1.Write a Python program to find sum of all elements, largest element and smallest element from the Tuple.Input a tuple from User

t1=()

s=int(input("Enter the size:"))

for k in range(s):

item=int(input())

t1=t1+(item,)

print("Sum of all element:",sum(t1))

print("Minimum values in t1:",min(t1))

print("Maximum values in t1:",max(t1))

2. Write a Python program to find the repeated items of a tuple. Input a tuple from User.

t1=()

s=int(input("Enter the size:"))

for k in range(s):

item=int(input())

t1=t1+(item,)

element=int(input("Enter the element to count:"))

count=t1.count(element)

print(count)

3. Print the number in words for Example: 1234 => One Two Three Four. Input a tuple from User.

'''l1 = ['Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine']

t1 = ()

s = int(input("Enter the size of the tuple: "))

for k in range(s):

item = int(input(f"Enter number {k+1}: "))

t1 = t1 + (item,)

for number in t1:

n = number

words = []

if n == 0:

words.append(l1[0])

else:

while n > 0:

d = n % 10

words.append(l1[d])

n = n // 10

words.reverse()

print(number, "=>", end=" ")

for word in words:

print(word, end=" ")

print()

4. Write a Python program to display Even numbers from a tuple. Input a tuple from User.

t1 = ()

s = int(input("Enter the size of the tuple: "))

for i in range(s):

item = int(input(f"Enter element {i+1}: "))

t1 = t1 + (item,)

print("Even numbers in the tuple are:")

for num in t1:

if num % 2 == 0:

print(num, end=" ")

5. Write a Python program to create a set, add member(s) in a set and remove one item from set.

size=int(input("Enter the size:"))

s=set()

for i in range(size):

item=int(input())

s.add(item)

#s.add(input())

print(s)

item=int(input("Enter item to discard:"))

s.discard(item)

print(s)

6.Write a Python program to perform following operations on set: intersection of sets, union of sets, set difference, symmetric difference, clear a set.

set1 = set()

n1 = int(input("Enter the number of elements in Set 1: "))

for i in range(n1):

item = int(input(f"Enter element {i+1} for Set 1: "))

set1.add(item)

set2 = set()

n2 = int(input("Enter the size of elements in Set 2: "))

for i in range(n2):

item = int(input(f"Enter size {i+1} for Set 2: "))

set2.add(item)

print("Set1:",set1)

print("Set2:",set2)

print("Intersections of sets:",set1&set2)

print("Union of sets:",set1|set2)

print("Difference of set1 and set2:",set1-set2)

print("Symmetric Difference:",set1^set2)

7.Write a Python program to find maximum and the minimum value in a set and to find the length of a set.

size=int(input("Enter the size:"))

s=set()

for i in range(size):

item=int(input())

s.add(item)

#s.add(input())

print(s)

print("Maximum number is set:",max(s))

print("Minimum number is set:",min(s))

print("length of set:",len(s))

8. Write a Python program to Input elements in the Set and find out only Even Numbers from Set and Display.

size=int(input("Enter the size:"))

s=set()

for i in range(size):

item=int(input())

s.add(item)

#s.add(input())

print(s)

print("Even number for sets:")

for num in s:

if num % 2 == 0:

print(num, end=" ")

9.Input two Sets from User and find the common elements between them without using any inbuilt function.

set1 = set()

n1 = int(input("Enter the number of elements in Set 1: "))

for i in range(n1):

item = int(input(f"Enter element {i+1} for Set 1: "))

set1.add(item)

set2 = set()

n2 = int(input("Enter the size of elements in Set 2: "))

for i in range(n2):

item = int(input(f"Enter size {i+1} for Set 2: "))

set2.add(item)

common = []

for i in set1:

for j in set2:

if i == j:

common.append(i)

if common:

print("Common elements are:")

for item in common:

print(item, end=" ")

else:

print("No common elements found.")

9. Write a Python script to sort (ascending and descending) a dictionary by value.

d=dict()

cnt=int(input("How many pass:"))

for i in range(cnt):

k=input("key:")

v=input("Value:")

d[k]=v

d1=sorted(d.key(),reverse=False)

d2=sorted(d.key(),reverse=True)

print(d1)

print(d2)

10.Write a Python program to print all even numbers between 1 to 100 using while loop.

num=1

print("Even nymber from 1 -100")

while num <= 100:

if num % 2 == 0:

print(num)

num=num+1

11. Write a Python program to check if the input year is a leap year of not.

year=int(input("enter year:-"))

if(year%4==0):

print("It is leap year")

else:

print("it is not")

12. Write a Python program to print the table of given no.

no=int(input("enter number :-"))

for i in range(1,11):

print(f"{no}\*{i}={no\*i}")

13. Write a Python program to print table from 1 to 10 numbers.

for i in range(1, 10 + 1):

for j in range(i, (i \* 10) + 1):

if j % i == 0:

print(j, end="\t")

print()

14. Write a Python program to check if the input number is prime or not.

no=int(input("enter number:-"))

flag=0

for i in range(2,no):

if(no%i==0):

flag=1

if(flag==0):

print("it i prime number")

elif (flag==1):

print("it is not")

15. Write a Python program to find the sum of first 10 natural numbers using for loop.

sum1=0

for i in range(1,11):

sum1=sum1+i

print(sum1)

16. Write a Python program to print Fibonacci series.

no1=0

no2=1

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

print(no1,"\n",no2)

for i in range(2,no+1):

no3=no1+no2

print(no3)

no1=no2

no2=no3

17. Write a Python program to calculate factorial of a number

no=int(input("enter number:-"))

fact=1

for i in range(1,no+1):

fact=fact\*i

print("factorial:-",fact)

18 Write a Python Program to Reverse a Given Number/String.

num=int(input("enter number:-"))

rev=0

while num > 0:

rem=num%10

rev=rem+rev\*10

num=num//10

print(rev)

str=input("enter string:=")

strrev=""

for char in str:

strrev=char+strrev

print("reverse string:-",strrev)

19.Write a Python program to input a number and find the sum of digits in a number.

num=int(input("enter number:-"))

sum1=0

while num > 0:

rem=num%10

sum1=rem+sum1

num=num//10

print(sum1)

20. Write a Python program that takes a number and checks whether it is a palindrome or not.

num=int(input("Enter number:-"))

num1=num

rev=0

while num > 0:

rem=num%10

rev=rem+rev\*10

num=num//10

print(rev)

if(rev==num1):

print("palindrome")

else:

print("not palindrome")

21. Write a Python program to sum all the items in a list. Input a list from User.

l=[]

size=int(input("Enter the size:"))

for i in range(size):

item=int(input())

l.append(item)

print("Sum:",sum(l))

22. Write a Python program to multiplies all the items in a list. Input a list from User.

l=[]

size=int(input("Enter the size:"))

for i in range(size):

item=int(input())

l.append(item)

mul=1

for i in l:

mul=mul\*i

print("Multiplication:",mul)

23. Write a Python program to get the largest number and smallest number from a list. Input a list from User.

l=[]

size=int(input("Enter the size:"))

for i in range(size):

item=int(input())

l.append(item)

print(max(l))

print(min(l))

24. Write a Python program to reverse a list. Input a list from User.

l=[]

size=int(input("Enter the size:"))

for i in range(size):

item=int(input())

l.append(item)

l.reverse()

print(l)

25. Write a Python program to find common items from two lists. Input a list from User.

l=[]

size=int(input("Enter the size:"))

for i in range(size):

item=int(input())

l.append(item)

print("Even number from the list")

for i in l:

if i % 2 == 0:

print(i)

26. Write a Python program to find common items from two lists. Input a list from User.

l=[]

size=int(input("Enter the size:"))

for i in range(size):

item=int(input())

l.append(item)

l1=[]

size1=int(input("Enter the size:"))

for i in range(size1):

item=int(input())

l1.append(item)

l3 = []

for i in l:

if i in l1 and i not in l3:

l3.append(i)

print("Common elements:", l3)

27. Write a Python script to concatenate the following dictionaries to create a new one.

o Sample Dictionary:

 dic1 = {1:10, 2:20}

 dic2 = {3:30, 4:40}

 dic3 = {5:50,6:60}

d1={1:10,2:20}

d2={3:30,4:40}

d3={5:50,6:60}

dict3=d1|d2|d3

print(dict3)

28. Write a Python program to combine two dictionary adding values for common Keys.

d1 = {'a': 100, 'b': 200, 'c':300}

 d2 = {'a': 300, 'b': 200, 'd':400}

d1={1:10,2:20}

d2={3:30,4:40}

combined={}

for key,value in d1.items():

combined[key]=value

for key,value in d2.items():

if key in combined:

combined[key]=combined[key]+value

else:

combined[key]=value

print(combined)

29. Write a Python program to print all unique values in a dictionary.

Sample Data: [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"},

 {"VII":"S005"}, {"V":"S009"}, {"VIII":"S007"}]

# Sample data

data = [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"},

{"VII":"S005"}, {"V":"S009"}, {"VIII":"S007"}]

# Use a set to store unique values

unique\_values = set()

for item in data:

for value in item.values():

unique\_values.add(value)

print("Unique Values:", unique\_values)

30. Write a Python program to find the highest 3 values in a dictionary.

d1={1:10,2:20,3:30,4:40,5:50}

d13=sorted(d1.key(),reverse=True)

print(d13[:3])

31. Python Program to input values from the user in a dictionary where keys will be automatically generated using range.

num=int(input("How many eleemnts youu want:-"))

d1={}

for i in range(num):

value=input("Enter value for elemnts-")

d1[i]=value

print(d1)

32. Write a Python program to generate a random float where the value is between 5 and 50 using the Python math module.

import math

import random

num=random.uniform(5,50)

num2=math.floor(num\*100)/100

print(num2)

33. Python program to convert given string into another case. If the string is in uppercase, convert to lowercase and vice-versa.

str1=input("Enter string:-")

if str1.isupper():

print(str1.lower())

elif str1.islower():

print(str1.upper())

34. Write a Python function that takes a number as a parameter and check whether the number is prime or not.

def isPrime(num):

flag=1

for i in range(2,num):

if num%2==0:

flag=0

if flag==1:

print("prime")

else:

print("not prime")

no=int(input("enter number:-"))

isPrime(no)

35. Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument.

def isfactorial(num):

fact=1

for i in range(1,num+1):

fact=fact\*i

print(fact)

no=int(input("enter number:-"))

isfactorial(no)

36. Write a Python function that accepts a string and calculate the number of uppercase letters and lowercase letters.

def stringfun(str1):

upper\_count=0

lower\_count=0

for i in str1:

if(i.isupper()):

upper\_count=upper\_count+1

elif(i.islower()):

lower\_count=lower\_count+1

print(upper\_count,"\n",lower\_count)

no=input("enter number:-")

stringfun(no)

37. Write a Python program with the user defined function which accepts a number & returns Fibonacci series of given numbers.

def fibbo(no):

no1=0

no2=1

print(no1,"\n",no2)

for i in range(2,no+1):

no3=no1+no2

print(no3)

no1=no2

no2=no3

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

fibbo(no)

38. Write a Python program to create a user defined module that will ask your college name and will display the name of the college.

#college.py

def sample():

cgl\_name=input("Enter your college name:")

print("Your college is:",cgl\_name)

#main.py

import college

college.sample()

39. Write a Python program to create a class 'Degree' having a method 'getDegree' that prints "I got a degree".It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.

class Degree:

def getDegree(self):

print("I got degree")

class Udergraduate(Degree):

def getDegree(self):

print("i am udergraduate")

class Postgraduate(Degree):

def getDegree(self):

print("I am Postgraduate")

d=Degree()

d.getDegree()

d1=Udergraduate()

d1.getDegree()

d2=Postgraduate()

d2.getDegree()

40. Create a class Employee with data members: name, department and salary. Create suitable methods for reading and printing employee information.

class Employee:

def \_\_init\_\_(self):

self.name = ""

self.department = ""

self.salary = 0.0

def read\_info(self):

self.name = input("Enter employee name: ")

self.department = input("Enter department: ")

self.salary = float(input("Enter salary: "))

def print\_info(self):

print("\nEmployee Information:")

print(f"Name: {self.name}")

print(f"Department: {self.department}")

print(f"Salary: {self.salary}")

emp = Employee()

emp.read\_info()

emp.print\_info()

41. Python program to read and print students information using two classes using simple inheritance.

**class Person:**

**def get\_info(self):**

**self.name = input("Enter name: ")**

**self.age = input("Enter age: ")**

**class Student(Person):**

**def get\_student\_info(self):**

**self.roll = input("Enter roll number: ")**

**self.marks = input("Enter marks: ")**

**def show\_info(self):**

**print("\n--- Student Info ---")**

**print("Name:", self.name)**

**print("Age:", self.age)**

**print("Roll Number:", self.roll)**

**print("Marks:", self.marks)**

**# Create object and call methods**

**s = Student()**

**s.get\_info()**

**s.get\_student\_info()**

**s.show\_info()**

**42.** **Write a Python program to implement multiple inheritance assuming suitable data.**

class A:

def get\_name(self):

self.name = input("Enter name: ")

class B:

def get\_age(self):

self.age = input("Enter age: ")

class C(A, B):

def display(self):

print("\nName:", self.name)

print("Age:", self.age)

# Create object and call methods

obj = C()

obj.get\_name()

obj.get\_age()

obj.display()

43. Write a Python program to Check for ZeroDivisionError Exception.

try:

no1=int(input("enter num1-"))

no2=int(input("Enter num2-"))

result=no1/no2

except ZeroDivisionError:

print("Divide by zero")

44. Write a Python program to create user defined exception that will check whether the password is correct or not using direct method.

class PasswordError(Exception):

pass

def check\_password(pwd):

correct\_password = "admin123"

if pwd != correct\_password:

raise PasswordError("Incorrect Password!")

else:

print("Access Granted ")

try:

password = input("Enter password: ")

check\_password(password)

except PasswordError as e:

print("Error:", e)

45. Write a Python program to create user defined exception that will check whether the username is correct or not using indirect method.

class UsernameError(Exception):

pass

class Login:

def \_\_init\_\_(self):

self.correct\_username = "admin"

def check\_username(self):

username = input("Enter username: ")

if username != self.correct\_username:

raise UsernameError("Invalid Username!")

else:

print("Username is correct ")

try:

obj = Login()

obj.check\_username()

except UsernameError as e:

print("Error:", e)

46. Write a Python program to create a User Defined Exception to input the age from a person & check whether he/she is an adult, if not raise exception.

class NotAnAdultError(Exception):

pass

def check\_age():

age = int(input("Enter your age: "))

if age < 18:

raise NotAnAdultError("You are not an adult!")

else:

print("You are an adult ")

try:

check\_age()

except NotAnAdultError as e:

print("Error:", e)

47.Write a Python program to check for value error & name error exception. Input the data from the user.

try:

# ValueError check

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

# NameError check (trying to print an undefined variable)

print(x)

except ValueError:

print("Error: Invalid number entered!")

except NameError:

print("Error: Variable is not defined!")

48. Implement a program to copy contents of one file to another but converting them into uppercase.

# Open the source file in read mode

with open("source.txt", "r") as src:

content = src.read() # Read the content of the file

# Convert the content to uppercase

content = content.upper()

# Open the destination file in write mode and write the uppercase content

with open("destination.txt", "w") as dest:

dest.write(content)

print("Contents copied and converted to uppercase.")

49. Implement a program to calculate no. of lines, words and characters in any file.

f1=open("abci.txt","r")

content=f1.read()

count=0

line\_count=0

word\_count=0

l=input("Enter char to find count")

for line in content.splitlines():

line\_count=line\_count+1

words=line.split()

for word in words:

word\_count+=1

for char in word:

if char==l:

count=count+1

print("count is:-",count)

print("line count is:-",line\_count)

print("word",word\_count)

50. Implement a program to create a new file “first.txt”, write some data in it and show its contents line by line.

# Create and write data to "first.txt"

with open("first.txt", "w") as f:

f.write("This is the first line.\n")

f.write("This is the second line.\n")

f.write("This is the third line.\n")

# Read and display contents of "first.txt"

with open("first.txt", "r") as f:

for line in f:

print(line, end="")

51. Implement a program to create a new file “first.txt”, write some data in it and show its contents line by line. In this file, append some another data in file and show whether data is updated or not.

# Step 1: Create and write initial data to "first.txt"

with open("first.txt", "w") as f:

f.write("This is the first line.\n")

f.write("This is the second line.\n")

f.write("This is the third line.\n")

# Display the contents of the file

print("Initial contents of 'first.txt':")

with open("first.txt", "r") as f:

for line in f:

print(line, end="")

# Step 2: Append more data to "first.txt"

with open("first.txt", "a") as f:

f.write("This is the appended fourth line.\n")

f.write("This is the appended fifth line.\n")

# Display the updated contents of the file

print("\nContents of 'first.txt' after appending new data:")

with open("first.txt", "r") as f:

for line in f:

print(line, end="")

52. Create a directory named “first\_dir” and apply various directory handling methods in Python on it.

import os

# Step 1: Create a directory "first\_dir"

dir\_name = "first\_dir"

if not os.path.exists(dir\_name):

os.mkdir(dir\_name)

print(f"Directory '{dir\_name}' created.")

else:

print(f"Directory '{dir\_name}' already exists.")

# Step 2: List the contents of the current directory (should show 'first\_dir' if created)

print("\nContents of the current directory:")

print(os.listdir("."))

# Step 3: Check if the directory exists

print("\nCheck if the directory exists:")

if os.path.exists(dir\_name):

print(f"Directory '{dir\_name}' exists.")

else:

print(f"Directory '{dir\_name}' does not exist.")

# Step 4: Rename the directory

new\_name = "renamed\_dir"

os.rename(dir\_name, new\_name)

print(f"\nDirectory renamed to '{new\_name}'.")

# Step 5: List the contents of the current directory after renaming

print("\nContents of the current directory after renaming:")

print(os.listdir("."))

# Step 6: Remove the directory

os.rmdir(new\_name)

print(f"\nDirectory '{new\_name}' removed.")

53. Input a list from user and perform all operations on it.

# Input a list from the user

user\_list = input("Enter a list of elements separated by space: ").split()

# Display the original list

print("\nOriginal List:", user\_list)

# Append a new element to the list

new\_element = input("Enter an element to append: ")

user\_list.append(new\_element)

print("List after appending:", user\_list)

# Remove an element from the list

remove\_element = input("Enter an element to remove: ")

if remove\_element in user\_list:

user\_list.remove(remove\_element)

print("List after removing:", user\_list)

else:

print(f"Element '{remove\_element}' not found.")

# Sort the list

user\_list.sort()

print("List after sorting:", user\_list)

# Display the length of the list

print("Length of the list:", len(user\_list))

54. Input a set from user and perform all operations on it.

# Input a set from the user

user\_set = set(map(int, input("Enter elements of the set separated by space: ").split()))

# Ask the user what operation to perform

print("\nChoose an operation:")

print("1. Add an element")

print("2. Remove an element")

print("3. Union with another set")

print("4. Intersection with another set")

print("5. Difference with another set")

# Perform the operation based on user choice

choice = int(input("\nEnter your choice (1/2/3/4/5): "))

if choice == 1:

add\_element = int(input("Enter an element to add: "))

user\_set.add(add\_element)

print("Updated set:", user\_set)

elif choice == 2:

remove\_element = int(input("Enter an element to remove: "))

user\_set.discard(remove\_element)

print("Updated set:", user\_set)

elif choice == 3:

second\_set = set(map(int, input("Enter another set to perform union: ").split()))

print("Union of sets:", user\_set | second\_set)

elif choice == 4:

second\_set = set(map(int, input("Enter another set to perform intersection: ").split()))

print("Intersection of sets:", user\_set & second\_set)

elif choice == 5:

second\_set = set(map(int, input("Enter another set to perform difference: ").split()))

print("Difference of sets:", user\_set - second\_set)

else:

print("Invalid choice!")

55. Input a dictionary from user and perform all operations on it.

# Input a set from the user

user\_set = set(map(int, input("Enter elements of the set separated by space: ").split()))

# Ask the user what operation to perform

print("\nChoose an operation:")

print("1. Add an element")

print("2. Remove an element")

print("3. Union with another set")

print("4. Intersection with another set")

print("5. Difference with another set")

# Perform the operation based on user choice

choice = int(input("\nEnter your choice (1/2/3/4/5): "))

if choice == 1:

add\_element = int(input("Enter an element to add: "))

user\_set.add(add\_element)

print("Updated set:", user\_set)

elif choice == 2:

remove\_element = int(input("Enter an element to remove: "))

user\_set.discard(remove\_element)

print("Updated set:", user\_set)

elif choice == 3:

second\_set = set(map(int, input("Enter another set to perform union: ").split()))

print("Union of sets:", user\_set | second\_set)

elif choice == 4:

second\_set = set(map(int, input("Enter another set to perform intersection: ").split()))

print("Intersection of sets:", user\_set & second\_set)

elif choice == 5:

second\_set = set(map(int, input("Enter another set to perform difference: ").split()))

print("Difference of sets:", user\_set - second\_set)

else:

print("Invalid choice!")

56.Write a Python program that will calculate area and circumference of circle using inbuilt Math Module.

import math

# Input radius from the user

r = float(input("Enter the radius of the circle: "))

# Calculate area and circumference

area = math.pi \* r \*\* 2

circumference = 2 \* math.pi \* r

# Display the results

print(f"Area of the circle: {area}")

print(f"Circumference of the circle: {circumference}")

57. Python program to generate random int & float within the given rang

import random

# Input range from user

low = int(input("Enter lower bound (int): "))

high = int(input("Enter upper bound (int): "))

# Generate and print

print("Random integer:", random.randint(low, high))

print("Random float: ", random.uniform(low, high))e using random module.

58. Use operator module & perform all arithmetic operations on given input.

import operator

# Input two numbers from the user

a = float(input("Enter first number: "))

b = float(input("Enter second number: "))

# Perform arithmetic operations using the operator module

print("Addition: ", operator.add(a, b))

print("Subtraction: ", operator.sub(a, b))

print("Multiplication: ", operator.mul(a, b))

print("True Division: ", operator.truediv(a, b))

print("Floor Division: ", operator.floordiv(a, b))

print("Modulus: ", operator.mod(a, b))

print("Exponentiation: ", operator.pow(a, b))

59. Use datetime built-in module & show fulldate, time, day, month and year.

import datetime

# Get current date and time

now = datetime.datetime.now()

# Display full date and time

print("Full date & time:", now.strftime("%Y-%m-%d %H:%M:%S"

# Display date parts

print("Date: ", now.date())

print("Time: ", now.time().strftime("%H:%M:%S"))

print("Year: ", now.year)

print("Month: ", now.month)

print("Day: ", now.day)

60. Write a Python program to create two matrices and perform addition, subtraction, multiplication and division operation on matrix.

import numpy as np

# Input matrices

a = np.array([[1, 2], [3, 4]])

b = np.array([[5, 6], [7, 8]])

# Operations

print("Matrix A:\n", a)

print("Matrix B:\n", b)

print("\nAddition:\n", a + b)

print("Subtraction:\n", a - b)

print("Multiplication:\n", a \* b)

print("Division:\n", a / b)

61. Write a Python program to concatenate two strings.

# Input two strings

str1 = input("Enter first string: ")

str2 = input("Enter second string: ")

# Concatenate

result = str1 + str2

# Display result

Print("Concatenated string:", result)

62. Write a NumPy program to generate six random integers between 10 and 30.

import numpy as np

# Generate 6 random integers from 10 to 30 (exclusive of 30)

random\_numbers = np.random.randint(10, 31, size=6)

# Print result

print("Random integers between 10 and 30:", random\_numbers)

63. Write a Python program using numpy Poly1d to generate the polynomial equation and to use det(), inv() on matrices.

65. Design a class to show the use of parameterized constructor.

class Student:

# Parameterized constructor

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

self.name = name

self.age = age

def display(self):

print("Name:", self.name)

print("Age:", self.age)

# Create object with parameters

s1 = Student("Alice", 20)

s1.display()

66. Write a Python program to create a class to operate on an integer and a character with two methodshaving the same name but different sequence of the integer and the character parameters. For example, ifthe parameters of the first method are of the form (int n, char c), then that of the second method will be ofthe form (char c, int n).

class MyClass:

def show(self, a, b):

print("Values are:", a, b)

# Create object

obj = MyClass()

# Call with (int, char)

obj.show(5, 'A')

# Call with (char, int)

obj.show('B', 10)

67. Write a Python program to create a class to print the area of a square and a rectangle. The class has twomethods with the same name but different number of parameters. The method for printing area ofrectangle has two parameters which are length and breadth respectively while the other method forprinting area of square has one parameter which is the side of square.

class Area:

def find\_area(self, a, b=None):

if b is None:

print("Area of Square:", a \* a)

else:

print("Area of Rectangle:", a \* b)

# Create object

shape = Area()

# Call method for square (one parameter)

shape.find\_area(5)

# Call method for rectangle (two parameters)

shape.find\_area(4, 6)

68. Implement Python program to Create Series from Array using pandas package.

import pandas as pd

import numpy as np

# Create a NumPy array

arr = np.array([10, 20, 30, 40, 50])

# Create a Pandas Series from the NumPy array

series = pd.Series(arr)

# Print the Series

print("Pandas Series from Array:")

print(series)

69. Implement Python program to Create Series from List using pandas package.

import pandas as pd

# Create a List

my\_list = [10, 20, 30, 40, 50]

# Create a Pandas Series from the List

series = pd.Series(my\_list)

# Print the Series

print("Pandas Series from List:")

print(series)

70. Implement Python program to Create DataFrame using List using pandas package.

import pandas as pd

# Create a List of Lists (2D list) for DataFrame

data = [['Alice', 24], ['Bob', 27], ['Charlie', 22]]

# Create DataFrame from List

df = pd.DataFrame(data, columns=['Name', 'Age'])

# Print the DataFrame

print("Pandas DataFrame from List:")

print(df)

71. Implement Python program to Create DataFrame using dictionary using pandas package.

import pandas as pd

# Create a Dictionary

data = {

'Name': ['Alice', 'Bob', 'Charlie'],

'Age': [24, 27, 22],

'City': ['New York', 'Los Angeles', 'Chicago']

}

# Create DataFrame from Dictionary

df = pd.DataFrame(data)

# Print the DataFrame

print("Pandas DataFrame from Dictionary:")

print(df)

72. Implement Python program to read csv file using pandas package.

import pandas as pd

df = pd.read\_csv('example.csv')

print("Data from CSV file:")

print(df)

73. Implement a python program using numpy package to perform arithmetic operations on 2D matrix.

import numpy as np

matrix1 = np.array([[1, 2], [3, 4]])

matrix2 = np.array([[5, 6], [7, 8]])

addition\_result = matrix1 + matrix2

print("Addition of matrices:\n", addition\_result)

subtraction\_result = matrix1 - matrix2

print("\nSubtraction of matrices:\n", subtraction\_result)

multiplication\_result = matrix1 \* matrix2

print("\nMultiplication of matrices (element-wise):\n", multiplication\_result)

division\_result = matrix1 / matrix2

print("\nDivision of matrices (element-wise):\n", division\_result)

74. Implement python program to use of matplotlib package to represent data in graphical form.

import matplotlib.pyplot as plt

# Data

x = [1, 2, 3, 4, 5] # X-axis data (e.g., time)

y = [2, 4, 6, 8, 10] # Y-axis data (e.g., value)

plt.plot(x, y)

plt.title("Simple Line Plot")

plt.xlabel("X Axis (Time)")

plt.ylabel("Y Axis (Value)")

plt.show()

75. Implement a python program to demonstrate the use of any ten functions in math module.

import math

# 1. Square root

print("Square root of 16:", math.sqrt(16))

# 2. Factorial

print("Factorial of 5:", math.factorial(5))

# 3. Power

print("2 raised to the power of 3:", math.pow(2, 3))

# 4. Logarithm (base 10)

print("Logarithm of 1000 (base 10):", math.log10(1000))

# 5. Exponential

print("Exponential of 2:", math.exp(2))

# 6. Sine (30 degrees)

print("Sine of 30 degrees:", math.sin(math.radians(30)))

# 7. Cosine (45 degrees)

print("Cosine of 45 degrees:", math.cos(math.radians(45)))

# 8. Tangent (60 degrees)

print("Tangent of 60 degrees:", math.tan(math.radians(60)))

# 9. Absolute value

print("Absolute value of -10.5:", math.fabs(-10.5))

# 10. GCD (Greatest common divisor)

print("GCD of 36 and 60:", math.gcd(36, 60))

76. Implement a python program to demonstrate the use of any 5 functions in random module.

import random

# 1. random() - Generates a random float number between 0 and 1

random\_float = random.random()

print("Random float between 0 and 1:", random\_float)

# 2. randint() - Generates a random integer within a specified range

random\_int = random.randint(1, 100)

print("Random integer between 1 and 100:", random\_int)

# 3. choice() - Returns a randomly selected element from a sequence (list)

random\_choice = random.choice(['apple', 'banana', 'cherry', 'date'])

print("Random choice from list:", random\_choice)

# 4. shuffle() - Shuffles the elements of a list in place

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

random.shuffle(my\_list)

print("Shuffled list:", my\_list)

# 5. sample() - Returns a specified number of random elements from a list

random\_sample = random.sample([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 3)

print("Random sample of 3 elements:", random\_sample)

77. Implement a python program to demonstrate the use of any 7 functions in os module.

import os

# 1. getcwd() - Get the current working directory

current\_dir = os.getcwd()

print("Current Working Directory:", current\_dir)

# 2. listdir() - List all files and directories in a specified path

files\_and\_dirs = os.listdir(current\_dir)

print("\nFiles and directories in the current directory:")

print(files\_and\_dirs)

# 3. makedirs() - Create a new directory, including intermediate directories

new\_dir = "test\_directory"

os.makedirs(new\_dir, exist\_ok=True) # If it exists, won't raise an error

print(f"\nDirectory '{new\_dir}' created!")

4. remove() - Remove a file (if it exists)

# Create a temporary file for demonstration

with open('temp\_file.txt', 'w') as file:

file.write("This is a temporary file.")

# 5. rename() - Rename a file or directory

os.rename(new\_dir, "renamed\_directory")

print("\nDirectory renamed to 'renamed\_directory'!")

# 6. path.exists() - Check if a path exists

path\_exists = os.path.exists("renamed\_directory")

print("\nDoes 'renamed\_directory' exist?", path\_exists)

# 7. getpid() - Get the process ID of the current process

pid = os.getpid()

print("\nCurrent Process ID:", pid)

78. Implement a Python program to demonstrate use of 1. lambda 2. map 3. Reduce.

from functools import reduce

# 1. Lambda: Adds 2 to a number

print("Lambda example:", (lambda x: x + 2)(5))

# 2. Map: Adds 2 to each element of a list

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

print("Map example:", list(map(lambda x: x + 2, numbers)))

# 3. Reduce: Sums up all elements of the list

print("Reduce example:", reduce(lambda x, y: x + y, numbers))

79. Implement a python program to demonstrate Method Overloading.

class Calculator:

# Method to add numbers

def add(self, a, b=0, c=0): # Default values simulate overloading

return a + b + c

# Creating an object of Calculator class

calc = Calculator()

# 1. Adding two numbers

print("Sum of 2 numbers:", calc.add(5, 10))

# 2. Adding three numbers

print("Sum of 3 numbers:", calc.add(5, 10, 15))

# 3. Adding one number (using default values for b and c)

print("Sum of 1 number:", calc.add(5))

80. Implement a python program to demonstrate Method Overriding.

# Parent class

class Animal:

def sound(self):

print("Some generic animal sound")

# Child class

class Dog(Animal):

def sound(self): # Overriding the method from Animal class

print("Bark")

# Another child class

class Cat(Animal):

def sound(self): # Overriding the method from Animal class

print("Meow")

# Creating objects of Dog and Cat

dog = Dog()

cat = Cat()

# Calling the overridden methods

dog.sound() # Output: Bark

cat.sound() # Output:Meow

81. Implement a python program to demonstrate constructor overloading.

class Vehicle:

# Constructor with default arguments to simulate overloading

def \_\_init\_\_(self, name="Unnamed Vehicle", type="Unknown"):

self.name = name

self.type = type

def display(self):

print(f"Vehicle Name: {self.name}, Type: {self.type}")

# Creating objects of Vehicle class with different constructor arguments

vehicle1 = Vehicle("Car")

vehicle2 = Vehicle("Truck", "Diesel")

vehicle3 = Vehicle()

# Displaying the vehicle information

vehicle1.display() # Output: Vehicle Name: Car, Type: Unknown

vehicle2.display() # Output: Vehicle Name: Truck, Type: Diesel

vehicle3.display() # Output: Vehicle Name: Unnamed Vehicle, Type: Unknown

82.Write a Python program to demonstrate various methods of parameter passing to the function in python.

# 1. Positional Arguments

def greet(name, age):

print(f"Hello {name}, you are {age} years old.")

# 2. Keyword Arguments

def describe\_pet(name, animal\_type="Dog"):

print(f"My {animal\_type} is named {name}.")

# 3. Default Arguments

def calculate\_area(length, width=5):

print(f"Area: {length \* width}")

# 4. Variable-length Arguments (\*args)

def add\_numbers(\*args):

print(f"Sum: {sum(args)}")

# 5. Variable-length Keyword Arguments (\*\*kwargs)

def person\_details(\*\*kwargs):

print(kwargs)

# Test the functions

greet("Alice", 30)

describe\_pet("Max", "Cat")

describe\_pet("Buddy")

calculate\_area(10)

add\_numbers(1, 2, 3)

person\_details(name="John", age=25)