**Python programming file**



**LAMRIN TECH SKILLS UNIVERSITY PUNJAB**

**Submitted by:**

**Name of the student: GANDEPALLI UPENDRA**

**Roll No : 24100030020**

**Branch : CYBER SECURITY**

**Submitted to : sukhwinder kaur**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Lab Exercise(List of Programs)** | **Date** | **Remarks** |  |  |  |
| **1.** | **Write a program to declare a string and perform all the basic functions of string on it.** | **27.08.2024** |  |  |  |  |
| **2.** | **Write a program to declare a list and perform all the basic functions of list on it.** | **27.08.2024** |  |  |  |  |
| **3.** | **Write a program to declare a tuple and perform all the basic functions of tuple on it.** | **02.09.2024** |  |  |  |  |
| **4.** | Write a program to declare a dictionary and perform all the basic functions of dictionary on it. | **02.09.2024** |  |  |  |  |
| **5.** | Write a program to declare a set and perform all the basic functions of set on it. | **13.09.2024** |  |  |  |  |
| **6.** | **Write a program to give grades according to percentage using if elif** |  |  |  |  |  |
| **7.** | **Write a program to print n number of natural numbers using for loop** | **13.09.2024** |  |  |  |  |
| **8.** | Write a program to draw any star pattern | **20.09.2024** |  |  |  |  |
| **9.** | **Write a program to implement calculator using class** | **20.09.2024** |  |  |  |  |
| **10.** | **Write a program to reverse a number using 2 methods** | **20.09.2024** |  |  |  |  |
| **11.** | **WAP to check whether a number is Armstrong number or not** | **04.10.2024** |  |  |  |  |
| **12.** | **WAP to create address book** | **04.10.2024** |  |  |  |  |
| **13.** | **WAP to implement math, data time, random built in module. Find GCD of any two number** | **11.10.2024** |  |  |  |  |
| **14.** | WAP to merge dictionary | **11.10.2024** |  |  |  |  |
| **15.** | WAP to output of any expression using eval method | **25.10.2024** |  |  |  |  |
| **16.** | **WAP to implement calculator using GUI** | **17.11.2024** |  |  |  |  |
| **17.** | **WAP to apply all operations of NumPy** | **17.11.2024** |  |  |  |  |
| **18.** | **WAP to implement charts using matplotlib** | **22.11.2024** |  |  |  |  |
| **19.** | **WAP to generate Pythagorus triplet** | **22.11.2024** |  |  |  |  |
| **20.** | **WAP to implement OS module** | **22.11.2024** |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**PROGRAM NO. 1**

**AIM : Write a program to declare a string and perform all the basic functions of string on it.**

#### Program: -

string = "Hello, world!" print(len(string)) print(string.upper()) print(string.lower())

print(string.replace("everyone!", "world!")) print(string.find("world!")) print(string.startswith("Hello")) print(string.endswith("!")) print(string[2:10:1])

print(string.split(", "))

**Output:-**

***A screenshot of a computer

Description automatically generated***

### PROGRAM NO. 2

**AIM : Write a program to declare a list and perform all the basic functions of list on it.**

#### Program:-

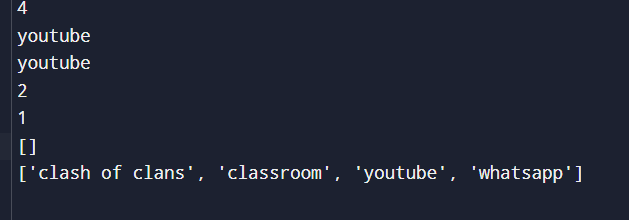
list1 = ["whatsapp","instagram","youtube","file manager"] print(len(list1))

print(list1[2]) print(list1[-2])

list1.append("classroom")

list1.insert(5,"clash of clans") list1.remove("file manager") print(list1.index("youtube")) print(list1.count("whatsapp")) list1.pop(1)

list1.reverse() list2 = list1.copy() list1.clear() print(list1) print(list2) **Output:-**



**PROGRAM NO.3**

### AIM :Write a program to declare a tuple and perform all the basic functions of tuple on it.

#### Program:-

tuple1 = ("whatsapp","instagram","clash of clans","file manager") print(len(tuple1))

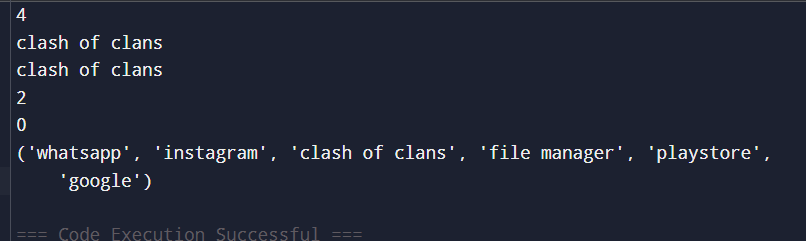
print(tuple1[2]) print(tuple1[-2])

print(tuplel.index("clash of clans”))

print(tuple1.count(2))

tuple2=tuple1+("playstore","google") print(tuple2)

#### Output:-

******

**PROGRAM NO.4**

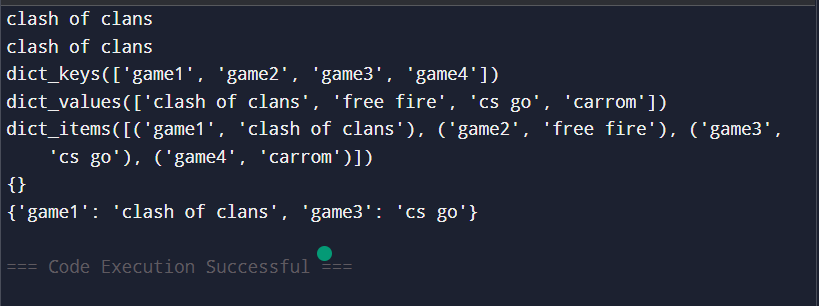
### AIM : Write a program to declare a dictionary and perform all the basic functions of dictionary on it.

#### Program:-

dict1 = {"game1": "clash of clans", "game2": "free fire","game3" : "call of duty"} print(dict1["game1"])

print(dict1.get("game1")) dict1["game3"] = "cs go" dict1["game4"] = "carrom" print(dict1.keys()) print(dict1.values()) print(dict1.items()) dict1.pop("game2") dict1.popitem()

dict2 = dict1.copy() dict1.clear() print(dict1) print(dict2) **Output:-**



**PROGRAM NO.5**

### AIM : Write a program to declare a set and perform all the basic functions of set on it.

#### Program:-

set1 = {"whatsapp","instagram","youtube","file manager"} set2 = {"carrom","cs go","valorant","free fire"} print(len(set1))

set1.add("ludo") set1.remove("file manager") print(set1.union(set2)) print(set1.intersection(set2)) print(set1.difference(set2)) print(set1.issubset(set2)) print(set1.issuperset(set2)) set1.clear()

print(set2) print(set1)

**Output:-**

A screenshot of a computer screen

Description automatically generated

### PROGRAM NO. 6

**AIM** : **Write a program to give grades according to percentage using if elif**

**Program:-**

percentage = float(input("Enter your percentage: ")) if percentage >= 95:

print("Grade: A") elif percentage >= 85: print("Grade: B")

elif percentage >= 75: print("Grade: C")

elif percentage >= 60: print("Grade: D") else:

print("Grade: F")

### Output:-

***A screen shot of a computer

Description automatically generated***

### PROGRAM NO. 7

**AIM : Write a program to print n number of natural numbers using for loop**

#### Program:-

n = int(input("Enter a number: ")) for i in range(1, n + 1):

print(i)

#### Output:-

***A screenshot of a computer

Description automatically generated***

**PROGRAM NO. 8**

### AIM:Write a program to draw any star pattern

#### Program:-

i=1

while i<=9: print(i\*"\*") i=i+1

#### Output:-

***A computer screen shot of a computer code

Description automatically generated***

**PROGRAM NO. 9**

### AIM : Write a program to implement calculator using class

#### Program:-

class Calculator:

def add(self, x, y):

return x + y

def subtract(self, x, y):

return x - y

def multiply(self, x, y):

return x \* y

def divide(self, x, y):

if y != 0:

return x / y

else:

return "Cannot divide by zero"

while True:

cal = Calculator()

print("Enter 0 for addition")

print("Enter 1 for subtraction")

print("Enter 2 for multiplication")

print("Enter 3 for division")

print("Enter 4 to exit")

choice = int(input("Choose an operation: "))

if choice == 4:

print("Exiting the calculator. Goodbye!")

break

x = float(input("Enter first value: "))

y = float(input("Enter second value: "))

if choice == 0:

result = cal.add(x, y)

elif choice == 1:

result = cal.subtract(x, y)

elif choice == 2:

result = cal.multiply(x, y)

elif choice == 3:

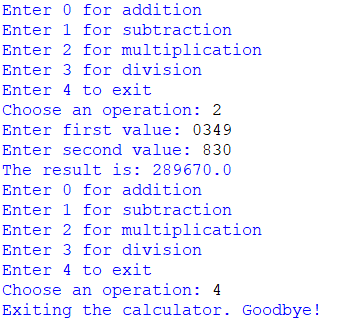
result = cal.divide(x, y)

else:

result = "Invalid choice"

print("The result is:", result)

#### output:-

****

**PROGRAM NO. 10**

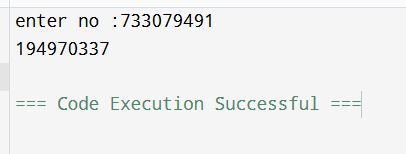
**AIM : Write a program to reverse a number using 2 methods**

#### Progrm:Method1:

num = (input("enter no :"))

print(str(num)[::-1])

#### output :-

******

**Method 2:-**

num = int(input("ENTER NO.: "))

reversed\_num = 0

while num != 0:

digit = num % 10

reversed\_num = reversed\_num \* 10 + digit

num //= 10

print("Reversed Number: " + str(reversed\_num))

#### Output:-

v A number on a white background

Description automatically generated

**PROGRAM NO. 11**

### AIM : WAP to check whether a number is Armstrong number or not

#### Program :-

num = int(input("Enter a number: ")) sum = 0

temp = num while temp > 0:

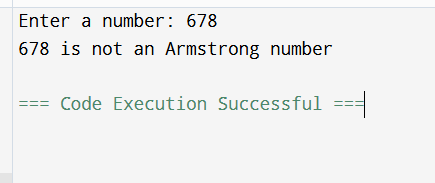
digit = temp % 10 sum += digit \*\* 3 temp //= 10

if num == sum:

print(num,"is an Armstrong number") else:

print(num,"is not an Armstrong number")

#### output:-

******

**PROGRAM NO. 12**

# AIM : WAP to create address book

**Program :-**

address\_book = []

def add\_contact(name, phone, address, email):

contact = {

'name': name,

'phone': phone,

'address': address,

'email': email

}

address\_book.append(contact)

print(f"Contact {name} added successfully.")

def view\_contacts():

if not address\_book:

print("No contacts found.")

else:

for contact in address\_book:

print(f"Name: {contact['name']}, Phone: {contact['phone']}, Address: {contact['address']}, Email: {contact['email']}")

while True:

print("\nAddress Book Menu:")

print("1. Add Contact")

print("2. View Contacts")

print("3. Exit")

choice = input("Enter your choice: ")

if choice == '1':

name = input("Enter name: ")

phone = input("Enter phone number: ")

address = input("Enter address: ")

email = input("Enter email ID: ")

add\_contact(name, phone, address, email)

elif choice == '2':

view\_contacts()

elif choice == '3':

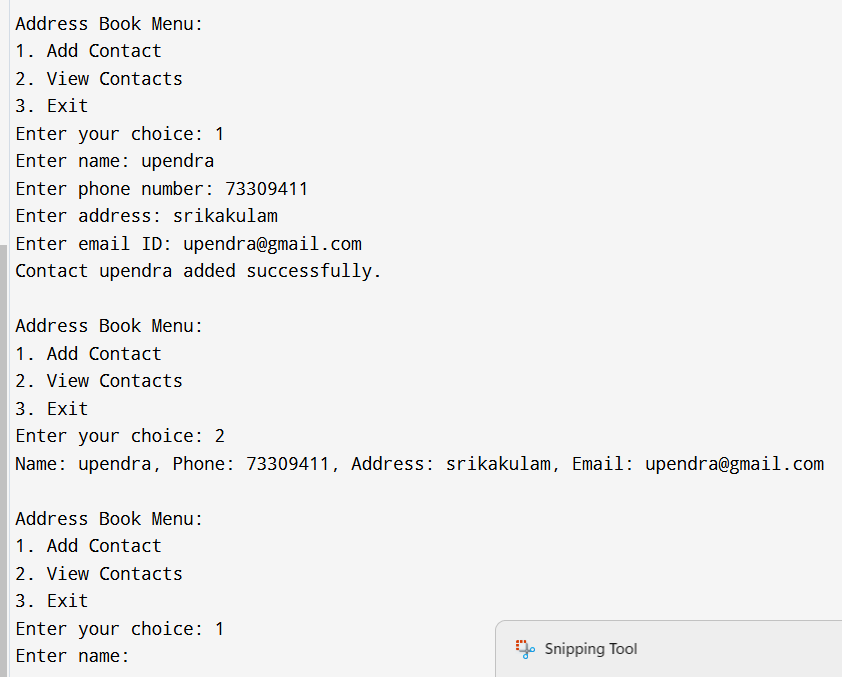
print("Exiting the Address Book.")

break

else:

print("Invalid choice. Please try again.")

#### output:-

******

**PROGRAM NO. 13**

# AIM : WAP to implement math, data time, random built in module. Find GCD of any two number.

#### Program :-

#### import math

#### import datetime

#### import random

#### print(math.sqrt(26))

#### print(datetime.datetime.now())

#### print(random.randint(3, 10))

#### num1 = 36

#### num2 = 18

#### print(math.gcd(num1, num2))

#### output:-

***A number on a white background

Description automatically generated***

**PROGRAM NO. 14**

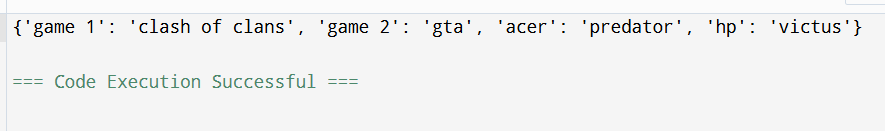
# AIM : WAP to merge dictionary

#### Program :-

dict1 = {'game 1': 'clash pof clans', 'game 2': 'gta'}

dict2 = {'acer': 'predator', 'hp': 'victus'} merged\_dict = dict1.copy() merged\_dict.update(dict2) print(merged\_dict)

#### output:-

******

**PROGRAM NO. 15**

# AIM : WAP to output of any expression using eval method

**Program :**

while True:

expression = input("Enter a mathematical expression (or type 'exit' to quit): ")

if expression.lower() == 'exit':

print("Exiting the program.")

break

try:

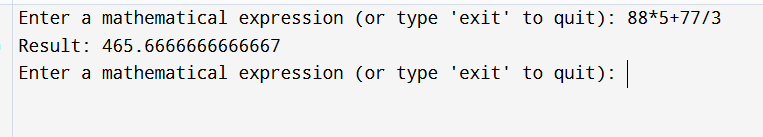
result = eval(expression)

print(f"Result: {result}")

except Exception as e:

print(f"Error: {e}")

#### output:-

******

**PROGRAM NO. 16**

**AIM : WAP to implement calculator using GUI**

**Program :**

import tkinter as tk

import tkinter.ttk as ttk

window = tk.Tk()

window.title("Simple Calculator")

window.geometry("300x400")

def calculate():

operation = combo.get()

num1 = int(numOneEntry.get())

num2 = int(numTwoEntry.get())

match operation:

case "Addition":

output = num1 + num2

case "Subtraction":

output = num1 - num2

case "Multiplication":

output = num1 \* num2

case "Division":

output = num1 / num2

result.config(text=output)

label = tk.Label(window, text="Simple Calculator")

label.pack()

frame = tk.Frame(window)

frame.pack()

numOneLabel = tk.Label(frame, text="Enter number 1: ")

numOneLabel.grid(row=0, column=0)

numOneEntry = tk.Entry(frame)

numOneEntry.grid(row=0, column=1)

numTwoLabel = tk.Label(frame, text="Enter number 2: ")

numTwoLabel.grid(row=1, column=0, pady=5)

numTwoEntry = tk.Entry(frame)

numTwoEntry.grid(row=1, column=1)

combo = ttk.Combobox(window, values=("Addition", "Subtraction", "Multiplication", "Division"))

combo.pack(pady=5)

combo.current(0)

result = tk.Label(window, text="Result will be displayed here")

result.pack(pady=10)

submit = tk.Button(window, text="Calculate", command=calculate)

submit.pack(pady=10)

window.mainloop()

#### output:-

#### 

**PROGRAM NO. 17**

**AIM : WAP to apply all operations of NumPy**

**Program :**

import numpy as np

# Creating a numpy array

arr = np.array([[1, 2, 3], [4, 5, 6]])

print("Size of the array:", arr.size)# Size of the array

# Dimensions of the array

print("Number of dimensions:", arr.ndim)

# Creating a 5x4 zero matrix

zeroes = np.zeros((5, 4))

print("5x4 Zero matrix:\n", zeroes)

# Creating a 3x3 identity matrix

ones = np.identity(3)

print("3x3 Identity matrix:\n", ones)

# Prints the sum of all the elements of the numpy array

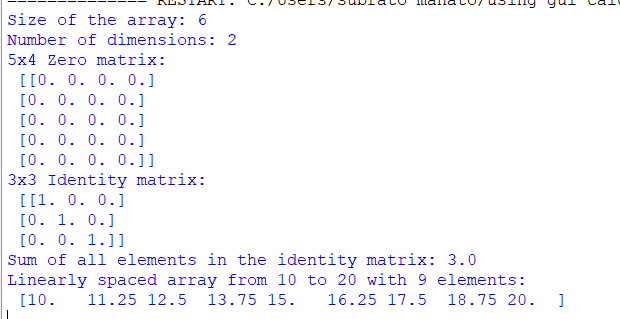
print("Sum of all elements in the identity matrix:", np.sum(ones))

# Creating a linearly spaced array

array = np.linspace(10, 20, num=9, endpoint=True, axis=0)

print("Linearly spaced array from 10 to 20 with 9 elements:\n", array)

#### output:-



**PROGRAM NO. 18**

**AIM : WAP to implement charts using matplotlib**

**Program :**

import matplotlib.pyplot as plt

months = ['January', 'February', 'March', 'April', 'May',

'June', 'July', 'August', 'September', 'October', 'November', 'December']

sales = [600, 800, 750, 900, 1100, 1300, 1200, 1400, 1500, 1600, 1300, 1100]

max\_sales = max(sales)

max\_index = sales.index(max\_sales)

plt.figure(figsize=(10, 5))

plt.plot([i[:3] for i in months], sales, label="Sales data", marker='o')

plt.scatter([months[max\_index][:3]], [max\_sales], color='red', zorder=5, label="Highest Sale")

plt.title("Monthly Sales")

plt.xlabel("Months")

plt.ylabel("Sales")

plt.grid(True)

plt.legend()

plt.show()

fig = plt.figure(figsize=(20, 8), facecolor="lightgrey")

fig.suptitle("Quarterly Reports")

aa = fig.add\_subplot(141)

ab = fig.add\_subplot(142)

ac = fig.add\_subplot(143)

ad = fig.add\_subplot(144)

aa.plot(months[:3], sales[:3], marker='o', label='Q1')

aa.set\_title('Q1 Sales')

aa.set\_xlabel('Months')

aa.set\_ylabel('Sales')

aa.grid(True)

ab.plot(months[3:6], sales[3:6], marker='o', label='Q2')

ab.set\_title('Q2 Sales')

ab.set\_xlabel('Months')

ab.set\_ylabel('Sales')

ab.grid(True)

ac.plot(months[6:9], sales[6:9], marker='o', label='Q3')

ac.set\_title('Q3 Sales')

ac.set\_xlabel('Months')

ac.set\_ylabel('Sales')

ac.grid(True)

ad.plot(months[9:12], sales[9:12], marker='o', label='Q4')

ad.set\_title('Q4 Sales')

ad.set\_xlabel('Months')

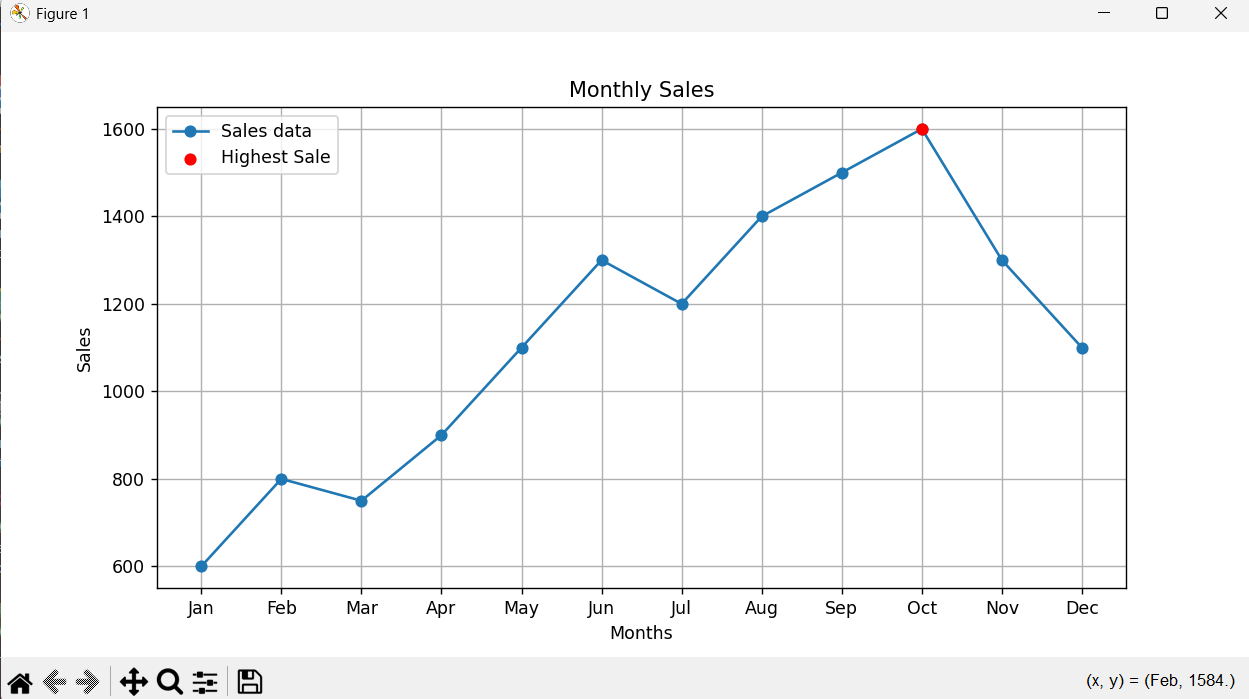
ad.set\_ylabel('Sales')

ad.grid(True)

fig.tight\_layout(rect=[0, 0, 1, 0.95])

plt.show()

**A graph of a line

Description automatically generated with medium confidenceoutput:-**

**PROGRAM NO. 19**

**AIM : WAP to generate Pythagorus triplet**

**Program:**

number = int(input("Enter a number: "))

for m in range(1, int(number\*\*0.5) + 1):

for n in range(1, m):

a = m\*\*2 - n\*\*2

b = 2 \* m \* n

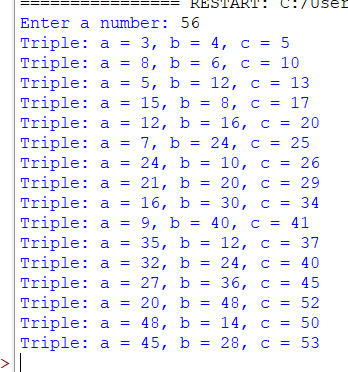
c = m\*\*2 + n\*\*2

if c > number:

break

print(f"Triple: a = {a}, b = {b}, c = {c}")

**output:**



**PROGRAM NO. 20**

**AIM : WAP to implement OS module**

**Program :**

import os

# Getting the current working directory

cwd = os.getcwd()

print(f"Current working directory is: {cwd}")

# Change the current working directory

#os.chdir("/home/zephex/Documents")

print(f"Working directory after changing the directory is: {os.getcwd()}")

# Creating a new directory

os.mkdir("new-directory")

print("Created a new directory named 'new-directory'.")

# Renaming the same directory

os.rename("new-directory", "temp")

print("Renamed 'new-directory' to 'temp'.")

# Deleting the directory

os.rmdir("temp") # Use os.rmdir instead of os.remove for directories

print("Deleted the directory named 'temp'.")

**output**:

