Python Programming - Lab - 1

March 11, 2025

[ ]:

Python Programming - 2301CS404

Lab - 1

OM BHUT | 23010101033 | 122

0.0.1 01) WAP to print “Hello World”

[ ]:
```
print("Hello World")
```

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

[2]:
```
# a =int(input("Enter number 1"))
# b = int(input("Enter number 2"))
# print(a+b)

# print(4+5)
```

```
9
```

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

[7]:
```
a = input("enter a")
print(type(5))
# print('{hi}')
```

```
{hi}
```

0.0.4 04) WAP to calculate simple interest.

[9]:
```
p = int(input("enter p"))
r = int(input("enter r"))
t = int(input("enter t"))
sI = (p*r*t)/100
print(sI)
```

```
enter p10
enter r10
```

```
enter t10
10.0
```

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

[14]:
```
import math
r = int(input("enter r"))
print(f"perimeter = {2*math.pi*r} \n area = {math.pi*r*r}")
```

```
enter r10
perimeter = 62.83185307179586
 area = 314.1592653589793
```

0.0.6 06) WAP to calculate area of a triangle.

[16]:
```
b = int(input("enter b"))
h = int(input("enter h"))
print((b*h)/2)
```

```
enter l2
enter b2
enter h2
4.0
```

0.0.7 07) WAP to compute quotient and remainder.

[17]:
```
divident = int(input("enter divident"))
divisor = int(input("enter divisor"))

print(f"quotient = {int(divident/divisor)} \n remainder = {divident%divisor}")
```

```
enter divident25
enter divisor6
quotient = 4
 remainder = 1
```

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

[18]:
```
f = float(input("enter f"))
c = float(input("enter c"))
print(f"c = {(c*9/5)+32} \n f = {(f-32)*5/9}")
```

```
enter f77
enter c25
c = 77.0
 f = 25.0
```

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

[19]:
```
import math
x1=int(input("enter x1"))
x2=int(input("enter x2"))
y1=int(input("enter y1"))
y2=int(input("enter y2"))
print(math.sqrt((x2-x1)**2 + (y2-y1)**2))
```

```
enter x11
enter x24
enter y12
enter y26
5.0
```

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

[24]:
```
n = int(input("enter n"))
sum=0
for i in range(1,n+1):
    sum+=i
print(sum)
```

```
enter n3
6
```

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

[25]:
```
n = int(input("enter n"))
sum=0
for i in range(1,n+1):
    sum+=i**2
print(sum)
```

```
enter n3
14
```

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

[28]:
```
first = input("enter first")
last = input("enter last")
print(first+last,sep=" ")
```

```
enter firstmeet
enter lastok
meetok
```

0.0.13 13) WAP to swap two numbers.

[2]:
```
a=10
b=20
temp=a
a=b
b=temp
print(a,b,sep=" ")
```

```
20 10
```

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

[4]:
```
km = float(input("enter km"))
print(f"meter = {km * 1000}",f"feet = {3280.84}",f"inches = {39370.
  ↪1}",f"centimeter = {100000}",sep="\n");
```

```
meter = 1000.0
feet = 3280.84
inches = 39370.1
centimeter = 100000
```

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

[5]:
```
day = int(input("day"))
month = int(input("month"))
year = int(input("year"))
print(f"{day}-{month}-{year}")
```

```
23-11-2024
```

[ ]:
```

```

Python Programming - Lab - 2

March 11, 2025

Python Programming - 2301CS404

Lab - 2

OM BHUT | 23010101033 | 122

1 if..else..

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

[4]:
```
a = int(input("enter number"))
if(a>=0):
    print("positive")
else:
    print("negative")
```

```
enter number-4
negative
```

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

[6]:
```
a = int(input("enter number"))
if(a%2==0):
    print("even")
else:
    print("odd")
```

```
enter number3
odd
```

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

[10]:
```
a = int(input("enter number 1"))
b = int(input("enter number 2"))
# if(a>b):
#     print("a is greater")
# else:
#     print("b is greater")
```

```
# a>b ? print("a is greater") : print("b is greater")
print("a is greater") if a>b else print("b is greater")
```

```
enter number 15
enter number 26
b is greater
```

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

[6]:
```
a = int(input("enter number 1"))
b = int(input("enter number 2"))
c = int(input("enter number 3"))

print("a is greater" if a>c else "c is greater" if a>b else "b is greater" if b␣
 ↪> c  else "c is greater")
# print("a is greater") if a>c else print("c is greater") if a>b else print("b␣
 ↪is greater") if b > c  else print("c is greater")
# print("a is greater" if a > b and a > c else "b is greater" if b > c else "c␣
 ↪is greater")
```

```
enter number 15
enter number 26
enter number 37
c is greater
```

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]

[9]:
```
n = int(input("enter year"))
if(n%4==0 and n%100==0 or n%4):
    print("leap year")
```

```
enter year2024
```

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

[10]:
```
n = int(input("enter number"))
match n:
    case 1:
        print("monday")
    case 2:
        print("tuesday")
    case 3:
        print("wednesday")
    case 4:
        print("thursday")
    case 5:
```

```
        print("friday")
    case 6:
        print("saturday")
    case 7:
        print("sunday")
```

```
enter number5
friday
```

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

[12]:
```
a = int(input("enter number 1"))
c = input("enter operator")
b = int(input("enter number 2"))

match c:
    case '+':
        print(a+b)
    case '-':
        print(a-b)
    case '*':
        print(a*b)
    case '/':
        print("b should not be 0") if b==0 else print(a/b)
```

```
enter number 15
enter operator/
enter number 20
b should not be 0
```

1.0.7 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

[14]:
```
totalSum = 0;
for i in range(5):
    n = float(input(f"enter marks of student {i+1}"))
    totalSum+=n
percentage = totalSum/5
if percentage<35:
    print("fail")
elif percentage>=35 and percentage<45:
    print("pass")
elif percentage>=45 and percentage<60:
    print("second")
```

```
elif percentage>=60 and percentage<70:
    print("first class")
else:
    print("distinction")
```

```
enter marks of student 150
enter marks of student 250
enter marks of student 350
enter marks of student 450
enter marks of student 550
second
```

1.0.8 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.

[17]:
```
a = int(input("Enter side 1: "))
b = int(input("Enter side 2: "))
c = int(input("Enter side 3: "))

if a==b and b==c and a==c:
    print("Equilateral")
elif a==b or b==c or a==c:
    print("isosceles")
elif a!=b and a!=c and b!=c:
    print("scalene")
elif (a**2 + b**2 == c**2 or a**2+c**2 == b**2 or c**2+b**2 == a**2):
    print("right-angled triangle")
```

```
Enter side 1: 3
Enter side 2: 4
Enter side 3: 5
scalene
```

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

[21]:
```
a = int(input("Enter  1: "))
b = int(input("Enter  2: "))
c = int(input("Enter  3: "))

if a>b:
    if a>c:
        print("c is second")
    else:
        print("a is second")
else:
    if b>c:
        print("c is second")
    else:
```

```
        print("b is second")
```

```
Enter  1: 2
Enter  2: 3
Enter  3: 5
b is second
```

1.0.10 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

[32]:
```
unit = int(input("Enter units: "))
totalSum = 0

# First 50 units
if unit > 0:
    if unit > 50:
        totalSum += 50 * 2.60
        unit -= 50
    else:
        totalSum += unit * 2.60
        unit = 0

# Next 50 units (51 to 100)
if unit > 0:
    if unit > 50:
        totalSum += 50 * 3.25
        unit -= 50
    else:
        totalSum += unit * 3.25
        unit = 0

# Next 100 units (101 to 200)
if unit > 0:
    if unit > 100:
        totalSum += 100 * 5.26
        unit -= 100
    else:
        totalSum += unit * 5.26
        unit = 0

# Above 200 units
if unit > 0:
    totalSum += unit * 8.45
```

```
# Print total sum
print("Total cost: Rs.", totalSum)
```

```
Enter units: 350
Total cost: Rs. 2086.0
```

[ ]:

Python Programming - Lab - 3

March 11, 2025

Python Programming - 2301CS404

Lab - 3

OM BHUT | 23010101033 | 122

1 for and while loop

1.0.1 01) WAP to print 1 to 10.

[4]:
```
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.

[7]:
```
n = int(input("enter n: "))
for i in range(n+1):
    print(i)
```

```
enter n: 50
0
1
2
3
4
5
6
7
8
```

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

[15]:
```
n = int(input("enter n: "))
for i in range(n+1):
    if(i%2!=0):
        print(i,end = " ")
```

```
enter n: 50
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
```

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

[16]:
```
a = int(input("enter a "))
b = int(input("enter b "))
for i in (a,b+1):
    if(i%2==0 and i%3!=0):
        print(i,end = " ")
```

```
enter a 1
enter b 50
50
```

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

[24]:
```
n = int(input("enter n "))
sum = 0
for i in range(1,n+1):
    sum+=i
print(sum)
```

```
enter n 5
15
```

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

[27]:
```
n = int(input("enter n "))
sum = 0
for i in range(1,n+1):
    sum+=i**2
print(sum)
```

```
enter n 3
14
```

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

[29]:
```
n = int(input("enter n "))
sum = 0
for i in range(1,n+1):
    if(i%2==0):
        sum-=i
    else:
        sum+=i
print(sum)
```

```
enter n 5
3
```

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

[31]:
```
n = int(input("enter n "))
for i in range(1,11):
    print(n*i,end=" ")
```

```
enter n 10
10 20 30 40 50 60 70 80 90 100
```

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

[33]:
```
def factorial(n):
    fact = 1
    ans = 1
    for i in range(2,n+1):
        ans = ans*i
    return ans
n = int(input("enter n"))
print(factorial(n))
```

```
enter n5
120
```

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

[3]:
```
def factors(n):
    list = []
    for i in range(1,n+1):
        if(n%i==0):
            list.append(i)
    return list
n = int(input("enter n "))
print(factors(n))
```

```
[1, 2, 3, 6]
```

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

[8]:
```
def checkPrime(n):
    if(n==1):
        return False
    for i in range(2,int(n/2)+1):
        if(i%n==0):
            return False

    return True

n = int(input("enter n "))
print(checkPrime(n))
```

```
False
```

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

[14]:
```
def sumOfDigits(n):
    sum=0
    while(n>0):
        lastDigit = n%10
        sum+=lastDigit
        n//=10
    return sum
n = int(input("enter n "))
print(sumOfDigits(n))
```

```
7
```

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

[17]:
```
def checkPalindrome(n):
    originalNum = n
    reversedNum = 0
    while (n>0):
       remainder = n%10
       reversedNum = reversedNum*10 + remainder
       n//=10
    return originalNum==reversedNum
n = int(input("enter n "))
print((checkPalindrome(n)))
```

```
False
```

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

[21]:
```
a = int(input("enter a "))
b = int(input("enter b"))
i=2
while(i<=a and i<=b):
    if(a%i==0 and b%i==0):
       print(i)
       break
    i+=1
```

```
5
```

Python Programming - Lab - 4

March 11, 2025

Python Programming - 2301CS404

Lab - 4

OM BHUT | 23010101033 | 122

1 String

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

[3]:
```
def palindromeCheck(s):
    return str(s) == str(s[::-1])
print(palindromeCheck("jaja"))
```

```
False
```

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

[11]:
```
s = input("enter s : ")
s = s.split(" ")
s = s[::-1]
newS = ""
for i in s:
    newS+=i+" "
print(newS)
```

```
world hello
```

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

[12]:
```
s = input("enter string ")
i = int(input("enter i "))
s = s.replace(s[i],"",1)
print(s)
```

```
hllo
```

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

[14]:
```
s = input("enter string ")
count = -1
for i in s:
    count+=1
print(count)
```

```
4
```

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

[19]:
```
s = input("enter ")
s = s.split()
for i in s:
    if len(i)%2==0:
        print(i)
```

```
hell
```

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

[21]:
```
s = input("enter")
count=0
for i in s:
    if (i == 'a' or i == 'e' or i=='i' or i=='o' or i=='u'):
       count+=1
print(count)
```

```
7
```

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

[40]:
```
s = input("enter ").title().split()
newS = ""
for i in s:
    reversed = i[::-1][0]
    newS += i.removesuffix(reversed) + reversed.capitalize() + " "
print(newS)
```

```
JaY HinD
```

1.0.8 08) WAP to convert given array to string.

[30]:
```
arr = [1,2,3,'om','jay']
s = ""
for i in arr:
    s+=str(i)+" "
print(s)
```

```
1 2 3 om jay
```

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.

[32]:
```
password = input("enter pass")
confirmPassword = input("enter curr pass")
if(password.lower() == confirmPassword.lower()):
    if(password == confirmPassword):
        print("correct")
    else:
        print("case is not correct")
else:
    print("wrong pass")
```

```
case is not correct
```

1.0.11 10) : Display credit card number.

1.0.12 card no. : 1234 5678 9012 3456

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

[36]:
```
cardNo = "1234 5678 9012 3456".split()
print(f"**** **** **** {cardNo[len(cardNo)-1]}")
```

```
**** **** **** 3456
```

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

1.0.15 s1 = decimal and s2 = medical are Anagram

[38]:
```
s1 = "decimal"
s2 = "medical"
if(sorted(s1) == sorted(s2)):
    print("anagram")
else:
    print("not anagram")
```

```
not anagram
```

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

1.0.17 input : EHlsarwiwhtwMV

1.0.18 output : lsarwiwhtwEHMV

[42]:
```
s = "EHILshshdWEWEsjsdj"
lower = ""
upper = ""
for i in s:
```

```
    if i.isupper():
        upper+=i
    else:
        lower+=i
print(lower+upper)
```

```
shshdsjsdjEHILWEWE
```

[ ]:

Python Programming - Lab - 5

March 11, 2025

Python Programming - 2301CS404

OM BHUT

OM BHUT | 23010101033 | 122

1 List

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

[3]:
```
l1 = [3,15,23,12]
ans = sum(l1)
print(ans)
```

```
53
```

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

[4]:
```
l1 = [3,15,23,12]
ans = max(l1)
print(ans)
```

```
23
```

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

[5]:
```
l1 = [3,15,23,12]
ans = len(l1)
print(ans)
```

```
4
```

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

[9]:
```
l1 = [3,15,23,12]
n = len(l1)-1
l1[0],l1[n] = l1[n],l1[0]
print(l1)
```

```
[12, 15, 23, 3]
```

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

[19]:
```
l1 = [3,15,23,12]
n = len(l1)//2
first,last = l1[:n], l1[n:]
last.extend(first)
print(last)
```

```
[23, 12, 3, 15]
```

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

[20]:
```
a = int(input("enter first index"))
b = int(input("enter second index"))
l1 = [3,15,23,12]
l1[a],l1[b] = l1[b],l1[a]
print(l1)
```

```
[3, 23, 15, 12]
```

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

[26]:
```
l1 = input("enter space seperated values").split()
l1 = reversed(l1)
l1 = [int(i) for i in l1]
print(l1)
```

```
[6, 5, 4, 3, 2, 1]
```

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

[32]:
```
l1 = [3,15,23,12,24]
l2 = filter(lambda x : x%2==0,l1)
print(list(l2))
```

```
[12, 24]
```

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

[39]:
```
l1 = [3,3,3,3,15,15,23,12,24]
# ans = []
# for i in l1:
#     if(i not in ans):
#         ans.append(i)
# print(ans)
dict = {}
for i in l1:
    if(i in dict):
        dict[i] += 1
```

```
    else:
        dict[i] = 1
# print(dict)
ans = []
for key,value in dict.items():
    if value==1:
        ans.append(key)
print(ans)
```

```
[23, 12, 24]
```

1.0.10 10) WAP to copy a list.

[40]:
```
l1 = [3,15,23,12,24]
l2 = l1.copy()
print(l2)
```

```
[3, 15, 23, 12, 24]
```

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

[43]:
```
a = int(input("enter first"))
b = int(input("enter second"))
for i in range(a,b):
    if i%3==0:
        print(i)
```

```
3
6
9
```

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

[44]:
```
l1 = [3,3,3,3,15,15,23,12,24]
dict = {}
for i in l1:
    if(i in dict):
        dict[i] += 1
    else:
        dict[i] = 1
print(dict)
```

```
{3: 4, 15: 2, 23: 1, 12: 1, 24: 1}
```

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

[61]:
```
l1 = [3,15,23,12,24]
largest = max(l1)
l1 = filter(lambda x: x<largest,l1)
l1 = list(l1)
print(max(l1))
```

```
23
```

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

[63]:
```
k = int(input("enter frequency"))
l1 = [3,3,3,3,15,15,23,12,24]
dict = {}
for i in l1:
    if(i in dict):
        dict[i] += 1
    else:
        dict[i] = 1
ans = []
for key,value in dict.items():
    if value>=k:
        ans.append(key)
print(ans)
```

```
[3]
```

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

[65]:
```
l1 = [3,15,23,12,24]
l1 = map(lambda x: x**2,l1)
print(list(l1))
```

```
[9, 225, 529, 144, 576]
```

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

[67]:
```
l1 = ['banana','apple','bonanaza']
l2 = filter(lambda x:x[0]=='b',l1)
print(list(l2))
```

```
['banana', 'bonanaza']
```

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

[72]:
```
l1 = [3,3,4,7,12,4]
l2 = [3,4,5,9]
s1 = set(l1)
s2 = set(l2)
print(s1.intersection(s2))
```

```
{3, 4}
```

Python Programming - Lab - 6

March 11, 2025

Python Programming - 2301CS404

Lab - 6

OM BHUT | 23010101033 | 122

1 Tuple

1.0.1 01) WAP to find sum of tuple elements.

[1]:
```
t1 = (1,2,3,4,5)
print(sum(t1))
```

```
15
```

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

[13]:
```
t1 = (1,2,3,4,5,9,12,2.5)
n = int(input("enter n"))
sortedTuple = sorted(t1)
print("min value ",end="=")
for i in range(0,n):
    print(sortedTuple[i] , end=" ")
print("max value ",end="=")
for i in range(len(t1)-1,len(t1)-n-1,-1):
    print(sortedTuple[i] , end=" ")
```

```
min value =1 2 2.5 max value =12 9 5
```

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

[29]:
```
l1 = [(1,2,3,4),(5,6,7,8),(9,10,11,12),(3,3,3,6)]
k = int(input())
for tuple in l1:
    check = False
    count = 0
    for i in tuple:
        if i%k==0:
```

```
            count+=1
    if(count==len(tuple)):
        print(tuple)
```

```
(3, 3, 3, 6)
```

[ ]:
```
def find_tuples_divisible_by_k(tuples_list, K):
    # Filter tuples where all elements are divisible by K
    result = [tup for tup in tuples_list if all(x % K == 0 for x in tup)]
    return result

# Example usage:
tuples_list = [(10, 20, 30), (5, 15, 25), (2, 4, 8), (7, 14, 21)]
K = 5
result = find_tuples_divisible_by_k(tuples_list, K)
print(result)
```

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

[30]:
```
l1 = [1,3,6,5,9,2]
t1 = [(i,i**3) for i in l1]
print(t1)
```

```
[(1, 1), (3, 27), (6, 216), (5, 125), (9, 729), (2, 8)]
```

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

[34]:
```
l1 = [(1,2,3,4),(-5,6,7,8),(9,10,11,12),(-3,3,3,6)]
ans = [tuple for tuple in l1 if(all(x>=0 for x in tuple))]
print(ans)
```

```
[(1, 2, 3, 4), (9, 10, 11, 12)]
```

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

[2]:
```
t1 = (3,4)
l1 = [1,2]
ansList = []
ansList.append(t1)
l2 = list(t1)
l1.append(l2)
ansTuple = tuple(l1)
print(ansList)
print(ansTuple)
```

```
[(3, 4)]
(1, 2, [3, 4])
```

1.0.7 07) WAP to remove list of tuples of length K.

[6]:
```
l1 = [(1,2,3,4,7,8),(-5,6,7,8),(9,10,11,12),(-3,3,3,6)]
k = int(input())
l2 = [tuple for tuple in l1 if len(tuple)!=k]
print(l2)
```

```
[(1, 2, 3, 4, 7, 8)]
```

1.0.8 08) WAP to remove duplicates from tuple.

[7]:
```
t1 = (-3,3,3,6)
ans = tuple(set(t1))
ans
```

[7]:
```
(3, -3, 6)
```

[17]:
```
t1 = (-3,3,3,6,6,5,5,6)
l1 = []
s1 = set()
for i in t1:
    if i not in s1:
        l1.append(i)
    s1.add(i)
print(tuple(l1))
```

```
(-3, 3, 6, 5)
```

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

[10]:
```
t1 = (1,2,3,4,5,6,7)
ans = tuple(t1[i]*t1[i+1] for i in range(0,len(t1)-1))
print(ans)
```

```
(2, 6, 12, 20, 30, 42)
```

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

[14]:
```
t1 = (1,2,3,4,5,6,7)
print(len(t1) == len(set(t1)))
```

```
True
```

[ ]:
```

```

Python Programming - Lab - 7

March 11, 2025

Python Programming - 2301CS404

Lab - 7

OM BHUT | 23010101033 | 122

1 Set & Dictionary

1.0.1 01) WAP to iterate over a set.

[ ]:
```
s = {1,2,3,4,5,6,7}
for i in s:
    print(i,end=" ")
```

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

[ ]:
```
s = {1,2,3,4,5,6,7}
l1 = list(s)
t1 = tuple(s)
s1 = ""
for i in s:
    s1 += str(i) + " "
print(l1)
print(t1)
print(s1)
```

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

[ ]:
```
s = {1,2,3,4,5,6,7}
print(max(s))
print(min(s))
```

1.0.4 04) WAP to perform union of two sets.

[ ]:
```
s1 = {1,2,3,4,5,6,7}
s2 = {4,5,6,7,8,9,10}
s3 = s1.union(s2)
s3
```

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

[ ]:
```
s1 = {1,2,3,4,5,6,7}
s2 = {4,5,6,7,8,9,10}
s3 = s1.intersection(s2)
print("yes" if len(s3) >= 1 else "no")
```

1.0.6 06) WAP to remove duplicates from list.

[ ]:
```
l1 = [1,1,1,1,2,2,2,3,3,3]
print(list(set(l1)))
```

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

[ ]:
```
s = "hello hello hi bye bye".split(" ")
s1 = " ".join(set(s))
s1
```

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

[ ]:
```
s1 = {1,2,3,4,5,6,7}
s2 = {4,5,6,7,8,9,10}
s3 = s1 - s2
s3
```

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

[13]:
```
def checkAnagram(s1,s2):
    return sorted(s1) == sorted(s2)

def checkAnagramUsingDictionary(s1,s2):
    d1 = {i:s1.count(i) for i in s1}
    d2 = {i:s2.count(i) for i in s1}
    return d1 == d2

s1 = "are you why"
s2 = "why are you"
print(checkAnagramUsingDictionary(s1,s2))
```

```
True
```

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

[20]:
```
l1=[1,2,3,5]
l2=[1,3,9]
l3=[1,5,7,3]
```

```
s1 = set(l1)
s2 = set(l2)
s3 = set(l3)
print(s1.intersection(s2,s3))
```

```
{1, 3}
```

[21]:
```
s1 = set(s1)
s1.intersection(l2,l3)
```

[21]:
```
{1, 3}
```

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

[22]:
```
def countVowels(set1,str1):
    count = 0
    for i in str1:
        if i in set1:
            count+=1
    return count
s1 = {'a','e','i','o','u'}
print(countVowels(s1,"aaabbdfgdu"))
```

```
4
```

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

[3]:
```
set1 = {'0','1'}
str1 = "010111010"

count1 = 0
for i in str1:
    if i in set1:
        count1+=1
print(count1==len(str1))
```

```
True
```

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

[10]:
```
d1 = {3:"bhavya",2:"avi",4:"chaman"}

print(sorted(d1.values()))
print(sorted(d1.keys()))
```

```
['avi', 'bhavya', 'chaman']
[2, 3, 4]
```

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

[11]:
```
d1 = {}
sum = 0
for i in range(5):
    key = input("enter key")
    value = int(input("enter value"))
    sum+=value
    d1[key] = value
print(sum)
```

```
21
```

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.

[19]:
```
dict1 = {'a': 5, 'c': 8, 'e': 2}
userInput = 'a'
valueOfKey = dict1.get(userInput)
if (valueOfKey == None):
    print("Key Not Found")
else:
    print(dict1[userInput])
```

```
5
```

Python Programming - Lab - 8

March 11, 2025

Python Programming - 2301CS404

Lab - 8

OM BHUT | 23010101033 | 122

1 User Defined Function

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

[ ]:
```
def calculateBmi(mass,height):
    return mass/height**2
```

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

[3]:
```
def addFirstN(n : int):
    return n*(n+1)/2
addFirstN(50)
```

[3]:
```
1275.0
```

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

[6]:
```
def checkPrime(n):
    if(n==2 or n==1 or n==0 or n==4):
        return 0
    for i in range(2,n//2):
        if n%i == 0:
            return 0
    return 1

print(checkPrime(8))
```

```
0
```

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

[7]:
```
def primeNumbersList(a,b):
    ans = []
    for i in range(a,b+1):
        if (checkPrime(i) == 1):
            ans.append(i)
    return ans
print(primeNumbersList(2,50))
```

```
[3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

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

[10]:
```
def checkPalindrome(s1):
    return s1 == s1[::-1]
print(checkPalindrome("abab"))
```

```
False
```

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

[ ]:
```
def sumOfAllElements(l1):
    return sum(l1)
```

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

[1]:
```
def sumOfFirstElementsOfTuple(l1):
    sum = 0
    for t in l1:
        sum += t[0]
    return sum
list1 = [(1,2,3),(3,4,1),(4,8,2)]
print(sumOfFirstElementsOfTuple(list1))
```

```
8
```

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

[8]:
```
def fibonacciSeries(n):
    first = 0
    second =1
    for i in range(n):
        print(first,end=" ")
        nextVal = first+second
        first = second
```

```
        second = nextVal


def fibonacciSeriesUsingRecursion(n):
    if n==0 or n==1:
        return n
    else:
        return fibonacciSeriesUsingRecursion(n - 1) +␣
 ↪fibonacciSeriesUsingRecursion(n - 2)

# fibonacciSeries(30)
fibonacciSeriesUsingRecursion(6)
```

[8]: 8

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

[12]:
```
def giveNameFromRollNumber(dict1 : dict,rollNo):
    return dict1[rollNo]

dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'}
print(giveNameFromRollNumber(dict1,102))
```

```
Rahul
```

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

[14]:
```
def sumOfEndingScores(l1):
    sum=0
    for i in l1:
        sum += i%10
    return sum
scores = [200, 456, 300, 100, 234, 678]
print(sumOfEndingScores(scores))
```

```
18
```

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'}

[4]:
```
def invertDictionary(d1 : dict):
    invertedDictionary = {value:key for key,value in d1.items()}
    return invertedDictionary


d1 = {
    0:"om",
    1:"yash",
    2:"raj"
}
print(invertDictionary(d1))
```

```
{'om': 0, 'yash': 1, 'raj': 2}
```

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.

[12]:
```
def checkPanagram(s : str):
    set1 = {chr(i) for i in range(97,123)}
    set2 = set(s)
    if " " in set2:
        set2.remove(' ')
    return set1 == set2

s = "the quick brown fox jumps over the lazy dog"
# s="hello"
print(checkPanagram(s))
```

```
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

[14]:
```
def countUpperLower(s : str):
    noOfUpper = 0
    noOfLower = 0
    for i in s:
        if i.isupper():
            noOfUpper+=1
        else:
            noOfLower+=1
    return {
        "upper":noOfUpper,
        "lower":noOfLower
    }
```

```
print(countUpperLower("AbcDEfgh"))
```

```
{'upper': 3, 'lower': 5}
```

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

[15]:
```
x = lambda a,b : min(a,b)

print(x(5,6))
```

```
5
```

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

[18]:
```
l1 = ["jayesh","rajnikant","raftaar"]
l2 = filter(lambda x: len(x)>=7,l1)
print(list(l2))
```

```
['rajnikant', 'raftaar']
```

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

[22]:
```
l1 = ["jayesh","rajnikant","raftaar","Ok","hello"]
l2 = map(lambda x : x[0].upper() + x[1::] , l1)
print(list(l2))
```

```
['Jayesh', 'Rajnikant', 'Raftaar', 'Ok', 'Hello']
```

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 Length Positional(*args) & variable length Keyword Arguments (**kwargs)
5. Keyword-Only & Positional Only Arguments

[3]:
```
def positionalArguments(a,b):
    return a+b
def defaultArguments(a=1,b=1):
    return a+b
def variableLengthPositional(a,*args):
    print(a,args)
def keywordArguments(a,**kwargs):
    print(a,kwargs)
def keywordOnly(*,a,b):
    return a+b
```

```
def positionalArgumentsOnly(a,b,/):
    return a+b
positionalArguments(1,2)
positionalArguments(b=2,a=1)
defaultArguments()
variableLengthPositional(10,20,30,40)
keywordArguments(10,b=20,c=30)
keywordOnly(a=10,b=20)
positionalArgumentsOnly(10,20)
```

[3]: 30

[ ]:

Python Programming - Lab - 9

March 11, 2025

Python Programming - 2301CS404

Lab - 9

OM BHUT | 23010101033 | 122

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

[2]:
```
fp1 = open("hello.txt")
data = fp1.read()
print(data)
fp1.close()
```

```
hello world
```

[18]:
```
fp1 = open("hello.txt")
for i in fp1:
    print(i)
fp1.close()
```

```
hello world

hello world

hello world

hello world

hello world

hello world
```

[17]:
```
fp1 = open("hello.txt")
data = fp1.readlines()
print(data)
fp1.close()
```

```
['hello world\n', 'hello world\n', 'hello world\n', 'hello world\n', 'hello
world\n', 'hello world']
```

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

[1]:
```
fp1 = open("new.txt","x")
fp1.close()
```

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

[6]:
```
fp1 = open("new.txt")
for i in range(0,5):
    data = fp1.readline()
    print(data)
fp1.close()
```

```
hello 1

hello 2

hello 3

hello 4

hello 5
```

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

[14]:
```
fp1 = open("new.txt")
s1 = fp1.read().split()
l1 = [len(i) for i in s1]
maxLength = max(l1)
ans = filter(lambda x: len(x)==maxLength,s1)
print(list(ans))
fp1.close()
```

```
['hello', 'hello', 'hello', 'hello', 'hello']
```

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

[28]:
```
with open("new.txt","r") as fp1:
    countWords = 0
    countCharacters = 0
```

```
    countLines = 0
    data = fp1.read().split()
    l1 = [len(i) for i in data]
    countWords = len(l1)
    countCharacters = sum(l1)
    fp1.seek(0)
    countLines = len(fp1.readlines())


    print(countWords,countCharacters,countLines,sep=" ")
```

```
10 30 5
```

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

[29]:
```
with open("new.txt","r") as fp1,open("new2.txt","w") as fp2:
    fp2.write(fp1.read())
```

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

[30]:
```
import os
print(os.path.getsize("new.txt"))
```

```
43
```

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

[34]:
```
def frequencyOfWord(wordToFind:str,fileName:str):
    with open(fileName,"r") as fp1:
        data = fp1.read().split()
    return data.count(wordToFind)
frequencyOfWord("hello","new.txt")
```

[34]:
```
5
```

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.

[44]:
```
marks = ["25","45","78","35","45"]
# for i in range(0,5):
#     mark = input(f"enter marks for {i+1}")
with open("marks.txt","w") as fp1:
    fp1.write(" ".join(marks))
with open("marks.txt","r") as fp2:
    data = fp2.read().split()
    l1 = [int(i) for i in data]
```

```
    print(max(l1))
```

78

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)

[49]:
```
def firstHundredPrime():
    l1 = []
    for i in range(0,100):
        check = True
        for j in range(2,i//2):
            if i%j==0:
                check=False
        if(check):
            l1.append(str(i))
    return l1
with open("primenumbers.txt","w") as fp1:
    l1 = firstHundredPrime()
    fp1.write("\n".join(l1))
```

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

[51]:
```
with open("new.txt","r") as fp1,open("marks.txt","r") as fp2, open("mergeAns.
 ↪txt","w") as fp3:
    data1 = fp1.read()
    data2 = fp2.read()
    ans = data1+"\n"+data2
    fp3.write(ans)
```

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

[58]:
```
data = ""
with open("mergeAns.txt","r") as fp1:
    data = fp1.read().replace("hello","helloBye")
with open("new4.txt","w") as fp1:
    fp1.write(data)
```

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 Programming - Lab - 10

March 11, 2025

Python Programming - 2301CS404

Lab - 10

OM BHUT | 23010101033 | 122

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.

[1]:
```
try:
    print(1/0)
except ZeroDivisionError:
    raise ZeroDivisionError
except ValueError:
    raise ValueError
except TypeError:
    raise TypeError
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
d:\om\python\Python Programming - Lab - 10.ipynb Cell 4 line 2
      <a href='vscode-notebook-cell:/d%3A/om/python/
↪Python%20Programming%20-%20Lab%20-%2010.ipynb#W3sZmlsZQ%3D%3D?line=0'>1</a>␣
↪try:
----> <a href='vscode-notebook-cell:/d%3A/om/python/
↪Python%20Programming%20-%20Lab%20-%2010.ipynb#W3sZmlsZQ%3D%3D?line=1'>2</a>
↪ print(1/0)                                                              ␣
      <a href='vscode-notebook-cell:/d%3A/om/python/
↪Python%20Programming%20-%20Lab%20-%2010.ipynb#W3sZmlsZQ%3D%3D?line=2'>3</a>␣
↪except ZeroDivisionError:

ZeroDivisionError: division by zero
```

```
During handling of the above exception, another exception occurred:

ZeroDivisionError                         Traceback (most recent call last)
d:\om\python\Python Programming - Lab - 10.ipynb Cell 4 line 4

      <a href='vscode-notebook-cell:/d%3A/om/python/
 ↪Python%20Programming%20-%20Lab%20-%2010.ipynb#W3sZmlsZQ%3D%3D?line=1'>2</a>
 ↪ print(1/0)
      <a href='vscode-notebook-cell:/d%3A/om/python/
 ↪Python%20Programming%20-%20Lab%20-%2010.ipynb#W3sZmlsZQ%3D%3D?line=2'>3</a>
 ↪except ZeroDivisionError:
----> <a href='vscode-notebook-cell:/d%3A/om/python/
 ↪Python%20Programming%20-%20Lab%20-%2010.ipynb#W3sZmlsZQ%3D%3D?line=3'>4</a>
 ↪ raise ZeroDivisionError
      <a href='vscode-notebook-cell:/d%3A/om/python/
 ↪Python%20Programming%20-%20Lab%20-%2010.ipynb#W3sZmlsZQ%3D%3D?line=4'>5</a>
 ↪except ValueError:
      <a href='vscode-notebook-cell:/d%3A/om/python/
 ↪Python%20Programming%20-%20Lab%20-%2010.ipynb#W3sZmlsZQ%3D%3D?line=5'>6</a>
 ↪ raise ValueError

ZeroDivisionError:
```

[4]:
```
try:
    print(1/0)
except Exception:
    raise Exception
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
d:\om\python\Python Programming - Lab - 10.ipynb Cell 5 line 2

      <a href='vscode-notebook-cell:/d%3A/om/python/
 ↪Python%20Programming%20-%20Lab%20-%2010.ipynb#X31sZmlsZQ%3D%3D?line=0'>1</a>
 ↪try:
----> <a href='vscode-notebook-cell:/d%3A/om/python/
 ↪Python%20Programming%20-%20Lab%20-%2010.ipynb#X31sZmlsZQ%3D%3D?line=1'>2</a>
 ↪  print(1/0)
      <a href='vscode-notebook-cell:/d%3A/om/python/
 ↪Python%20Programming%20-%20Lab%20-%2010.ipynb#X31sZmlsZQ%3D%3D?line=2'>3</a>
 ↪except Exception:

ZeroDivisionError: division by zero

During handling of the above exception, another exception occurred:

Exception                                 Traceback (most recent call last)
```

```
d:\om\python\Python Programming - Lab - 10.ipynb Cell 5 line 4
      <a href='vscode-notebook-cell:/d%3A/om/python/
↪Python%20Programming%20-%20Lab%20-%2010.ipynb#X31sZmlsZQ%3D%3D?line=1'>2</a>
↪   print(1/0)
      <a href='vscode-notebook-cell:/d%3A/om/python/
↪Python%20Programming%20-%20Lab%20-%2010.ipynb#X31sZmlsZQ%3D%3D?line=2'>3</a>
↪except Exception:
----> <a href='vscode-notebook-cell:/d%3A/om/python/
↪Python%20Programming%20-%20Lab%20-%2010.ipynb#X31sZmlsZQ%3D%3D?line=3'>4</a>
↪   raise Exception

Exception:
```

1.0.2 02) WAP to handle following exceptions:

1. IndexError
2. KeyError

[6]:
```
try:
    d1 = {"jay":1}
    d1["tata"]
except IndexError:
    print("wrong index")
except KeyError:
    print("wrong key")
```

```
wrong key
```

1.0.3 03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

[9]:
```
try:
    import abcd
    # f1 = open("abc.txt")
except (FileNotFoundError,ModuleNotFoundError) as e:
    print(e)
```

```
No module named 'abcd'
```

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

[10]:
```
try:
    print("hello")
except Exception as e:
    print(e)
```

```
hello
```

1.0.5 05) WAP to demonstrate else and finally block.

[18]:
```
try:
    print("1/0")
except Exception as e:
    print(e)
else:
    print("from else")
finally:
    print("from finally")
```

```
1/0
from else
from finally
```

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.

[22]:
```
def getList(s1: str):
    try:
        l1 = s1.split()
        ans = [int(i) for i in l1]
        return ans
    except ValueError:
        print("enter only integers")
s1 = input("enter : ")
print(getList(s1))
```

```
enter only integers
None
```

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

[24]:
```
def divide(a,b):
    try:
        print(a/b)
    except ZeroDivisionError as ze:
        print(ze)
divide(1,0)
```

```
division by zero
```

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.

[28]:
```
try:
    age = int(input("enter age"))
    if age<18:
        raise ValueError("Enter Valid Age")
except ValueError as ve:
    print(ve)
```

```
Enter Valid Age
```

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.

[35]:
```
class InvalidUsernameError(Exception):
    print

try:
    userName = input("enter user name")
    if len(userName) <=5 or len(userName) >= 15:
        raise InvalidUsernameError("username must between 5 and 15")
    print(userName)
except InvalidUsernameError as iue:
    print(iue)
```

```
username must between 5 and 15
```

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.

[38]:
```
class NegativeNumberError(Exception):
    print('Cannot calculate the square root of a negative number')

try:
    n = 14
    if n<0:
        raise NegativeNumberError
except NegativeNumberError as ne:
```

```
    print(ne)
```

```
Cannot calculate the square root of a negative number
```

Python Programming - Lab - 11

March 11, 2025

Python Programming - 2301CS404

Lab - 11

OM BHUT | 23010101033 | 122

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.

[ ]:

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

[2]:
```
import random

s = "abcdef"

print(random.choice(s))
```

```
f
```

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

[4]:
```
import random

s = [1,34,56,7,4]

print(random.choice(s))
```

```
34
```

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

[11]:
```
import random

random.seed(1)
s = random.randint(1,6)
print(s)

random.seed(2)
s = random.randint(1,6)
print(s)


random.seed(1)
s = random.randint(0,6)
print(s)
```

```
2
1
1
```

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

[13]:
```
import random

count = 1
while count<=3:
    n = random.randint(100,999)
    if n%5==0:
        print(n)
        count+=1
```

```
885
705
425
```

1.1.2 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
t = random.sample(range(100,999),100)
ticket1 = random.choice(t)
t.remove(ticket1)
ticket2 = random.choice(t)
print(ticket1,ticket2,sep=" ")
```

```
873 544
```

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

[19]:
```
from datetime import datetime
val = datetime.now()
print(val)
```

```
2025-02-19 21:22:36.199047
```

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

[26]:
```
from datetime import datetime,timedelta
val = datetime.now() - timedelta(days=7)
print(val)
```

```
2025-02-12 21:27:42.663669
```

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

[28]:
```
date1 = datetime(2025, 2, 19)
date2 = datetime(2025, 2, 26)

date_diff = date2 - date1
print(date_diff.days)
```

```
7
```

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

[33]:
```
date1 = datetime.now()
days_of_week = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",␣
 ↪"Saturday", "Sunday"]

dayOfWeek = days_of_week[date1.weekday()]
print(dayOfWeek)
```

```
Wednesday
```

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

[1]:
```
import datetime

# Get the current date and time
now = datetime.datetime.now()
print(f"Current Date and Time: {now}")

# Get today's date
```

```
today = datetime.date.today()
print(f"Today's Date: {today}")

# Extract specific parts of the date and time
year = now.year
month = now.month
day = now.day
hour = now.hour
minute = now.minute
second = now.second
print(f"Year: {year}, Month: {month}, Day: {day}")
print(f"Hour: {hour}, Minute: {minute}, Second: {second}")

# Creating a specific date object
specific_date = datetime.date(2025, 2, 19)
print(f"Specific Date: {specific_date}")

# Add 10 days to the current date
future_date = today + datetime.timedelta(days=10)
print(f"10 days from Today: {future_date}")

# Subtract 5 days from the current date
past_date = today - datetime.timedelta(days=5)
print(f"5 days ago: {past_date}")

# Format a date into a string
formatted_date = now.strftime("%A, %B %d, %Y %I:%M%p")
print(f"Formatted Date: {formatted_date}")
```

```
Current Date and Time: 2025-02-19 21:46:57.961491
Today's Date: 2025-02-19
Year: 2025, Month: 2, Day: 19
Hour: 21, Minute: 46, Second: 57
Specific Date: 2025-02-19
10 days from Today: 2025-03-01
5 days ago: 2025-02-14
Formatted Date: Wednesday, February 19, 2025 09:46PM
```

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

[2]:
```
import math

# 1. Getting the value of pi
pi_value = math.pi
print(f"Value of Pi: {pi_value}")

# 2. Square root of a number
```

```python
num = 16
sqrt_value = math.sqrt(num)
print(f"Square root of {num}: {sqrt_value}")

# 3. Power of a number
base = 2
exponent = 3
power_value = math.pow(base, exponent)
print(f"{base} raised to the power of {exponent}: {power_value}")

# 4. Trigonometric functions
angle_deg = 45  # degrees
angle_rad = math.radians(angle_deg)  # Convert degrees to radians
sin_value = math.sin(angle_rad)
cos_value = math.cos(angle_rad)
print(f"Sine of {angle_deg} degrees: {sin_value}")
print(f"Cosine of {angle_deg} degrees: {cos_value}")

# 5. Logarithmic functions
log_value = math.log(100, 10)  # log base 10
print(f"Logarithm of 100 to the base 10: {log_value}")

# 6. Factorial of a number
fact_value = math.factorial(5)
print(f"Factorial of 5: {fact_value}")

# 7. Absolute value
neg_num = -7
abs_value = math.fabs(neg_num)
print(f"Absolute value of {neg_num}: {abs_value}")

# 8. Rounding a number
number = 4.567
rounded_value = round(number, 2)
print(f"{number} rounded to 2 decimal places: {rounded_value}")

# 9. Greatest common divisor (GCD)
gcd_value = math.gcd(36, 60)
print(f"GCD of 36 and 60: {gcd_value}")

# 10. Converting radians to degrees
radian_value = math.pi / 4
degree_value = math.degrees(radian_value)
print(f"{radian_value} radians is equal to {degree_value} degrees")
```

```
Value of Pi: 3.141592653589793
Square root of 16: 4.0
2 raised to the power of 3: 8.0
```

```
Sine of 45 degrees: 0.7071067811865476
Cosine of 45 degrees: 0.7071067811865476
Logarithm of 100 to the base 10: 2.0
Factorial of 5: 120
Absolute value of -7: 7.0
4.567 rounded to 2 decimal places: 4.57
GCD of 36 and 60: 12
0.7853981633974483 radians is equal to 45.0 degrees
```

Python Programming - Lab - 12

March 11, 2025

Python Programming - 2301CS404

Lab - 12

OM BHUT | 23010101033 | 122

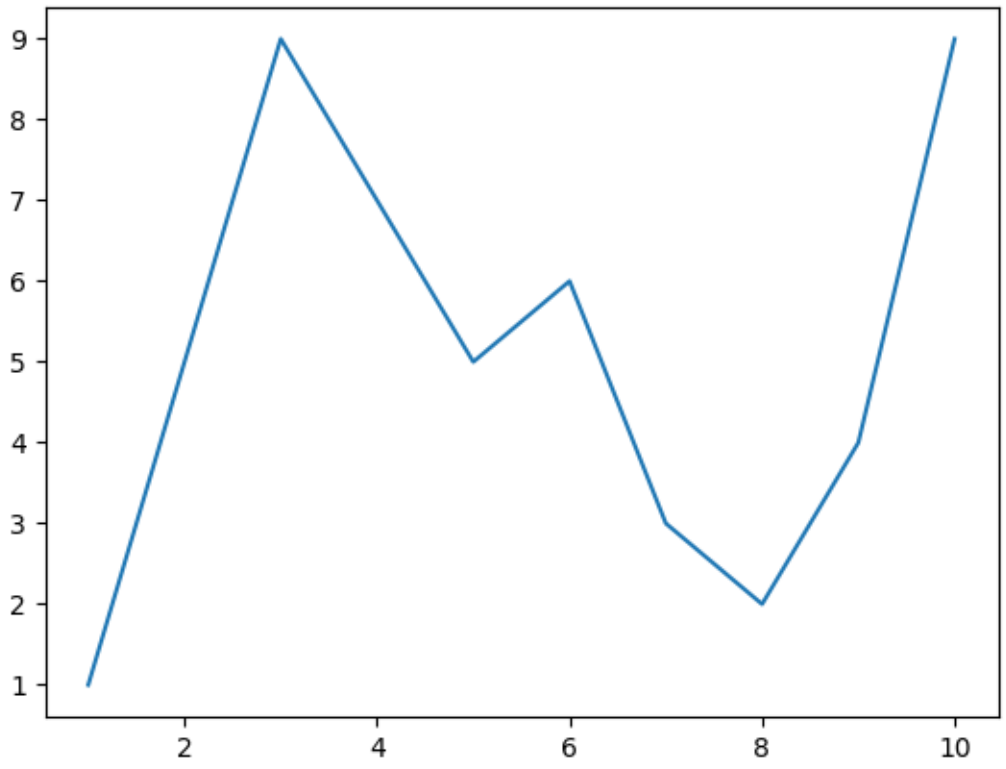
[4]:
```
!pip install matplotlib
```

```
Collecting matplotlib
  Using cached matplotlib-3.10.1-cp311-cp311-win_amd64.whl (8.1 MB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Using cached contourpy-1.3.1-cp311-cp311-win_amd64.whl (219 kB)
Collecting cycler>=0.10 (from matplotlib)
  Using cached cycler-0.12.1-py3-none-any.whl (8.3 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Using cached fonttools-4.56.0-cp311-cp311-win_amd64.whl (2.2 MB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Using cached kiwisolver-1.4.8-cp311-cp311-win_amd64.whl (71 kB)
Requirement already satisfied: numpy>=1.23 in c:\python311\lib\site-packages
(from matplotlib) (2.2.3)
Requirement already satisfied: packaging>=20.0 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in c:\python311\lib\site-packages (from
matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\python311\lib\site-
packages (from matplotlib) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\student\appdata\roaming\python\python311\site-packages (from
matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in
c:\users\student\appdata\roaming\python\python311\site-packages (from python-
dateutil>=2.7->matplotlib) (1.17.0)
Installing collected packages: kiwisolver, fonttools, cycler, contourpy,
matplotlib

ERROR: Could not install packages due to an OSError: [WinError 5] Access is
denied: 'C:\\Python311\\share'
Consider using the `--user` option or check the permissions.
```

```
[notice] A new release of pip is available: 23.1.2 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

[15]:
```
#import matplotlib below

import matplotlib.pyplot as plt
import random
```

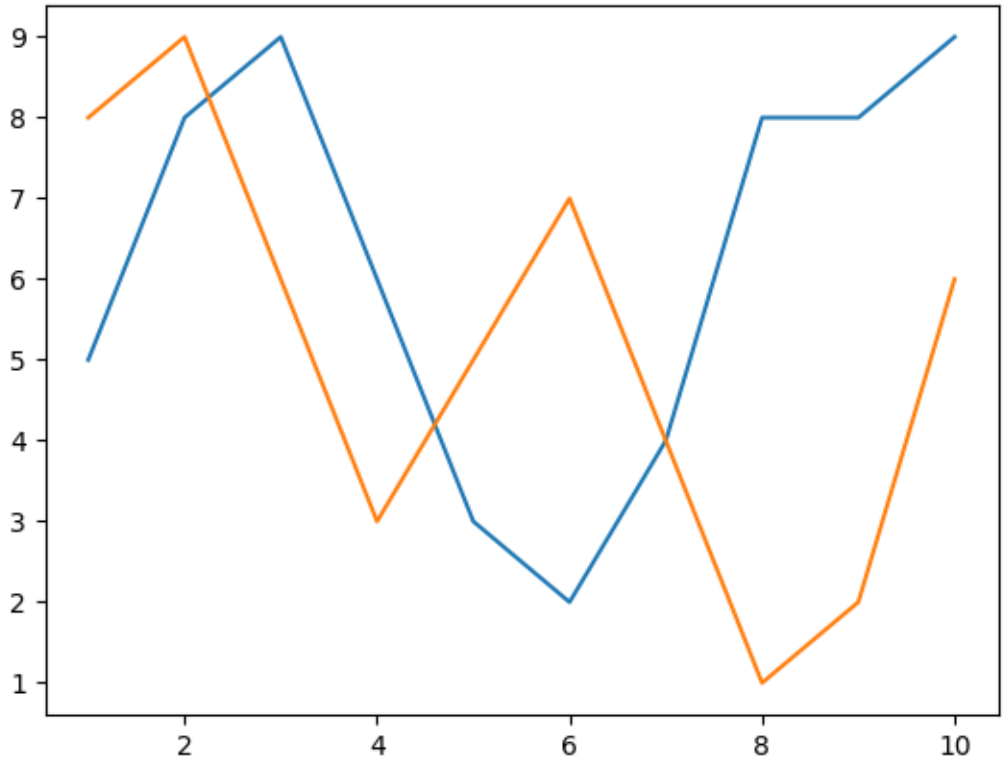
[3]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]

# write a code to display the line chart of above x & y

plt.plot(x,y)
plt.show()
```



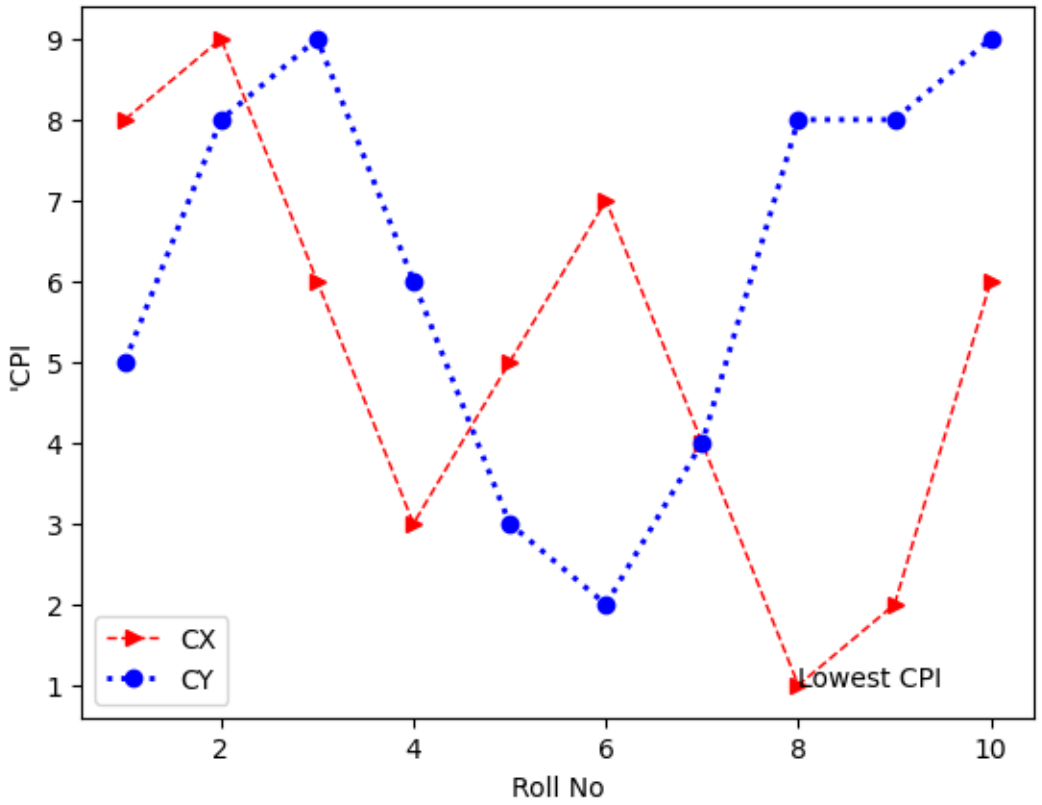
[4]:
```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarks = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]

# write a code to display two lines in a line chart (data given above)
```

```
plt.plot(x,cxMarks)
plt.plot(x,cyMarks)
```

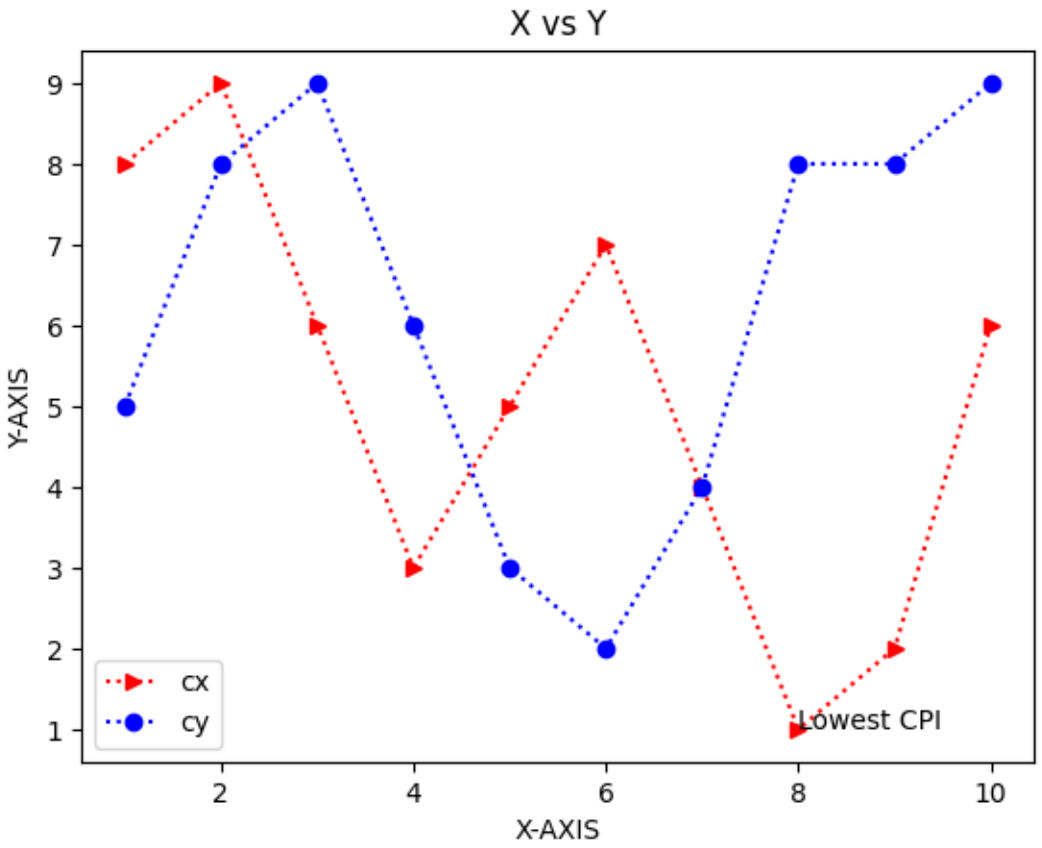
[4]: [<matplotlib.lines.Line2D at 0x1f28a0adf10>]



[13]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]


# write a code to generate below graph
```

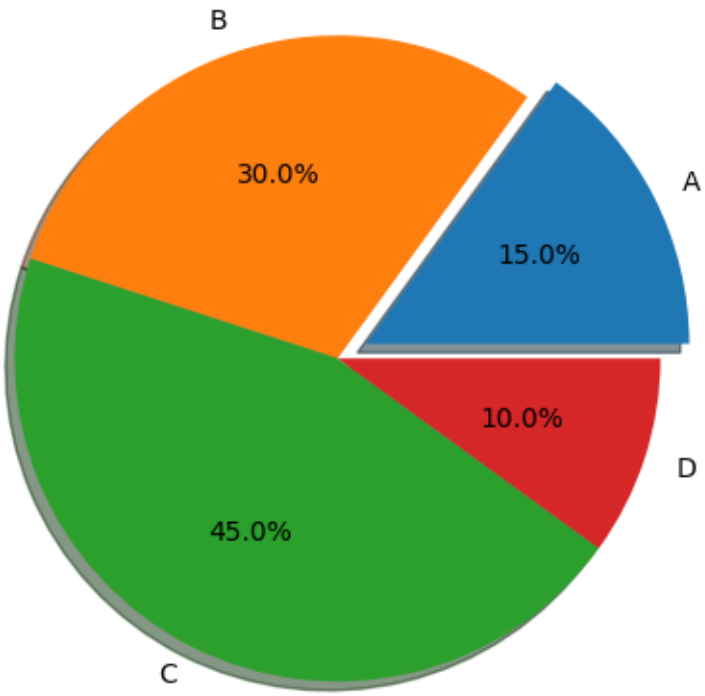


[11]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]


# write a code to generate below graph
plt.plot(x,cxMarks,c="r",marker=">",ls=":",label="cx")
plt.plot(x,cyMarks,c="b",marker="o",ls=":",label="cy")
plt.xlabel("X-AXIS")
plt.ylabel("Y-AXIS")
plt.title("X vs Y")
plt.annotate("Lowest CPI", xy=[8,1])
plt.legend()
plt.show()
```



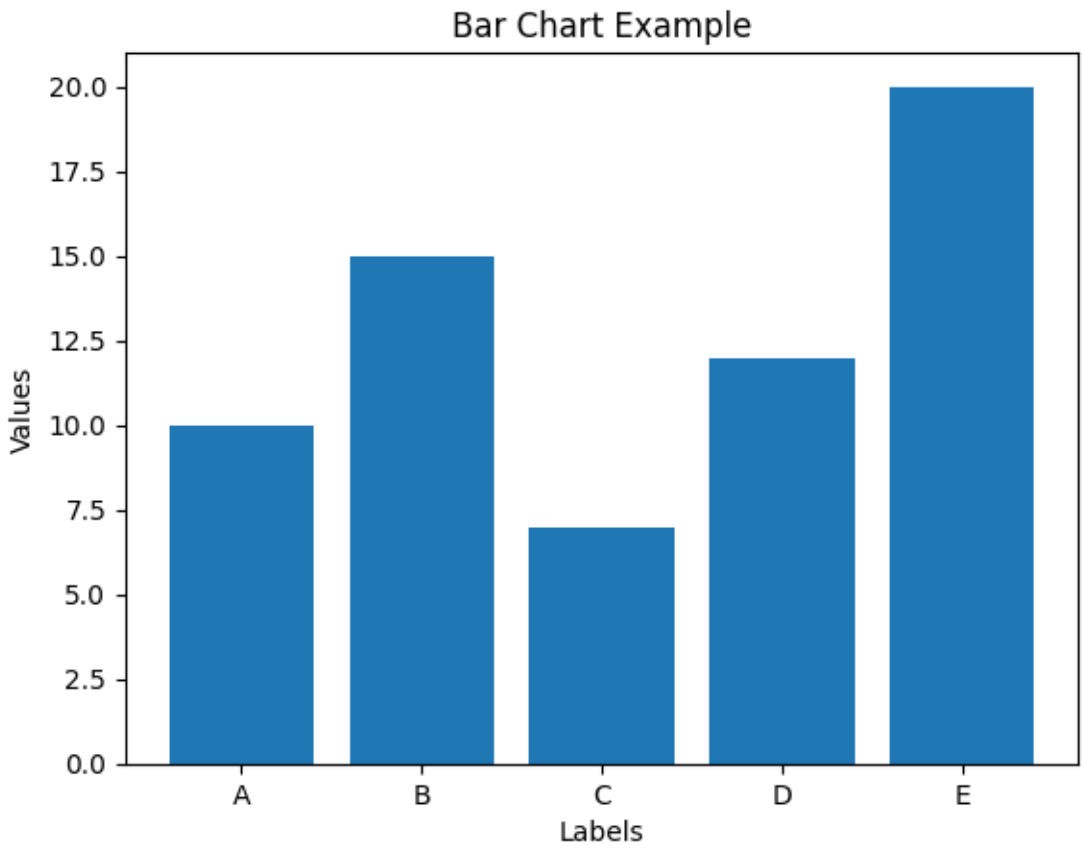
0.0.1 04) WAP to demonstrate the use of Pie chart.

[12]:
```
labels = ['A', 'B', 'C', 'D']
sizes = [15, 30, 45, 10]
explode = [0.1, 0, 0, 0]

plt.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True)
plt.axis('equal')
plt.show()
```



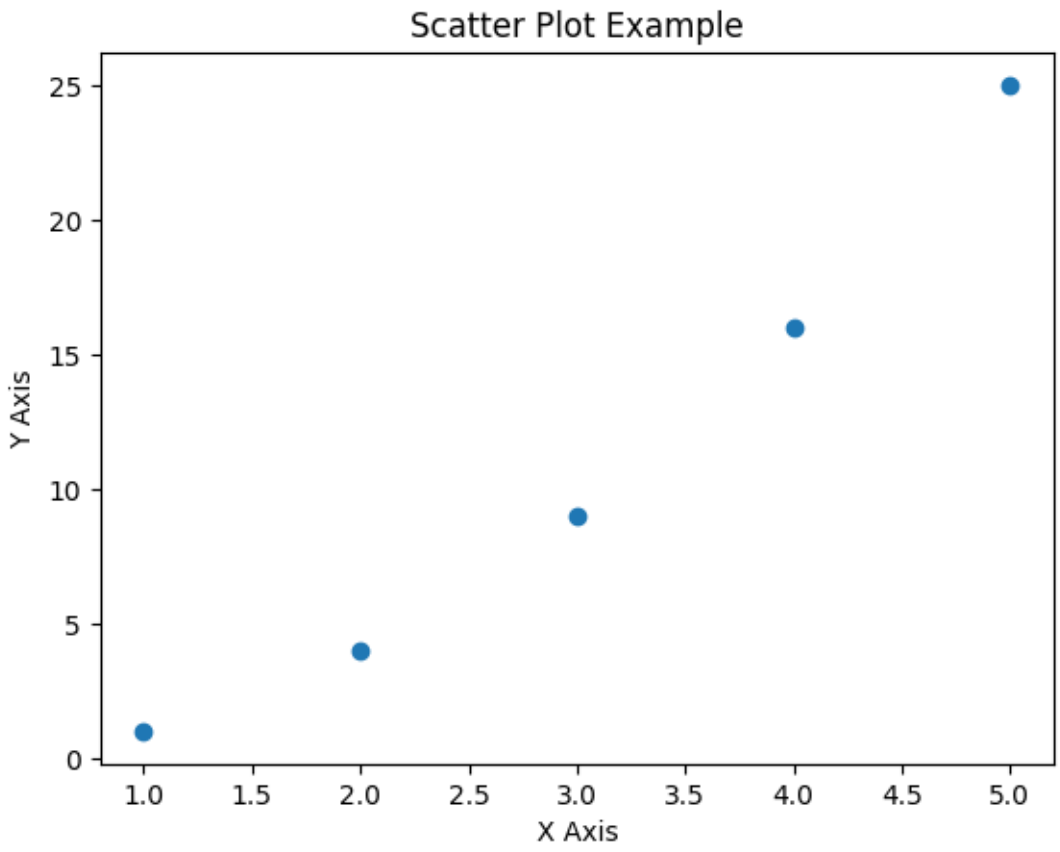
0.0.2 05) WAP to demonstrate the use of Bar chart.

[13]:
```
labels = ['A', 'B', 'C', 'D', 'E']
values = [10, 15, 7, 12, 20]

plt.bar(labels, values)

plt.title('Bar Chart Example')
plt.xlabel('Labels')
plt.ylabel('Values')

plt.show()
```



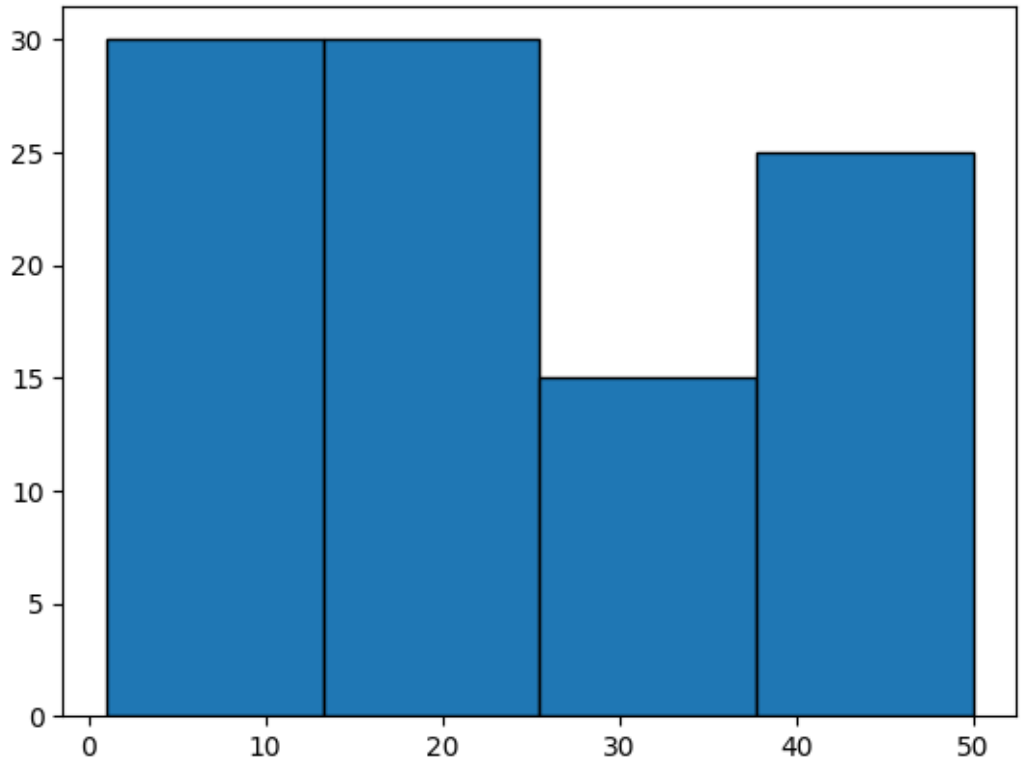
0.0.3 06) WAP to demonstrate the use of Scatter Plot.

[14]:
```
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

plt.scatter(x, y)

plt.title('Scatter Plot Example')
plt.xlabel('X Axis')
plt.ylabel('Y Axis')

plt.show()
```



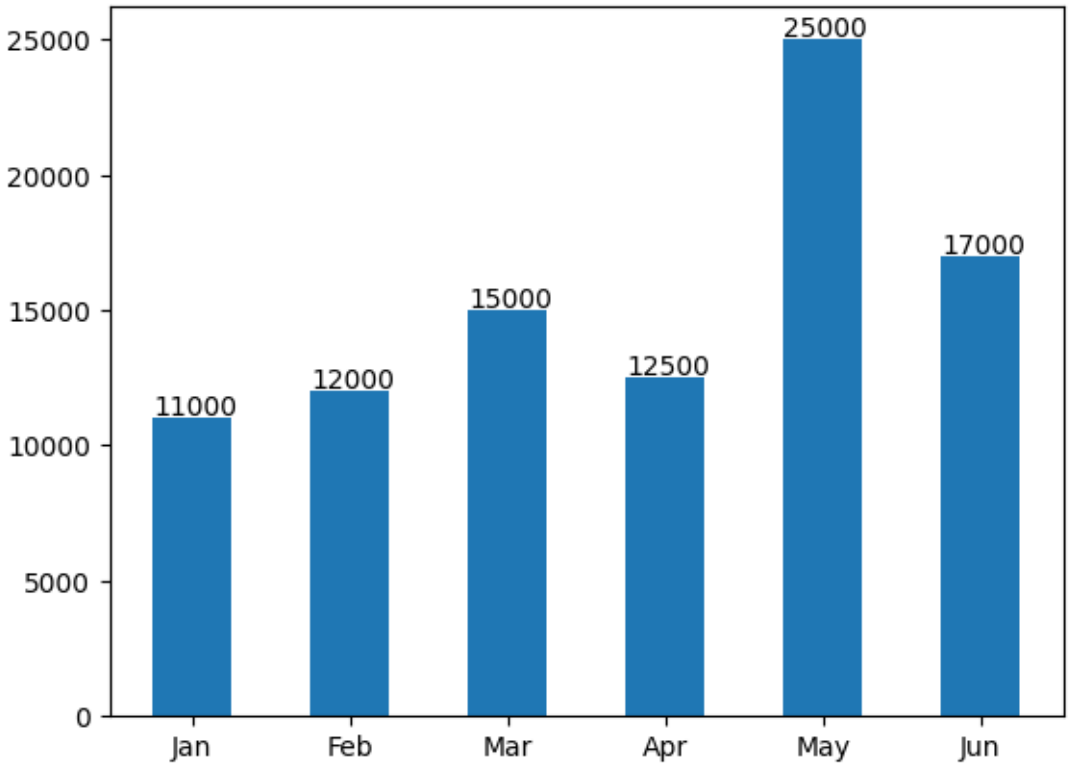
0.0.4 07) WAP to demonstrate the use of Histogram.

[17]:
```
random.seed(5)
age = [random.randint(1,50) for i in range(100)]
plt.hist(age, edgecolor="k", bins=6,histtype="bar")
plt.show()
```



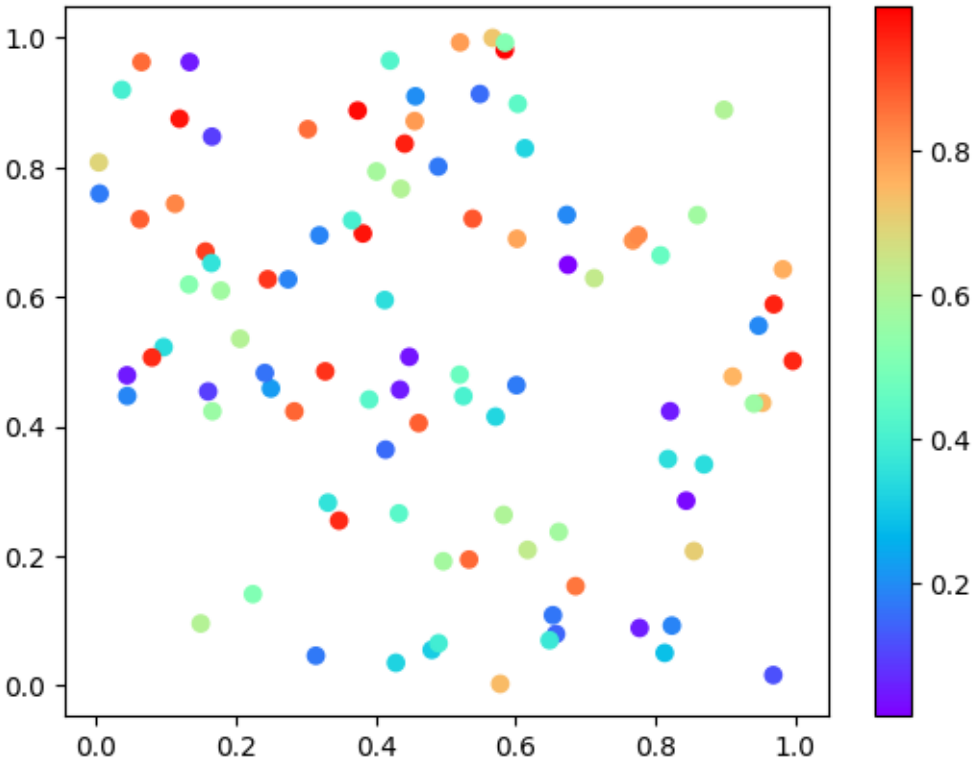
0.0.5 08) WAP to display the value of each bar in a bar chart using Matplotlib.

[18]:
```
mon = ["Jan", "Feb", "Mar", "Apr", "May", "Jun"]
visiters = [11000, 12000, 15000, 12500, 25000, 17000]
bars = plt.bar(mon, visiters, width=0.5)
for i in bars:
    yc = i.get_height()
    plt.text(i.get_x(), yc+100, f"{yc}")
plt.show()
```



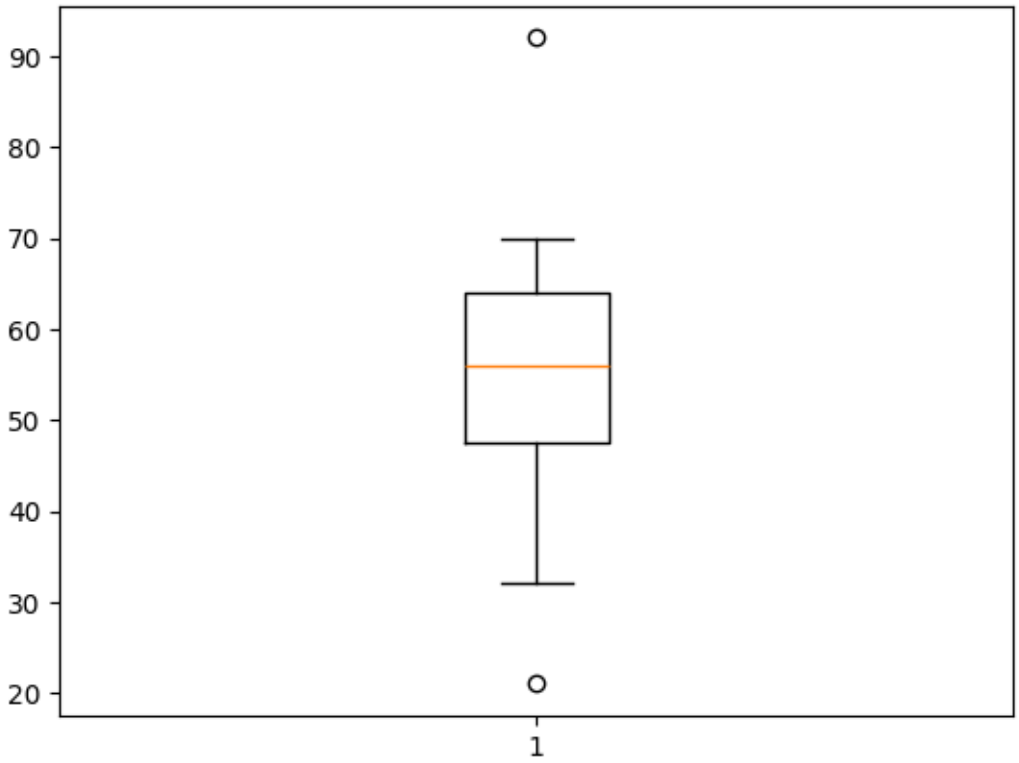
0.0.6 09) WAP create a Scatter Plot with several colors in Matplotlib?

[21]:
```
random.seed(10)
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="rainbow")
plt.colorbar()
plt.show()
```



0.0.7 10) WAP to create a Box Plot.

[32]:
```
# random.seed(10)
# x = [random.random() for i in range(5000)]
# y = [random.random() for i in range(100)]
plt.boxplot([50,45,52,63,70,21,56,68,54,57,35,62,65,92,32])

# plt.boxplot(x,vert=True,widths=0.3)
plt.show()
```


90
80
70
60
50
40
30
20
1

[ ]:

lab_13

March 11, 2025

Python Programming - 2301CS404

Lab - 13

OM BHUT | 23010101033 | 122

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.

[11]:
```
class Student:
    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade

    def display_info(self):
        print("Name:",self.name)
        print("Age:",self.age)
        print("Grade:",self.grade)

student1 = Student("Yash", 20, "O")
student1.display_info()
```

```
Name: Yash
Age: 20
Grade: O
```

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.

[12]:
```
class BankAccount:
    def __init__(self, Account_No, User_Name, Email, Account_Type,␣
 ↪Account_Balance):
        self.Account_No = Account_No
        self.User_Name = User_Name
```

```
        self.Email = Email
        self.Account_Type = Account_Type
        self.Account_Balance = Account_Balance

    def GetAccountDetails(self):
        return self.Account_No, self.User_Name, self.Email, self.Account_Type,␣
 ↪self.Account_Balance

    def DisplayAccountDetails(self):
        print("Account Number:", self.Account_No)
        print("User Name:", self.User_Name)
        print("Email:", self.Email)
        print("Account Type:", self.Account_Type)
        print("Account Balance:", self.Account_Balance)

account1 = BankAccount(1234567890, "Yash", "yash77@gmail.com", "Savings",␣
 ↪70000000)

account_details = account1.GetAccountDetails()

account1.DisplayAccountDetails()
```

```
Account Number: 1234567890
User Name: Yash
Email: yash77@gmail.com
Account Type: Savings
Account Balance: 70000000
```

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

[13]:
```
import math
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi* (self.radius ** 2)

    def perimeter(self):
        return 2 * math.pi * self.radius

circle1 = Circle(10)

print("Area:", circle1.area())
print("Perimeter:", circle1.perimeter())
```

```
Area: 314.1592653589793
```

```
Perimeter: 62.83185307179586
```

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.

[14]:
```python
class Employee:
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary

    def update_info(self, name=None, age=None, salary=None):
        if name:
            self.name = name
        if age:
            self.age = age
        if salary:
            self.salary = salary

    def display_info(self):
        print("Name:", self.name)
        print("Age:", self.age)
        print("Salary:", self.salary)

employee1 = Employee("Yash Kakadiya", 20, 70000)

employee1.display_info()

employee1.update_info(salary=100000)

employee1.display_info()

employee1.update_info(age=21)

employee1.display_info()
```

```
Name: Yash Kakadiya
Age: 20
Salary: 70000
Name: Yash Kakadiya
Age: 20
Salary: 100000
Name: Yash Kakadiya
Age: 21
Salary: 100000
```

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

[15]:
```
class BankAccount:
    def __init__(self, account_number, initial_balance):
        self.account_number = account_number
        self.balance = initial_balance

    def deposit(self, amount):
        self.balance += amount
        print("Deposited:", amount)
        self.check_balance()

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print("Withdrawn:", amount)
            self.check_balance()
        else:
            print("Insufficient balance.")

    def check_balance(self):
        print("Current Balance:", self.balance)

account1 = BankAccount(1234567890, 10000)

account1.deposit(5000)

account1.withdraw(2000)

account1.withdraw(3000)

account1.check_balance()
```

```
Deposited: 5000
Current Balance: 15000
Withdrawn: 2000
Current Balance: 13000
Withdrawn: 3000
Current Balance: 10000
Current Balance: 10000
```

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.

[16]:
```
class Product:
    def __init__(self, item_name, price, quantity):
        self.item_name = item_name
        self.price = price
        self.quantity = quantity

    def add_item(self, quantity):
        self.quantity += quantity
        print("Item added:", self.item_name)
        self.display_info()

    def remove_item(self, quantity):
        if quantity <= self.quantity:
            self.quantity -= quantity
            print("Item removed:", self.item_name)
            self.display_info()
        else:
            print("Not enough items in stock.")

    def update_price(self, new_price):
        self.price = new_price
        print("Price updated:", self.item_name)
        self.display_info()

    def display_info(self):
        print("Item Name:", self.item_name)
        print("Price:", self.price)
        print("Quantity:", self.quantity)

product1 = Product("Laptop", 10000, 5)

product1.add_item(3)

product1.remove_item(2)

product1.update_price(12000)

product1.display_info()
```

```
Item added: Laptop
Item Name: Laptop
Price: 10000
Quantity: 8
Item removed: Laptop
Item Name: Laptop
```

```
Price: 10000
Quantity: 6
Price updated: Laptop
Item Name: Laptop
Price: 12000
Quantity: 6
Item Name: Laptop
Price: 12000
Quantity: 6
```

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

[22]:
```python
class MyClass:
    def __init__(self, attribute1, attribute2):
        self.attribute1 = attribute1
        self.attribute2 = attribute2

    def display_attributes(self):
        print(f'Attribute 1: {self.attribute1}')
        print(f'Attribute 2: {self.attribute2}')

my_object = MyClass('Hello', 'World')

my_object.display_attributes()
```

```
Attribute 1: Hello
Attribute 2: World
```

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.

[21]:
```python
class StudentKit:
    PrincipalName='Mr.ABC'
    def __init__(self,name):
        self.name=name
        self.attendance=0
        self.certificate=0

    def attendance_method(self,days):
        self.attendance=days
        print(f'{self.name} has attended {self.attendance} days in class.')
```

```
    def certificate_method(self,days):
        self.certificate=days
        print(f'{self.name} has obtained {self.certificate} days of certificate.
 ↪')

    def display_info(self):
        print(f'Student Name: {self.name}\nPrincipal Name: {self.
 ↪PrincipalName}\nAttendance: {self.attendance} days\nCertificate: {self.
 ↪certificate} days')

student1=StudentKit('Yash Kakaiya')

student1.attendance_method(25)

student1.certificate_method(30)

student1.display_info()
```

```
Yash Kakaiya has attended 25 days in class.
Yash Kakaiya has obtained 30 days of certificate.
Student Name: Yash Kakaiya
Principal Name: Mr.ABC
Attendance: 25 days
Certificate: 30 days
```

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

[20]:
```
class Time:
    def __init__(self, hour, minute):
        self.hour = hour
        self.minute = minute

    def add_time(self, other_time):
        new_minute = self.minute + other_time.minute
        new_hour = self.hour + other_time.hour + new_minute // 60
        new_minute = new_minute % 60
        return Time(new_hour, new_minute)

    def display_time(self):
        print(f"{self.hour:02d}:{self.minute:02d}")

time1 = Time(10, 30)

time2 = Time(5, 45)
```

```
sum_time = time1.add_time(time2)

sum_time.display_time()
```

```
16:15
```

lab_13_2

March 11, 2025

Python Programming - 2301CS404

Lab - 13

OM BHUT | 23010101033 | 122

0.1 Continued..

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

[10]:
```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width


def calculate_area(obj):
    area= obj.length*obj.width
    print(f"Area calculated using object is: {area}")
r1 = Rectangle(5, 10)

calculate_area(r1)
```

```
Area calculated using object is: 50
```

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().

[8]:
```
class Square:
    def __init__(self, side):
        self.side = side

    def area(self):
        area_value = self.side ** 2
        self.output(area_value)

    def output(self, area_value):
        print(f"Area of square with side {self.side} is: {area_value}")
```

```
square = Square(4)
square.area()
```

```
Area of square with side 4 is: 16
```

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.

[ ]:
```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

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

    def output(self, area_value):
        print(f"Area of rectangle with length {self.length} and width {self.
 ↪width} is: {area_value}")

    @classmethod
    def create_rectangle(cls, length, width):
        if length == width:
            print("THIS IS SQUARE.")
            return

        return cls(length, width)


rect1 = Rectangle.create_rectangle(5, 3)
if rect1:
    rect1.area()

rect2 = Rectangle.create_rectangle(4, 4)
```

```
Area of rectangle with length 5 and width 3 is: 15
THIS IS SQUARE.
```

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.

[13]:
```
class Square:
    def __init__(self, side):
        self.__side = side

    def get_side(self):
        return self.__side

    def set_side(self, side):
        self.__side = side

    def calculate_area(self):
        return self.__side ** 2


sq = Square(5)
print(f"Side of square: {sq.get_side()}")
print(f"Area of square: {sq.calculate_area()}")

sq.set_side(7)
print(f"New side of square: {sq.get_side()}")
print(f"New area of square: {sq.calculate_area()}")
```

```
Side of square: 5
Area of square: 25
New side of square: 7
New area of square: 49
```

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().

[2]:
```
class Profit:
    def __init__(self):
        self.profit = 0

    def getProfit(self):
        self.profit = float(input("Enter profit amount: "))
        return self.profit
```

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

    def getLoss(self):
        self.loss = float(input("Enter loss amount: "))
        return self.loss

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

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

    def printBalance(self):
        print(f"Profit: ${self.profit}")
        print(f"Loss: ${self.loss}")
        print(f"Balance: ${self.balance}")

balance_sheet = BalanceSheet()
balance_sheet.getProfit()
balance_sheet.getLoss()
balance_sheet.getBalance()
balance_sheet.printBalance()
```

```
Profit: $50.0
Loss: $25.0
Balance: $25.0
```

0.1.12 15) WAP to demonstrate all types of inheritance.

[19]:
```
class Parent:
    def __init__(self):
        self.parent_attribute = "This is from parent"

    def parent_method(self):
        print("This is parent method")

class Child(Parent):
    def __init__(self):
        super().__init__()
        self.child_attribute = "This is from child"
```

```
    def child_method(self):
        print("This is child method")

class Father:
    def father_method(self):
        print("This is father method")

class Mother:
    def mother_method(self):
        print("This is mother method")

class Child2(Father, Mother):
    def child_method(self):
        print("This is child2 method")

class Grandparent:
    def grandparent_method(self):
        print("This is grandparent method")

class Parent2(Grandparent):
    def parent_method(self):
        print("This is parent2 method")

class Child3(Parent2):
    def child_method(self):
        print("This is child3 method")

class Parent3:
    def parent_method(self):
        print("This is parent3 method")

class ChildA(Parent3):
    def child_a_method(self):
        print("This is childA method")

class ChildB(Parent3):
    def child_b_method(self):
        print("This is childB method")


class Base:
    def base_method(self):
        print("This is base method")

class Derived1(Base):
    def derived1_method(self):
```

```
        print("This is derived1 method")

class Derived2(Base):
    def derived2_method(self):
        print("This is derived2 method")

class DerivedOfDerived(Derived1, Derived2):
    def derived_of_derived_method(self):
        print("This is derived of derived method")


print("\nSingle Inheritance:")
child = Child()
child.parent_method()
child.child_method()

print("\nMultiple Inheritance:")
child2 = Child2()
child2.father_method()
child2.mother_method()
child2.child_method()

print("\nMultilevel Inheritance:")
child3 = Child3()
child3.grandparent_method()
child3.parent_method()
child3.child_method()

print("\nHierarchical Inheritance:")
childA = ChildA()
childB = ChildB()
childA.parent_method()
childA.child_a_method()
childB.parent_method()
childB.child_b_method()

print("\nHybrid Inheritance:")
derived_of_derived = DerivedOfDerived()
derived_of_derived.base_method()
derived_of_derived.derived1_method()
derived_of_derived.derived2_method()
derived_of_derived.derived_of_derived_method()
```

```
Single Inheritance:
This is parent method
This is child method
```

```
Multiple Inheritance:
This is father method
This is mother method
This is child2 method

Multilevel Inheritance:
This is grandparent method
This is parent2 method
This is child3 method

Hierarchical Inheritance:
This is parent3 method
This is childA method
This is parent3 method
This is childB method

Hybrid Inheritance:
This is base method
This is derived1 method
This is derived2 method
This is derived of derived method
```

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.

[23]:
```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

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

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

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

```
# Create an employee
employee = Employee("John Doe", 30, 70000)
employee.display_info()
```

```
Name: John Doe, Age: 30
Salary:  70000
```

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.

[ ]:
```
class Shape:
    def draw(self):
        raise NotImplementedError("Subclass must implement abstract method")

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
```

[7]:
```
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
```

[ ]: