Bhilai Institute of Technology, Durg.

Department of Computer Science and Engineering

PYTHON WITH DJANGO LAB 102393CS CRN: A59

Date : 09/09/2022

Experiment No: 1.1

Aim:

WAP to input your name and print a welcome message along with your name.

Code:

```
#WAP to input your name and print a welcome message along with your name.
name=input("enter your name:")
print("Welcome",name)
```

Output:

```
enter your name:Aditi
Welcome Aditi
```

Date : 09/09/2022

Experiment No: 1.2

Aim:

WAP to demonstrate the use of replacement operators {}, {index} and {alphabet} in format() used in formatted string.

Code:

```
#WAP to demonstrate the use of replacement operators {}, {index} and {alphabet} in format()
used in formatted string.
a=10
b=20
c=a+b
s1="Addition of {} and {} is {}"
s2=s1.format(a,b,c)
print(s2)
s1="Addition of {0} and {1} is {2}"
s2=s1.format(a,b,c)
print(s2)
s1="Addition of {x} and {y} is {z}"
s2=s1.format(x=a,y=b,z=c)
print(s2)
```

Output:

```
Addition of 10 and 20 is 30
Addition of 10 and 20 is 30
Addition of 10 and 20 is 30
```

Date : 09/09/2022

Experiment No: 1.3

Aim:

WAP to demonstrate the use of “%d”, “%f”, “%s” in a formatted string to print values of specific data types.

Code:

```
#WAP to demonstrate the use of “%d”, “%f”, “%s” in a formatted string to print values of specific data types.
a=10
b=123.456
s1="Welcome"
print("a=%d b=%f s1=%s"%(a,b,s1))
print("a=%d b=%f s1=%s"%(b,a,s1))
```

Output:

```
a=10 b=123.456000 s1=Welcome
a=123 b=10.000000 s1=Welcome
```

Date : 09/09/2022

Experiment No: 1.4

Aim:

WAP to find the sum of 3 numbers taking input from user. Print the 3 numbers and their sum as formatted string in the print() function.

Code:

```
#WAP to find the sum of 3 numbers taking input from user. Print the 3 numbers and their sum as formatted string in the print() function.
a=int(input("Enter 1st no.:"))
b=int(input("Enter 2nd no.:"))
c=int(input("enter 3rd no.:"))
sum=a+b+c
print("a={}, b={}, c={}".format(a,b,c))
print("{0}+{1}+{2}={3}".format(a,b,c,sum))
```

Output:

```
Enter 1st no.:12
Enter 2nd no.:13
enter 3rd no.:14
a=12, b=13, c=14
12+13+14=39
```

Date : 09/09/2022

Experiment No: 1.5

Aim:

WAP to find the average if 3 numbers taking input from user. Print the 3 numbers and their average as formatted string in the print() function.

Code:

```
#WAP to find the average if 3 numbers taking input from user. Print the 3 numbers and their average as formatted string in the print() function.
a=float(input("Enter 1st no.:"))
b=float(input("Enter 2nd no.:"))
c=float(input("enter 3rd no.:"))
avg=(a+b+c)/3
print("a=%f, b=%f, c=%f"%(a,b,c))
s1="Average of {},{},{} is {}"
s2=s1.format(a,b,c,avg)
print(s2)
```

Output:

```
Enter 1st no.:2.876
Enter 2nd no.:3.987
enter 3rd no.:4.879
a=2.876000, b=3.987000, c=4.879000
Average of 2.876,3.987,4.879 is 3.9139999999999997
```

Date : 09/09/2022

Experiment No: 1.6

Aim:

WAP to swap two variables. Take necessary inputs from user.

Code:

```
#WAP to swap two variables. Take necessary inputs from user.
a=int(input("Enter 1st no.:"))
b=int(input("Enter 2nd no.:"))
print("a={} and b={}".format(a,b))
c=a
a=b
b=c
print("a={} and b={}".format(a,b))
```

Output:

```
Enter 1st no.:21
Enter 2nd no.:23
a=21 and b=23
a=23 and b=21
```

Date : 09/09/2022

Experiment No: 1.7

Aim:

WAP to swap two variables without using a third variable. Take necessary inputs from user. (hint: use comma in between)

Code:

```
#WAP to swap two variables without using a third variable. Take necessary inputs from user. (hint: use
comma in between)
a=int(input("Enter 1st no.:"))
b=int(input("Enter 2nd no.:"))
print("a={} and b={}".format(a,b))
(a,b)=(b,a)
print("a={} and b={}".format(a,b))
```

Output:

```
Enter 1st no.:12
Enter 2nd no.:33
a=12 and b=33
a=33 and b=12
```

Date : 16/09/2022

Experiment No: 2.1

Aim:

WAP to use eval() to evaluate an arithmetic expression as a string input from user.

Code:

```
#WAP to use eval() to evaluate an arithmetic expression as a string input from user.
a=eval(input("enter 1st no.:"))
b=eval(input("enter 2nd no.:"))
c=a+b
print("sum=",c)
sum=eval("100+200")
print("sum=",sum)
res=eval(input("enter any arithmatic operations with values"))
print("result=",res)
```

Output:

```
enter 1st no.:23
enter 2nd no.:45
sum= 68
sum= 300
enter any arithmatic operations with values230+300
result= 530
```

Date : 16/09/2022

Experiment No: 2.2

Aim:

WAP to calculate area of rectangle, square, circle and triangle. Take necessary inputs from user.

Code:

```
#WAP to calculate area of rectangle, square, circle and triangle. Take necessary inputs from user.
import math
#area of rectangle
l=eval(input("enter length of rectangle:"))
b=eval(input("enter breadth of rectangle:"))
rarea=l*b
print("area of rectangle:",rarea)
#area of circle
r=eval(input("enter radius of circle:"))
carea=math.pi*(r**2)
print("area of circle:",carea)
#area of triangle
b=eval(input("enter base of triangle:"))
h=eval(input("enter height of triangle:"))
tarea=0.5*b*h
print("area of triangle:",tarea)
#area of square
s=eval(input("enter side of square:"))
sarea=s*s
print("area of square:",sarea)
```

Output:

```
enter length of rectangle:2
enter breadth of rectangle:3
area of rectangle: 6
enter radius of circle:4
area of circle: 50.26548245743669
enter base of triangle:5
enter height of triangle:4
```

```
area of triangle: 10.0
enter side of square:4
area of square: 16
```

Date : 16/09/2022

Experiment No: 2.3

Aim:

WAP for height taken in cms then covert into feet and inches.(1 foot=12 inches and 1 inch=2.54 cm)

Code:

```
#WAP for height taken in cms then covert  into feet and inches.(1 foot=12 inches and 1 inch=2.54 cm)
cm=eval(input("enter height in cm:"))
inch=cm/2.54
feet=inch/12
print("height :",feet)
feet2=int(feet)
inch2=feet-feet2
print("height=",feet2,"feet and",inch2,"inches")
```

Output:

```
enter height in cm:345
height : 11.318897637795276
height= 11 feet and 0.3188976377952759 inches
```

Date : 16/09/2022

Experiment No: 2.4

Aim:

Accept as input the basic salary of a person. His dearness allowance (DA) is 40% of the basic salary and the house rent allowance (HRA) is 20% of the basic salary. Calculate and show the Gross salary.

Code:

```
#Accept as input the basic salary of a person. His dearness allowance (DA) is 40% of the basic salary
#and the house rent allowance (HRA) is 20% of the basic salary. Calculate and show the Gross salary.
sal=eval(input("enter salary:"))
DA=0.4*sal
HRA=0.2*sal
print("DA=",DA,"and HRA=",HRA)
sum=DA+HRA
gs=sal-sum
print("gross sal=",gs)
```

Output:

```
enter salary:1000
DA= 400.0 and HRA= 200.0
gross sal= 400.0
```

Date : 16/09/2022

Experiment No: 2.5

Aim:

Accept as Input the marks obtained by a student in 5 subjects. Show the Aggregate marks and Percentage marks

Code:

```
# Accept as Input the marks obtained by a student in 5 subjects. Show the Aggregate marks and
#Percentage marks
a=eval(input("enter marks for 1st subject: "))
b=eval(input("enter marks for 2nd subject: "))
c=eval(input("enter marks for 3rd subject :"))
d=eval(input("enter marks for 4th subject :"))
e=eval(input("enter marks for 5th subject: "))
sum=a+b+c+d+e
per=sum/5
print("total marks for student is:",sum)
print("total percentage is:",per)
```

Output:

```
enter marks for 1st subject: 92
enter marks for 2nd subject: 95
enter marks for 3rd subject: 23
enter marks for 4th subject: 4
enter marks for 5th subject: 45
total marks for student is: 259
total percentage is: 51.8
```

Date : 16/09/2022

Experiment No: 2.6

Aim:

Write a program to read age from keyboard and print whether the person is child,adult or elderly.

Code:

```
#Write a program to read age from keyboard and print whether the person is child,adult or elderly.
age=eval(input("enter age of person:"))
if age<18:
    print("person is child as age=",age)
elif age>=18 and age<=59:
    print("person is adult as age=",age)
else:
    print("person is elderly as age=",age)
```

Output:

```
enter age of person:60
person is elderly as age= 60
```

Date : 16/09/2022

Experiment No: 2.7

Aim:

WAP to find whether a given no is even or odd

Code:

```
#WAP to find whether a given no is even or odd.
num=eval(input("enter any no.: "))
if num%2==0:
    print(num, "is even")
else:
    print(num,"is odd")
```

Output:

```
enter any no.: 23
23 is odd
```

Date : 16/09/2022

Experiment No: 2.8

Aim:

WAP to find whether a given number is -ve , +ve or zero.

Code:

```
#WAP to find whether a given number is  -ve , +ve or zero.
num=eval(input("enter any number:"))
if num>0:
    print(num,"is +ve")
elif num<0:
    print(num,"is -ve")
else:
    print(num, "is zero")
```

Output:

```
enter any number:23
23 is +ve
```

Date : 16/09/2022

Experiment No: 2.9

Aim:

WAP to find the greatest of 2 numbers taking input from user.

Code:

```
#WAP to find the greatest of 2 numbers taking input from user.
num1=eval(input("enter 1st no.: "))
num2=eval(input("enter 2nd no.: "))
if num1>num2:
    print(num1,"is greater than",num2)
elif num1==num2:
    print(num1,"is equal to",num2)
else:
    print(num2,"is greater than",num1)
```

Output:

```
enter 1st no.: 23
enter 2nd no.: 4
23 is greater than 4
```

Date : 16/09/2022

Experiment No: 2.10

Aim:

WAP to find the smallest of 3 numbers taking input from user.

Code:

```
#WAP to find the smallest of 3 numbers taking input from user.
a=eval(input("enter 1st no.: "))
b=eval(input("enter 2nd no.: "))
c=eval(input("enter 3rd no.: "))
if a<b and a<c:
   print(a,"is smallest")
elif b<a and b<c:
   print(b,"is smallest")
else:
   print(c,"is smallest")
```

Output:

```
enter 1st no.: 12
enter 2nd no.: 45
enter 3rd no.: 67
12 is smallest
```

Bhilai Institute of Technology, Durg.

Department of Computer Science and Engineering

PYTHON WITH DJANGO LAB 102393CS CRN: A59

Date : 23/09/2022

Experiment No: 3.1

Aim:

WAP to find whether an input number is prime or composite.

Code:

```
#WAP to find whether an input number is prime or composite.
num = int(input("Enter any number : "))
if num > 1:
    for i in range(2, num):
        if (num % i) == 0:
            print(num, "is NOT a prime number")
            break
    else:
        print(num, "is a PRIME number")
elif num == 0 or 1:
    print(num, "is a neither prime NOR composite number")
else:
    print(num, "is NOT a prime number it is a COMPOSITE number")
```

Output:

```
Enter any number : 31
31 is a PRIME number
```

Date : 23/09/2022

Experiment No: 3.2

Aim:

WAP to determine whether the input number is an Armstrong number or not. (Hint: sum of cubes = number itself)

Code:

```
#WAP to determine whether the input number is an Armstrong number or not
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:

```
Enter a number: 407
407 is an Armstrong number
```

Date : 23/09/2022

Experiment No: 3.3

Aim:

Accept a five digit number and reverse the number. Show whether the reversed number is same as the original number or not

Code:

```
# Accept a five digit number and reverse the number. Show whether the reversed number is same as the
#original number or not
num = int(input("Enter a five digit number: "))
reversed_num = 0
while num != 0:
    digit = num % 10
    reversed_num = reversed_num * 10 + digit
    num //= 10

print("Reversed Number: " + str(reversed_num))
```

Output:

```
Enter a five digit number: 12345
Reversed Number: 54321
```

Date : 23/09/2022

Experiment No: 3.4

Aim:

WAP to print the Fibonacci sequence up to N terms. Input N from user.

Code:

```
# WAP to print the Fibonacci sequence up to N terms. Input N from user.
num = int(input("Enter number of terms: "))
n1, n2 = 0, 1
print("Fibonacci Series:", n1, n2, end=" ")
for i in range(2, num):
    n3 = n1 + n2
    n1 = n2
    n2 = n3
    print(n3, end=" ")
print()
```

Output:

```
Enter number of terms: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
```

Date : 23/09/2022

Experiment No: 3.5

Aim:

WAP to print the following using nested for loop:

```
*
* *
* * *
* * * *
* * * * *
```

Code:

```
rows = int(input("Enter number of rows: "))
# outer loop
for i in range(1, rows + 1):
    # inner loop
    for j in range(1, i + 1):
        print("*", end=" ")
    print(")
```

Output:

```
Enter number of rows: 5
*
* *
* * *
* * * *
* * * * *
```

Bhilai Institute of Technology, Durg.

Department of Computer Science and Engineering

PYTHON WITH DJANGO LAB 102393CS CRN: A59

Date : 23/09/2022

Experiment No: 3.6

Aim:

Write a program to print the following pattern .

1
12
123
1234

Code:

```
n = int(input("Enter number of rows: "))
for i in range(1,n+1):
    for j in range(1, i+1):
        print(j, end="")
    print()
```

Output:

```
Enter number of rows: 4
1
12
123
1234
```

Date : 23/09/2022

Experiment No: 3.7

Aim:

Write a program to print the following pattern .
A
BB
CCC
DDDD

Code:

```
n = int(input("Enter number of rows: "))
A = 65
for i in range(1,n+1):
    for j in range(1, i+1):
        print("%c" %(A), end="")
    A +=1
    print()
```

Output:

```
Enter number of rows: 4
A
BB
CCC
DDDD
```

Date : 23/09/2022

Experiment No: 3.8

Aim:

WAP to assume that uname="ABC" and pswd="123". Ask user to enter the correct combination of uname and pswd. Print "Welcome to Python" only when both the uname and pswd are correct, otherwise keep on asking user to enter correct uname and pswd.

Code:

```
uname = "ABC"
pswd = "123"

while True:
    username = str(input("Enter your username: "))
    password = str(input("Enter your password: "))
    if username == uname and password == pswd:
        print("Welcome to Python")
        break
    print("Invalid Credentials, Try again.")
```

Output:

```
Enter your username: ABC
Enter your password: 123
Welcome to Python
```

Date: 23/09/2022

Experiment No: 3.9

Aim:

WAP to demonstrate the use of lstrip(), rstrip(), strip(), count(sub_string), count(sub_string,start_index,end_index) and replace(old_string,new_string) on Strings.

Code:

```
string = "---ABCDEF---"
print(f"String after stripping all '-' is {string.strip('-')}")
print(f"String after stripping all leading '-' is {string.lstrip('-')}")
print(f"String after stripping all trailing '-' is {string.rstrip('-')}")
new_string = "Python is awesome, it really is"
print(f"The count is {new_string.count('is')}")
print(f"The count of 'i' is {new_string.count('i', 8, 25)}")
print(f"String after replacing 'ABCDEF' with 'PQRSTUV' is {string.replace('ABCDEF', 'PQRSTUV')}")
```

Output:

```
String after stripping all '-' is ABCDEF
String after stripping all leading '-' is ABCDEF---
String after stripping all trailing '-' is ---ABCDEF
The count is 2
The count of 'i' is 1
String after replacing 'ABCDEF' with 'PQRSTUV' is ---PQRSTUV---
```

Date: 23/09/2022

Experiment No: 3.10

Aim:

WAP to demonstrate the use of count(), replace(), split(), rsplit(), join() on Strings.

Code:

```
string = "Python is a programming language"
print(string.split())
string = "tic-tac-toe"
print(string.rsplit('-'))
charlist = ['A','B','C','D','E','F']
print("".join(charlist))
```

Output:

```
['Python', 'is', 'a', 'programming', 'language']
['tic', 'tac', 'toe']
ABCDEF
```

Date: 23/09/2022

Experiment No: 3.11

Aim:

WAP to demonstrate the use of upper(), swapcase(), title(), capitalize(), startswith(), endswith() on Strings.

Code:

```
string = "Python"
print("Uppercase: ", string.upper())
print("Swapcase: ", string.swapcase())
print("Title: ", "python".title())
print("Capitalize: ", "pyThOn".capitalize())
string = "Python is fun"
print(string.startswith("Python"))
print(string.endswith("fun"))
```

Output:

```
Uppercase:  PYTHON
Swapcase:  pYTHON
Title:  Python
Capitalize:  Python
True
True
```

Date: 23/09/2022

Experiment No: 3.12

Aim:

WAP to demonstrate the use of isalpha(), isnumeric(), sorted(), chr() and ord() on Strings.

Code:

```
string = "Python"
print(string.isalpha())
print(string.isnumeric())
print(f"Before sorting '{string}', after sorting '{sorted(string)}'")
print(chr(68))
print(ord('c'))
```

Output:

```
True
False
Before sorting 'Python', after sorting '['P', 'h', 'n', 'o', 't', 'y']'
D
99
```

Date: 30/09/2022

Experiment No: 4.1

Aim:

Write a program to read a string from user and convert it into a list.

Code:

```
s1=input("Enter any text:")
print("Input string:",s1)
l1=s1.split()
print(l1,type(l1))
```

Output:

```
Enter any text:aditi
Input string: aditi
['aditi'] <class 'list'>
```

Date: 30/09/2022

Experiment No: 4.2

Aim:

WAP to demonstrate how to traverse a list 1) using print(), 2) using while loop and 3) using for loop

Code:

```
l1=[10,20,30,40,50]
print("1) using print()",l1)
print(l1)
i=0
n=len(l1)
print("2) using while loop",l1)
while i<n:
   print(l1[i], end=" ")
   i+=1
print()
print("3) using for loop",l1)
for n in l1:
   print(n,end=" ")
```

Output:

```
1) using print() [10, 20, 30, 40, 50]
[10, 20, 30, 40, 50]
2) using while loop [10, 20, 30, 40, 50]
10 20 30 40 50
3) using for loop [10, 20, 30, 40, 50]
10 20 30 40 50
```

Date: 30/09/2022

Experiment No: 4.3

Aim:

WAP to print only the even numbers present in a List of integers

Code:

```
l1=list(range(1,11))
print("Original List:",l1)
for n in l1:
    if n%2==0:
        print(n,end=" ")
```

Output:

```
Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 4 6 8 10
```

Date: 30/09/2022

Experiment No: 4.4

Aim:

WAP to print the List elements using positive and negative indexing

Code:

```
l1=list(range(10,60,10))
print("Original List:",l1)
i=0
n=len(l1)
while i<n:
    print("l1[{}] = {} or l1[{}] ={}".format(i,l1[1],i-n,l1[i-n]))
    i+=1
```

Output:

```
Original List: [10, 20, 30, 40, 50]
l1[0] = 20 or l1[-5] =10
l1[1] = 20 or l1[-4] =20
l1[2] = 20 or l1[-3] =30
l1[3] = 20 or l1[-2] =40
l1[4] = 20 or l1[-1] =50
```

Date: 30/09/2022

Experiment No: 4.5

Aim:

WAP to print the sum and average of all the elements in a List of numbers

Code:

```
l1=list(range(10,60,10))
print("Original List:",l1)
sum=0
avg=0
k=len(l1)
for n in l1:
   sum=sum+n
avg= sum/k
print("sum = ",sum," Average = ",avg)
```

Output:

```
Original List: [10, 20, 30, 40, 50]
sum =  150  Average =  30.0
```

Date: 30/09/2022

Experiment No: 4.6

Aim:

WAP to demonstrate the use of append(), clear(), copy() and count() methods

Code:

```
l1=[]
print("Original List:", l1)
l1.append(10)
l1.append(20)
l1.append(20)
l1.append(30)
l1.append(40)
print("After append:", l1)
l2=l1.copy()
print("Copy of l1:", l2)
print("l1.count(20):", l1.count(20))
l1.clear()
print("l1.clear():", l1)
```

Output:

```
Original List: []
After append: [10, 20, 20, 30, 40]
Copy of l1: [10, 20, 20, 30, 40]
l1.count(20): 2
l1.clear(): []
```

Date: 30/09/2022

Experiment No: 4.7

Aim:

WAP to create two Lists, the first List should contain only even numbers and second List should only contain odd numbers from a single main List of numbers

Code:

```
l1=list(range(1,11))
print("Original List:",l1)
evenlist=[n for  n in l1 if n%2==0]
oddlist=[n for n in l1 if n%2!=0]
print("Even List:",evenlist)
print("Odd List:",oddlist)
```

Output:

```
Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Even List: [2, 4, 6, 8, 10]
Odd List: [1, 3, 5, 7, 9]
```

Date: 30/09/2022

Experiment No: 4.8

Aim:

WAP to create a List of numbers from 1 to 100, where each element should be completely divisible by 10. (hint: use comprehension)

Code:

```
l1=list(range(1,101))
print("Original List:",l1)
l2=[n for n in l1 if n%10==0]
print("Filtered List:",l2)
```

Output:

```
Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99,
100]
Filtered List: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

Date: 30/09/2022

Experiment No: 4.9

Aim:

WAP to demonstrate the use of sort() of List class, to sort the elements of the List containing Strings based on string length

Code:

```
s1=("Welcome to Python Lab")
l1=s1.split()
print("Original List:",l1)
def f1(s):
   return len(s)
l1.sort(key=f1)
print("Sorted List:",l1)
```

Output:

```
Original List: ['Welcome', 'to', 'Python', 'Lab']
Sorted List: ['to', 'Lab', 'Python', 'Welcome']
```

Date: 30/09/2022

Experiment No: 4.10

Aim:

WAP to find the greatest list from the following nested list
l1=[[10,20,30],[40,50,60],[70,80,90]]

Code:

```
l1=[[10,20,30],[40,50,60],[70,80,90]]
print("Original List:",l1)
print(max(l1))
```

Output:

```
Original List: [[10, 20, 30], [40, 50, 60], [70, 80, 90]]
[70, 80, 90]
```

Date: 30/09/2022

Experiment No: 4.11

Aim:

WAP to create a list whose elements are squares of integers from 1 to 10. 1) Implement without list comprehension and 2) Implement with list comprehension

Code:

```
l1=list(range(1,11))
print("Original List:",l1)
l2=[]
for n in l1:
    l2.append(n**2)
print("Without list comprehension:",l2)
l2=[n**2 for n in l1]
print("With list comprehension:",l2)
```

Output:

```
Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Without list comprehension: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
With list comprehension: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Date: 30/09/2022

Experiment No: 4.12

Aim:

WAP to create a list L1 of integers from 1 to 10. From L1 create another list L2 with the condition that at positions having even numbers in L1 that even number will be inserted in L2 else 0 will be inserted. Use list comprehension.

Code:

```
l1=list(range(1,11))
print("Original List:",l1)
l2=[n if n%2==0 else 0 for n in l1]
print("Result:",l2)
```

Output:

```
Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Result: [0, 2, 0, 4, 0, 6, 0, 8, 0, 10]
```

Date: 30/09/2022

Experiment No: 4.13

Aim:

WAP to create a list containing only the String length from each string present in another list of strings. Use list comprehension.

Code:

```
s1="Welcome to Python Programming Lab"
l1=s1.split()
print("Original List:", l1)
l2=[len(n) for n in l1]
print("Result:",l2)
```

Output:

```
Original List: ['Welcome', 'to', 'Python', 'Programming', 'Lab']
Result: [7, 2, 6, 11, 3]
```

Date: 30/09/2022

Experiment No: 4.14

Aim:

WAP to convert the following string "the quick brown fox jumps over the lazy dog" into list of words and then create another list that contains uppercase words extracted from the previous list. Create another list that contains sub-list having uppercase words and its length. Use list comprehension.

Code:

```
s1="the quick brown fox jumps over the lazy dog"
l1=s1.split()
print("Original List:", l1)
l2=[n.upper() for n in l1]
print("Uppercase List:",l2)
l3=[(n,len(n)) for n in l2]
print("String with String length:",l3)
```

Output:

```
Original List: ['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
Uppercase List: ['THE', 'QUICK', 'BROWN', 'FOX', 'JUMPS', 'OVER', 'THE', 'LAZY', 'DOG']
String with String length: [('THE', 3), ('QUICK', 5), ('BROWN', 5), ('FOX', 3), ('JUMPS', 5),
('OVER', 4), ('THE', 3), ('LAZY', 4), ('DOG', 3)]
```

Date: 30/09/2022

Experiment No: 4.15

Aim:

WAP to remove duplicate elements from a list and create a new list with those unique elements. (Hint: use in and not in)

Code:

```
l1=[10,20,20,30,30,30,40,40,40]
print("Original List:", l1)
l2=[]
for n in l1:
    if n not in l2:
        l2.append(n)
print("Result:",l2)
```

Output:

```
Original List: [10, 20, 20, 30, 30, 30, 40, 40, 40]
Result: [10, 20, 30, 40]
```

Date: 07/10/2022

Experiment No: 5.1

Aim:

Write a program to read a string from user and convert it into a tuple.

Code:

```
s1=input("enter any string:")
print("input string is: ",s1)
t1=tuple(s1)
print(t1)
```

Output:

```
enter any string:aditi sharma
input string is:  aditi sharma
('a', 'd', 'i', 't', 'i', ' ', 's', 'h', 'a', 'r', 'm', 'a')
```

Date: 07/10/2022

Experiment No: 5.2

Aim:

WAP to print the sum and average of all the elements in a Tuple of numbers.

Code:

```
t1=(10,20,30,40,50)
sum=0
avg=0
n=len(t1)
for i in t1:
    sum=sum+i
avg=sum/n
print("sum=",sum,"average=",avg)
```

Output:

```
sum= 150 average= 30.0
```

Date: 07/10/2022

Experiment No: 5.3

Aim:

WAP to create a tuple t2 from an existing tuple t1 = (11, 22, 33, 44, 55, 66) by extracting 33,44 & 55 using slicing

Code:

```
t1=(11,22,33,44,55,66)
print("original tuple:",t1)
t2=t1[2:5]
print(t2)
```

Output:

```
original tuple: (11, 22, 33, 44, 55, 66)
(33, 44, 55)
```

Date: 07/10/2022

Experiment No: 5.4

Aim:

WAP to modify the first item (22) of a list inside a following tuple to 222 where t1=(11, [22, 33], 44, 55)

Code:

```
t1=(11,[22,33],44,55)
print("original tuple:",t1)
t1[1][0]=222
print(t1)
```

Output:

```
original tuple: (11, [22, 33], 44, 55)
(11, [222, 33], 44, 55)
```

Date: 07/10/2022

Experiment No: 5.5

Aim:

WAP to create two Tuples, the first Tuple should contain only even numbers and second Tuple should only contain odd numbers from a single main Tuple of numbers. Use tuple comprehension

Code:

```
t1=tuple(n for n in range(1,11))
print("original tuple:",t1)
t2=tuple(n for n in t1 if n%2==0)
print(t2)
t3=tuple(n for n in t1 if n%2!=0)
print(t3)
```

Output:

```
original tuple: (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
(2, 4, 6, 8, 10)
(1, 3, 5, 7, 9)
```

Date: 07/10/2022

Experiment No: 5.6

Aim:

WAP to build a tuple T1 containing ages ranging from 10 to 70 with a gap of 5 years in between. Extract a tuple T2 containing ages below 30. Extract another tuple T3 containing ages above 30 but below 50 and extract another tuple T4 having ages above 50. Use Tuple comprehension

Code:

```
t1=tuple(range(10,75,5))
print("original tuple ",t1)
t2=tuple(n for n in t1 if n<30)
print(t2)
t3=tuple(n for n in t1 if n>30 and n<50)
print(t3)
t4=tuple(n for n in t1 if n>50)
print(t4)
```

Output:

```
original tuple  (10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70)
(10, 15, 20, 25)
(35, 40, 45)
(55, 60, 65, 70)
```

Date: 07/10/2022

Experiment No: 5.7

Aim:

WAP to read a string from user and convert it into a set.

Code:

```
s1=input("enter the value:")
print("the value:",s1)
st1=set(s1)
print(st1,type(st1))
```

Output:

```
enter the value:python
the value: python
{'h', 't', 'n', 'o', 'p', 'y'} <class 'set'>
```

Date: 07/10/2022

Experiment No: 5.8

Aim:

WAP to convert a list of numbers with duplicate entries into a set and display the result. Use set comprehension

Code:

```
l1=[1,23,4,56,7,69,90,34,8,6]
print("original value",l1)
st1={n for n in l1}
print(st1,type(st1))
```

Output:

```
original value [1, 23, 4, 56, 7, 69, 90, 34, 8, 6]
{1, 34, 4, 69, 6, 7, 8, 23, 56, 90} <class 'set'>
```

Date: 07/10/2022

Experiment No: 5.9

Aim:

WAP to create a list containing unique elements from another list. Use set comprehension

Code:

```
l1=[1,2,3,4,5,6,7,6,4,4,6,5]
print("original value of
l1:",l1) l2=list({n for n in
l1}) print(l2,type(l2))
```

Output:

```
original value of l1: [1, 2, 3, 4, 5, 6, 7, 6, 4, 4, 6, 5]
[1, 2, 3, 4, 5, 6, 7] <class 'list'>
```

Date: 07/10/2022

Experiment No: 5.10

Aim:

WAP to create a set containing even numbers from 1 to 10 and insert 0 if odd numbers are encountered. Use set comprehension.

Code:

```
st1={n if n%2==0 else 0 for n in range(1,11)}
print(st1,type(st1))
```

Output:

```
{0, 2, 4, 6, 8, 10} <class 'set'>
```

Date: 07/10/2022

Experiment No: 5.11

Aim:

WAP to input values from user into two sets and perform set union, intersection and difference.

Code:

```
l1=[]
for n in range(1,5):
    v=input("enter the no.")
    l1.append(v)
print("list1: ", l1)
l2=[]
for n in range(1,5):
    v=input("enter the no.")
    l2.append(v)
print("list2: ",l2)
st1=set(l1)
st2=set(l2)
print(st1,st2)
print("union",st1.union(st2))
print("intersection",st1.intersection(st2))
print("difference",st1.difference(st2))
```

Output:

```
enter the no.12
enter the no.11
enter the no.32
enter the no.23
list1:  ['12', '11', '32', '23']
enter the no.12
enter the no.34
enter the no.41
enter the no.14
```

```
list2:  ['12', '34', '41', '14']
{'12', '23', '11', '32'} {'34', '12', '41', '14'}
union {'12', '34', '32', '11', '23', '41', '14'}
intersection {'12'}
difference {'11', '32', '23'}
```

Date: 07/10/2022

Experiment No: 5.12

Aim:

WAP to implement some general purpose functions in python.

Code:

```
#abs(n):Returns the absolute value of a number
x = int(input("enter any value:"))
print(abs(x))
#all(iterable):Returns True if all items in an iterable object are true
l1 = [0, 1, 1]
print(all(l1))
#any(iterable):Returns True if any item in an iterable object is true
t1 = (0, 1, False)
print(any(t1))
#chr(number):Returns a character from the specified Unicode code.
x = chr(97)
print(x)
#dir(object):Returns a list of the specified object's properties and methods
name = "John"
print(dir(name))
print()
#eval(string):evaluates and executes an expression
x = 'print(55)'
eval(x)
#help():Executes the built-in help system
help(print)
#id(object):Returns the id of an object
x = ('apple', 'banana', 'cherry')
print(id(x))
#input(message):Allowing user input and returns it as a string object
x = input("Enter your name:")
print("Hello, ",x)
```

```
#len(object):Returns the length of an object
l1 = ["apple", "orange", "cherry"]
print(len(l1))
#max(iterable):Returns the largest item in a homogeneous iterable
x = max(5, 10)
print(x)
#min(iterable):Returns the smallest item in a homogeneous iterable
x = min(5, 10)
print(x)
#ord(character):Convert an integer representing the Unicode of the specified
#character
x = ord("h")
print(x)
#pow(x,y):Returns the value of x to the power of y
x = pow(4, 3)
print(x)
#print(message):Prints to the standard output device
print("Hello World")
#reversed(iterable):Returns a reversed iterator
t1 = ('g', 'e', 'e', 'k', 's')
print(list(reversed(t1)))
#round(n):Rounds a number
x = round(5.76543, 2)
print(x)
#sorted(iterable, key=key,reverse=reverse):Returns a sorted list by sorting any
#iterator
a = ("b", "g", "a", "d", "f", "c", "h", "e")
print(sorted(a))
#sum(iterable):Sums the items of an iterator containing numeric values
a = (1, 2, 3, 4, 5)
print(sum(a))
#type(object):Returns the type of an object
a = ('apple', 'banana', 'cherry')
print(type(a))
#zip(iterable):Returns tuples containing the pairing of elements from all the
#iterables given as the input parameter.
a = ("John", "Charles", "Mike")
b = ("Jenny", "Christy", "Monica")
x = zip(a, b)
print(tuple(x))
```

Output:

```
enter any value:-567
567
False
True
a
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__',
'__init_ser__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__', sefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find',
'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier',
'islower', 'isnumeric', 'ispri', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition',
'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlh', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

55
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

2077320025536
Enter your name:aditi
Hello,  aditi
3
10
5
104
64
Hello World
['s', 'k', 'e', 'e', 'g']
5.77
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
15
<class 'tuple'>
```

```
(('John', 'Jenny'), ('Charles', 'Christy'), ('Mike', 'Monica'))
```

Date: 14/10/2022

Experiment No: 6.1

Aim:

WAP to build a dictionary from two lists, one containing names of 5 usernames and another list containing their passwords. Use dictionary comprehension. (Hint: use zip())

Code:

```
l1=["user1","user2","user3","user4"]
l2=["p1","p2","p3","p4"]
d1={k:v for k,v in zip(l1,l2)}
print(d1)
```

Output:

```
{'user1': 'p1', 'user2': 'p2', 'user3': 'p3', 'user4': 'p4'}
```

Date: 14/10/2022

Experiment No: 6.2

Aim:

WAP to create two dictionaries, one should contain keys as even numbers and their respective values and another dictionary should contain as odd numbers and their respective values. List 1 contents are [1,2,3,4,5,6,7,8,9]. List 2 contents are extracted from the string "One Two Three Four Five Six Seven Eight Nine". Use Dictionary comprehension (Hint: use zip())

Code:

```
l1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]
s1 = "One Two Three Four Five Six Seven Eight Nine"
l2 = s1.split()
print()
deven = {k: v for k, v in zip(l1, l2) if k % 2 == 0}
print(deven)
print()
dodd = {k: v for k, v in zip(l1, l2) if k % 2 != 0}
print(dodd)
```

Output:

```
{2: 'Two', 4: 'Four', 6: 'Six', 8: 'Eight'}

{1: 'One', 3: 'Three', 5: 'Five', 7: 'Seven', 9: 'Nine'}
```

Bhilai Institute of Technology, Durg.

Department of Computer Science and Engineering

PYTHON WITH DJANGO LAB 102393CS CRN: A59

Date: 14/10/2022

Experiment No: 6.3

Aim:

WAP to read five subject names and their corresponding marks and store it in a dictionary. Display that dictionary

Code:

```
l1 = {}
try:
    for i in range(1,6):
        print("Enter the name of subject" , i , ": ")
        a = input()
        print("Enter the marks of subject" , i , ": ")
        b = float(input())
        l1[a] = b
    print(l1)
except:
    print("Check Your entered data")
```

Output:

```
Enter the name of subject 1 :
python
Enter the marks of subject 1 :
78
Enter the name of subject 2 :
DE
Enter the marks of subject 2 :
56
Enter the name of subject 3 :
Maths
Enter the marks of subject 3 :
```

```
34
Enter the name of subject 4 :
DSA
Enter the marks of subject 4 :
48
Enter the name of subject 5 :
OS
Enter the marks of subject 5 :
89
{'python': 78.0, 'DE': 56.0, 'Maths': 34.0, 'DSA': 48.0, 'OS': 89.0}
```

Date: 14/10/2022

Experiment No: 6.4

Aim:

WAP to read five subject names and their corresponding marks and store it in a dictionary. Display subject name with marks that is maximum and minimum.

Code:

```
try:
    l1 = {}
    for i in range(5):
        a = input("Enter Subject Name :")
        b = float(input("Enter marks obtained :"))
        c = b
        l1[a] = b

    values = list(l1.values())
    max = values[0]
    min = values[0]

    for i in values:
        if i > max:
            max = i
        if i < min:
            min = i

    for i in l1:
        if l1[i] == max:
            print("max marks : {} {}".format(i, l1[i]))

    for i in l1:
        if l1[i] == min:
            print("min marks : {} {}".format(i, l1[i]))
except:
    print("Check the entered marks")
```

Output:

```
Enter Subject Name :maths
Enter marks obtained :49
Enter Subject Name :python
Enter marks obtained :45
Enter Subject Name :DSA
Enter marks obtained :78
Enter Subject Name :DE
Enter marks obtained :56
Enter Subject Name :OS
Enter marks obtained :78
max marks : DSA 78.0
max marks : OS 78.0
min marks : python 45.0
```

Date: 14/10/2022

Experiment No: 6.5

Aim:

WAP to create a Login validator. Use Dictionary to hold all users and their passwords. The existing users should be able to login by entering correct username and password.

Code:

```
dic = {'Arpita': 'best', 'Ganga': 'nice', 'Tushar': 'cool', 'Gopika': 'good', 'Shipra': 'kind'}

username = input("Enter the username : ")
password = input("Enter the password : ")

if username in dic and password == dic[username]:
    print('Correct Login')
else:
    print('Invalid Login... Please Try Again')
```

Output:

```
Enter the username : Arpita
Enter the password : best
Correct Login
```

Date: 14/10/2022

Experiment No: 6.6

Aim:

WAP to build a Dictionary to hold Name, Dept, Salary, DA and Gross of an employee.
i.Input the Name, Dept and Salary details from the user.
ii.Calculate DA as 20% of the Salary
iii.Gross = Salary + DA
iv.Display all the contents

Code:

```
t1 = ("Name", "Dept", "Salary")
dic = {key:input("Enter %s :"%key) for key in t1}

dic["Salary"] = int(dic["Salary"])

dic["DA"] = 0.2 * dic["Salary"]    # 20% of salary
dic["Gross"] = dic["Salary"] + dic["DA"]    # (Salary + DA) in Gross

print(dic)
```

Output:

```
Enter Name :shaheen
Enter Dept :cse
Enter Salary :24000
{'Name': 'shaheen ', 'Dept': 'cse', 'Salary': 24000, 'DA': 4800.0, 'Gross': 28800.0}
```

Bhilai Institute of Technology, Durg.

Department of Computer Science and Engineering

PYTHON WITH DJANGO LAB 102393CS CRN: A59

Date: 14/10/2022

Experiment No: 6.7

Aim:

WAP to build a Dictionary to hold Name, Dept, AggMarks, AggPer and Div for a student.
i.Input the Name and Dept details from the user.
ii.Input marks of 5 subjects (out of 100) and store the aggregate in AggMarks.
iii.percentage out of 500 and store in AggMarks.
iv.Display all the contents

Code:

```
det = {}
det["Name"] = input("Enter the name: ")
det["Dept"] = input("Enter the department: ")

aggr = 0
for i in range(1,6):
    score = int(input("Enter the score of subject %d :"%i))
    aggr += score
det["AggMarks"] = aggr
det["Percentage"] = aggr/5

print(det)
```

Output:

```
Enter the name: shaheen
Enter the department: cse
Enter the score of subject 1 :76
Enter the score of subject 2 :65
Enter the score of subject 3 :85
Enter the score of subject 4 :98
Enter the score of subject 5 :65
{'Name': 'shaheen', 'Dept': 'cse', 'AggMarks': 389, 'Percentage': 77.8}
```

Date: 07/11/2022

Experiment No: 7.1

Aim:

WAP to demonstrate the application of class, objects, data members and member methods.

Code:

```
class Test:
    def f1(self):# instance member method: object.f1()
        self.nm="CSE" #instance data member:object.nm
        print("Name =",self.nm)

ob1=Test()    #create a instance/object of a class
ob1.f1()
print("Name =",ob1.nm)
```

Output:

```
Name= CSE
Name= CSE
```

Date: 07/11/2022

Experiment No: 7.2

Aim:

WAP to demonstrate the application of constructor.

Code:

```
class Test():
    def __init__(self):
        print(“Constructor of class Test called”)
ob1=Test()
```

Output:

```
Constructor of class Test called
```

Date: 07/11/2022

Experiment No: 7.3

Aim:

WAP to demonstrate the application of destructor
del , None

Code:

```
class Test:
    def __init__(self):    # a constructor method
        print("init() called")
    def __del__(self):    # a destructor method
        print("del() called")
ob1=Test()    #init() called
ob2=Test()    #init() called
ob1=None    #del() called
del ob2   #del() called
```

Output:

```
init() called
init() called
del() called
del() called
```

Date: 07/11/2022

Experiment No: 7.4

Aim:

WAP to demonstrate two ways of calling the overridden parent class methods from child classes

Code:

```
class Parent:
    def f1(self):
        print("f1() is Parent class")
class Child(Parent):
    def f1(self):
        print(“Method-1”)
        Parent.f1(self) #Parent_class_name.method_name(self,argument_list)
        print(“Method-2”)
        super().f1() #super().method_name(argument_list)
        print("f1() is Child class")
ob1=Child()
ob1.f1()
```

Output:

```
Method-1
f1() is Parent class
Method-2
f1() is Parent class
f1() is Child class
```

Date: 07/11/2022

Experiment No: 7.5

Aim:

WAP to print the unique ID of each object created and destroyed. Also print the number of objects created. (hint: use instance and static members)

Code:

```
class A:
    cnt=0 #static/class data member
    def __init__(self):
        A.cnt+=1
        self.id=A.cnt
        print("Newly created object : ",self.id)
    def showCount():
        print("Number of Objects = ",A.cnt)
    def __del__(self):
        print("Object deleted : ",self.id)
ob1=A()
ob2=A()
ob3=A()
A.showCount()
ob3=None
ob2=None
ob1=None
```

Output:

```
Newly created object :  1
Newly created object :  2
Newly created object :  3
Number of Objects =  3
Object deleted :  3
Object deleted :  2
Object deleted :  1
```

Date: 07/11/2022

Experiment No: 7.6

Aim:

WAP to create a list of 5 objects and call the member methods of each object in the list.

Code:

```
class A:
   def f1(self):
      print(“f1() called”)
l1=[]
for n in range(5):
   ob1=A()
   l1.append(ob1)
for k in l1:
   k.f1()
```

Output:

```
f1() called
f1() called
f1() called
f1() called
f1() called
```

Date: 07/11/2022

Experiment No: 7.7

Aim:

WAP to create a list of 5 students and each student should have its roll number, name and aggregate marks. Implement necessary methods for filling the student details, displaying the student details and also display the student with highest and lowest aggregate marks.

Code:

```
class Student:
    def __init__(self):
        self.name = input("Enter the student’s name:")
        self.roll_no = int(input("Enter the student’s Roll number:"))
        self.aggregate_marks = int(input("Enter the student’s Aggregate Marks:"))

    def __str__(self):
        return "Name:{} Roll number:{} Aggregate
Marks:{}".format(self.name,self.roll_no,self.aggregate_marks)
class Result:
    def Marks_comparison(self):
        new_list=[]
        for student in l1:
            new_list.append(student.aggregate_marks)

        # For max marks
        print("Student with most Aggregate Marks")
        max_student = student1

        for student in l1:
            if student.aggregate_marks > max_student.aggregate_marks:
                max_student = student
        print(max_student)

        # For min marks
        print("Student with least Aggregate Marks")
        min_student = student1
        for student in l1:
            if student.aggregate_marks < min_student.aggregate_marks:
                min_student = student
        print(min_student)
print("List of students:")
l1=[]
for n in range(5):
    student1=Student()
    l1.append(student1)
for k in l1:
    print(k)

result = Result()
result.Marks_comparison()
```

Output:

```
Enter the student's name: Anwesha
Enter the student's Roll number: 1
Enter the student's Aggregate Marks: 99
Enter the student's name: Kushagra
Enter the student's Roll number: 2
Enter the student's Aggregate Marks: 98
Enter the student's name: Parth
Enter the student's Roll number: 3
Enter the student's Aggregate Marks: 100
Enter the student's name: Ganga
Enter the student's Roll number: 4
Enter the student's Aggregate Marks: 87
Enter the student's name: Ram
Enter the student's Roll number: 5
Enter the student's Aggregate Marks: 93
List of students:
Name:Anwesha Roll number:1 Aggregate Marks:99
Name:Kushagra Roll number:2 Aggregate Marks:98
Name:Parth Roll number:3 Aggregate Marks:100
Name:Ganga Roll number:4 Aggregate Marks:87
Name:Ram Roll number:5 Aggregate Marks:93
Student with most Aggregate Marks
Name:Parth Roll number:3 Aggregate Marks:100
```

```
Student with least Aggregate Marks
Name:Ganga Roll number:4 Aggregate Marks:87
```

Date: 07/11/2022

Experiment No: 7.8

Aim:

WAP to demonstrate Single level inheritance.

Code:

```
class A:
   def f1(self): print(“class-A”)
class B(A):
   def f2(self): print(“class-B”)
ob1=B()
ob1.f1()
ob2.f2()
```

Output:

```
class-A
class-B
```

Date: 07/11/2022

Experiment No: 7.9

Aim:

WAP to demonstrate Multi level inheritance.

Code:

```
class A:
   def f1(self): print(“f1()”)
class B(A):
   def f2(self): print(“f2()”)
class C(B):
   def f3(self): print(“f3()”)
ob1=C()
ob1.f1()
ob1.f2()
ob1=f3()
```

Output:

```
f1()
f2()
f3()
```

Date: 07/11/2022

Experiment No: 7.10

Aim:

WAP to demonstrate Multiple level inheritance.

Code:

```
class A:
   def f1(self): print(“f1()”)
class B:
   def f2(self): print(“f2()”)
class C(A,B):
   def f3(self): print(“f3()”)
ob1=C()
ob1.f1()
ob1.f2()
ob1.f3()
```

Output:

```
f1()
f2()
f3()
```

Date: 07/11/2022

Experiment No: 7.11

Aim:

WAP to demonstrate Hierarchical level inheritance.

Code:

```
class A:
   def f1(self): print(“f1()”)
class B(A):
   def f2(self): print(“f2()”)
class C(A):
   def f3(self): print(“f3()”)
ob1=B()
ob2=C()
ob1.f1()
ob1.f2()
ob1.f1()
ob1.f3()
```

Output:

```
f1()
f2()
f1()
f3()
```

Date: 07/11/2022

Experiment No: 7.12

Aim:

WAP to demonstrate Hybrid level inheritance.

Code:

```
class A:
   def f1(self): print(“f1()”)
class B(A):
   def f2(self): print(“f2()”)
class C(A):
   def f3(self): print(“f3()”)
class D(A):
   def f4(self): print(“f4()”)
ob1=D()
ob1.f1()
ob1.f2()
ob1.f3()
ob1.f4()
```

Output:

```
f1()
f2()
f3()
f4()
```

Date: 07/11/2022

Experiment No: 7.13

Aim:

WAP to create a parent class named Shape with a member method area() which has no definition, derive two child classes Circle and Triangle, implement the area() in both the child classes and necessary member methods to input necessary data. Display the area of Circle

Code:

```
class Shape:
    def area(self):
        pass
class Circle(Shape):
    def area(self):
        r=int(input(“Enter the radius of the circle:”))
        import math
        a=(math.pi)*r**2
        print(“Area of circle is:”,a)
class Triangle(Shape):
    def area(self):
        b=int(input(“Enter the base of the triangle:”))
        h=int(input(“Enter the height of the triangle:”))
        ar=0.5*h*b
ob1=Circle()
ob1.area()
ob2=Triangle()
ob2.area()
```

Output:

```
 Enter the radius of the circle:2
Area of circle is: 12.566370614359172
Enter the base of the triangle:6
Enter the height of the triangle:3
```

Date: 07/11/2022

Experiment No: 7.14

Aim:

WAP to create a parent class named Person with two member methods about() and biodata() without any definitions, derive two child classes named Actor and Actress, both should implement the about() to store the person’s name along with other details and biodata() details of the cinema they have done. Also include necessary members so that we can store and view 5 Actor and Actresses details.

Code:

```
class Person:
   def about(self):
      pass
   def biodata(self):
      pass
class Actor(Person):
   def about(self):
      self.nm=input(“Enter the actor’s name:”)
      self.age=int(input(“Enter the actor’s age:”))
      print(“Actor’s name is:”, self.nm)
      print(“Actor’s age is:”, self.age)
   def biodata(self):
      self.film=input(“Enter the films in which the actor has appeared:”)
      print(“Films in which the actor has appeared:”, self.film)
class Actress(Person):
   def about(self):
      self.nm=input(“Enter the actress’ name:”)
      self.age=int(input(“Enter the actress’ age:”))
      print(“Actress’ name is:”, self.nm)
      print(“Actress’ age is:”, self.age)
   def biodata(self):
      self.film=input(“Enter the films in which the actress has appeared:”)
      print(“Films in which the actress has appeared:”, self.film)

l1=[]
for n in range(5):
   ob1=Actor()
   l1.append(ob1)
```

```
for k in l1:
   k.about()
   k.biodata()
print()
l2=[]
for m in range(5):
   ob2=Actress()
   l1.append(ob2)
for j in l2:
   j.about()
   j.biodata()
```

Output:

```
Enter the actor’s name: Shahrukh Khan
Enter the actor’s age: 57
Enter the films in which the actor has appeared: Ra-One
Actor’s name is: Shahrukh Khan
Actor’s age is: 57
Films in which the actor has appeared: Ra-One
Enter the actor’s name: Salman Khan
Enter the actor’s age: 56
Enter the films in which the actor has appeared: Bodyguard
Actor’s name is: Salman Khan
Actor’s age is: 56
Films in which the actor has appeared: Bodyguard
Enter the actor’s name: Hrithik Roshan
Enter the actor’s age: 48
Enter the films in which the actor has appeared: Bang Bang
Actor’s name is: Hrithik Roshan
Actor’s age is: 48
Films in which the actor has appeared: Bang Bang
Enter the actor’s name: Amitabh Bachchan
Enter the actor’s age: 80
Enter the films in which the actor has appeared: Sholay
Actor’s name is: Amitabh Bachchan
Actor’s age is: 80
Films in which the actor has appeared: Sholay
Enter the actor’s name: Aamir Khan
Enter the actor’s age: 57
Enter the films in which the actor has appeared: Taare Zameen Par
Actor’s name is: Aamir Khan
```

```
Actor’s age is: 57
Films in which the actor has appeared: Taare Zameen Par

Enter the actress’ name: Katrina Kaif
Enter the actress’ age: 39
Enter the films in which the actress has appeared: Jagga Jasoos
Actress’ name is: Katrina Kaif
Actress’ age is: 39
Films in which the actress has appeared: Jagga Jasoos
Enter the actress’ name: Kareena Kapoor
Enter the actress’ age: 42
Enter the films in which the actress has appeared: Udta Punjab
Actress’ name is: Kareena Kapoor
Actress’ age is: 42
Films in which the actress has appeared: Udta Punjab
Enter the actress’ name: Anushka Sharma
Enter the actress’ age: 34
Enter the films in which the actress has appeared: Sultan
Actress’ name is: Anushka Sharma
Actress’ age is: 34
Films in which the actress has appeared: Sultan
Enter the actress’ name: Emma Watson
Enter the actress’ age: 32
Enter the films in which the actress has appeared: Noah
Actress’ name is: Emma Watson
Actress’ age is: 32
Films in which the actress has appeared: Noah
Enter the actress’ name: Deepika Padukone
Enter the actress’ age: 36
Enter the films in which the actress has appeared: Padmavat
Actress’ name is: Deepika Padukone
Actress’ age is: 36
Films in which the actress has appeared: Padmavat
```

Date: 14/11/2022

Experiment No: 8.1

Aim:

WAP to demonstrate the application of try, except, else and finally.

Code:

```
try:
   print(“Welcome to Python”)
except:
   print(“Except block”)
else:
   print(“Else block”)
finally:
   print(“Finally block”)
```

Output:

```
Welcome to Python
Else block
Finally block
```

Date: 14/11/2022

Experiment No: 8.2

Aim:

WAP to demonstrate how to raise an in-built exception and a customized exception.

Code:

```
#raising in-built exception
print(“Raising in-built exception:”)
try:
   raise NameError
except:
   print(“In-built exception caught”)

#raising customized exception
print(“Raising customized exception:”)
class Exp(Exception):
   def __str__(self):
      return “My Customized Exception”
try:
   raise Exp
except Exp as e:
   print(“Caught:”,e)
```

Output:

```
Raising in-built exception:
In-built exception caught
Raising customized exception:
My Customized Exception
```

Date: 14/11/2022

Experiment No: 8.3

Aim:

WAP to generate a user defined exception whenever a numeric value is found in a string which is input from the user.

Code:

```
class Exp(Exception):
    def __str__(self):
        return “Exception: Numeric value found”

def numerror():
    s1=input(“Enter a string without numeric value:”)
    s=s1.isalpha()
    if s==False:
        ob1=Exp()
        raise ob1
    else:
        print(s1)
try:
    numerror()
except Exp as e:
    print(e)
```

Output:

```
Enter a string without numeric value: abc1
Exception: Numeric value found
```

Date: 14/11/2022

Experiment No: 8.4

Aim:

WAP to generate a user defined exception whenever a character value is found in a string which is input from the user.

Code:

```
class Exp(Exception):
    def __str__(self):
        return “Exception: Character value found”)
def charerror():
    s1=input(“Enter a string without character value:”)
    s=s1.isnumeric()
    if s==False:
        ob1=Exp()
        raise ob1
    else:
        print(s1)
try:
    charerror()
except Exp as e:
    print(e)
```

Output:

```
Enter a string without character value: 123b
Exception: Character value found
```

Date: 14/11/2022

Experiment No: 8.5

Aim:

WAP to generate a user defined exception whenever a special symbol is found in a string which is input from the user.

Code:

```
class Exp(Exception):
    def __str__(self):
        return “Exception: Special symbol found”)
def specialerror():
    s1=input(“Enter a string without special symbols:”)
    s=s1.isalnum()
    if s==False:
        ob1=Exp()
        raise ob1
    else:
        print(s1)
try:
    specialerror()
except Exp as e:
    print(e)
```

Output:

```
Enter a string without special symbols: bcd@
Exception: Special symbol found
```

Date: 14/11/2022

Experiment No: 8.6

Aim:

WAP to input name and age from user. The program should generate an exception if the name and age are not in proper format i.e. age should be numeric and name should only contain alphabets.

Code:

```
class Exp(Exception):
    def __str__(self):
        return “Name or Age are not in proper format”
def formaterror():
    nm=input(“Enter your name:”)
    n=nm.isalpha()
    age=input(“Enter your age:”)
    a=age.isnumeric()
    if n==False or a==False:
        ob1=Exp()
        raise ob1
    else:
        print(“Name:”,nm)
        print(“Age:”,age)
try:
    formaterror()
except Exp as e:
    print(e)
```

Output:

```
Enter your name: abc!
Enter your age: 12d
Name or Age are not in proper format
```

Date: 14/11/2022

Experiment No: 8.7

Aim:

WAP to input age from user. The program should generate an exception if age contains any non-numeric data or if the age entered is below 1 or above 90.

Code:

```
class Exp(Exception):
    def __str__(self):
        return “Age Exception”
def agerror():
    age=input(“Enter your age:”)
    a=age.isnumeric()
    if a==False or int(age)<1 or int(age)>90:
        ob1=Exp()
        raise ob1
    else:
        print(“Age:”,age)
try:
    agerror()
except Exp as e:
    print(e)
```

Output:

```
Enter your age: 100
Age Exception
```

Date: 14/11/2022

Experiment No: 8.8

Aim:

WAP to develop a Name verifier, if the name contains any special symbol or numeric value then the program should generate a customized exception to indicate that only alphabets are allowed. The verification should be done using a dedicated method (eg. verifyName(string nm))

Code:

```
class Exp(Exception):
   def __str__(self):
      return “Name Verification failed: Only alphabets are allowed”
def  verifyName(nm):
   n=nm.isalpha()
   if n==False:
      ob1=Exp()
      raise ob1
   else:
      print(“Name:”,nm)
try:
   verifyName(“Shah@”)
except Exp as e:
   print(e)
```

Output:

```
Name Verification failed: Only alphabets are allowed
```