python-programming-lab-1

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 25-12-2024

Lab-1

0.0.1 01) WAP to print "Hello World"

[1]:
```
print('Hello World')
```

```
Hello World
```

0.0.2 02) WAP to print addition of two numbers with and without using input().

[3]:
```
a,b = 10,10
print(a+b)

n1 = int(input('Enter number 1'))
n2 = int(input('Enter number 2'))
print(n1+n2)
```

```
20
```

```
Enter number 1 10
Enter number 2 10
```

```
20
```

0.0.3 03) WAP to check the type of the variable.

[2]:
```
b = True
print(type(b))
```

```
<class 'bool'>
```

0.0.4 04) WAP to calculate simple interest.

[3]:
```
p = float(input("Enter the principal amount: "))
r = float(input("Enter the rate of interest (in percentage): "))
t = float(input("Enter the time period (in years): "))

si = (p * r * t) / 100
print(f"The simple interest is: {si}")
```

```
Enter the principal amount:  1000
Enter the rate of interest (in percentage):  25
Enter the time period (in years):  65
```

```
The simple interest is: 16250.0
```

0.0.5 05) WAP to calculate area and perimeter of a circle.

[6]:
```
r = float(input("Enter Radius "))
peri = 2 * 3.14 * r
area = 3.14 * r * r

print("Perimeter is",peri)
print("Area is",area)
```

```
Enter Radius  3
```

```
Perimeter is 18.84
Area is 28.259999999999998
```

0.0.6 06) WAP to calculate area of a triangle.

[7]:
```
base = float(input("Enter base "))
height = float(input("Enter Height "))
area  = 0.5 * base * height
print("Area of Triangle is",area)
```

```
Enter base  1
Enter Height  1
```

```
Area of Triangle is 0.5
```

0.0.7 07) WAP to compute quotient and remainder.

[8]:
```
dividend = int(input("Enter the dividend: "))
divisor = int(input("Enter the divisor: "))

quotient = dividend // divisor
remainder = dividend % divisor
```

```
print("Quotient:", quotient)
print("Remainder:", remainder)
```

```
Enter the dividend:  45
Enter the divisor:  5
```

```
Quotient: 9
Remainder: 0
```

0.0.8 08) WAP to convert degree into Fahrenheit and vice versa.

[5]:
```
f = float(input("Enter temp in Fahrenheit: "))
c = (f - 32) * (5 / 9)
print(str(f) + " degrees Fahrenheit is equal to " + str(c) + " degrees Celsius.
  ↪")
```

```
Enter temp in Fahrenheit:  100
```

```
100.0 degrees Fahrenheit is equal to 37.77777777777778 degrees Celsius.
```

1 09) WAP to find the distance between two points in 2-D space.

[26]:
```
import math

x1 = float(input("Enter x1"));
y1 = float(input("Enter y1"));
x2 = float(input("Enter x2"));
y2 = float(input("Enter y2"));

distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)

print(f"The distance between the points ({x1}, {y1}) and ({x2}, {y2}) is:␣
  ↪{distance}")
```

```
Enter x1 5
Enter y1 5
Enter x2 5
Enter y2 5
```

```
The distance between the points (5.0, 5.0) and (5.0, 5.0) is: 0.0
```

1.0.1 10) WAP to print sum of n natural numbers.

[16]:
```
n = int(input("Enter number n: "))

sum_n = (n * (n + 1)) // 2

print(f"The sum of the first {n} natural numbers is: {sum_n}")
```

```
Enter number n:  5

The sum of the first 5 natural numbers is: 15
```

1.0.2 11) WAP to print sum of square of n natural numbers.

[17]:
```
n = int(input("Enter number n: "))

sum_of_squares = (n * (n + 1) * (2 * n + 1)) // 6

print(f"The sum of squares of the first {n} natural numbers is:␣
  ↪{sum_of_squares}")
```

```
Enter number n:  55

The sum of squares of the first 55 natural numbers is: 56980
```

1.0.3 12) WAP to concate the first and last name of the student.

[21]:
```
fn = input("Enter the first name: ")
ln = input("Enter the last name: ")

full = fn + " " + ln

print(f"The full name of the student is: {full}")
```

```
Enter the first name:  rtn
Enter the last name:  \srtj

The full name of the student is: rtn \srtj
```

1.0.4 13) WAP to swap two numbers.

[22]:
```
a = float(input("Enter the first number: "))
b = float(input("Enter the second number: "))

temp = a
a = b
b = temp

print(f"After swapping: a = {a}, b = {b}")
```

```
Enter the first number:  54
Enter the second number:  56

After swapping: a = 56.0, b = 54.0
```

1.0.5 14) WAP to get the distance from user into kilometer, and convert it into meter, feet, inches and centimeter.

[24]:
```
kilometers = float(input("Enter the distance in kilometers: "))

meters = kilometers * 1000
feet = kilometers * 3280.84
inches = kilometers * 39370.1
centimeters = kilometers * 100000

print(f"{meters} meters")
print(f"{feet} feet")
print(f"{inches} inches")
print(f"{centimeters} centimeters")
```

```
Enter the distance in kilometers:  4
```

```
4000.0 meters
13123.36 feet
157480.4 inches
400000.0 centimeters
```

1.0.6 15) WAP to get day, month and year from the user and print the date in the given format: 23-11-2024.

[27]:
```
day = int(input("Enter the day: "))
month = int(input("Enter the month: "))
year = int(input("Enter the year: "))

formatted_date = f"{day:02d}-{month:02d}-{year}"

print("The date in the given format is:", formatted_date)
```

```
Enter the day:  32
Enter the month:  32
Enter the year:  32
```

```
The date in the given format is: 32-32-32
```

python-programming-lab-2

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 02-12-2024

Lab-2

1 if..else..

1.0.1 01) WAP to check whether the given number is positive or negative.

[3]:
```python
num = float(input('Enter Number'))
if(num > 0):
    print('Number is positive')
else:
    print('Number is Negative')
```

```
Enter Number -10

Number is Negative
```

1.0.2 02) WAP to check whether the given number is odd or even.

[5]:
```python
num = float(input('Enter Number'))
if(num % 2 == 0):
    print('Number is Even')
else:
    print('Number is Odd')
```

```
Enter Number 3

Number is Odd
```

1.0.3 03) WAP to find out largest number from given two numbers using simple if and ternary operator.

[7]:
```python
num1 = float(input('Enter Number'))
num2 = float(input('Enter Number'))
print('num1 is largest') if num1 > num2 else print('Num2 is Largest')
```

```
Enter Number 20
Enter Number 10
```

```
num1 is largest
```

1.0.4 04) WAP to find out largest number from given three numbers.

[10]:
```python
num1 = float(input('Enter Number'))
num2 = float(input('Enter Number'))
num3 = float(input('Enter Number'))
if (num1 > num2 and num1 > num3):
    print('Num1 is largest')
elif (num2 > num1 and num2 > num3):
    print('Num2 is largest')
else:
    print('Num3 is largest')
```

```
Enter Number 100
Enter Number 10
Enter Number 20
```

```
Num1 is largest
```

1.0.5 05) WAP to check whether the given year is leap year or not.

[If a year can be divisible by 4 but not divisible by 100 then it is leap year but if it is divisible by 400 then it is leap year]

[13]:
```python
year = int(input('Enter Year'))
if (year % 4 == 0 and year % 100 != 0 or year % 400 == 0 ):
    print('Leap year')
else:
    print('Not Leap year')
```

```
Enter Year 2021
```

```
Not Leap year
```

1.0.6 06) WAP in python to display the name of the day according to the number given by the user.

1.0.7 07) WAP to implement simple calculator which performs (add,sub,mul,div) of two no. based on user input.

[28]:
```
op = input("Enter oprator")
num1 = int(input("Enter number 1"))
num2 = int(input("Enter number 2"))
if (op == '+'):
    ans = int(num1+num2)
    print(ans)
elif (op == '-'):
    ans = int(num1 - num2)
    print(ans)
elif (op == '*'):
    ans = int(num1 * num2)
    print(ans)
elif (op == '/'):
    if(num2 == 0):
        print('Not possible')
    else:
        ans = num1 / num2
        print(ans)
else:
    print("Invalid oprator")
```

```
Enter oprator /
Enter number 1 10
Enter number 2 0

Not possible
```

1.0.8 08) WAP to read marks of five subjects. Calculate percentage and print class accordingly.

Fail below 35 Pass Class between 35 to 45 Second Class between 45 to 60 First Class between 60 to 70 Distinction if more than 70

[32]:
```
num1 = int(input("Enter marks 1"))
num2 = int(input("Enter marks 2"))
num3 = int(input("Enter marks 3"))
num4 = int(input("Enter marks 4"))
num5 = int(input("Enter marks 5"))
total = num1+num2+num3+num4+num5
per = (total/500) * 100
if per < 35:
    print("Fail")
elif per < 45 and per > 35:
    print("pass")
elif per > 45 and per < 60:
    print("Second class")
elif per > 60 and per < 70:
    print("First class")
else:
    print("Distinction")
```

```
Enter marks 1 20
Enter marks 2 40
Enter marks 3 50
Enter marks 4 30
Enter marks 5 10

Fail
```

1.0.9 09) Three sides of a triangle are entered through the keyboard, WAP to check whether the triangle is isosceles, equilateral, scalene or right-angled triangle.

[2]:
```
a = float(input("Enter Number "))
b = float(input("Enter Number "))
c = float(input("Enter Number "))
if(a==b and a == c):
 print("Triangle is Equilateral")
elif(a==b or b == c or c==a):
 print("Triangel is isoceles")
elif((a*a+b*b)==(c*c) or (a*a+c*c)==(b*b) or (c*c+b*b)==(a*a)):
 print("Right Angled")
else :
 print("Scalene")
```

```
Enter Number  10
Enter Number  10
Enter Number  10

Triangle is Equilateral
```

1.0.10 10) WAP to find the second largest number among three user input numbers.

[47]:
```
num1 = float(input('Enter Number 1'))
num2 = float(input('Enter Number 2'))
num3 = float(input('Enter Number 3'))

if num1 > num2 and num1 > num3 and num2 > num3:
    print('Num2 is second largest')
elif num2 > num1 and num2 > num3 and num1 > num3:
    print('Num1 is second largest')
else:
    print('Num3 is second largest')
```

```
Enter Number 1 100
Enter Number 2 100
Enter Number 3 100
```

```
Num3 is second largest
```

1.0.11 11) WAP to calculate electricity bill based on following criteria. Which takes the unit from the user.

a. First 1 to 50 units – Rs. 2.60/unit
b. Next 50 to 100 units – Rs. 3.25/unit
c. Next 100 to 200 units – Rs. 5.26/unit
d. above 200 units – Rs. 8.45/unit

[3]:
```
a = float(input("Enter Number"))
if(a>200):
 print("Bill is : ",{(50*2.6)+(50*3.25)+(100*5.26)+((a-200)*8.45)})
elif(a>100 and a<200):
 print("Bill is : ",{(50*2.6)+(50*3.25)+((a-100)*5.26)})
elif(a>50 and a<100):
 print("Bill is : ",{(50*2.6)+((a-50)*3.25)})
else:
 print("Bill is : ",{(a-50)*2.6})
```

```
Enter Number 1000
```

```
Bill is :  {7578.499999999999}
```

[ ]:
```

```

python-programming-lab-3

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 09-12-2024

Lab - 3

1 for and while loop

1.0.1 01) WAP to print 1 to 10.

[19]:
```python
for i in range(1,11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

1.0.2 02) WAP to print 1 to n.

[18]:
```python
n = int(input("Enter number"))
for i in range(1,n+1):
    print(i)
```

```
Enter number 5
```

```
1
2
3
```

```
4
5
```

1.0.3 03) WAP to print odd numbers between 1 to n.

[17]:
```
n = int(input("Enter number"))
for i in range(1,n+1,2):
    print(i)
```

```
Enter number 10
```

```
1
3
5
7
9
```

1.0.4 04) WAP to print numbers between two given numbers which is divisible by 2 but not divisible by 3.

[16]:
```
n1 = int(input("Enter number 1 "))
n2 = int(input("Enter number 2 "))

for i in range(n1,n2+1):
    if(i%2 == 0 and i%3 != 0):
        print(i)
```

```
Enter number 1  5
Enter number 2  15
```

```
8
10
14
```

1.0.5 05) WAP to print sum of 1 to n numbers.

[13]:
```
n = int(input("Enter number "))
ans = 0
for i in range(1,n+1):
    ans += i
print(ans)
```

```
Enter number  5
```

```
15
```

1.0.6 06) WAP to print sum of series 1 + 4 + 9 + 16 + 25 + 36 + ...n.

[12]:
```
n = int(input("Enter number "))
ans = 0
for i in range(1,n+1):
    ans = ans+(i*i)
print(ans)
```

```
Enter number  5

55
```

1.0.7 07) WAP to print sum of series 1 − 2 + 3 − 4 + 5 − 6 + 7 ... n.

[11]:
```
n = int(input("Enter number "))
ans = 0
for i in range(1,n+1):
    if(i%2==0):
        ans -= i
    else:
        ans+=i
print(ans)
```

```
Enter number  4

-2
```

1.0.8 08) WAP to print multiplication table of given number.

[10]:
```
n = int(input("Enter number "))
for i in range(1,11):
    print(n ,'x',i,'=',n*i)
```

```
Enter number  5

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

1.0.9 09) WAP to find factorial of the given number.

[8]:
```
n = int(input("Enter number "))
ans = 1
for i in range(1,n+1):
    ans*=i
print(n,'factorial is',ans)
```

```
Enter number  5

5 factorial is 120
```

1.0.10 10) WAP to find factors of the given number.

[6]:
```
n = int(input("Enter number "))
for i in range(1,n+1):
    if(n%i==0):
        print(i)
```

```
Enter number  4

1
2
4
```

1.0.11 11) WAP to find whether the given number is prime or not.

[5]:
```
n = int(input("Enter number "))
count = 0
for i in range(2,n):
    if(n%i==0):
        count+=1
if(count == 0):
    print(n,'is prime number')
else:
    print(n,'is not prime number')
```

```
Enter number  13

13 is prime number
```

1.0.12 12) WAP to print sum of digits of given number.

[ ]:
```
n = int(input("Enter number "))
ans=0
while(n!=0):
    ans+=(n%10)
    n //= 10
print(ans)
```

1.0.13 13) WAP to check whether the given number is palindrome or not

[21]:
```
n = int(input("Enter number "))
temp = n
reverse = 0
while n > 0:
    digit = n % 10
    reverse = reverse * 10 + digit
    n = n // 10
if temp == reverse:
    print("Palindrome")
else:
    print("Not Palindrome")
```

```
Enter number  153
```

```
Not Palindrome
```

1.0.14 14) WAP to print GCD of given two numbers.

[24]:
```
a = int(input("Enter number "))
b = int(input("Enter number "))
while b != 0:
    a, b = b, a % b
print(a)
```

```
Enter number  15
Enter number  5
```

```
5
```

python-programming-lab-4

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 16-12-2024

Lab - 4

1 String

1.0.1 01) WAP to check whether the given string is palindrome or not.

[2]:
```python
str = input("Enter String ")
temp = str[::-1]
if temp == str:
    print(f"{str} is palindrome")
else:
    print(f"{str} is not palindrome")
```

```
Enter String  aba
```

```
aba is palindrome
```

1.0.2 02) WAP to reverse the words in the given string.

[6]:
```python
str = input("Enter String ")
temp = str.split(" ")
ans = temp[::-1]
print(ans)
```

```
Enter String  neel gohel
```

```
['gohel', 'neel']
```

1.0.3 03) WAP to remove ith character from given string.

[7]:
```
str = input("Enter String ")
index = int(input("Enter index "))
temp = str[:index:] + str[index+1::]
print(temp)
```

```
Enter String  neel
Enter index  1
```

```
nel
```

1.0.4 04) WAP to find length of string without using len function.

[8]:
```
str = input("Enter String ")
count = 0
for i in str:
    count += 1
print(f'Length of {str} is {count} ')
```

```
Enter String  neel
```

```
Length of neel is 4
```

1.0.5 05) WAP to print even length word in string.

[9]:
```
str = input("Enter String ")
temp = str.split(" ")
for i in temp:
    if(len(i)%2==0):
        print(i)
```

```
Enter String  neel gohel
```

```
neel
```

1.0.6 06) WAP to count numbers of vowels in given string.

[10]:
```
str = input("Enter String ")
count=0
for i in str:
    if(i=='a' or i=='e' or i=='o' or i=='i' or i=='u'):
        count+=1
print(count)
```

```
Enter String  neel
```

```
2
```

1.0.7 07) WAP to capitalize the first and last character of each word in a string.

[12]:
```
str = input("Enter String ")
str = str.title()
result = ""
temp = str.split(" ")
for word in temp:
    result = result + word[:-1] + word[-1].upper()+" "
print(result)
```

```
Enter String  neel gohel darshan

NeeL GoheL DarshaN
```

1.0.8 08) WAP to convert given array to string.

[15]:
```
arr = ['Darshan' , 'University' , 'Rajkot']
ans = " ".join(arr)
print(ans)
```

```
Darshan University Rajkot
```

1.0.9 09) Check if the password and confirm password is same or not.

1.0.10 In case of only case's mistake, show the error message.

[19]:
```
password = input("Enter your password: ")
confirmPass = input("Confirm your password: ")

if password == confirmPass:
    print("Password confirmed successfully!")
else:
    if password.lower() == confirmPass.lower():
        print("Error: Passwords do not match due to case sensitivity.")
    else:
        print("Error: Passwords do not match.")
```

```
Enter your password:  neel
Confirm your password:  neel

Error: Passwords do not match.
```

1.0.11 10) : Display credit card number.

1.0.12 card no. : 1234 5678 9012 3456

1.0.13 display as : **** **** **** 3456

[22]:
```python
card_number = "1234 5678 9012 3456"

masked_card_number = "**** **** **** " + card_number[-4:]

print("Displayed card number:", masked_card_number)
```

```
Displayed card number: **** **** **** 3456
```

1.0.14 11) : Checking if the two strings are Anagram or not.

1.0.15 s1 = decimal and s2 = medical are Anagram

[25]:
```python
s1 = "decimal"
s2 = "medical"

if len(s1) == len(s2):
    if sorted(s1) == sorted(s2):
        print(f"{s1} and {s2} are anagrams.")
    else:
        print(f"{s1} and {s2} are not anagrams.")
else:
    print(f"{s1} and {s2} are not anagrams.")
```

```
decimal and medical are anagrams.
```

1.0.16 12) : Rearrange the given string. First lowercase then uppercase alphabets.

1.0.17 input : EHlsarwiwhtwMV

1.0.18 output : lsarwiwhtwEHMV

[26]:
```python
input_string = "EHlsarwiwhtwMV"

lowercase_letters = [char for char in input_string if char.islower()]
uppercase_letters = [char for char in input_string if char.isupper()]

output_string = ''.join(lowercase_letters) + ''.join(uppercase_letters)

print("Rearranged string:", output_string)
```

```
Rearranged string: lsarwiwhtwEHMV
```

python-programming-lab-5

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 23-12-2024

Lab - 5

1 List

1.0.1 01) WAP to find sum of all the elements in a List.

[1]:
```
l1 = [1,2,3,4,5]
sum = 0
for i in l1:
    sum+=i
print(sum)
```

```
15
```

1.0.2 02) WAP to find largest element in a List.

[2]:
```
l1 = [1,20,15,87,3,4,5]
max = l1[0]
for i in l1:
    if(i > max):
        max = i
print(max)
```

```
87
```

1.0.3 03) WAP to find the length of a List.

[3]:
```
l1 = [1,20,15,87,3,4,5]
print(len(l1))
```

```
7
```

1.0.4 04) WAP to interchange first and last elements in a list.

[4]:
```
l1 = [1,20,15,87,3,4,5]
first = l1[0]
last = l1.pop()
l1.append(first)
l1[0] = last
print(l1)
```

```
[5, 20, 15, 87, 3, 4, 1]
```

1.0.5 05) WAP to split the List into two parts and append the first part to the end.

[6]:
```
l1 = [1,2,3,4,5,6,7,8,9,10]
n = int(len(l1)/2)

firstpart = l1[:n]
secondpart = l1[n:]

print(firstpart)
print(secondpart)

secondpart.extend(firstpart)
print(secondpart)
```

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

1.0.6 06) WAP to interchange the elements on two positions entered by a user.

[7]:
```
l1 = [1,20,15,87,3,4,5]

n1 = int(input("Enter 1st position "))
n2 = int(input("Enter 2st position "))

temp = l1[n1]
l1[n1] = l1[n2]
l1[n2] = temp

print(l1)
```

```
Enter 1st position  1
Enter 2st position  2
```

```
[1, 15, 20, 87, 3, 4, 5]
```

1.0.7 07) WAP to reverse the list entered by user.

[8]:
```
n1 = int(input("Enter size of list "))
l1 = []
for i in range(0,n1):
    l1.append(int(input("Enter number ")))
l1.reverse()
print(l1)
```

```
Enter size of list  3
Enter number  1
Enter number  2
Enter number  3
```

```
[3, 2, 1]
```

1.0.8 08) WAP to print even numbers in a list.

[9]:
```
n1 = int(input("Enter size of list "))
l1 = []
l2 = []
for i in range(0,n1):
    l1.append(int(input("Enter number ")))
for i in l1:
    if(i%2 == 0):
        l2.append(i)
print(l2)
```

```
Enter size of list  5
Enter number  1
Enter number  2
Enter number  3
Enter number  4
Enter number  5
```

```
[2, 4]
```

1.0.9 09) WAP to count unique items in a list.

[10]:
```
n1 = int(input("Enter size of list "))
l1 = []
l2 = []
count = 0
for i in range(0,n1):
    l1.append(int(input("Enter number ")))
for i in l1:
    if i not in l2:
        l2.append(i)
        count+=1
```

```
print(count)
print(l2)
```

```
Enter size of list  5
Enter number  1
Enter number  2
Enter number  3
Enter number  4
Enter number  5
```

```
5
[1, 2, 3, 4, 5]
```

1.0.10 10) WAP to copy a list.

[11]:
```
n1 = int(input("Enter size of list "))
l1 = []
l2 = []
for i in range(0,n1):
    l1.append(int(input("Enter number ")))
for i in l1:
    l2.append(i)
print(l2)
```

```
Enter size of list  5
Enter number  4
Enter number  4
Enter number  4
Enter number  4
Enter number  4
```

```
[4, 4, 4, 4, 4]
```

1.0.11 11) WAP to print all odd numbers in a given range.

[12]:
```
n1 = int(input("Enter size of list "))
l1 = []
l2 = []
for i in range(0,n1):
    l1.append(int(input("Enter number ")))
ran = int(input("Enter range "))
for i in range(0,ran):
    if(l1[i]%2 != 0):
        l2.append(l1[i])
print(l2)
```

```
Enter size of list  5
Enter number  1
Enter number  2
Enter number  3
Enter number  4
Enter number  5
Enter range  4
```

```
[1, 3]
```

1.0.12 12) WAP to count occurrences of an element in a list.

[13]:
```python
n1 = int(input("Enter size of list "))
l1 = []
for i in range(0,n1):
    l1.append(int(input("Enter number ")))
ele = int(input("Enter element to count "))
count = l1.count(ele)
print(count)
```

```
Enter size of list  5
Enter number  1
Enter number  1
Enter number  1
Enter number  2
Enter number  3
Enter element to count  2
```

```
1
```

1.0.13 13) WAP to find second largest number in a list.

[14]:
```python
n1 = int(input("Enter size of list "))
l1 = []
for i in range(0,n1):
    l1.append(int(input("Enter number ")))
l1.sort()
n = int(len(l1))
print(l1[n-2])
```

```
Enter size of list  5
Enter number  1
Enter number  2
Enter number  3
Enter number  4
Enter number  5
```

```
4
```

1.0.14 14) WAP to extract elements with frequency greater than K.

[16]:
```
l1 = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5]
k = 2

freq = {}

for num in l1:
    if num in freq:
        freq[num] += 1
    else:
        freq[num] = 1

result = []

for num, count in freq.items():
    if count > k:
        result.append(num)

print("Elements with frequency greater than", k, ":", result)
```

```
Elements with frequency greater than 2 : [3, 4]
```

1.0.15 15) WAP to create a list of squared numbers from 0 to 9 with and without using List Comprehension.

[15]:
```
l1 = [0,1,2,3,4,5,6,7,8,9]
l2 = []
for i in l1:
    l2.append(i**2)
print(l2)

l3 = [i**2 for i in l1]
print(l3)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

1.0.16 16) WAP to create a new list (fruit whose name starts with 'b') from the list of fruits given by user.

[19]:
```
fruits = input("Enter a list of fruits : ").split()

b_fruits = [fruit for fruit in fruits if fruit.lower().startswith('b')]

print("Fruits starting with 'b':", b_fruits)
```

```
Enter a list of fruits :  bb
```

```
Fruits starting with 'b': ['bb']
```

1.0.17 17) WAP to create a list of common elements from given two lists.

[18]:
```
n1 = int(input("Enter size of list 1 "))
l1 = []
for i in range(0,n1):
    l1.append(int(input("Enter number ")))
n2 = int(input("Enter size of list 2 "))
l2 = []
for i in range(0,n2):
    l2.append(int(input("Enter number ")))

common_elements = [element for element in l1 if element in l2]

print("Common elements:", common_elements)
```

```
Enter size of list 1  5
Enter number  1
Enter number  2
Enter number  3
Enter number  4
Enter number  5
Enter size of list 2  5
Enter number  2
Enter number  3
Enter number  4
Enter number  5
Enter number  6
```

```
Common elements: [2, 3, 4, 5]
```

[ ]:
```

```

python-programming-lab-6

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 30-12-2024

Lab - 6

1 Tuple

1.0.1 01) WAP to find sum of tuple elements.

[4]:
```python
t1 = ((1,2,2,3,4,5,6))
sum = 0
for i in t1 :
    sum += i
print(sum)
```

```
23
```

1.0.2 02) WAP to find Maximum and Minimum K elements in a given tuple.

[45]:
```python
t1 = (15,1,2,1,2,3,26,21,3,4,5,6)
k = int(input("Enter k "))
s = {i for i in t1}
l1 = list(s)
min = l1[:k]
max = l1[-k:]
print(max)
print(min)
```

```
Enter k 3
[15, 21, 26]
[1, 2, 3]
```

1.0.3 03) WAP to find tuples which have all elements divisible by K from a list of tuples.

[59]:
```
l1 = [(1,2,-33) , (2,6,4) , (4,5,6)]
k = int(input("Enter k "))
res = [t for t in l1 if all(i%k==0 for i in t)]
print(res)
```

```
Enter k 2
[(2, 6, 4)]
```

1.0.4 04) WAP to create a list of tuples from given list having number and its cube in each tuple.

[52]:
```
n = int(input("Enter size of list "))
l1 = []
for i in range(0,n):
    n1 = int(input("Enter number "))
    l1.append(n1)

l1 = [(i,i**3) for i in l1]
print(l1)
```

```
Enter size of list 3
Enter number 1
Enter number 2
Enter number 3
[1, 2, 3]
[(1, 1), (2, 8), (3, 27)]
```

1.0.5 05) WAP to find tuples with all positive elements from the given list of tuples.

[57]:
```
l1 = [(1,2,-33) , (2,3,4) , (4,5,6)]
res = [t for t in l1 if all(i>0 for i in t)]
print(res)
```

```
[(2, 3, 4), (4, 5, 6)]
```

1.0.6 06) WAP to add tuple to list and vice – versa.

[69]:
```
l1 = [(1,2,-33) , (2,3,4) , (4,5,6)]
t = (7,8)
l1.append(t)
print(l1)
```

```
[(1, 2, -33), (2, 3, 4), (4, 5, 6), (7, 8)]
```

1.0.7 07) WAP to remove tuples of length K.

[68]:
```
t4 = (15,1,3,26,21,3,4,5,6)
k = int(input("Enter k "))
legth = 0
for i in t4:
    legth+=1
new = t4[:legth - k]
print(new)
```

```
Enter k 6
(15, 1, 3)
```

1.0.8 08) WAP to remove duplicates from tuple.

[62]:
```
t5 = (15,1,2,1,2,3,26,21,3,4,5,6)
s = {i for i in t5}
tu = tuple(s)
print(tu)
```

```
(1, 2, 3, 4, 5, 6, 15, 21, 26)
```

1.0.9 09) WAP to multiply adjacent elements of a tuple and print that resultant tuple.

[1]:
```
tup = (1,2,3,4,3,2,1)
result = ()
for i in range(len(tup) - 1):
    result += (tup[i] * tup[i + 1],)
print(result)
```

```
(2, 6, 12, 12, 6, 2)
```

1.0.10 10) WAP to test if the given tuple is distinct or not.

[80]:
```
t = (1,2,3,3,4,5)
temp = 0
for i in t:
    if(t.count(i) > 1):
        temp = i
print(temp,"is multiple time tuple not distinct")
```

```
3 is multiple time tuple not distinct
```

[ ]:
```

```

python-programming-lab-7

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 06-01-2025

Lab - 7

1 Set & Dictionary

1.0.1 01) WAP to iterate over a set.

[1]:
```
s1 = set()
n = int(input("Enter number of Element "))
for i in range(n):
    x = int(input("Enter number "))
    s1.add(x)
print(s1)
s2 = {i**2 for i in s1}
print(s2)
```

```
Enter number of Element  3
Enter number  1
Enter number  2
Enter number  3

{1, 2, 3}
{1, 4, 9}
```

1.0.2 02) WAP to convert set into list, string and tuple.

[2]:
```
s1 = set()
n = int(input("Enter number of Element "))
for i in range(n):
    x = int(input("Enter number "))
    s1.add(x)
print(s1)
```

```

l1 = list(s1)
print(l1)
print(type(l1))

t1 = tuple(s1)
print(t1)
print(type(t1))

str1 = str(s1)
print(str1)
print(type(str1))
```

```
Enter number of Element  5
Enter number  1
Enter number  2
Enter number  34
Enter number  5
Enter number  6

{1, 2, 34, 5, 6}
[1, 2, 34, 5, 6]
<class 'list'>
(1, 2, 34, 5, 6)
<class 'tuple'>
{1, 2, 34, 5, 6}
<class 'str'>
```

1.0.3 03) WAP to find Maximum and Minimum from a set.

[3]:
```python
s1 = set()
n = int(input("Enter number of Element "))
for i in range(n):
    x = int(input("Enter number "))
    s1.add(x)
print(s1)
maximum = 0
for i in s1:
    if(i>maximum):
        maximum = i
print('Maximum element is ',maximum)
minimum = maximum
for i in s1:
    if(i < minimum):
        minimum = i
print('Minimum element is ',minimum)
```

```
Enter number of Element  5
Enter number  1
Enter number  2
Enter number  3
Enter number  4
Enter number  6
```

```
{1, 2, 3, 4, 6}
Maximum element is  6
Minimum element is  1
```

1.0.4 04) WAP to perform union of two sets.

[4]:
```python
s1 = set()
n1 = int(input("Enter number of Element "))
for i in range(n1):
    x1 = int(input("Enter number "))
    s1.add(x1)
print(s1)

s2 = set()
n2 = int(input("Enter number of Element "))
for i in range(n2):
    x2 = int(input("Enter number "))
    s2.add(x2)
print(s2)

print('Union of two set is ',s1.union(s2))
```

```
Enter number of Element  1
Enter number  1
```

```
{1}
```

```
Enter number of Element  2
Enter number  1
Enter number  2
```

```
{1, 2}
Union of two set is  {1, 2}
```

1.0.5 05) WAP to check if two lists have at-least one element common.

[5]:
```python
s1 = set()
n1 = int(input("Enter number of Element "))
for i in range(n1):
    x1 = int(input("Enter number "))
    s1.add(x1)
print(s1)
```

```
s2 = set()
n2 = int(input("Enter number of Element "))
for i in range(n2):
    x2 = int(input("Enter number "))
    s2.add(x2)
print(s2)

print('Intersection of two set is ',s1.intersection(s2))
```

```
Enter number of Element  5
Enter number  1
Enter number  4
Enter number  5
Enter number  6
Enter number  4

{1, 4, 5, 6}

Enter number of Element  5
Enter number  4
Enter number  7
Enter number  8
Enter number  9
Enter number  6

{4, 6, 7, 8, 9}
Intersection of two set is  {4, 6}
```

1.0.6 06) WAP to remove duplicates from list.

[6]:
```
l1 = []
n1 = int(input("Enter number of Element "))
for i in range(n1):
    x1 = int(input("Enter number "))
    l1.append(x1)
print(l1)

s1 = set(l1)
print(s1)
```

```
Enter number of Element  2
Enter number  1
Enter number  1

[1, 1]
{1}
```

1.0.7 07) WAP to find unique words in the given string.

[8]:
```
str1 = input("Enter a String ")

words = str1.split()

ans = []

for word in words:
    if word not in ans:
        ans.append(word)
print(ans)
```

```
Enter a String  neel neel neel
```

```
['neel']
```

1.0.8 08) WAP to remove common elements of set A & B from set A.

[9]:
```
s1 = set()
n1 = int(input("Enter number of Element "))
for i in range(n1):
    x1 = int(input("Enter number "))
    s1.add(x1)
print(s1)

s2 = set()
n2 = int(input("Enter number of Element "))
for i in range(n2):
    x2 = int(input("Enter number "))
    s2.add(x2)
print(s2)

s3 = s1.intersection(s2)
print(s3)

ans = s1.difference(s3)
print(ans)
```

```
Enter number of Element  5
Enter number  1
Enter number  2
Enter number  3
Enter number  4
Enter number  5
```

```
{1, 2, 3, 4, 5}
```

```
Enter number of Element  5
Enter number  5
Enter number  4
Enter number  16
Enter number  9
Enter number  5

{16, 9, 4, 5}
{4, 5}
{1, 2, 3}
```

1.0.9 09) WAP to check whether two given strings are anagram or not using set.

[11]:

```python
# s1 = input("Enter a String ")
# s2 = input("Enter a String ")

# if set(s1) == set(s2) and len(s1) == len(s2):
#     print("Anagram")
# else:
#     print("not anagram")

str1 = input("Enter a String ")
str2 = input("Enter a String ")

if len(str1) != len(str2):
    print("Not Anagram")
else:
    dict1 = {}
    for char in str1:
        dict1[char] = dict1.get(char, 0) + 1

    dict2 = {}
    for char in str2:
        dict2[char] = dict2.get(char, 0) + 1

    if dict1 == dict2:
        print("Anagram")
    else:
        print("Not Anagram")
```

```
Enter a String  neel
Enter a String  leen

Anagram
```

1.0.10 10) WAP to find common elements in three lists using set.

[12]:
```
s1 = {1,2,3,4,5}
s2 = {1,2}
s3 = {4,1}

print(s1.intersection(s2,s3))
```

```
{1}
```

1.0.11 11) WAP to count number of vowels in given string using set.

[13]:
```
str1 = input("Enter a String ")
count = 0
for i in str1:
    if i in set('aeiouAEIOU'):
        count+=1
print('Number of vowel is' , count)
```

```
Enter a String  neel

Number of vowel is 2
```

1.0.12 12) WAP to check if a given string is binary string or not.

[14]:
```
str1 = input("Enter a String ")
length = len(str1)
count = 0

for i in str1:
    if i in set('01'):
        count+=1
if count == length:
    print("yes",str1, "is binary string")
else:
    print("No",str1, "is not binary string")
```

```
Enter a String  0110

yes 0110 is binary string
```

1.0.13 13) WAP to sort dictionary by key or value.

[15]:
```
d1 = {5 : 'neel' , 3 : 'shubham'}
sort = {i : d1[i] for i in sorted(d1)}
print(sort)
```

```
{3: 'shubham', 5: 'neel'}
```

1.0.14 14) WAP to find the sum of all items (values) in a dictionary given by user. (Assume: values are numeric)

[16]:
```
d1 = dict()
sum = 0
for i in range(5):
    temp = int(input("Enter number "))
    d1[i] = temp
print(d1)

l1 = d1.values()
for i in l1:
    sum += i
print(sum)
```

```
Enter number  2
Enter number  3
Enter number  4
Enter number  5
Enter number  6
```

```
{0: 2, 1: 3, 2: 4, 3: 5, 4: 6}
20
```

1.0.15 15) WAP to handle missing keys in dictionaries.

Example : Given, dict1 = {'a': 5, 'c': 8, 'e': 2}

if you look for key = 'd', the message given should be 'Key Not Found', otherwise print the value of 'd' in dict1.

[18]:
```
dict1 = {'a': 5, 'c': 8, 'e': 2}
key = input("Enter key ")

if key in dict1:
    print(dict1[key])
else:
    print("Key Not Found")
```

```
Enter key  a
```

```
5
```

python-programming-lab-8

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 13-01-2025

Lab - 8

1 User Defined Function

1.0.1 01) Write a function to calculate BMI given mass and height. (BMI = mass/h**2)

[2]:
```
mass = int(input("Enter Mass: "))
h = int(input("Enter height: "))

def BMI(mass , h):
    return mass/h**2
BMI(mass  ,  h )
```

```
Enter Mass:  50
Enter height:  123
```

[2]:
```
0.0033049110978914670
```

1.0.2 02) Write a function that add first n numbers.

[11]:
```
def add(n):
    ans = 0
    for i in range(0,n+1):
        ans = ans + i
    return ans

n = int(input("Enter n"))
add(n)
```

```
Enter n 2
```

[11]:
```
3
```

1.0.3 03) Write a function that returns 1 if the given number is Prime or 0 otherwise.

[18]:
```python
def is_prime(n):
    for i in range(2,n):
        if n % i == 0:
            return 0
    return 1
n = int(input("Enter number: "))
is_prime(n)
```

```
Enter number:  5
```

[18]:
```
1
```

1.0.4 04) Write a function that returns the list of Prime numbers between given two numbers.

[20]:
```python
def prime_in_range(start, end):
    l1 = []
    for n in range(start, end + 1):
        if n > 1:
            for i in range(2, n):
                if n % i == 0:
                    break
            else:
                l1.append(n)
    return l1

start = int(input("Enter the starting number: "))
end = int(input("Enter the ending number: "))
print(prime_in_range(start, end))
```

```
Enter the starting number:  1
Enter the ending number:  4
```

```
[2, 3]
```

1.0.5 05) Write a function that returns True if the given string is Palindrome or False otherwise.

[22]:
```python
def is_palindrome(s):
    return s == s[::-1]

s = input("Enter a string: ")
```

```
print(is_palindrome(s))
```

```
Enter a string:  121
```

```
True
```

1.0.6 06) Write a function that returns the sum of all the elements of the list.

[23]:
```
def sum_of_list(lst):
    return sum(lst)

lst = list(map(int, input("Enter numbers separated by space: ").split()))
print(sum_of_list(lst))
```

```
Enter numbers separated by space:  1 2 3 4 5
```

```
15
```

1.0.7 07) Write a function to calculate the sum of the first element of each tuples inside the list.

[ ]:
```
def sum_first_elements(lst):
    return sum(t[0] for t in lst)

lst = [(1, 2), (3, 4), (5, 6)]
print(sum_first_elements(lst))
```

1.0.8 08) Write a recursive function to find nth term of Fibonacci Series.

[25]:
```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

n = int(input("Enter the term number: "))
print(fibonacci(n))
```

```
Enter the term number:  4
```

```
3
```

1.0.9 09) Write a function to get the name of the student based on the given rollno.

Example: Given dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'} find name of student whose rollno = 103

[26]:
```
def get_student_name(rollno):
    students = {
        101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'
    }
    return students.get(rollno, "Student not found")

rollno = int(input("Enter roll number: "))
print(get_student_name(rollno))
```

```
Enter roll number:  101
```

```
Ajay
```

1.0.10 10) Write a function to get the sum of the scores ending with zero.

Example : scores = [200, 456, 300, 100, 234, 678]

Ans = 200 + 300 + 100 = 600

[27]:
```
def sum_scores_ending_with_zero(scores):
    return sum(score for score in scores if score % 10 == 0)

scores = [200, 456, 300, 100, 234, 678]
print(sum_scores_ending_with_zero(scores))
```

```
600
```

1.0.11 11) Write a function to invert a given Dictionary.

hint: keys to values & values to keys

Before : {'a': 10, 'b':20, 'c':30, 'd':40}

After : {10:'a', 20:'b', 30:'c', 40:'d'}

[38]:
```
def invert_dict(d):
    return {v: k for k, v in d.items()}

d = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
print(invert_dict(d))
```

```
{10: 'a', 20: 'b', 30: 'c', 40: 'd'}
```

1.0.12 12) Write a function to check whether the given string is Pangram or not.

hint: Pangram is a string containing all the characters a-z atlest once.

"the quick brown fox jumps over the lazy dog" is a Pangram string.

[6]:
```
def is_pangram(s):
    s = s.lower()
    alphabet_set = set('abcdefghijklmnopqrstuvwxyz')
    string_set = set(char for char in s if char.isalpha())
    return string_set == alphabet_set
```

```
s = input("Enter a string: ")
is_pangram(s)
```

```
Enter a string:  the quick brown fox jumps over the lazy dog
```

```
[6]: True
```

1.0.13 13) Write a function that returns the number of uppercase and lowercase letters in the given string.

example : Input : s1 = AbcDEfgh ,Ouptput : no_upper = 3, no_lower = 5

```
[9]: def count_case(s):
         upper = sum(1 for char in s if char.isupper())
         lower = sum(1 for char in s if char.islower())
         return upper, lower

     s = "AbcDEfgh"
     upper, lower = count_case(s)
     print(f"no_upper = {no_upper}, no_lower = {no_lower}")
```

```
no_upper = 3, no_lower = 5
```

1.0.14 14) Write a lambda function to get smallest number from the given two numbers.

```
[37]: smallest = lambda a, b: a if a < b else b

      n1 = int(input("Enter first number: "))
      n2 = int(input("Enter second number: "))
      print(smallest(n1, n2))
```

```
Enter first number:  1
Enter second number:  5

1
```

1.0.15 15) For the given list of names of students, extract the names having more that 7 characters. Use filter().

```
[1]: name = ["abc" , "abcdefghi" , "abcfdretjdbsnkwlsngdl" , "nenk"]
     ans = list(filter(lambda i:len(i)>7 , name))
     print(ans)
```

```
['abcdefghi', 'abcfdretjdbsnkwlsngdl']
```

1.0.16 16) For the given list of names of students, convert the first letter of all the names into uppercase. use map().

[2]:
```
def nameUpper(s):
    return s.title()
name = ["abc" , "abcdefghi" , "abcfdretjdbsnkwlsngdl" , "nenk"]
ans = list(map(nameUpper , name))
print(ans)
```

```
['Abc', 'Abcdefghi', 'Abcfdretjdbsnkwlsngdl', 'Nenk']
```

1.0.17 17) Write udfs to call the functions with following types of arguments:

1. Positional Arguments
2. Keyword Arguments
3. Default Arguments
4. Variable Legngth Positional(*args) & variable length Keyword Arguments (**kwargs)
5. Keyword-Only & Positional Only Arguments

[3]:
```
# Positional
def add(n1,n2):
    return n1+n2
add(5,3)

# Keyword
def add(n1,n2):
    return n1+n2
add(n2 = 5 , n1 = 10)

# Default
def add(n1,n2 = 2):
    return n1+n2
add(5)

# Variable Legngth Positional
def add(n1 , *n):
    sum = n1
    for i in n:
        sum = sum + i
    print(sum)
add(1,2,3,4,5)
```

```
15
```

[ ]:
```

```

python-programming-lab-9

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 27-01-2025

Lab - 9

1 File I/O

1.0.1 01) WAP to read and display the contents of a text file. (also try to open the file in some other directory)

- in the form of a string

- line by line

- in the form of a list

[1]:
```python
file_path = 'file1.txt'

with open(file_path, 'r') as file:
    content = file.read()
    print(content)

with open(file_path, 'r') as file:
    for line in file:
        print(line, end='')

with open(file_path, 'r') as file:
    lines = file.readlines()
    print(lines)
```

```
hello from the file
hello from the file['hello from the file']
```

1.0.2 02) WAP to create file named “new.txt” only if it doesn’t exist.

[2]:
```
fp = open("new.txt","w")
s1 = "Hello form file write method"
fp.write(s1)
fp.close()
```

1.0.3 03) WAP to read first 5 lines from the text file.

[3]:
```
with open('file1.txt', 'r') as file:
    for i in range(5):
        line = file.readline()
        print(line, end='')
```

```
hello from 1st line
hello from 2nd line
hello from 3rd line
hello from 4th line
hello from 5th line
```

1.0.4 04) WAP to find the longest word(s) in a file

[4]:
```
with open('file1.txt', 'r') as file:
    words = file.read().split()
    max_length = max(len(word) for word in words)
    longest_words = [word for word in words if len(word) == max_length]

print(longest_words)
```

```
['bsjbebrjbgjebtrjbwrjbjbtbj']
```

1.0.5 05) WAP to count the no. of lines, words and characters in a given text file.

[5]:
```
lines_count = 0
words_count = 0
chars_count = 0

with open('file1.txt', 'r') as file:
    for line in file:
        lines_count += 1
        words_count += len(line.split())
        chars_count += len(line)

print(f"Lines: {lines_count}")
print(f"Words: {words_count}")
print(f"Characters: {chars_count}")
```

```
Lines: 7
Words: 28
Characters: 160
```

1.0.6 06) WAP to copy the content of a file to the another file.

[6]:
```python
with open('file1.txt', 'r') as source_file:
    content = source_file.read()

with open('newCopyFile.txt', 'w') as destination_file:
    destination_file.write(content)
```

1.0.7 07) WAP to find the size of the text file.

[10]:
```python
with open('file1.txt', 'r') as file:
    data = file.read()
    size = len(data)

print(size)
```

```
160
```

1.0.8 08) WAP to create an UDF named frequency to count occurances of the specific word in a given text file.

[14]:
```python
def frequency(file_path, word_to_find):
    count = 0
    with open(file_path, 'r') as file:
        for line in file:
            count += line.lower().split().count(word_to_find.lower())
    return count

file_path = 'file1.txt'
word_to_find = 'hello'

word_count = frequency(file_path, word_to_find)
print(f"The word '{word_to_find}' appears {word_count} times in the file.")
```

```
The word 'hello' appears 5 times in the file.
```

1.0.9 09) WAP to get the score of five subjects from the user, store them in a file. Fetch those marks and find the highest score.

[15]:
```python
file_path = 'marks.txt'

scores = []
for i in range(1, 6):
    score = int(input(f"Enter marks subject {i}: "))
    scores.append(score)

with open(file_path, 'w') as file:
    for score in scores:
        file.write(f"{score}\n")

with open(file_path, 'r') as file:
    scores_from_file = [int(line.strip()) for line in file]

highest_score = max(scores_from_file)
print(f"The highest score is: {highest_score}")
```

```
Enter marks subject 1:  12
Enter marks subject 2:  6526
Enter marks subject 3:  261
Enter marks subject 4:  6261
Enter marks subject 5:  66161
```

```
The highest score is: 66161
```

1.0.10 10) WAP to write first 100 prime numbers to a file named primenumbers.txt

(Note: each number should be in new line)

[18]:
```
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True

file_path = 'primenumbers.txt'
prime_numbers = []
num = 2

while len(prime_numbers) < 100:
    if is_prime(num):
        prime_numbers.append(num)
    num += 1

with open(file_path, 'w') as file:
    for prime in prime_numbers:
        file.write(f"{prime}\n")
```

1.0.11 11) WAP to merge two files and write it in a new file.

[19]:
```python
file1_path = 'new.txt'
file2_path = 'newCopyFile.txt'
merged_file_path = 'merged_file.txt'

with open(file1_path, 'r') as file1, open(file2_path, 'r') as file2:
    content1 = file1.read()
    content2 = file2.read()

with open(merged_file_path, 'w') as merged_file:
    merged_file.write(content1)
    merged_file.write(content2)
```

1.0.12 12) WAP to replace word1 by word2 of a text file. Write the updated data to new file.

[ ]:
```python
file_path = 'new.txt'
new_file_path = 'updated_file.txt'
word1 = ''
word2 = 'newword'

with open(file_path, 'r') as file:
    content = file.read()

updated_content = content.replace(word1, word2)

with open(new_file_path, 'w') as new_file:
    new_file.write(updated_content)
```

1.0.13 13) Demonstrate tell() and seek() for all the cases(seek from beginning-end-current position) taking a suitable example of your choice.

[ ]:
```python

```

python-programming-lab-10

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 03-02-2025

Lab - 10

1 Exception Handling

1.0.1 01) WAP to handle following exceptions:

1. ZeroDivisionError
2. ValueError
3. TypeError #### Note: handle them using separate except blocks and also using single except block too.

[10]:
```
try:
    a = int(input("Enter 1st number "))
    b = int(input("Enter 2nd number "))
    c = a//b
    print(c)
except (ZeroDivisionError , ValueError , TypeError ) as e:
    print(e)
```

```
Enter 1st number 10
Enter 2nd number 0
integer division or modulo by zero
```

1.0.2 02) WAP to handle following exceptions:

1. IndexError
2. KeyError

[3]:
```
l1 = [1,2,3]
d1 = {1 : 'a' , 2:'b'}
try:
    print(l1[3])
    print(d1[3])
except (IndexError ,KeyError ) as e:
    print(e)
```

```
'c'
```

1.0.3 03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

[35]:
```
try:
    fp = open("temp.txt" , "r")
except FileNotFoundError as e:
    print(e)
try:
    import neel
except ModuleNotFoundError as err:
    print(err)
```

```
No module named 'neel'
```

1.0.4 04) WAP that catches all type of exceptions in a single except block.

[14]:
```
try:
    a = int(input("Enter Number "))
    b = int(input("Enter Number "))
    c = a // b
    print(c)
except Exception as e:
    print(e)
```

```
Enter Number 10
Enter Number ru
invalid literal for int() with base 10: 'ru'
```

1.0.5 05) WAP to demonstrate else and finally block.

[16]:
```
try:
    fp = open('temp.txt' , "r")
except Exception as e:
    print(e)
else:
    print(fp.read())
    fp.close()
finally:
    print("hello from the finally block")
```

```
hello
hello from the finally block
```

1.0.6 06) Create a short program that prompts the user for a list of grades separated by commas.

1.0.7 Split the string into individual grades and use a list comprehension to convert each string to an integer.

1.0.8 You should use a try statement to inform the user when the values they entered cannot be converted.

[ ]:

```

```

1.0.9 07) WAP to create an udf divide(a,b) that handles ZeroDivisionError.

[26]:

```
class divideError(Exception):
    pass
try:
    a = int(input("Enter Number "))
    b = int(input("Enter Number "))

    if(b != 5):
        print(a // b)
    else:
        raise divideError

except divideError:
    print("error")
```

```
Enter Number 10
Enter Number 5
error
```

1.0.10 08) WAP that gets an age of a person form the user and raises ValueError with error message: "Enter Valid Age" :

If the age is less than 18.

otherwise print the age.

[30]:

```
class ageError(Exception):
    pass

try:
    age = int(input("Enter age "))
    if(age > 18):
        print(age)
    else:
        raise ageError
except ageError:
    print("Enter Valid Age")
```

```
Enter age 19
19
```

1.0.11 09) WAP to raise your custom Exception named InvalidUsernameError with the error message : "Username must be between 5 and 15 characters long":

if the given name is having characters less than 5 or greater than 15.

otherwise print the given username.

[34]:
```
class InvalidUsernameError(Exception):
    pass

try:
    name = input("Enter your name ")
    if(len(name) > 5 and len(name) < 15):
        print(name)
    else:
        raise InvalidUsernameError
except InvalidUsernameError:
    print("Username must be between 5 and 15 characters long")
```

```
Enter your name neel
Username must be between 5 and 15 characters long
```

1.0.12 10) WAP to raise your custom Exception named NegativeNumberError with the error message : "Cannot calculate the square root of a negative number" :

if the given number is negative.

otherwise print the square root of the given number.

[40]:
```
import math
class NegativeNumberError(Exception):
    def __init__(self, msg):
        self.msg = msg
try:
    a = int(input("Enter number "))
    if(a > 0):
        ans = (math.sqrt(a))
        print(ans)
    else:
        raise NegativeNumberError("Cannot calculate the square root of a␣
↪negative number")
```

```python
except NegativeNumberError as e:
    print(e)
```

```
Enter number 56
7.483314773547883
```

```
[ ]:
```

python-programming-lab-11

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 10-02-2025

Lab - 11

1 Modules

1.0.1 01) WAP to create Calculator module which defines functions like add, sub,mul and div.

1.0.2 Create another .py file that uses the functions available in Calculator module.

[1]:
```
import calculator as cal

a = int(input("Enter 1st Number "))
b = int(input("Enter 2nd Number "))

print(f"Sum of two number is",cal.add(a,b))
print(f"Substraction of two number is",cal.sub(a,b))
print(f"Multiplication of two number is",cal.multiply(a,b))
print(f"Division of two number is",cal.divide(a,b))
```

```
Enter 1st Number  11
Enter 2nd Number  22
```

```
Sum of two number is 33
Substraction of two number is -11
Multiplication of two number is 242
Division of two number is 0
```

1.0.3 02) WAP to pick a random character from a given String.

[7]:
```
import random as rd

temp = input("Enter a String ")

print(rd.choice(temp))
```

```
Enter a String neel gohel
e
```

1.0.4 03) WAP to pick a random element from a given list.

[2]:
```
import random as rd

l1 = ["Neel", "Raj", "Shubham"]

print(rd.choice(l1))
```

```
Neel
```

1.0.5 04) WAP to roll a dice in such a way that every time you get the same number.

[3]:
```
print(rd.randrange(1,2))
```

```
1
```

1.0.6 05) WAP to generate 3 random integers between 100 and 999 which is divisible by 5.

[4]:
```
import random

l1 = []

while len(l1)<3:
    a = random.randint(100,999)
    if(a%5==0):
        l1.append(a)
print(l1)
```

```
[210, 605, 140]
```

1.0.7 06) WAP to generate 100 random lottery tickets and pick two lucky tickets from it and announce them as Winner and Runner up respectively.

[5]:
```
import random as rd

l1 = []

for i in range(0,100):
    l1.append(rd.randrange(100,999))


print("Winner is ",rd.choice(l1))
print("Runner up is ",rd.choice(l1))
```

```
Winner is  194
Runner up is  580
```

1.0.8 07) WAP to print current date and time in Python.

[6]:
```
from datetime import datetime
print(datetime.now())
```

```
2025-02-10 15:56:58.660911
```

1.0.9 08) Subtract a week (7 days) from a given date in Python.

[7]:
```
import datetime

d1 = datetime.datetime.now()

print("Today_Date", d1)

d2 = d1 - datetime.timedelta(days=7)

print('before 7days:', d2)
```

```
Today_Date 2025-02-10 15:57:00.504418
before 7days: 2025-02-03 15:57:00.504418
```

1.0.10 09) WAP to Calculate number of days between two given dates.

[8]:
```
import random

d1 = datetime.datetime.now()

print("Today_Date", d1)

d2 = d1 + datetime.timedelta(days= random.randrange(1,7))
```

```
print("After days:", d2)

print("Day Diffrence" , d2-d1)
```

```
Today_Date 2025-02-10 15:57:02.459487
After days: 2025-02-11 15:57:02.459487
Day Diffrence 1 day, 0:00:00
```

1.0.11 10) WAP to Find the day of the week of a given date.(i.e. wether it is sunday/monday/tuesday/etc.)

[9]:
```
import datetime

d1 = datetime.datetime.now()

# print("Today_Date", d1)

d2 = d1 + datetime.timedelta(days= random.randrange(1,7))

# print("After days:", d2)

print(d2.strftime("%A"))
```

```
Thursday
```

1.0.12 11) WAP to demonstrate the use of date time module.

[ ]:
```
from datetime import datetime
print(datetime.datetime.now())
```

1.0.13 12) WAP to demonstrate the use of the math module.

[4]:
```
import math

print(math.factorial(5))
print(math.gcd(10,5))
print(math.ceil(5.25))
print(math.lcm(23,5))
print(math.pow(23,5))
print(math.sqrt(81))
```

```
120
5
6
115
```

```
6436343.0
9.0
```

[ ]:

```

```

python-programming-lab-12

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

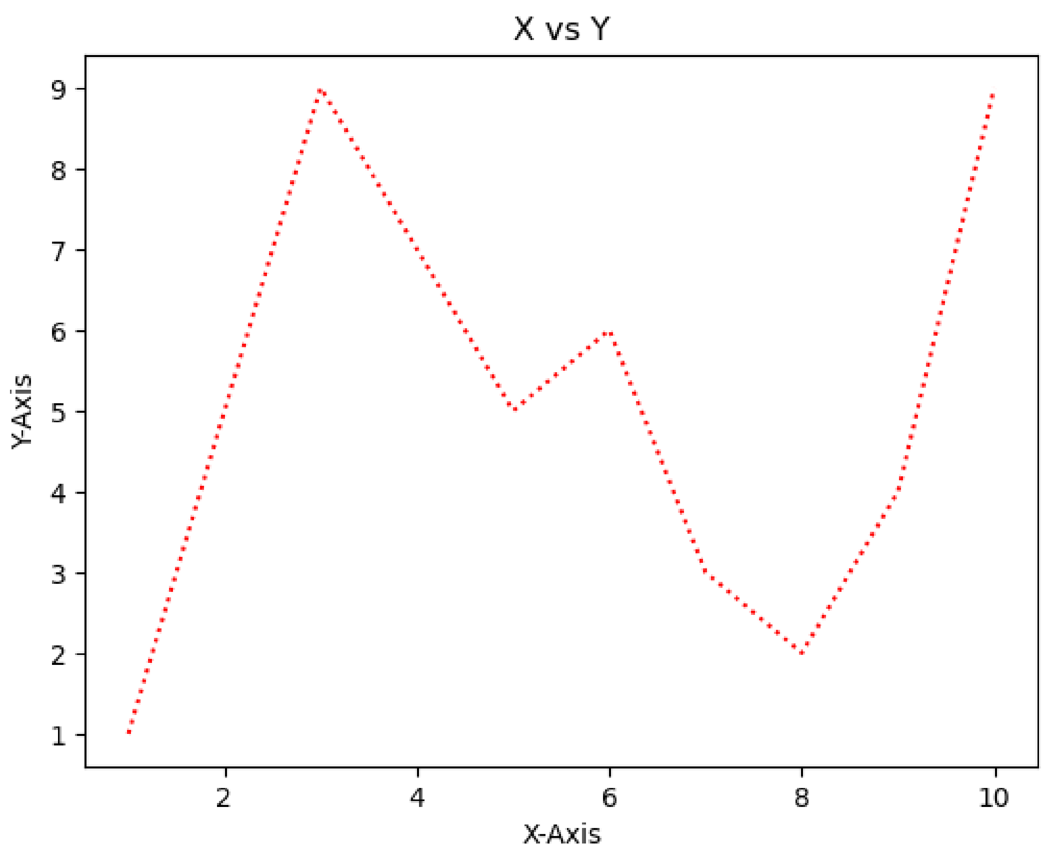
Roll No. 340

Date: 24-02-2025

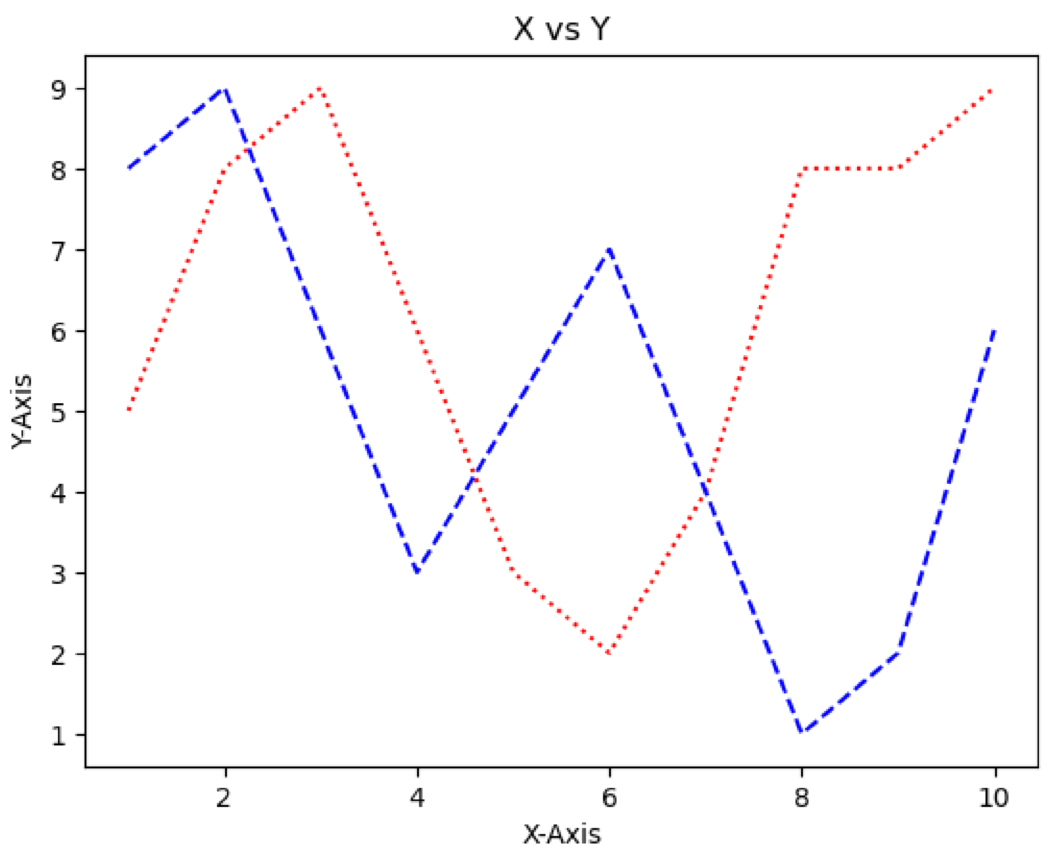
Lab - 12

[2]:
```
#import matplotlib below
import matplotlib.pyplot as plt
```

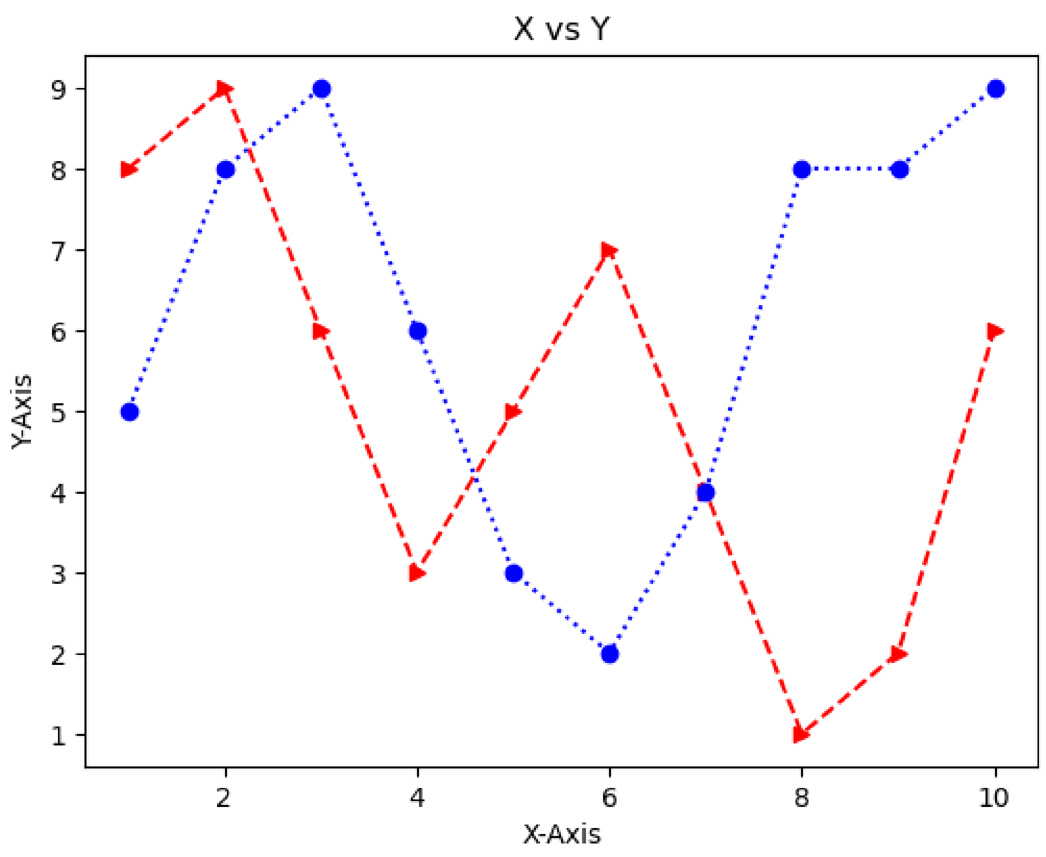
[13]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]

plt.plot(x,y , ls = ":" , c = "r")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.title("X vs Y")
plt.show()
# write a code to display the line chart of above x & y
```



[14]:
```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarks = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]

plt.plot(x,cxMarks , ls = ":" , c = "r")
plt.plot(x,cyMarks , ls = "--" , c = "b")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.title("X vs Y")
plt.show()
# write a code to display two lines in a line chart (data given above)
```



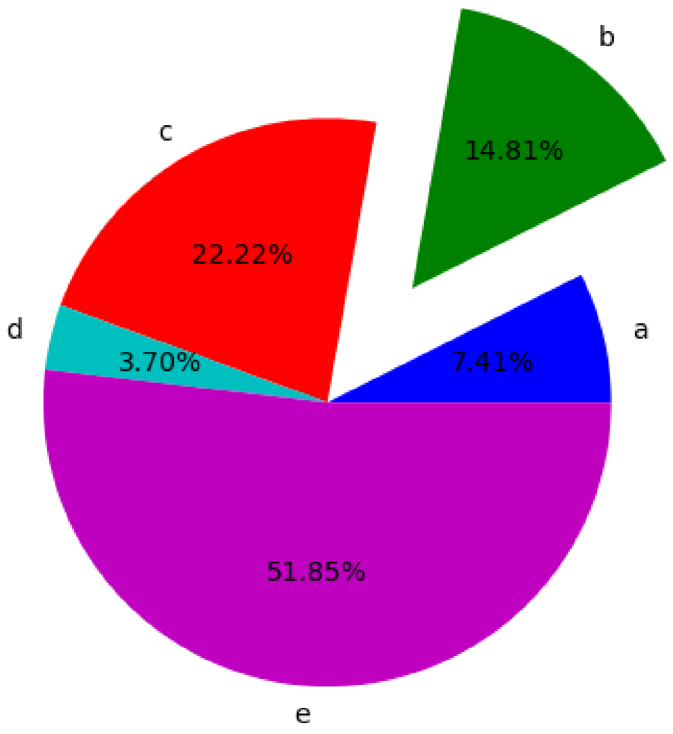
[19]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]

plt.plot(x,cxMarks , ls = "--" , c = "r" , marker=">")
plt.plot(x,cyMarks , ls = ":" , c = "b" , marker="o")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.title("X vs Y")
plt.show()
# write a code to generate below graph
```



```
[ ]:
```

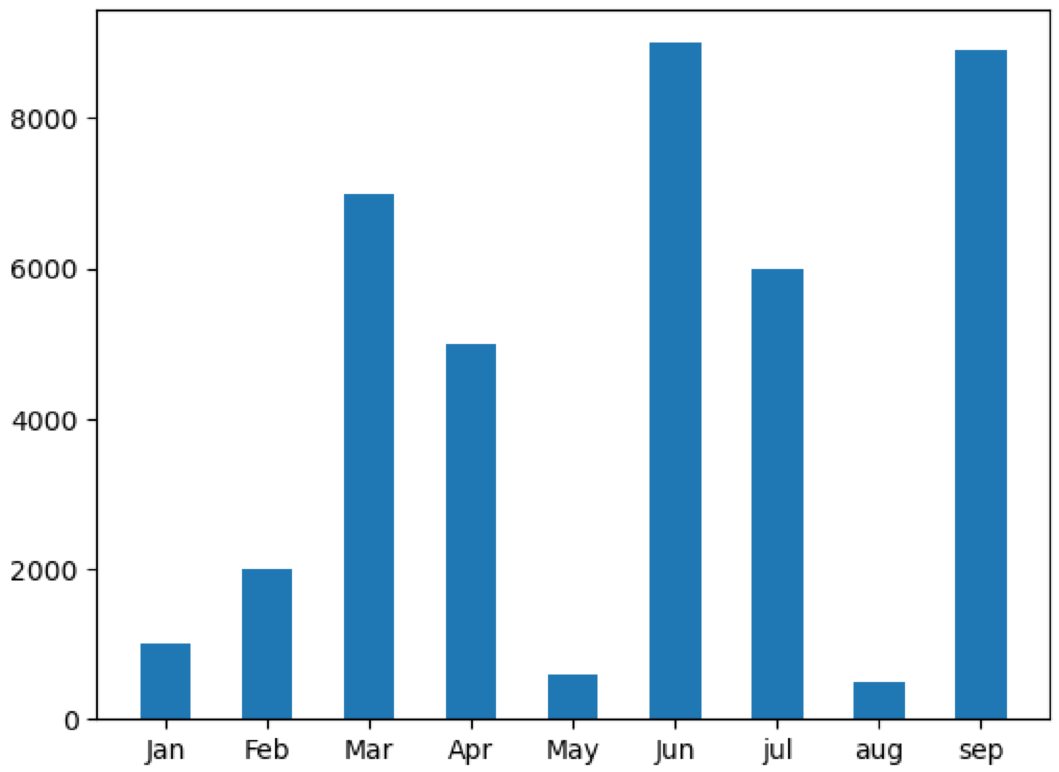
0.0.1 04) WAP to demonstrate the use of Pie chart.

```
[38]: dept = ["a", "b", "c", "d", "e"]
      stud = [100, 200, 300, 50, 700]
      x = ["b","g","r","c","m"]
      e = [0,0.5,0,0,0]
      plt.pie(stud, labels = dept, colors=x, autopct="%1.2f%%" , explode = e)
      plt.show()
```



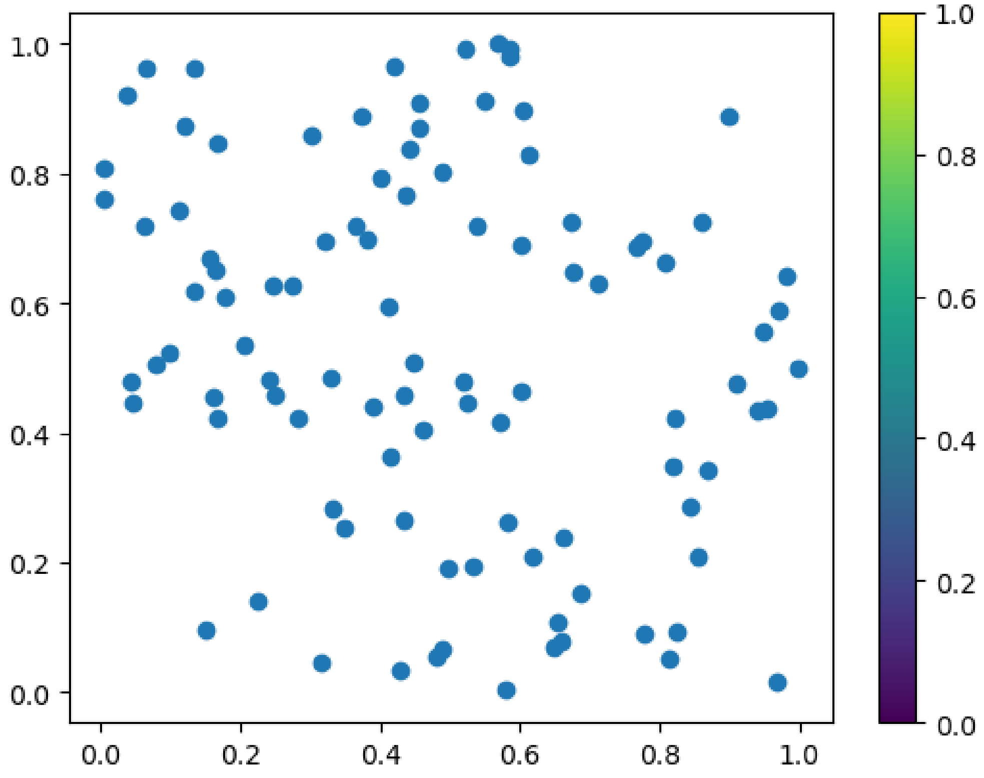
0.0.2 05) WAP to demonstrate the use of Bar chart.

```
[42]: a = ["Jan", "Feb", "Mar", "Apr", "May", "Jun" , "jul" , "aug" , "sep"]
      b = [1000, 2000, 7000, 5000, 600, 9000 , 6000 , 500 , 8900]
      plt.bar(a, b, width=0.5)
      plt.show()
```



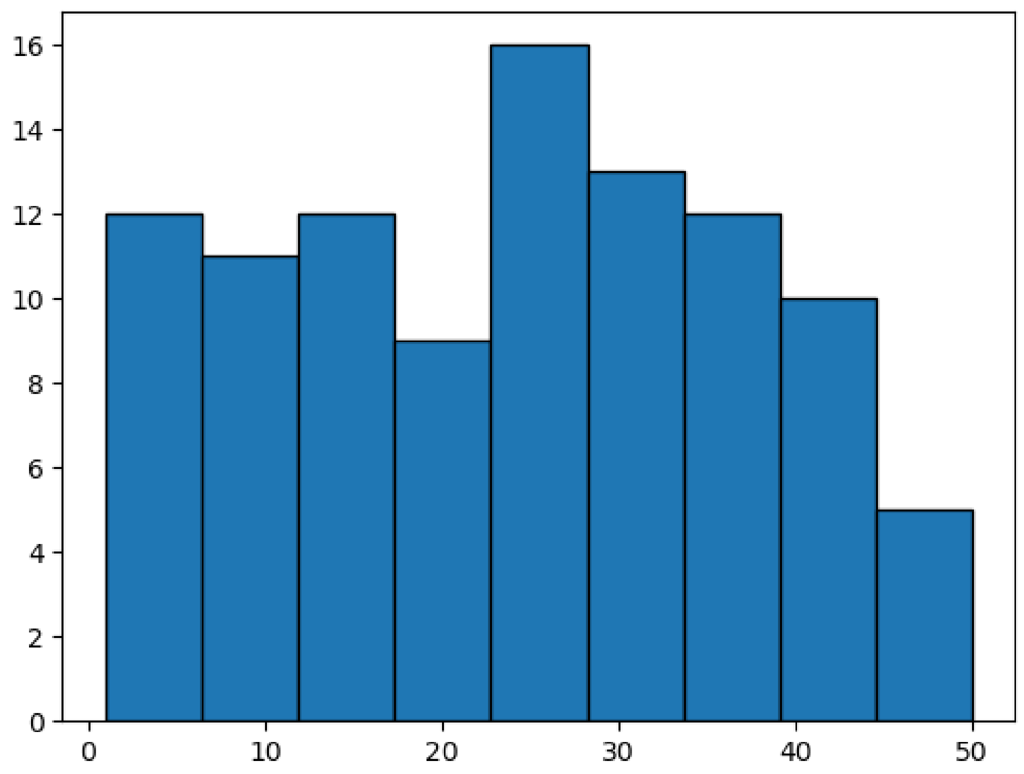
0.0.3 06) WAP to demonstrate the use of Scatter Plot.

[26]:
```
import random

random.seed(10)
x = [random.random() for i in range(100)]
y = [random.random() for i in range(100)]
plt.scatter(x,y)
plt.colorbar()
plt.show()
```



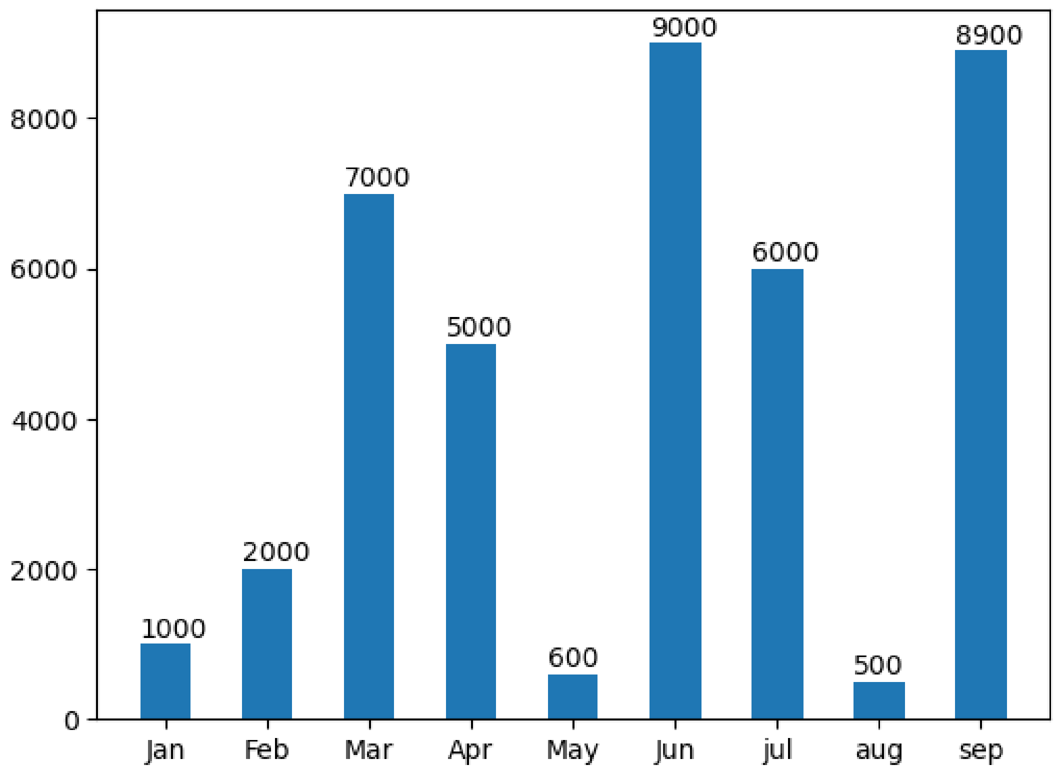
0.0.4 07) WAP to demonstrate the use of Histogram.

```
[48]: random.seed(10)
age = [random.randint(1,50) for i in range(100)]
plt.hist(age, edgecolor="k", bins=9,histtype="bar")
plt.show()
```



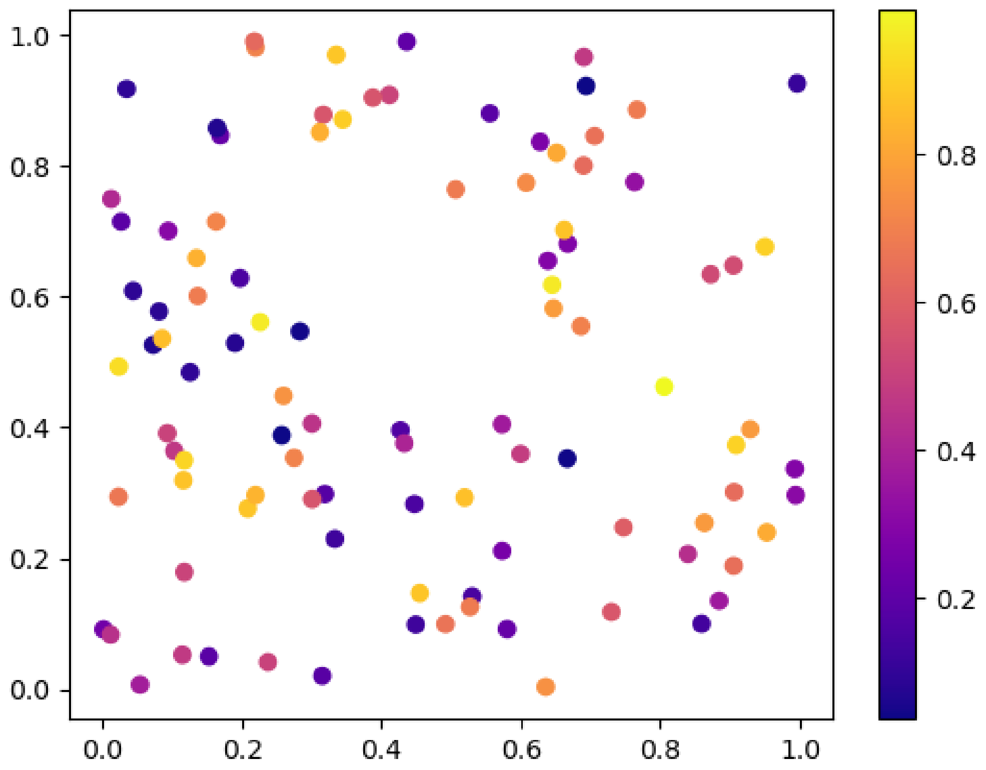
0.0.5 08) WAP to display the value of each bar in a bar chart using Matplotlib.

[43]:
```python
a = ["Jan", "Feb", "Mar", "Apr", "May", "Jun" , "jul" , "aug" , "sep"]
b = [1000, 2000, 7000, 5000, 600, 9000 , 6000 , 500 , 8900]

bars = plt.bar(a, b, width=0.5)
for i in bars:
    yc = i.get_height()
    plt.text(i.get_x(), yc+100, f"{yc}")
plt.show()
```



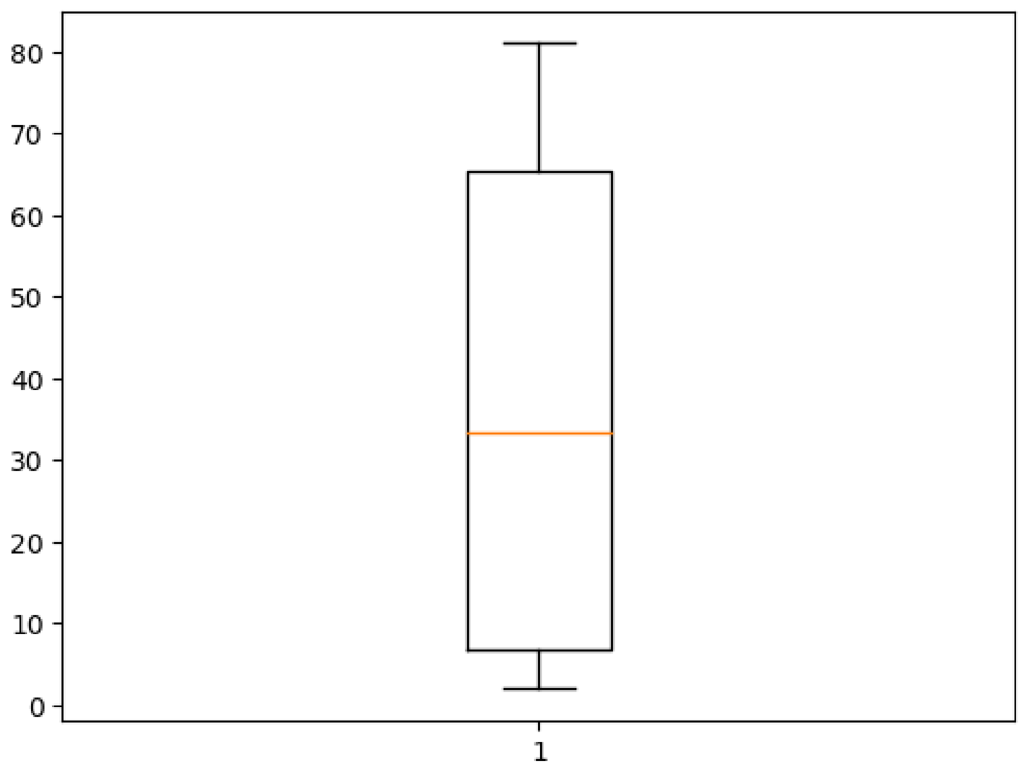
0.0.6 09) WAP create a Scatter Plot with several colors in Matplotlib?

[44]:
```
random.seed(20)
x = [random.random() for i in range(100)]
y = [random.random() for i in range(100)]
z = [random.random() for i in range(100)]
plt.scatter(x,y, c=z,cmap="plasma")
plt.colorbar()
plt.show()
```



0.0.7 10) WAP to create a Box Plot.

[50]:
```
plt.boxplot([81,72,63,24,43,8,2,3])
plt.show()
```


80
70
60
50
40
30
20
10
0
1

```
[ ]:
```

python-programming-lab-13-1

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 03-03-2025

Lab - 13_1

1 OOP

1.0.1 01) Write a Program to create a class by name Students, and initialize attributes like name, age, and grade while creating an object.

[5]:
```
class Student:
    school = "DIET"
    def __init__(self , name , age , grade):
        self.name = name
        self.age = age
        self.grade = grade
    def printStudent(self):
        print(self.name)
        print(self.age)
        print(self.grade)
stu1 = Student("Neel" , "20" , "A")
stu1.printStudent()
print(Student.school)
```

```
Neel
20
A
DIET
```

1.0.2 02) Create a class named Bank_Account with Account_No, User_Name, Email,Account_Type and Account_Balance data members. Also create a method GetAccountDetails() and DisplayAccountDetails(). Create main method to demonstrate the Bank_Account class.

[18]:
```python
class Bank_Account:

    def getAccountDetail(self):
        self.account_No = input("Enter account_No ")
        self.user_name = input("Enter user_name ")
        self.email = input("Enter email ")
        self.account_type = input("Enter account_type ")
        self.account_balance = input("Enter account_balance ")

    def displayAccountDetail(self):
        print("Your account_No is" , self.account_No)
        print("Your user_name ",self.user_name)
        print("Your email ",self.email)
        print("Your account_type ",self.account_type)
        print("Your account_balance ",self.account_balance)
B1 = Bank_Account()
B1.getAccountDetail()
B1.displayAccountDetail()
```

```
Enter account_No abc
Enter user_name abc
Enter email abc
Enter account_type abc
Enter account_balance abc
Your account_No is abc
Your user_name  abc
Your email  abc
Your account_type  abc
Your account_balance  abc
```

1.0.3 03) WAP to create Circle class with area and perimeter function to find area and perimeter of circle.

[26]:
```python
import math

class Circle:

    def __init__(self):
        self.r = int(input("Enter Radius "))
    def area(self):
        area = math.pi * self.r * self.r
        print("Area is " ,area)
    def perimeter(self):
        peri = 2 * math.pi * self.r
        print("Perimeter is " ,peri)
c1 = Circle()
c1.area()
c1.perimeter()
```

```
Enter Radius 5
Area is  78.53981633974483
Perimeter is  31.41592653589793
```

1.0.4 04) Create a class for employees that includes attributes such as name, age, salary, and methods to update and display employee information.

[27]:
```
class Employee:

    def __init__(self):
        self.name = input("Enter name ")
        self.age = int(input("Enter Age "))
        self.salary = int(input("Enter Salary "))
    def updateEmployee(self):
        n1 = input("Enter Updated Name ")
        a1 = int(input("Enter updated age "))
        s1 = int(input("Enter updated Salary "))
        self.name = n1
        self.age = a1
        self.salary = s1
    def display(self):
        print("Employee name is ",self.name)
        print("Employee age is " ,self.age)
        print("Employee salary is ",self.salary)
e1 = Employee()
e1.display()
e1.updateEmployee()
e1.display()
```

```
Enter name neel
Enter Age 20
Enter Salary 12345
Employee name is  neel
Employee age is  20
Employee salary is  12345
Enter Updated Name neeeeel
Enter updated age 21
Enter updated Salary 9876
Employee name is  neeeeel
Employee age is  21
Employee salary is  9876
```

1.0.5 05) Create a bank account class with methods to deposit, withdraw, and check balance.

[29]:
```
class bank_account():

    def __init__(self):
        self.user_name = input("Enter user_name ")
        self.account_balance = int(input("Enter account_balance "))

    def deposit(self):
        money = int(input("Enter money to deposit "))
        self.account_balance+=money
        print("Your account has ",self.account_balance)

    def withdraw(self):
        money = int(input("Enter money to withdraw "))
        if(money >self.account_balance ):
            print("Invalid Money")
        else:
            self.account_balance-=money
        print("Your account has ",self.account_balance)


    def checkBalance(self):
        print("Your Money in bank is " ,self.account_balance)

user1 = bank_account()
user1.deposit()
user1.withdraw()
user1.checkBalance()
```

```
Enter user_name neel
Enter account_balance 500
Enter money to deposit 100
Your account has  600
Enter money to withdraw 200
Your account has  400
Your Money in bank is  400
```

1.0.6 06) Create a class for managing inventory that includes attributes such as item name, price, quantity, and methods to add, remove, and update items.

[ ]:
```
class managing_inventory():

    def __init__(self):
        self.name = input("Enter name ")
        self.price = int(input("Enter price "))
        self.quantity = int(input("Enter quantity "))
    def add(self):
        add = int(input("Enter no of add items "))
        self.quantity += add
        print("Updated " ,self.quantity )
```

1.0.7 07) Create a Class with instance attributes of your choice.

[3]:
```
class Car:
    def __init__(self, make, model, year, color):
        self.make = make
        self.model = model
        self.year = year
        self.color = color

    def display_car_info(self):
        print(f"Car Information:")
        print(f"Make: {self.make}")
        print(f"Model: {self.model}")
        print(f"Year: {self.year}")
        print(f"Color: {self.color}")

car1 = Car("Toyota", "Century", 2021, "Blue")
car1.display_car_info()
car2 = Car("Honda", "Civic", 2020, "Red")
car2.display_car_info()
```

```
Car Information:
Make: Toyota
Model: Century
Year: 2021
Color: Blue
Car Information:
Make: Honda
Model: Civic
Year: 2020
Color: Red
```

1.0.8 08) Create one class student_kit

Within the student_kit class create one class attribute principal name (Mr ABC)

Create one attendance method and take input as number of days.

While creating student take input their name .

Create one certificate for each student by taking input of number of days present in class.

[1]:
```python
class StudentKit:
    principal = "Mr ABC"

    def __init__(self, student_name):
        self.student_name = student_name
        self.attendance = 0

    def record_attendance(self, days_present):
        self.attendance = days_present
        print(f"Attendance recorded: {self.attendance} days")

    def generate_certificate(self):
        if self.attendance >= 75:
            print(f"Certificate of Attendance\n")
            print(f"Principal: {StudentKit.principal}")
            print(f"Student: {self.student_name}")
            print(f"Days Present: {self.attendance}")
            print(f"Status: Passed (Attendance is sufficient)")
        else:
            print(f"Certificate of Attendance\n")
            print(f"Principal: {StudentKit.principal}")
            print(f"Student: {self.student_name}")
            print(f"Days Present: {self.attendance}")
            print(f"Status: Failed (Attendance is insufficient)")

student_name = input("Enter the student's name: ")
student = StudentKit(student_name)
days_present = int(input("Enter the number of days present in class: "))
student.record_attendance(days_present)
student.generate_certificate()
```

```
Enter the student's name:  neel
Enter the number of days present in class:  28

Attendance recorded: 28 days
Certificate of Attendance


Principal: Mr ABC
Student: neel
Days Present: 28
Status: Failed (Attendance is insufficient)
```

1.0.9 09) Define Time class with hour and minute as data member. Also define addition method to add two time objects.

[2]:
```python
class Time:
    def __init__(self, hour=0, minute=0):
        self.hour = hour
        self.minute = minute

    def add_time(self, other):
        total_minutes = (self.hour * 60 + self.minute) + (other.hour * 60 +␣
↪other.minute)
        total_hour = total_minutes // 60
        total_minute = total_minutes % 60
        return Time(total_hour, total_minute)

    def display_time(self):
        print(f"{self.hour} hour(s) and {self.minute} minute(s)")

time1 = Time(2, 45)
time2 = Time(3, 30)
total_time = time1.add_time(time2)
print("Time 1:")
time1.display_time()
print("Time 2:")
time2.display_time()
print("Total Time after adding:")
total_time.display_time()
```

```
Time 1:
2 hour(s) and 45 minute(s)
Time 2:
3 hour(s) and 30 minute(s)
Total Time after adding:
6 hour(s) and 15 minute(s)
```

[ ]:

python-programming-lab-13-2

March 13, 2025

Python Programming - 2301CS404

Gohel Neel

Enrollnment No. : 23010101089

Roll No. 340

Date: 10-03-2025

Lab - 13_2

0.1 Continued..

0.1.1 10) Calculate area of a ractangle using object as an argument to a method.

[3]:
```
class Rectangle:
    def __init__(self):
        self.length = int(input("Enter Length "))
        self.width = int(input("Enter Width "))

def calculate_area(rectangle):
    return rectangle.length * rectangle.width

R1 = Rectangle()
area = calculate_area(R1)

print("The area is: ",area)
```

```
Enter Length  10
Enter Width  10

The area is:  100
```

0.1.2 11) Calculate the area of a square.

0.1.3 Include a Constructor, a method to calculate area named area() and a method named output() that prints the output and is invoked by area().

[4]:
```
class Square:
    def __init__(self):
        self.l = int(input("Enter of square: "))

    def area(self):
        area = self.l * self.l
        self.output(area)

    def output(self, area):
        print("area of square is: ",area)

square = Square()
square.area()
```

```
Enter of square:  10
```

```
area of square is:  100
```

0.1.4 12) Calculate the area of a rectangle.

0.1.5 Include a Constructor, a method to calculate area named area() and a method named output() that prints the output and is invoked by area().

0.1.6 Also define a class method that compares the two sides of reactangle. An object is instantiated only if the two sides are different; otherwise a message should be displayed : THIS IS SQUARE.

[17]:
```
class Rectangle:
    def __init__(self):
        self.length = int(input("Enter the length: "))
        self.width = int(input("Enter the width: "))
        if self.is_square():
            print("THIS IS A SQUARE")
        else:
            self.area()

    def area(self):
        area = self.length * self.width
        self.output(area)

    def output(self, area):
        print(f"The area of the rectangle is: {area}")

    def is_square(self):
        return self.length == self.width

R1 = Rectangle()
```

```
Enter the length:  4
Enter the width:  5

The area of the rectangle is: 20
```

0.1.7 13) Define a class Square having a private attribute "side".

0.1.8 Implement get_side and set_side methods to accees the private attribute from outside of the class.

[25]:
```
class Square:
    def __init__(self):
        self._side = int(input("Enter side of square: "))

    def get_side(self):
        return self._side

    def set_side(self):
        new_side = int(input("Enter new side of square: "))
        if new_side > 0:
            self._side = new_side
        else:
            print("Side must be positive.")

square = Square()

print("Side of square:", square.get_side())

square.set_side()

print("Updated side :", square.get_side())
```

```
Enter side of square:  10

Side of square: 10

Enter new side of square:  45

Updated side : 45
```

0.1.9 14) Create a class Profit that has a method named getProfit that accepts profit from the user.

0.1.10 Create a class Loss that has a method named getLoss that accepts loss from the user.

0.1.11 Create a class BalanceSheet that inherits from both classes Profit and Loss and calculates the balanace. It has two methods getBalance() and printBalance().

[26]:
```
class Profit:
    def __init__(self):
        self.profit = 0

    def getProfit(self):
        self.profit = int(input("Enter the profit: "))

class Loss:
    def __init__(self):
        self.loss = 0

    def getLoss(self):
        self.loss = int(input("Enter the loss: "))

class BalanceSheet(Profit, Loss):
    def __init__(self):
        Profit.__init__(self)
        Loss.__init__(self)
        self.balance = 0

    def getBalance(self):
        self.balance = self.profit - self.loss

    def printBalance(self):
        print("The balance is: ",self.balance)

balance_sheet = BalanceSheet()

balance_sheet.getProfit()
balance_sheet.getLoss()

balance_sheet.getBalance()
balance_sheet.printBalance()
```

```
Enter the profit:  100
Enter the loss:  20

The balance is:  80
```

0.1.12 15) WAP to demonstrate all types of inheritance.

[1]:
```python
# Single Inheritance
class Animal:
    def speak(self):
        print("Animal speaks")

class Dog(Animal):
    def bark(self):
        print("Dog barks")

# Multiple Inheritance
class Person:
    def __init__(self, name):
        self.name = name

    def introduce(self):
        print(f"Hello, my name is {self.name}")

class Employee:
    def __init__(self, employee_id):
        self.employee_id = employee_id

    def show_id(self):
        print(f"My employee ID is {self.employee_id}")

class Manager(Person, Employee):
    def __init__(self, name, employee_id):
        Person.__init__(self, name)
        Employee.__init__(self, employee_id)

    def work(self):
        print(f"{self.name} is working as a Manager.")

# Multilevel Inheritance
class Vehicle:
    def start_engine(self):
        print("Vehicle engine started")

class Car(Vehicle):
    def drive(self):
        print("Car is driving")

class SportsCar(Car):
    def speed(self):
        print("Sports car is speeding")
```

```
# Hierarchical Inheritance
class Shape:
    def area(self):
        pass

class Circle(Shape):
    def area(self, radius):
        return 3.14 * radius * radius

class Rectangle(Shape):
    def area(self, length, width):
        return length * width

# Hybrid Inheritance (Combination of Multiple and Multilevel Inheritance)
class School:
    def __init__(self, name):
        self.name = name

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

class Teacher(School):
    def __init__(self, name, subject):
        School.__init__(self, name)
        self.subject = subject

    def teach(self):
        print(f"Teaching {self.subject}")

class HeadTeacher(Teacher):
    def __init__(self, name, subject, head_teacher_name):
        Teacher.__init__(self, name, subject)
        self.head_teacher_name = head_teacher_name

    def manage(self):
        print(f"{self.head_teacher_name} manages the teaching process.")


# 1. Single Inheritance
print("Single Inheritance:")
dog = Dog()
dog.speak()
dog.bark()
print()

# 2. Multiple Inheritance
print("Multiple Inheritance:")
```

```python
manager = Manager("ABC", 101)
manager.introduce()
manager.show_id()
manager.work()
print()

# 3. Multilevel Inheritance
print("Multilevel Inheritance:")
sports_car = SportsCar()
sports_car.start_engine()
sports_car.drive()
sports_car.speed()
print()

# 4. Hierarchical Inheritance
print("Hierarchical Inheritance:")
circle = Circle()
print(f"Circle Area: {circle.area(5)}")
rectangle = Rectangle()
print(f"Rectangle Area: {rectangle.area(10, 5)}")
print()

# 5. Hybrid Inheritance
print("Hybrid Inheritance:")
head_teacher = HeadTeacher("DEF", "Maths", "ABC")
head_teacher.show_name()
head_teacher.teach()
head_teacher.manage()
```

```
Single Inheritance:
Animal speaks
Dog barks

Multiple Inheritance:
Hello, my name is ABC
My employee ID is 101
ABC is working as a Manager.

Multilevel Inheritance:
Vehicle engine started
Car is driving
Sports car is speeding

Hierarchical Inheritance:
Circle Area: 78.5
Rectangle Area: 50
```

```
Hybrid Inheritance:
School Name: DEF
Teaching Maths
ABC manages the teaching process.
```

0.1.13 16) Create a Person class with a constructor that takes two arguments name and age.

0.1.14 Create a child class Employee that inherits from Person and adds a new attribute salary.

0.1.15 Override the init method in Employee to call the parent class's init method using the super() and then initialize the salary attribute.

[2]:
```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display_info(self):
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")

class Employee(Person):
    def __init__(self, name, age, salary):
        super().__init__(name, age)
        self.salary = salary

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

emp = Employee("ABC", 30, 50000)
emp.display_employee_info()
```

```
Name: ABC
Age: 30
Salary: 50000
```

0.1.16 17) Create a Shape class with a draw method that is not implemented.

0.1.17 Create three child classes Rectangle, Circle, and Triangle that implement the draw method with their respective drawing behaviors.

0.1.18 Create a list of Shape objects that includes one instance of each child class, and then iterate through the list and call the draw method on each object.

[3]:
```python
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def draw(self):
        pass

class Rectangle(Shape):
    def draw(self):
        print("Drawing a rectangle")

class Circle(Shape):
    def draw(self):
        print("Drawing a circle")

class Triangle(Shape):
    def draw(self):
        print("Drawing a triangle")

shapes = [Rectangle(), Circle(), Triangle()]

for shape in shapes:
    shape.draw()
```

```
Drawing a rectangle
Drawing a circle
Drawing a triangle
```

[ ]:
```
```