | **Script Mode** |
| --- | --- | --- |
| Execution Style | One line at a time | Whole file at once |
| Output | Immediate | Output after complete execution |
| Use Case | Testing, learning | Developing programs |
| Tool | IDLE, Python shell | Text editor + terminal |

**✅ Example in Interactive Mode:**

python

CopyEdit

>>> print("Hello")

Hello

**✅ Example in Script Mode:**

python

CopyEdit

# file: greet.py

print("Hello from script!")

Run in terminal:

bash

CopyEdit

python greet.py

**UNIT III - CONTROL FLOW, FUNCTIONS, STRINGS**

## ****1. Explain in detail various types of operators. (Jan 2019)****

Operators are used to perform operations on variables and values.

### Types of Operators in Python:

### ****1. Arithmetic Operators:****

Used to perform mathematical operations.

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| + | Addition | 10 + 5 = 15 |
| - | Subtraction | 10 - 3 = 7 |
| \* | Multiplication | 4 \* 2 = 8 |
| / | Division | 10 / 2 = 5.0 |
| // | Floor Division | 10 // 3 = 3 |
| % | Modulus (Remainder) | 10 % 3 = 1 |
| \*\* | Exponentiation | 2 \*\* 3 = 8 |

### ****2. Relational / Comparison Operators:****

Used to compare values.

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| == | Equal to | 5 == 5 → True |
| != | Not equal to | 5 != 3 → True |
| > | Greater than | 10 > 5 → True |
| < | Less than | 4 < 2 → False |
| >= | Greater or equal | 4 >= 4 → True |
| <= | Less or equal | 3 <= 4 → True |

### ****3. Logical Operators:****

Used to combine conditional statements.

python

CopyEdit

a = True

b = False

print(a and b) # False

print(a or b) # True

print(not a) # False

### ****4. Assignment Operators:****

Used to assign values.

| **Operator** | **Example** | **Meaning** |
| --- | --- | --- |
| = | x = 10 | Assign 10 to x |
| += | x += 5 | x = x + 5 |
| -= | x -= 3 | x = x - 3 |
| \*= | x \*= 2 | x = x \* 2 |
| /= | x /= 2 | x = x / 2 |

### ****5. Bitwise Operators:****

Used to perform bit-level operations.

python

CopyEdit

a = 5 # 0101

b = 3 # 0011

print(a & b) # 0001 → 1

print(a | b) # 0111 → 7

### ****6. Identity Operators:****

Used to compare memory locations.

python

CopyEdit

x = [1, 2]

y = [1, 2]

print(x is y) # False

print(x == y) # True

### ****7. Membership Operators:****

Check membership of a value.

python

CopyEdit

fruits = ["apple", "banana"]

print("apple" in fruits) # True

## ****2. Discuss conditional alternative and chained conditional in detail. (Jan 2019)****

### ✅ ****Conditional Statements in Python****

### ****Simple**** if ****Statement:****

python

CopyEdit

x = 10

if x > 5:

print("x is greater than 5")

### ****Alternative (****if-else****):****

python

CopyEdit

x = 3

if x > 5:

print("x is greater")

else:

print("x is smaller or equal")

### ****Chained Conditionals (****if-elif-else****):****

Used to handle multiple conditions.

python

CopyEdit

marks = 85

if marks >= 90:

print("Grade A")

elif marks >= 75:

print("Grade B")

elif marks >= 60:

print("Grade C")

else:

print("Fail")

### 🔁 ****Nested**** if ****Conditions:****

python

CopyEdit

age = 25

if age > 18:

if age < 30:

print("Young Adult")

### 🔎 ****Example: Leap Year Check****

python

CopyEdit

year = 2024

if year % 4 == 0:

if year % 100 != 0 or year % 400 == 0:

print("Leap year")

else:

print("Not a leap year")

## ****3. Explain in detail about iterations.****

### ✅ ****Iteration (Looping):****

Used to repeat a block of code multiple times.

### ****1.**** while ****Loop:****

python

CopyEdit

i = 1

while i <= 5:

print(i)

i += 1

### ****2.**** for ****Loop:****

python

CopyEdit

for i in range(1, 6):

print(i)

### ****3. Loop Control Statements:****

* **break** – exits the loop
* **continue** – skips current iteration
* **pass** – placeholder

python

CopyEdit

for i in range(5):

if i == 3:

continue

print(i)

### ****Nested Loops:****

python

CopyEdit

for i in range(3):

for j in range(2):

print(f"{i},{j}")

## ****4. Explain in detail about Fruitful Functions.****

### ✅ ****Fruitful Function:****

A **fruitful function** returns a value using the return statement.

### ****Syntax:****

python

CopyEdit

def add(a, b):

return a + b

result = add(5, 3)

print(result) # Output: 8

### 🔹 ****Benefits:****

* Makes code reusable
* Can be used in expressions
* Supports modular programming

### 🧠 ****Example: Check Even/Odd****

python

CopyEdit

def is\_even(num):

return num % 2 == 0

print(is\_even(4)) # True

### 🧮 ****Function Returning Multiple Values****

python

CopyEdit

def calc(a, b):

return a+b, a-b

add, sub = calc(10, 5)

print(add, sub)

## ****5. Describe in detail about Recursion.****

### ✅ ****Recursion:****

A function that **calls itself** to solve smaller instances of a problem.

### ****Basic Example:****

python

CopyEdit

def countdown(n):

if n <= 0:

print("Done!")

else:

print(n)

countdown(n-1)

### 🔁 ****Factorial using Recursion:****

python

CopyEdit

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

print(factorial(5)) # Output: 120

### ****Working:****

matlab

CopyEdit

factorial(3)

= 3 \* factorial(2)

= 3 \* 2 \* factorial(1)

= 3 \* 2 \* 1 \* factorial(0)

= 3 \* 2 \* 1 \* 1 = 6

**UNIT IV LISTS, TUPLES, DICTIONARIES**

## ****1. Discuss in detail about list methods and list loops with examples.****

### What is a List?

A **list** is an ordered, mutable, and iterable collection of items in Python. Lists can contain different data types.

python

CopyEdit

my\_list = [1, 2, 3, 4, 5]

### List Methods (Commonly Used)

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| append(x) | Adds an item to end | list.append(6) |
| insert(i, x) | Inserts at index | list.insert(2, 10) |
| remove(x) | Removes first x | list.remove(3) |
| pop(i) | Removes item at i | list.pop(1) |
| sort() | Sorts the list | list.sort() |
| reverse() | Reverses the list | list.reverse() |
| index(x) | Returns index of x | list.index(3) |
| count(x) | Count occurrences | list.count(2) |

python

CopyEdit

numbers = [1, 2, 2, 3, 4]

numbers.append(5)

numbers.remove(2)

print(numbers) # [1, 2, 3, 4, 5]

### List Loops

**For loop:**

python

CopyEdit

for item in numbers:

print(item)

**While loop:**

python

CopyEdit

i = 0

while i < len(numbers):

print(numbers[i])

i += 1

### List Comprehension

python

CopyEdit

squares = [x\*\*2 for x in range(5)]

print(squares) # [0, 1, 4, 9, 16]

## ****2. Explain in detail about mutability and tuples with a Python program.****

### Mutability

* **Mutable** objects can be changed (e.g., lists).
* **Immutable** objects cannot be changed (e.g., tuples, strings).

### Tuple

* A **tuple** is an immutable, ordered collection.

python

CopyEdit

my\_tuple = (1, 2, 3)

### 🔸 Example – List (Mutable) vs Tuple (Immutable)

python

CopyEdit

# List (Mutable)

list1 = [1, 2, 3]

list1[0] = 10

print(list1) # [10, 2, 3]

# Tuple (Immutable)

tuple1 = (1, 2, 3)

# tuple1[0] = 10 # ❌ Error

### 🔸 Why Tuples?

* Faster than lists
* Used in dictionary keys
* Safe from unintended modification

## ****3. What is tuple assignment? Explain it with an example.****

### ✅ Tuple Assignment:

Tuple assignment allows **multiple variables** to be assigned values **simultaneously**.

python

CopyEdit

(a, b) = (10, 20)

print(a) # 10

print(b) # 20

### 🔹 With Lists:

python

CopyEdit

data = [1, 2, 3]

(x, y, z) = data

print(x, y, z) # 1 2 3

### 🔹 Swap Example:

python

CopyEdit

a, b = 5, 10

a, b = b, a

print(a, b) # 10 5

Tuple assignment is **cleaner** and **easier** than traditional assignments.

## ****4. Is it possible to return tuple as values? Justify your answer with an example.****

### ✅ Yes, tuples can be returned from functions.

### 🔸 Example:

python

CopyEdit

def calculate(a, b):

add = a + b

diff = a - b

return (add, diff)

result = calculate(10, 5)

print(result) # (15, 5)

print(result[0]) # 15

### 🔹 Multiple Return Assignment:

python

CopyEdit

sum, difference = calculate(10, 5)

print(sum) # 15

print(difference)# 5

Tuples are a simple way to **return multiple values** from a function.

## ****5. Explain in detail about dictionaries and its operations. (Jan-2018)****

### ✅ What is a Dictionary?

* A **dictionary** is an unordered collection of **key-value** pairs.
* Keys must be **unique and immutable** (strings, numbers, tuples).

python

CopyEdit

student = {

"name": "John",

"age": 21,

"department": "CSE"

}

### 🔸 Accessing Values:

python

CopyEdit

print(student["name"]) # John

### 🔸 Dictionary Operations:

| **Operation** | **Example** |
| --- | --- |
| Access value | dict[key] |
| Add/Update | dict["age"] = 22 |
| Delete | del dict["age"] |
| Get keys | dict.keys() |
| Get values | dict.values() |
| Get items | dict.items() |
| Check key | "name" in dict |
| Clear all | dict.clear() |

### 🔹 Looping through Dictionary:

python

CopyEdit

for key, value in student.items():

print(key, ":", value)

### 🔹 Nested Dictionary:

python

CopyEdit

students = {

"101": {"name": "Alice", "mark": 90},

"102": {"name": "Bob", "mark": 85}

}

### 🔹 Built-in Functions:

* len(), str(), type(), dict.get()

### 🔸 Example Program:

python

CopyEdit

student = {"name": "John", "age": 21}

student["mark"] = 95

del student["age"]

for k, v in student.items():

print(k, v)

**UNIT V FILES, MODULES, PACKAGES**

## ****1. Write a function that copies a file reading and writing up to 50 characters at a time. (or) Explain the commands used to read and write into a file with example. (Jan 2019)****

### Function to Copy File Reading and Writing 50 Characters at a Time:

To copy a file in Python, we can read and write the file content in chunks. In this case, we'll read and write 50 characters at a time.

python

CopyEdit

def copy\_file(src, dest):

try:

# Open source file in read mode and destination file in write mode

with open(src, 'r') as infile, open(dest, 'w') as outfile:

while True:

# Read up to 50 characters

chunk = infile.read(50)

# If no content is read, stop

if not chunk:

break

# Write the chunk to the destination file

outfile.write(chunk)

print(f"File '{src}' successfully copied to '{dest}'.")

except FileNotFoundError:

print("Source file not found.")

except Exception as e:

print(f"An error occurred: {e}")

# Example Usage

copy\_file("source.txt", "destination.txt")

### Explanation of File Commands in Python:

* **Opening Files**:
  + open(filename, mode): Opens a file.
  + Modes: 'r' (read), 'w' (write), 'a' (append), 'b' (binary mode), 'rb' (read binary), 'wb' (write binary).
  + Example: f = open('file.txt', 'r')
* **Reading from File**:
  + read(size): Reads the specified number of bytes/characters.
  + readlines(): Reads the entire file into a list of lines.
  + Example: content = f.read(50)
* **Writing to File**:
  + write(data): Writes data to a file.
  + writelines(lines): Writes a list of lines to a file.
  + Example: f.write("Hello World!")
* **Closing Files**:
  + close(): Closes an opened file.
  + Example: f.close()

## ****2. (a). Write a program to perform exception handling.****

### Example Program for Exception Handling:

python

CopyEdit

def safe\_division(a, b):

try:

result = a / b

print("Result:", result)

except ZeroDivisionError:

print("Error: Division by zero is not allowed.")

except TypeError:

print("Error: Invalid type. Please provide numbers.")

except Exception as e:

print(f"An unexpected error occurred: {e}")

else:

print("Division successful!")

finally:

print("Execution complete.")

# Example Usage

safe\_division(10, 2) # Works fine

safe\_division(10, 0) # Will raise ZeroDivisionError

safe\_division(10, "a") # Will raise TypeError

### 🔹 Explanation of Exception Handling:

* **Try**: The block of code that might raise an exception.
* **Except**: This block handles the exception if it occurs.
* **Else**: If no exception occurs, this block will execute.
* **Finally**: This block always runs, regardless of exceptions.

## ****2. (b). Write a Python program to handle multiple exceptions.****

### Example Program to Handle Multiple Exceptions:

python

CopyEdit

def handle\_multiple\_exceptions(a, b, c):

try:

result1 = a / b

result2 = c[2]

except ZeroDivisionError:

print("Error: Division by zero.")

except IndexError:

print("Error: Index out of range.")

except Exception as e:

print(f"An unexpected error occurred: {e}")

else:

print("Both operations were successful!")

finally:

print("Execution complete.")

# Example Usage

handle\_multiple\_exceptions(10, 0, [1, 2]) # Division by zero error

handle\_multiple\_exceptions(10, 5, [1]) # Index out of range error

In this program, **multiple exceptions** are handled:

* **ZeroDivisionError**: Handles division by zero.
* **IndexError**: Handles out-of-range index access in a list.

## ****3. Write a python program to count number of lines, words and characters in a text file. (May 2019)****

### 🔹 Python Program to Count Lines, Words, and Characters in a Text File:

python

CopyEdit

def count\_file\_content(file\_path):

try:

with open(file\_path, 'r') as file:

lines = file.readlines()

num\_lines = len(lines)

num\_words = sum(len(line.split()) for line in lines)

num\_chars = sum(len(line) for line in lines)

print(f"Lines: {num\_lines}")

print(f"Words: {num\_words}")

print(f"Characters: {num\_chars}")

except FileNotFoundError:

print("The file was not found.")

except Exception as e:

print(f"An error occurred: {e}")

# Example Usage

count\_file\_content("example.txt")

### 🔹 Explanation:

* **Reading file**: The readlines() method is used to read all lines from the file.
* **Counting lines**: The length of the list returned by readlines().
* **Counting words**: The split() method splits each line into words, and len() counts the words.
* **Counting characters**: The length of each line is added together.

## ****4. Write a Python program to illustrate the use of command-line arguments.****

### 🔹 Python Program to Illustrate Command-Line Arguments:

python

CopyEdit

import sys

def print\_arguments():

print("Total Arguments:", len(sys.argv))

print("Arguments List:", sys.argv)

for i, arg in enumerate(sys.argv):

print(f"Argument {i}: {arg}")

# Run the program like: python program.py arg1 arg2 arg3

if \_\_name\_\_ == "\_\_main\_\_":

print\_arguments()

### 🔹 Explanation:

* **sys.argv**: A list in Python, which contains the command-line arguments passed to the script. The first item is the script name.
* To run the script: python script\_name.py arg1 arg2

## ****5. Mention the commands and their syntax for the following:****

### 🔹 ****Commands for File Operations:****

1. **Get current directory**:

bash

CopyEdit

pwd

1. **Change directory**:

bash

CopyEdit

cd directory\_name

1. **List directories and files**:

bash

CopyEdit

ls

1. **Make a new directory**:

bash

CopyEdit

mkdir directory\_name

1. **Rename a directory**:

bash

CopyEdit

mv old\_directory\_name new\_directory\_name

1. **Remove a directory**:

bash

CopyEdit

rmdir directory\_name

### 🔹 Example usage in Python:

* To get the current directory:

python

CopyEdit

import os

print(os.getcwd()) # Gets the current working directory

* To change directory:

python

CopyEdit

os.chdir('path/to/directory')

* To create a new directory:

python

CopyEdit

os.mkdir('new\_directory')

* To remove a directory:

python

CopyEdit

os.rmdir('old\_directory')

These commands and their equivalents can be used within the terminal or Python scripts for managing directories and files.