**Introduction to Python**

**1. Write a Python script that accepts user input for name and age and prints a greeting message.**

**Program:**

print("Enter Name:")

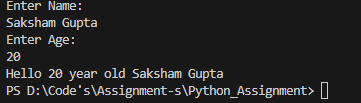
name = input()

print("Enter Age:")

age = input()

print("Hello " + age + " year old " + name)

**Output:**

****

**2. Create a program to find the largest of three input numbers using conditional statements.**

**Program:**

print("Enter Three Numbers")

print("Enter first number: ")

num1 = input()

print("Enter second number: ")

num2 = input()

print("Enter third number: ")

num3 = input()

if num1>num2 and num1>num3:

    print("The largest number is: " + num1)

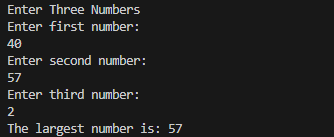
elif num2>num1 and num2>num3:

    print("The largest number is: " + num2)

else:

    print("The largest number is: " + num3)

**Output:**

****

**3. Write a script to check if a given year is a leap year.**

**Program:**

print("Enter Year:")

year = int(input()) #Need to be typecasted to int for calculation

if year%4 == 0:

    if year%100 == 0:

        if year%400 == 0:

            print(f"{year} is a leap year")

        else:

            print(f"{year} is not a leap year")

    else:

        print(f"{year} is a leap year")

**Output:**

****

**4. Develop a Python program that reverses a given integer.**

**Program:**

print("Enter Number:")

num = int(input())

rev\_num = 0

while num>0:

    rev\_num = rev\_num \* 10 + num % 10

    num = num // 10

print(f"The reverse of the number is: {rev\_num}")

**Output:**

****

**5. Write a script that swaps two variables without using a third variable.**

**Program:**

print("Enter first number: ")

num1 = int(input())

print("Enter second number: ")

num2 = int(input())

print("--- Before Swapping ---")

print(f"First number is: {num1}")

print(f"Second number is: {num2}")

num1 = num1 + num2

num2 = num1 - num2

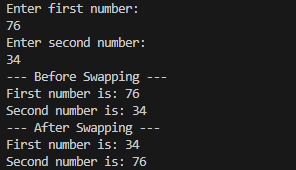
num1 = num1 - num2

print("--- After Swapping ---")

print(f"First number is: {num1}")

print(f"Second number is: {num2}")

**Output:**

****

**6. Create a program that simulates a simple calculator supporting +, -, \*, / with input parsing.**

**Program:**

print("-- Calculator --")

print("Select an operation:")

print("1. Addition(+)")

print("2. Subtraction(-)")

print("3. Multiplication(\*)")

print("4. Division(/)")

print("5. Exit")

while True:

    print("Enter choice :")

    choice = int(input())

    if choice == 5:

        break

    if choice in (1, 2, 3, 4):

        print("Enter first number:")

        num1 = float(input())

        print("Enter second number:")

        num2 = float(input())

        if choice == 1:

            result = num1 + num2

            print(f"Result: {num1} + {num2} = {result}")

        elif choice == 2:

            result = num1 - num2

            print(f"Result: {num1} - {num2} = {result}")

        elif choice == 3:

            result = num1 \* num2

            print(f"Result: {num1} \* {num2} = {result}")

        elif choice == 4:

            if num2 == 0:

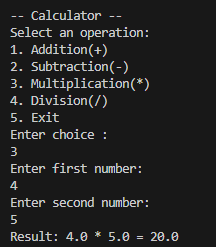
                print("Error: Division by zero")

            else:

                result = num1 / num2

                print(f"Result: {num1} / {num2} = {result}")

**Output:**

****

**7. Write a Python script to determine if a given number is a prime number.**

**Program:**

print("Enter Number:")

num = int(input())

flag = False

if num > 1:

    for i in range(2, num):

        if (num % i) == 0:

            flag = True

            break

if flag:

    print(f"{num} is not a prime number")

else:

    print(f"{num} is a prime number")

**Output:**

****

**8. Develop a program to convert a given temperature from Celsius to Fahrenheit and vice versa.**

**Program:**

print("Choice :")

print("1. Celsius to Fahrenheit")

print("2. Fahrenheit to Celsius")

print("Enter your choice:")

choice = int(input())

if choice == 1:

    print("Enter temperature in Celsius:")

    cel = float(input())

    fah = (cel \* 9/5) + 32

    print(f"{cel} \*C is equal to {round(fah,1)} \*F")

elif choice == 2:

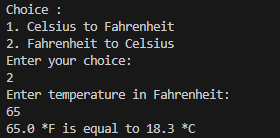
    print("Enter temperature in Fahrenheit:")

    fah = float(input())

    cel = (fah - 32) \* 5/9

    print(f"{fah} \*F is equal to {round(cel,1)} \*C")

**Output:**

****

**9. Create a Python program that prints the Fibonacci sequence up to n terms using iteration.**

**Program:**

print("Enter the number of terms for Fibonacci sequence:")

n = int(input())

if n <= 0:

    print("Enter a positive integer")

else:

    print("Fibonacci sequence:")

    a = 0

    b = 1

    for i in range(n):

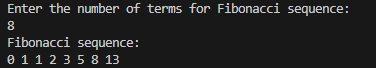
        print(a, end=" ")

        temp = a + b

        a = b

        b = temp

**Output:**

****

**10. Implement a basic number guessing game where the computer selects a random number.**

**Program:**

import random

num = random.randint(1, 100)

print("Enter your guess between 1 and 100:")

guess = int(input())

if guess == num:

    print("Your guess is right")

else:

    print(f"Your guess is wrong, The correct number is {num}")

**Output:**

****

**Python Functions**

**1. Write a function to calculate the factorial of a number (non-recursive).**

**Program:**

def factorial(num):

    for i in range(1, num + 1):

        if i == 1:

            res = 1

        else:

            res = res \* i

    return res

print("Enter Number:")

num = int(input())

res = factorial(num)

print(f"Factorial of {num} is {res}")

**Output:**

****

**2. Define a function that checks whether a string is a palindrome.**

**Program:**

def is\_palindrome(str):

    rev\_str = ""

    for char in str:

        rev\_str = char + rev\_str

    return str == rev\_str

print("Enter a string:")

str = input()

if is\_palindrome(str):

    print(f"{str} is a palindrome")

else:

    print(f"{str} is not a palindrome")

**Output:**

****

**3. Write a function that accepts a list and returns the sum and average of the numbers.**

**Program:**

def sum\_and\_average(numbers, n):

    total = 0

    for num in numbers:

        total += num

    avg = total / n

    return total, avg

print("Enter the number of elements: ")

n = int(input())

print("Enter the elements: ")

numbers = []

for i in range(n):

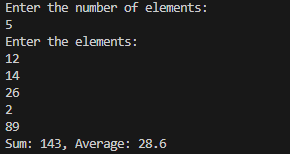
    num = int(input())

    numbers.append(num)

total, avg = sum\_and\_average(numbers, n)

print(f"Sum: {total}, Average: {avg}")

**Output:**

****

**4. Create a function that returns the nth Fibonacci number using recursion.**

**Program:**

def fibonacci(n):

    if n <= 0:

        return 0

    elif n == 1:

        return 1

    else:

        return fibonacci(n - 1) + fibonacci(n - 2)

print("Enter which Fibonacci number you want:")

n = int(input())

result = fibonacci(n)

print(f"The {n}th Fibonacci number is {result}")

**Output:**

****

**5. Define a function to count the number of vowels in a given string.**

**Program:**

def count\_vowels(str):

    count = 0

    for char in str:

        if char == 'a' or char == 'e' or char == 'i' or char == 'o' or char == 'u' or char == 'A' or char == 'E' or char == 'I' or char == 'O' or char == 'U':

            count += 1

    return count

print("Enter a string:")

str = input()

count = count\_vowels(str)

print(f"The number of vowels in '{str}' is {count}")

**Output:**

****

**6. Implement a decorator that measures execution time of any function.**

**Program:**

import time

def time\_count\_decorator(func):

    def wrapper():

        start = time.time()

        func()

        end = time.time()

        print(f"Execution time is: {round(end - start, 4)} seconds")

    return wrapper

@time\_count\_decorator

def temp\_func():

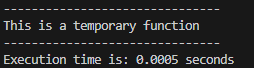
    print("-------------------------------")

    print("This is a temporary function")

    print("-------------------------------")

temp\_func()

**Output:**

****

**7. Write a recursive function to solve the Tower of Hanoi problem.**

**Program:**

def tower\_of\_hanoi(n, src, aux, target):

    if n == 1:

        print(f"Move disk 1 from {src} to {target}")

    else:

        tower\_of\_hanoi(n - 1, src, target, aux)

        print(f"Move disk {n} from {src} to {target}")

        tower\_of\_hanoi(n - 1, aux, src, target)

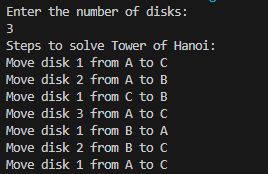
print("Enter the number of disks:")

num\_disks = int(input())

print("Steps to solve Tower of Hanoi:")

tower\_of\_hanoi(num\_disks, 'A', 'B', 'C')

**Output:**

****

**8. Implement a function that uses variable-length arguments to sum any number of inputs.**

**Program:**

def sum\_and\_average(numbers, n):

    total = 0

    for num in numbers:

        total += num

    avg = total / n

    return total, avg

print("Enter the number of elements: ")

n = int(input())

print("Enter the elements: ")

numbers = []

for i in range(n):

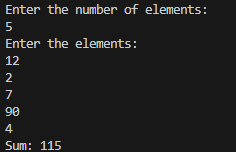
    num = int(input())

    numbers.append(num)

total, avg = sum\_and\_average(numbers, n)

print(f"Sum: {total}, Average: {avg}")

**Output:**

****

**9. Write a function that flattens a nested list using recursion.**

**Program:**

def flatten\_list(nested\_list):

    flat = []

    for item in nested\_list:

        if isinstance(item, list):

            flat.extend(flatten\_list(item))

        else:

            flat.append(item)

    return flat

nested = [1, [2, [3, 4], 5], [6, 7], 8]

print("List Before Flattening:")

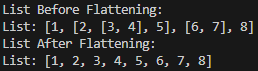
print("List:", nested)

flattened = flatten\_list(nested)

print("List After Flattening:")

print("List:", flattened)

**Output:**

****

**10. Implement a memoized version of the Fibonacci sequence.**

**Program:**

fib\_cache = {}

def memoized\_fibonacci(n):

    if n in fib\_cache:

        return fib\_cache[n]

    if n <= 1:

        fib\_cache[n] = n

    else:

        fib\_cache[n] = memoized\_fibonacci(n - 1) + memoized\_fibonacci(n - 2)

    return fib\_cache[n]

print("Enter which Fibonacci number you want:")

n = int(input())

result = memoized\_fibonacci(n)

print(f"The {n}th Fibonacci number is {result}")

**Output:**

****

**Modules in Python**

**1. Create a custom module with functions to add, subtract, multiply, and divide two numbers.**

**Program:**

def add(a, b):

    return a + b

def subtract(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    return a / b

**2. Use the `math` module to calculate square root, factorial, and power of a number.**

**Program:**

import math

print("\nSelect an operation:")

print("1. Square Root")

print("2. Factorial")

print("3. Power")

print("4. Exit")

while True:

    print("\nEnter your choice:")

    choice = int(input())

    if choice == 1:

        print("Enter a Number: ")

        num = float(input())

        if num < 0:

            print("Error: Cannot calculate square root of a negative number.")

        else:

            result = math.sqrt(num)

            print(f"The square root of {num} is {result}")

    elif choice == 2:

        print("Enter a Number: ")

        num = int(input())

        if num < 0:

            print("Error: Factorial is not defined for negative numbers.")

        else:

            result = math.factorial(num)

            print(f"The factorial of {num} is {result}")

    elif choice == 3:

        print("Enter the base number: ")

        base = float(input())

        print("Enter the exponent: ")

        exponent = float(input())

        result = math.pow(base, exponent)

        print(f"{base} raised to the power of {exponent} is {result}")

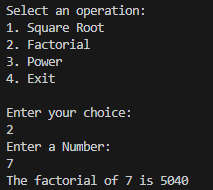
    elif choice == 4:

        break

    else:

        print("Invalid choice. Please select a valid option.")

**Output:**

****

**3. Write a program that uses `random` to generate a password of given length.**

**Program:**

import random

import string

def generate\_password(length):

    if length <= 0:

        print("Error: Password length must be greater than 0")

    characters = string.ascii\_letters + string.digits + string.punctuation

    password = random.choices(characters, k=length)

    return ''.join(password)

print("Enter password length:")

length = int(input())

password = generate\_password(length)

if password:

    print("Generated Password:", password)

**Output:**

****

**4. Create a program using the `datetime` module to display the current date and time.**

**Program:**

import datetime

now = datetime.datetime.now()

print("Current Date and Time:")

print(now.strftime("%Y-%m-%d %H:%M:%S"))

**Output:**

****

**5. Import a custom module and use its functions in another script.**

**Program:**

import module as m

print("-- Calculator --")

print("Select an operation:")

print("1. Addition(+)")

print("2. Subtraction(-)")

print("3. Multiplication(\*)")

print("4. Division(/)")

print("5. Exit")

while True:

    print("Enter choice :")

    choice = int(input())

    if choice == 5:

        break

    if choice in (1, 2, 3, 4):

        print("Enter first number:")

        num1 = float(input())

        print("Enter second number:")

        num2 = float(input())

        if choice == 1:

            result = m.add(num1, num2)

            print(f"Result: {num1} + {num2} = {result}")

        elif choice == 2:

            result = m.subtract(num1, num2)

            print(f"Result: {num1} - {num2} = {result}")

        elif choice == 3:

            result = m.multiply(num1, num2)

            print(f"Result: {num1} \* {num2} = {result}")

        elif choice == 4:

            if num2 == 0:

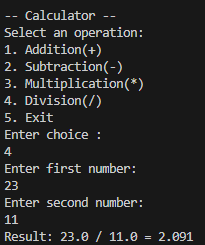
                print("Error: Division by zero")

            else:

                result = m.divide(num1, num2)

                print(f"Result: {num1} / {num2} = {round(result,3)}")

**Output:**

****

**6. Build a command-line utility using `argparse` to perform arithmetic operations.**

**Program:**

import argparse

parser = argparse.ArgumentParser()

parser.add\_argument("x", type=float)

parser.add\_argument("y", type=float)

parser.add\_argument("op", choices=["add", "sub", "mul", "div"])

args = parser.parse\_args()

if args.op == "add":

    print(args.x + args.y)

elif args.op == "sub":

    print(args.x - args.y)

elif args.op == "mul":

    print(args.x \* args.y)

elif args.op == "div":

    if args.y == 0:

        print("Error: Division by zero")

    else:

        print(args.x / args.y)

**Output:**

****

**7. Create and use a package with multiple modules in it.**

**Program:**

**Output:**

**8. Develop a program that uses `os` and `sys` modules to list files and command-line args.**

**Program:**

import os

import sys

print("Files in current directory:")

for file in os.listdir():

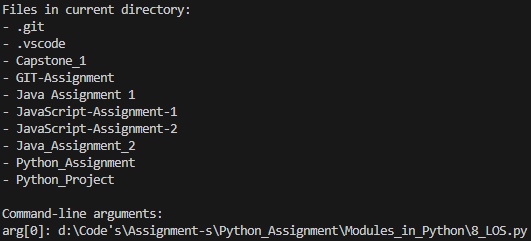
    print("-", file)

print("\nCommand-line arguments:")

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

    print(f"arg[{i}]: {arg}")

**Output:**

****

**9. Use `importlib` to dynamically import a module and invoke a function.**

**Program:**

**Operations.py**

def greet(name):

    return f"Hello, {name}!"

**main.py**

import importlib

module\_name = "operations"

function\_name = "greet"

mod = importlib.import\_module(module\_name)

func = getattr(mod, function\_name)

result = func("Saksham")

print(result)

**Output:**

****

**10. Implement a Python script that uses `glob` to search for all `.txt` files in a directory.**

**Program:**

import glob

import os

directory = "."

pattern = os.path.join(directory, "\*.txt")

txt\_files = glob.glob(pattern)

print("Found .txt files:")

for file in txt\_files:

    print("-", file)

**Output:**

****

**Data Structures**

**1. Implement a function to reverse a list without using built-in reverse().**

**Program:**

print("Enter numbers for List:")

nums = input().split()

result = []

for num in nums:

    result = [num] + result

print("Reversed List:", result)

**Output:**

****

**2. Write a function to merge two dictionaries.**

**Program:**

dict1 = {"a": 1, "b": 2}

dict2 = {"b": 3, "c": 4}

merged\_dict = dict1.copy()

merged\_dict.update(dict2)

print("Merged Dictionary:", merged\_dict)

**Output:**

****

**3. Develop a function that removes duplicate elements from a list.**

**Program:**

list = [1, 2, 2, 3, 4, 4, 5]

print("List Before Removing Duplicates:")

print("List:", list)

seen = set()

result = []

for item in list:

    if item not in seen:

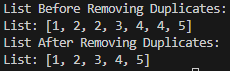
        seen.add(item)

        result.append(item)

print("List After Removing Duplicates:")

print("List:", result)

**Output:**

****

**4. Create a function that counts the frequency of each word in a list.**

**Program:**

List = ["apple", "banana", "apple", "orange", "banana", "apple"]

freq = {}

for w in List:

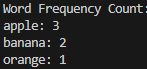
    freq[w] = freq.get(w, 0) + 1

print("Word Frequency Count:")

for word, count in freq.items():

    print(f"{word}: {count}")

**Output:**

****

**5. Write a program to sort a list of tuples based on the second element.**

**Program:**

tuples = [("a", 3), ("b", 1), ("c", 2)]

sorted\_tuples = sorted(tuples, key=lambda x: x[1])

print("Sorted tuples:", sorted\_tuples)

**Output:**

****

**6. Implement a stack using list with push, pop, and peek operations.**

**Program:**

class Stack:

    def \_\_init\_\_(self):

        self.items = []

    def push(self, val):

        self.items.append(val)

    def pop(self):

        if not self.items:

            return "Empty stack"

        return self.items.pop()

    def peek(self):

        if not self.items:

            return "Empty stack"

        return self.items[-1]

s = Stack()

s.push(10)

s.push(20)

print(s.peek())

print(s.pop())

print(s.pop())

**Output:**

****

**7. Create a queue using collections.deque and implement enqueue and dequeue.**

**Program:**

from collections import deque

class Queue:

    def \_\_init\_\_(self):

        self.q = deque()

    def enqueue(self, val):

        self.q.append(val)

    def dequeue(self):

        if not self.q:

            return "Empty queue"

        return self.q.popleft()

q = Queue()

q.enqueue(1)

q.enqueue(2)

print(q.dequeue())

print(q.dequeue())

**Output:**

****

**8. Write a function to find the intersection of two lists.**

**Program:**

l1,l2 = [1, 2, 3], [2, 3, 4]

print("Intersection of two lists is:", list(set(l1) & set(l2)))

**Output:**

****

**9. Create a program that uses a dictionary to implement a phonebook.**

**Program:**

def phonebook():

    book = {}

    book["Sam"] = "123456789"

    book["AG"] = "234567890"

    return book

print(phonebook())

**Output:**

****

**10. Implement a function to check if a list is a palindrome.**

**Program:**

def is\_palindrome(lst):

    return lst == lst[::-1]

print(is\_palindrome([1, 2, 3, 2, 1]))

print(is\_palindrome([1, 2, 3]))

**Output:**

****

**String Formatting & Manipulation**

**1. Write a function that capitalizes the first letter of each word in a string.**

**Program:**

def capitalize\_words(text):

    return ' '.join(word.capitalize() for word in text.split())

print(capitalize\_words("saksham is doing this assignment!"))

**Output:**

****

**2. Create a program that finds all substrings of a given string.**

**Program:**

s = "sam"

substrings = []

for i in range(len(s)):

    for j in range(i + 1, len(s) + 1):

        substrings.append(s[i:j])

print("Substrings:", substrings)

**Output:**

****

**3. Write a function that replaces all vowels in a string with '\*' symbol.**

**Program:**

text = "Saksham is doing this assignment!"

vowels = "aeiouAEIOU"

replaced = ""

for char in text:

    if char in vowels:

        replaced += "\*"

    else:

        replaced += char

print("Replaced vowels:", replaced)

**Output:**

****

**4. Develop a function that counts words, characters, and lines in a string.**

**Program:**

sample = """Saksham is doing this assignment!

This is a program to count words, lines, and characters."""

words = sample.split()

lines = sample.split('\n')

characters = len(sample)

print("Words:", len(words))

print("Lines:", len(lines))

print("Characters:", characters)

**Output:**

****

**5. Write a script to format a number as currency (e.g., 1000000 -> 1,000,000).**

**Program:**

number = 1000000

num\_str = str(number)

result = ""

count = 0

for char in reversed(num\_str):

    if count and count % 3 == 0:

        result = "," + result

    result = char + result

    count += 1

print("Formatted currency:", result)

**Output:**

****

**6. Implement a function that validates a strong password based on given criteria.**

**Program:**

password = "Saksham@123"

has\_upper = False

has\_lower = False

has\_digit = False

has\_special = False

for c in password:

    if c.isupper():

        has\_upper = True

    elif c.islower():

        has\_lower = True

    elif c.isdigit():

        has\_digit = True

    elif c in "!@#$%^&\*\_+":

        has\_special = True

if len(password) >= 8 and has\_upper and has\_lower and has\_digit and has\_special:

    is\_valid = True

else:

    is\_valid = False

print("Is strong password?", is\_valid)

**Output:**

****

**7. Write a script that encodes a string using Caesar cipher.**

**Program:**

text = "Saksham"

shift = 3

cipher = ''

for char in text:

    if char.isalpha():

        base = ord('A') if char.isupper() else ord('a')

        cipher += chr((ord(char) - base + shift) % 26 + base)

    else:

        cipher += char

print("Encoded text:", cipher)

**Output:**

****

**8. Create a function to remove HTML tags from a string.**

**Program:**

html = "<p>Hello <b>World</b></p>"

text = ""

inside\_tag = False

for char in html:

    if char == "<":

        inside\_tag = True

    elif char == ">":

        inside\_tag = False

    elif not inside\_tag:

        text += char

print("Without HTML Tags:", text)

**Output:**

****

**9. Develop a function that finds the longest palindromic substring.**

**Program:**

s = "babad"

longest = ""

for i in range(len(s)):

    for j in range(i, len(s)):

        sub = s[i:j+1]

        if sub == sub[::-1] and len(sub) > len(longest):

            longest = sub

print("Longest palindrome:", longest)

**Output:**

****

**10. Implement a string compression algorithm (e.g., aabcccccaaa -> a2b1c5a3).**

**Program:**

s = "aabcccccaaa"

compressed = ""

count = 1

for i in range(1, len(s)):

    if s[i] == s[i - 1]:

        count += 1

    else:

        compressed += s[i - 1] + str(count)

        count = 1

compressed += s[-1] + str(count)

print("Compressed string:", compressed)

**Output:**

****

**File Handling**

**1. Write a script that reads a file and prints each line with line numbers.**

**Program:**

with open("Python\_Assignment/File\_Handling/Description.txt", "r") as file:

    for i, line in enumerate(file, 1):

        print(f"{i}: {line.strip()}")

**Output:**

****

**2. Create a function to count the number of words in a text file.**

**Program:**

with open("Python\_Assignment/File\_Handling/Description.txt", "r") as file:

    text = file.read()

    words = text.split()

    print("Word count:", len(words))

**Output:**

****

**3. Write a program to copy the contents of one file into another.**

**Program:**

with open("Python\_Assignment/File\_Handling/Description.txt", "r") as src, open("Python\_Assignment/File\_Handling/Copy.txt", "w") as dst:

    for line in src:

        dst.write(line)

print("File copied.")

**Output:**

****

**4. Implement a script that appends user input to a file.**

**Program:**

user\_input = "This is a new line from 4\_AUI.py"

with open("Python\_Assignment/File\_Handling/Description.txt", "a") as file:

    file.write(user\_input + "\n")

print("Line appended.")

**Output:**

****

**5. Develop a function to read a file and remove all empty lines.**

**Program:**

with open("Python\_Assignment/File\_Handling/Description.txt", "r") as file:

    lines = file.readlines()

with open("Python\_Assignment/File\_Handling/clean.txt", "w") as out:

    for line in lines:

        if line.strip():

            out.write(line)

print("Empty lines removed.")

**Output:**

****

**6. Create a script to merge multiple text files into one.**

**Program:**

filenames = ["Python\_Assignment/File\_Handling/Description.txt", "Python\_Assignment/File\_Handling/Copy.txt"]

with open("Python\_Assignment/File\_Handling/merge.txt", "w") as outfile:

    for name in filenames:

        with open(name, "r") as infile:

            outfile.write(infile.read() + "\n")

print("Files merged.")

**Output:**

****

**7. Write a program to read a CSV file and calculate column averages.**

**Program:**

import csv

with open("Python\_Assignment/File\_Handling/temp.csv", "r") as file:

    reader = csv.reader(file)

    next(reader)

    sums = []

    count = 0

    for row in reader:

        nums = list(map(float, row))

        if not sums:

            sums = [0] \* len(nums)

        for i in range(len(nums)):

            sums[i] += nums[i]

        count += 1

    averages = [s / count for s in sums]

    print("Column averages:", averages)

**Output:**

****

**8. Implement a program that creates a log file with timestamped entries.**

**Program:**

from datetime import datetime

log\_entry = "Saksham was here and completed the assignment on File Handling in Python."

timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

with open("Python\_Assignment/File\_Handling/log.txt", "a") as log:

    log.write(f"[{timestamp}] {log\_entry}\n")

print("Log updated.")

**Output:**

****

**9. Write a script that reads a file and counts the frequency of each character.**

**Program:**

from collections import Counter

with open("Python\_Assignment/File\_Handling/character.txt", "r") as file:

    text = file.read()

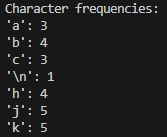
freq = Counter(text)

print("Character frequencies:")

for char, count in freq.items():

    print(f"{repr(char)}: {count}")

**Output:**

****

**10. Create a function to replace specific words in a file with user-provided values.**

**Program:**

**Output:**

**Object-Oriented Programming (OOP) in Python**

**1. Create a class representing a Bank Account with deposit and withdraw methods.**

**Program:**

class BankAccount:

    def \_\_init\_\_(self, balance=0):

        self.balance = balance

    def deposit(self, amount):

        self.balance += amount

    def withdraw(self, amount):

        if amount <= self.balance:

            self.balance -= amount

acc = BankAccount(10000)

acc.deposit(5000)

acc.withdraw(100)

print("Balance:", acc.balance)

**Output:**

****

**2. Implement a class for a Rectangle with methods to calculate area and perimeter.**

**Program:**

class Rectangle:

    def \_\_init\_\_(self, width, height):

        self.width = width

        self.height = height

    def area(self):

        return self.width \* self.height

    def perimeter(self):

        return 2 \* (self.width + self.height)

r = Rectangle(5, 3)

print("Area:", r.area(), "Perimeter:", r.perimeter())

**Output:**

****

**3. Create a Student class that stores name and grades, and can compute the average.**

**Program:**

class Student:

    def \_\_init\_\_(self, name, grades):

        self.name = name

        self.grades = grades

    def average(self):

        return sum(self.grades) / len(self.grades)

s = Student("Saksham", [90, 80, 85, 95, 100])

print("Name:", s.name)

print("Average:", s.average())

**Output:**

****

**4. Implement inheritance between a base class Animal and subclasses Dog and Cat.**

**Program:**

class Animal:

    def message(self):

        return "Animal class"

class Dog(Animal):

    def message(self):

        return "Dog Class"

class Cat(Animal):

    def message(self):

        return "Cat Class"

print(Dog().message(),Cat().message(),Animal().message())

**Output:**

****

**5. Write a class with a class variable shared among all instances.**

**Program:**

class Counter:

    count = 0

    def \_\_init\_\_(self):

        Counter.count += 1

a = Counter()

b = Counter()

print("Instances created:", Counter.count)

**Output:**

****

**6. Use magic methods to implement a custom class that mimics a list.**

**Program:**

class MyList:

    def \_\_init\_\_(self):

        self.data = []

    def \_\_getitem\_\_(self, index):

        return self.data[index]

    def \_\_setitem\_\_(self, index, value):

        self.data[index] = value

    def \_\_len\_\_(self):

        return len(self.data)

    def append(self, value):

        self.data.append(value)

lst = MyList()

lst.append(10)

lst.append(20)

print(lst[0], "Length:", len(lst))

**Output:**

****

**7. Implement method overriding in a subclass.**

**Program:**

class Parent:

    def greet(self):

        print("Hello from Parent")

class Child(Parent):

    def greet(self):

        print("Hello from Child")

Child().greet()

**Output:**

****

**8. Create a class with private attributes and getter/setter methods.**

**Program:**

class Person:

    def \_\_init\_\_(self, name):

        self.\_\_name = name

    def get\_name(self):

        return self.\_\_name

    def set\_name(self, name):

        self.\_\_name = name

p = Person("Saksham")

p.set\_name("Om")

print("Name:", p.get\_name())

**Output:**

****

**9. Design a class structure for a library system with books and members.**

**Program:**

class Book:

    def \_\_init\_\_(self, title):

        self.title = title

class Member:

    def \_\_init\_\_(self, name):

        self.name = name

        self.books = []

    def borrow(self, book):

        self.books.append(book)

book1 = Book("Fourty Rules of Love")

book2 = Book("The Alchemist")

member = Member("Saksham")

member.borrow(book1)

print("Books borrowed by", member.name + ":", [b.title for b in member.books])

**Output:**

****

**10. Implement multiple inheritance and demonstrate method resolution order.**

**Program:**

class A:

    def show(self):

        print("A")

class B(A):

    def show(self):

        print("B")

class C(A):

    def show(self):

        print("C")

class D(B, C):  # MRO: D → B → C → A

    pass

d = D()

d.show()

**Output:**

****

**Exception Handling**

**1. Write a script that handles division by zero using try/except.**

**Program:**

**Output:**

**2. Create a program that handles invalid input from the user.**

**Program:**

**Output:**

**3. Implement nested try/except blocks to handle multiple error types.**

**Program:**

**Output:**

**4. Develop a function that raises a custom exception for invalid age input.**

**Program:**

**Output:**

**5. Write a script using try/except/finally to open and safely close a file.**

**Program:**

**Output:**

**6. Create a context manager using a class to handle file open/close with exception support.**

**Program:**

**Output:**

**7. Write a program that logs exceptions to a file.**

**Program:**

**Output:**

**8. Implement exception chaining using raise from syntax.**

**Program:**

**Output:**

**9. Develop a decorator that catches and logs exceptions in any function.**

**Program:**

**Output:**

**10. Define multiple custom exceptions and handle them in different scenarios.**

**Program:**

**Output:**

**Iterators and Generators**

**1. Create a generator that yields even numbers up to a given number.**

**Program:**

**Output:**

**2. Write an iterator class that returns Fibonacci numbers up to n terms.**

**Program:**

**Output:**

**3. Develop a generator that yields the prime numbers under 100.**

**Program:**

**Output:**

**4. Implement a custom iterable class for iterating characters of a string.**

**Program:**

**Output:**

**5. Write a generator to yield lines from a file one by one.**

**Program:**

**Output:**

**6. Create a nested generator to yield Cartesian product of two lists.**

**Program:**

**Output:**

**7. Write a generator expression to filter out palindromes from a list of words.**

**Program:**

**Output:**

**8. Implement an infinite generator for Fibonacci numbers.**

**Program:**

**Output:**

**9. Develop a program using generator pipelines to process a text file.**

**Program:**

**Output:**

**10. Create a context manager that uses a generator to manage a resource.**

**Program:**

**Output:**

**Functional Programming**

**1. Write a lambda function to compute the square of a number and use it in a list comprehension.**

**Program:**

**Output:**

**2. Use `map()` to convert all strings in a list to uppercase.**

**Program:**

**Output:**

**3. Use `filter()` to remove empty strings from a list.**

**Program:**

**Output:**

**4. Write a function that takes a function and a list, and applies it to all items.**

**Program:**

**Output:**

**5. Create a higher-order function that returns a power function.**

**Program:**

**Output:**

**6. Implement currying using closures in Python.**

**Program:**

**Output:**

**7. Use `reduce()` to calculate the factorial of a number.**

**Program:**

**Output:**

**8. Develop a function that uses both `map()` and `filter()` in a pipeline.**

**Program:**

**Output:**

**9. Create a decorator that memoizes the result of a function.**

**Program:**

**Output:**

**10. Implement a generic compose() function that chains multiple functions.**

**Program:**

**Output:**

**Regular Expressions**

**1. Write a regex to extract all email addresses from a string.**

**Program:**

**Output:**

**2. Use regex to validate a phone number format (e.g., 123-456-7890).**

**Program:**

**Output:**

**3. Create a script to extract all hashtags from a given text.**

**Program:**

**Output:**

**4. Use regex to find all words starting with a capital letter.**

**Program:**

**Output:**

**5. Write a regex to replace all whitespace with hyphens.**

**Program:**

**Output:**

**6. Build a regex to validate complex passwords (at least 1 digit, 1 symbol, 8+ chars).**

**Program:**

**Output:**

**7. Write a script using regex to extract dates in dd-mm-yyyy format.**

**Program:**

**Output:**

**8. Create a regex pattern that identifies valid IPv4 addresses.**

**Program:**

**Output:**

**9. Use regex to parse and extract key-value pairs from a query string.**

**Program:**

**Output:**

**10. Implement a tokenizer using regular expressions that separates punctuation from words.**

**Program:**

**Output:**