3su9m6tyv

March 10, 2025

Python Programming - 2301CS404

Lab - 1

Raj Dedakiya | 23010101056 | 325

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().

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

c = 10
d = 20
print(c+d)
```

```
Enter number 1:  2
Enter number 2:  6
8
30
```

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

[4]:
```
a = 10
b = 10.5
c = 'Hello'
d = True
print(type(a))
print(type(b))
print(type(c))
print(type(d))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
```

0.0.4 04) WAP to calculate simple interest.

[6]:
```
principle = float(input('Enter the principle amount: '))
time = float(input('Enter the time period: '))
rate = float(input('Enter the rate of interest: '))
simpleInterest = (principle*time*rate)/100

print('Simple interest is:', simpleInterest)
```

```
Enter the principle amount:  1000
Enter the time period:  2
Enter the rate of interest:  5
Simple interest is: 100.0
```

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

[8]:
```
import math
r = float(input('Enter the radius of the circle: '))

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

print('Perimeter of the circle is:', perimeter)
print('Area of the circle is:', area)
```

```
Enter the radius of the circle:  7
Perimeter of the circle is: 43.982297150257104
Area of the circle is: 153.93804002589985
```

0.0.6 06) WAP to calculate area of a triangle.

[10]:
```
base = float(input('Enter the base of the triangle: '))
height = float(input('Enter the height of the triangle: '))
area = 1/2 * (base* height)

print('Area of the triangle is:', area)
```

```
Enter the base of the triangle:  6
Enter the height of the triangle:  8
Area of the triangle is: 24.0
```

0.0.7 07) WAP to compute quotient and remainder.

[12]:
```
a = int(input('Enter number 1: '))
b = int(input('Enter number 2: '))

print(f'Quotient is {a/b}')
print(f'Remainder is {a%b}')
```

```
Enter number 1:  5
Enter number 2:  3
Quotient is 1.6666666666666667
Remainder is 2
```

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

[14]:
```
temp = input('Enter "c" for celcius or "f" for fahrenheit: ')
if temp == c:
    c = float(input('Enter the temperature in celcius: '))
    f = c * (9/5) + 32
    print(f'{c} degrees celcius is equal to {f} degrees fahrenheit')
else:
    f = float(input('Enter the temperature in fahrenheit: '))
    c = (5/9) * (f-32)
    print(f'{f} degrees fahrenheit is equal to {c} degrees celcius')
```

```
Enter "c" for celcius or "f" for fahrenheit:  c
Enter the temperature in fahrenheit:  32
32.0 degrees fahrenheit is equal to 0.0 degrees celcius
```

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

[16]:
```
x1, x2 = 1, 4
y1, y2 = 2, 6

distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
print(f'Distance between 2 points in 2-D space is {distance}')
```

```
Distance between 2 points in 2-D space is 5.0
```

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

[18]:
```
n = int(input('Enter any natural number: '))
sum = 0
for i in range(n+1):
    sum += i

print(f'Sum of first {n} natural numbers is {sum}')
```

```
Enter any natural number:  5
Sum of first 5 natural numbers is 15
```

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

[20]:
```
n = int(input('Enter any natural number: '))
sum = 0
for i in range(n+1):
    sum += (i*i)

print(f'Sum of square of first {n} natural numbers is {sum}')
```

```
Enter any natural number:  8
Sum of square of first 8 natural numbers is 204
```

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

[1]:
```
first = input('Enter your first name: ')
last = input('Enter your last name: ')
name = first + ' ' + last
print(f'Your full name is {name}')
```

```
Enter your first name: abc
Enter your last name: xyz
Your full name is abc xyz
```

0.0.13 13) WAP to swap two numbers.

[24]:
```
a, b = 5, 10
print(f'a is {a} and b is {b}')
a, b = b, a
print(f'a is {a} and b is {b}')
```

```
a is 5 and b is 10
a is 10 and b is 5
```

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

[26]:
```
distance = float(input('Enter distance in kilometer: '))
print(f'The distance in meter is {distance * 1000} meters')
print(f'The distance in feet is {distance * 3280.84} feet')
print(f'The distance in inches is {distance * 39370.08} inches')
print(f'The distance in centimeters is {distance * 100000} centimeters')
```

```
Enter distance in kilometer:  8
The distance in meter is 8000.0 meters
The distance in feet is 26246.72 feet
```

```
The distance in inches is 314960.64 inches
The distance in centimeters is 800000.0 centimeters
```

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.

[28]:
```
day = int(input('Enter the day: '))
month = int(input('Enter the month: '))
year = int(input('Enter the year: '))

print(f'The date is {day}-{month}-{year}')
```

```
Enter the day:  5
Enter the month:  6
Enter the year:  2025
The date is 5-6-2025
```

[ ]:
```

```

hyhoinb8c

March 10, 2025

Python Programming - 2301CS404

Lab - 2

Raj Dedakiya | 23010101056 | 325

1 if..else..

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

[1]:
```
n=int(input("Enter a number:"));
if(n>=0):
    print("Number is positive!!");
else:
    print("Number is negative!!");
```

```
Enter a number: -8
Number is negative!!
```

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

[3]:
```
n=int(input("Enter a number:"));
if(n%2==0):
    print("Number is even!!");
else:
    print("Number is odd!!");
```

```
Enter a number: 8
Number is even!!
```

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

[7]:
```
#if..else..
n=int(input("Enter a number(n):"));
m=int(input("Enter a number(m):"));
if(n>m):
    print("Number n is larger!");
else:
```

```
    print("Number m is larger!");

#Ternary operator
n=int(input("Enter a number(n):"));
m=int(input("Enter a number(m):"));
a="n is larger!!" if n>m else "m is larger!!";
print(a);
```

```
Enter a number(n): 8
Enter a number(m): 9
Number m is larger!
Enter a number(n): 8
Enter a number(m): 9
m is larger!!
```

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

[9]:
```
n1=int(input("Enter a number(n1):"));
n2=int(input("Enter a number(n2):"));
n3=int(input("Enter a number(n3):"));
if(n1>=n2):
    if(n1>=n3):
        print(f"{n1} is largest!!");
    else:
        print(f"{n3} is largest!!");
else:
    if(n2>=n3):
        print(f"{n2} is largest!!");
    else:
        print(f"{n3} is largest!!");
```

```
Enter a number(n1): 7
Enter a number(n2): 8
Enter a number(n3): 9
9 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]

[11]:
```
year = int(input('Enter the year: '));

if ((year%4 == 0 and year%100 != 0) or year%400 == 0):
    print(f'{year} is Leap year')
else:
    print(f'{year} is not Leap year');
```

```
Enter the year:  2024
2024 is Leap year
```

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

[13]:
```
day = int(input('Enter the number of day: '))

match day:
    case 1:
        print('Today is monday')
    case 2:
        print('Today is tuesday')
    case 3:
        print('Today is wednesday')
    case 4:
        print('Today is thusday')
    case 5:
        print('Today is friday')
    case 6:
        print('Today is saturday')
    case 7:
        print('Today is sunday')
    case _:
        print('Invalid number')
```

```
Enter the number of day:  7
Today is sunday
```

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

[15]:
```
n=int(input("Enter a number(n):"));
m=int(input("Enter a number(m):"));
p=str(input("Enter a operator:"));
if(p=="+"):
    print("Addition:",n+m);
elif(p=="-"):
    print("Subtraction:",n-m);
elif(p=="*"):
    print("Multiplication:",n*m);
elif(p=="/"):
    print("Division:",n/m);
```

```
Enter a number(n): 7
Enter a number(m): 5
Enter a operator: /
Division: 1.4
```

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

[19]:
```
s1 = int(input('Enter marks of subject1: '));
s2 = int(input('Enter marks of subject2: '))
s3 = int(input('Enter marks of subject3: '))
s4 = int(input('Enter marks of subject4: '))
s5 = int(input('Enter marks of subject5: '))

result=(s1+s2+s3+s4+s5)/5;
print(result);

if(result<35):
    print("Class:Fail");
elif(result>=35 and result<45):
    print("Class Pass")
elif(result>=45 and result<60):
    print("Class:Second");
elif(result>=60 and result<70):
    print("Class:First");
elif(result>=70):
    print("Class:Distinction")
```

```
Enter marks of subject1:  55
Enter marks of subject2:  66
Enter marks of subject3:  88
Enter marks of subject4:  77
Enter marks of subject5:  99
77.0
Class:Distinction
```

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.

[4]:
```
a = float(input("Enter the first side: "))
b = float(input("Enter the second side: "))
c = float(input("Enter the third side: "))

if a + b > c and a + c > b and b + c > a:
    if a**2 + b**2 == c**2 or a**2 + c**2 == b**2 or b**2 + c**2 == a**2:
        print("The triangle is Right-angled.")
    if a == b == c:
        print("The triangle is Equilateral.")
    elif a == b or b == c or a == c:
        print("The triangle is Isosceles.")
```

```
    else:
        print("The triangle is Scalene.")
else:
    print("The given sides do not form a valid triangle.")
```

```
Enter the first side:  7
Enter the second side:  24
Enter the third side:  25
The triangle is Right-angled.
The triangle is Scalene.
```

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

[23]:
```
a = int(input('Enter the first number: '))
b = int(input('Enter the second number: '))
c = int(input('Enter the third number: '))

if (a > b and a < c):
    print('First number is second largest');
elif (b > a and b < c):
    print('Second number is second largest');
else:
    print('Third number is second largest');
```

```
Enter the first number:  7
Enter the second number:  8
Enter the third number:  9
Second number 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

[25]:
```
units = int(input('Enter the number of units: '))
case = 0
if units > 0 and units < 50:
    case = 1
elif units >= 50 and units < 100:
    case = 2
elif units >= 100 and units < 200:
    case = 3
elif units > 200:
    case = 4
rate = 0
```

```
match case:
    case 1:
        rate = units * 2.6
        print(f'Your electricity bill is {rate}')
    case 2:
        rate = (50 * 2.6) + ((units - 50) * 3.25)
        print(f'Your electricity bill is {rate}')
    case 3:
        rate = (50 * 2.6) + (50 * 3.25) + ((units - 100) * 5.26)
        print(f'Your electricity bill is {rate}')
    case 4:
        rate = (50 * 2.6) + (50 * 3.25) + (100 * 5.26) + ((units - 200) * 8.45)
        print(f'Your electricity bill is {rate}')
```

```
Enter the number of units:  50
Your electricity bill is 130.0
```

[ ]:

xvzhud9kx

March 10, 2025

Python Programming - 2301CS404

Lab - 3

Raj Dedakiya | 23010101056 | 325

1 for and while loop

1.0.1 01) WAP to print 1 to 10.

[6]:
```
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.

[12]:
```
n=int(input("enter a number:"))
for i in range (1,n+1):
    print(i)
```

```
enter a number: 10
1
2
3
4
5
6
7
8
```

```
9
10
```

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

[18]:
```
n=int(input("enter a number:"))
for i in range (1,n+1,2):
    print(i)
```

```
enter a 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.

[22]:
```
a=int(input("enter a number:"))
b=int(input("enter a number:"))
for i in range(a,b):
    if(i%2==0 and i%3!=0):
        print(i)
```

```
enter a number: 8
enter a number: 80
8
10
14
16
20
22
26
28
32
34
38
40
44
46
50
52
56
58
62
64
68
```

```
70
74
76
```

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

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

```
Enter a number: 15
1
3
6
10
15
21
28
36
45
55
66
78
91
105
120
```

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

[50]:
```
n=int(input("Enter a number:"))
sum=0;
for i in range(1,n+1):
    sum+=(i*i);
print(sum)
```

```
Enter a number: 5
55
```

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

[52]:
```
n=int(input("Enter a number:"));
sum=0;
for i in range (1,n+1):
    if i%2==0:
        sum-=i
    else:
```

```
        sum+=i
print(sum)
```

```
Enter a number: 4
-2
```

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

[54]:
```
n=int(input("Enter a number:"));
for i in range(1,11):
    print(n*i);
```

```
Enter a number: 5
5
10
15
20
25
30
35
40
45
50
```

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

[64]:
```
n=int(input("Enter a number:"));
fac=1;
for i in range(1,n+1):
        fac*=i;
print(fac)
```

```
Enter a number: 5
120
```

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

[60]:
```
n=int(input("Enter a number:"));
fac=0;
for i in range(1,n+1):
    if(n%i==0):
        print(i)
```

```
Enter a number: 20
1
2
4
5
```

```
10
20
```

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

[82]:
```
n=int(input("Enter a number:"));
for i in range(2,n):
    if(n%i==0):
        print("Not prime!!")
        break;
else:
    print("Prime!!")
```

```
Enter a number: 9
Not prime!!
```

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

[98]:
```
#Method-1
n = int(input('Enter the number: '))
sum = 0
while n > 0:
    sum += n % 10
    n = n // 10
print(f"The sum of the digits is {sum}")

#Method-2
n = input('Enter the number: ')
sum=0;
for i in n:
    sum+=int(i)
print(f"The sum of the digits is {sum}")
```

```
Enter the number:  596
The sum of the digits is 20
Enter the number:  596
The sum of the digits is 20
```

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

[86]:
```
n = input('Enter the number: ')

if n == n[::-1]:
    print('The number is a palindrome')
else:
    print('The number is not a palindrome')
```

```
Enter the number:  16461
The number is a palindrome
```

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

[92]:
```python
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))

while b != 0:
    a, b = b, a % b

print(f"The GCD is: {a}")
```

```
Enter the first number:  20
Enter the second number:  15
The GCD is: 5
```

ysxndfxf5

March 10, 2025

Python Programming - 2101CS405

Lab - 4

Raj Dedakiya | 23010101056 | 325

1 String

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

[1]:
```
str = "HelloolleH"

if str == str[::-1] :
    print("palindrom")
else :
    print("not palindrom")
```

```
palindrom
```

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

[17]:
```
str = "Hello World"
out_str = ''
words = str.split(' ')
for i in words:
    out_str += i[::-1]+' '
out_str = out_str[0:len(out_str)]
print(out_str)
```

```
olleH dlroW
```

1.0.3 03) WAP to remove ith character from given string

[18]:
```
index = 1;
str = "Hello Wolrd"

str = str[0:index]+str[index+1:]
print(str)
```

[18]:
```
'Hllo Wolrd'
```

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

[3]:
```
str = input("Enter a string :- ")
length = 0

for i in str:
    length +=1
print(length)
```

```
Enter a string :- harsh
5
```

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

[5]:
```
str = "Hello Wolrd abc asd dfdf sada"
words = str.split(' ')

for i in words:
    if len(i)%2 == 0 :
        print(i)
```

```
dfdf
sada
```

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

[43]:
```
string = "Hello Wolrd abc asd dfdf sada"
vovels = {'a', 'e', 'i', 'o', 'u'}
vovle_count = 0
for i in string:
    if(i in vovels):
        vovle_count += 1
print(vovle_count)
```

```
7
```

1.0.7 07) WAP to convert given array to string.

[1]:
```
arr = [1,2,3,4,5]
out_str = ''
for i in arr:
    out_str += str(i)+" "
print(out_str)
print(type(arr))
```

```
1 2 3 4 5
<class 'list'>
```

1.0.8 01) WAP to find out duplicate characters in given string.

[49]:
```
string = "Hello Wolrd abc asd dfdf sada"
unique_char = set()

for i in string:
    if i == ' ': continue
    if i in unique_char:
        print(i, end = ' ')
    else:
        unique_char.add(i)
```

```
l o l a d d d f s a d a
```

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

[59]:
```
string = "Hello Wolrd abc asd dfdf sada"
words = string.split(' ')
string = ''
for i in words:
    if(ord(i[0]) >= 97 and ord(i[0]) <= 122):
        i = chr(ord(i[0])-32)+i[1:]
    if(ord(i[-1]) >= 97 and ord(i[-1]) < 122):
        i = i[0:len(i)-1]+chr(ord(i[-1])-32)
    string += i+' '
string
```

```
A
A
D
S
```

[59]:
```
'HellO WolrD AbC AsD DfdF SadA '
```

[ ]:
```

```

1.0.10 03) WAP to find Maximum frequency character in String.

[81]:
```
char_fre = {}
string = "Hello Wolrd abc asd dfdf sada"

# for i in string:

#     if(char_fre.get(i)):
```

```
#         print(char_fre.get(i))
#         char_fre[i] = 1
#     else:
#         char_fre[i] += 1
print(char_fre.get(i) == 'None')
```

```
False
```

1.0.11 04) WAP to find Minimum frequency character in String.

[ ]:

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

[ ]:

uufamjlcj

March 10, 2025

Python Programming - 2301CS404

Lab - 5

Raj Dedakiya | 23010101056 | 325

1 List

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

[2]:
```
l1=[1,2,3,4,5]
print(sum(l1))
```

```
15
```

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

[4]:
```
l1=[1,2,3,4,5]
print(max(l1))
```

```
5
```

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

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

```
10
```

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

[10]:
```
l1=[1,2,3,4,5,6]
l1[0],l1[-1]=l1[-1],l1[0]
print(l1)
```

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

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

[24]:
```
l1=[1,2,3,4,5,6]
mid=len(l1)//2
x=l1[:mid]
y=l1[mid:]
print(y+x)
```

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

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

[47]:
```
n=int(input('Enter length of list:'))
l1=[]
for i in range(n):
    a=int(input(f'Element {i+1}:'))
    l1.append(a)
x=int(input('Enter a position:'))
y=int(input('Enter a position:'))
l1[x], l1[y] = l1[y], l1[x]
print(f'List after interchange is: {l1}')
```

```
Enter length of list: 5
Element 1: 0
Element 2: 1
Element 3: 2
Element 4: 3
Element 5: 4
Enter a position: 1
Enter a position: 3
List after interchange is: [0, 3, 2, 1, 4]
```

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

[49]:
```
n=int(input('Enter length of list:'))
l1=[]
for i in range(n):
    a=int(input(f'Element {i+1}:'))
    l1.append(a)
l2=l1[::-1]
print(l2)
```

```
Enter length of list: 4
Element 1: 1
Element 2: 2
Element 3: 3
Element 4: 4
[4, 3, 2, 1]
```

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

[53]:
```
n=int(input('Enter length of list:'))
l1=[]
for i in range(n):
    a=int(input(f'Element {i+1}:'))
    l1.append(a)
l2=[i for i in l1 if(i%2==0)]
print(l2)
```

```
Enter length of list: 4
Element 1: 0
Element 2: 1
Element 3: 2
Element 4: 3
[0, 2]
```

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

[79]:
```
l1=[1,1,1,1,2,2,3,3,4,5,6,8,8,9]
l2=[]
count=0
for i in l1:
    if i not in l2:
        count+=1
        l2.append(i)
print(count)
```

```
8
```

1.0.10 10) WAP to copy a list.

[81]:
```
l1=[1,2,3,4,5,6,7,8,9]
l2=l1.copy()
print(l2)
```

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

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

[91]:
```
x=int(input('Enter a number:'))
y=int(input('Enter a number:'))
l1=[]
for i in range(x,y):
    if(i%2!=0):
        l1.append(i)
print(l1)
```

```
Enter a number: 1
Enter a number: 12
[1, 3, 5, 7, 9, 11]
```

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

[9]:
```
n=int(input('Enter length of list:'))
l1=[]
for i in range(n):
    a=int(input(f'Element {i+1}:'))
    l1.append(a)
k=int(input('Enter number to be counted:'))
l2=l1.count(k)
print(l2)
```

```
Enter length of list: 6
Element 1: 1
Element 2: 11
Element 3: 1
Element 4: 2
Element 5: 3
Element 6: 4
Enter number to be counted: 1
2
```

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

[11]:
```
n=int(input('Enter length of list:'))
l1=[]
for i in range(n):
    a=int(input(f'Element {i+1}:'))
    l1.append(a)
l1.sort()
l2=l1[len(l1)-2]
print(l2)
```

```
Enter length of list: 8
Element 1: 1
Element 2: 2
Element 3: 3
Element 4: 4
Element 5: 5
Element 6: 6
Element 7: 7
Element 8: 8
7
```

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

[7]:
```
list = []
flag = True
while flag == True:
    element = input('Enter a element for list (Enter "f" to terminate): ')
    if element == 'f':
        break
    else:
        element = int(element)
    list.append(element)

k = int(input('Enter the frequency: '))

frequencyDict = {}
for i in list:
    if i in frequencyDict:
        frequencyDict[i] += 1
    else:
        frequencyDict[i] = 1

result = [key for key, value in frequencyDict.items() if value > k]
print(f'The element that occured more than {k} times are: {result}')
```

```
Enter a element for list (Enter "f" to terminate):  1
Enter a element for list (Enter "f" to terminate):  1
Enter a element for list (Enter "f" to terminate):  11
Enter a element for list (Enter "f" to terminate):  1
Enter a element for list (Enter "f" to terminate):  2
Enter a element for list (Enter "f" to terminate):  2
Enter a element for list (Enter "f" to terminate):  23
Enter a element for list (Enter "f" to terminate):  3
Enter a element for list (Enter "f" to terminate):  2
Enter a element for list (Enter "f" to terminate):  f
Enter the frequency:  2
The element that occured more than 2 times are: [1, 2]
```

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

[109]:
```
#With List Comprehension
list = [i**2 for i in range(10)]
print(list)

#Without List Comprehension
list = []
for i in range(10):
    list.append(i**2)
```

```
print(f'The list with square fom 0 to 9 is: {list}')
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
The list with square fom 0 to 9 is: [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.

[113]:
```
fruit_list = []
flag = True
while flag == True:
    element = input('Enter a fruit for list (Enter "f" to terminate): ')
    if element == 'f':
        break
    fruit_list.append(element)

b_fruits = [i for i in fruit_list if i.lower().startswith('b')]
print(f"Fruits whose names start with 'b': {b_fruits}")
```

```
Enter a fruit for list (Enter "f" to terminate):  apple
Enter a fruit for list (Enter "f" to terminate):  banana
Enter a fruit for list (Enter "f" to terminate):  orange
Enter a fruit for list (Enter "f" to terminate):  f
Fruits whose names start with 'b': ['banana']
```

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

[1]:
```
list1 = []
flag = True
while flag == True:
    element = input('Enter a element for list1 (Enter "f" to terminate): ')
    if element == 'f':
        break
    else:
        element = int(element)
    list1.append(element)

list2 = []
flag = True
while flag == True:
    element = input('Enter a element for list2 (Enter "f" to terminate): ')
    if element == 'f':
        break
    else:
        element = int(element)
    list2.append(element)
```

```
common = [element for element in list1 if element in list2]
print(f'Enter common elements in both list are: {common}')
```

```
Enter a element for list1 (Enter "f" to terminate):  1
Enter a element for list1 (Enter "f" to terminate):  2
Enter a element for list1 (Enter "f" to terminate):  3
Enter a element for list1 (Enter "f" to terminate):  4
Enter a element for list1 (Enter "f" to terminate):  5
Enter a element for list1 (Enter "f" to terminate):  f
Enter a element for list2 (Enter "f" to terminate):  4
Enter a element for list2 (Enter "f" to terminate):  5
Enter a element for list2 (Enter "f" to terminate):  6
Enter a element for list2 (Enter "f" to terminate):  7
Enter a element for list2 (Enter "f" to terminate):  8
Enter a element for list2 (Enter "f" to terminate):  f
Enter common elements in both list are: [4, 5]
```

j7p48hw72

March 10, 2025

Python Programming - 2301CS404

Lab - 6

Raj Dedakiya | 23010101056 | 325

1 Tuple

1.0.1 01) WAP to find sum of tuple elements.

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

print("sum of tuple is: ", sum)
```

```
sum of tuple is:  21
```

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

[2]:
```
t2 = (1, 2, 3, 4, 5, 6, 7 ,8, 9, 10)
print("max of tuple is: ", max(t2))
print("min of tuple is: ", min(t2))
```

```
max of tuple is:  10
min of tuple is:  1
```

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

[23]:
```
t3 = [(1, 4, 6, 1, 3), (-1, 3, 5, 7, 9), (2, 4, 6, 8, 10), (1, -3, 5, 7, 9)]
t4 = []
k = int(input("Enter the number to find divisiblity in tuple: "))
for t in t3:
    all_div=True
    for i in t:
        if i % k != 0:
            all_div=False
```

```
            break
    if all_div:
        t4.append(t)

print("Tuple of divisible by ", k, " is: ", t4)
```

```
Enter the number to find divisiblity in tuple: 2
Tuple of divisible by  2  is:  [(2, 4, 6, 8, 10)]
```

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

[6]:
```
l1 = [1, 2, 3, 4, 5]
cubes = [(num, num**3) for num in l1]
l2 =[]
for i in l1:
    l2.append((i, i**3))
print("List of tuples with number and its cube: ", cubes)
print("List of tuples with number and its cube: ", l2)
```

```
List of tuples with number and its cube:  [(1, 1), (2, 8), (3, 27), (4, 64), (5,
125)]
List of tuples with number and its cube:  [(1, 1), (2, 8), (3, 27), (4, 64), (5,
125)]
```

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

[20]:
```
l3 = [(1, 4, 6, 1, 3), (-1, 3, 5, 7, 9), (2, 4, 6, 8, 10), (1, -3, 5, 7, 9)]
positive_tuples = [t for t in l3 if all(x > 0 for x in t)]
print("Tuples with all positive elements: ", positive_tuples)
```

```
Tuples with all positive elements:  [(1, 4, 6, 1, 3), (2, 4, 6, 8, 10)]
```

[12]:
```
l3 = [(1,4,6,1,3),(-1,3,5,7,9),(2,4,6,8,10),(1,-3,5,7,9)]
positive_tuples = []
for t in l3:
    all_positive = True
    for x in t:+
        if x <= 0:
            all_positive = False
            break
    if all_positive:
        positive_tuples.append(t)

print("Tuples with all positive elements: ", positive_tuples)
```

```
Tuples with all positive elements:  [(1, 4, 6, 1, 3), (2, 4, 6, 8, 10)]
```

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

[37]:
```
li = [5, 6, 7]
tupl= (9, 10)
print("List: ", li)
print("Tuple: ", tupl)
li.append(tupl)
print("After adding: ",li)
```

```
List:  [5, 6, 7]
Tuple:  (9, 10)
After adding:  [5, 6, 7, (9, 10)]
```

1.0.7 07) WAP to remove tuples of length K.

[29]:
```
test_list = [(4, 5), (4, ), (8, 6, 7), (1, ), (3, 4, 6, 7)]
print("The original list : " + str(test_list))
K = int(input("Enter the number to remove tuples of lenght: "))
res = [ele for ele in test_list if len(ele) != K]
print("Filtered list : " + str(res))
```

```
The original list : [(4, 5), (4,), (8, 6, 7), (1,), (3, 4, 6, 7)]
Enter the number to remove tuples of lenght: 2
Filtered list : [(4,), (8, 6, 7), (1,), (3, 4, 6, 7)]
```

1.0.8 08) WAP to remove duplicates from tuple.

[24]:
```
t6 = (1,1,2,3,3,4,4,5,5,6,6,6,7,7,8,8,9,9,10,10)
t7 = tuple(set(t6))
print("Tuple after removing duplicates: ", t7)
```

```
Tuple after removing duplicates:  (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

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

[25]:
```
t8 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
l8 = list(t8)
t9 = ()

for i in range(0, len(l8)-1):
    t9 = t9 + (l8[i] * l8[i+1],)

print("Tuple after multiplying adjacent elements: ", t9)
```

```
Tuple after multiplying adjacent elements:  (2, 6, 12, 20, 30, 42, 56, 72, 90)
```

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

[26]:
```
t10 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
if len(t10) == len(set(t10)):
    print("tuple is distinct")
else:
    print("tuple is not distinct")
```

```
tuple is distinct
```

[ ]:
```

```

nyo1r3el2

March 10, 2025

Python Programming - 2301CS404

Lab - 7

Raj Dedakiya | 23010101056 | 325

1 Set & Dictionary

1.0.1 01) WAP to iterate over a set.

[2]:
```
S1={1,2,3,4,5,6,7,8,9}
for i in S1:
    print(i)
```

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

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

[6]:
```
s1={1,2,3,4,5,6,7,8,9}
print(list(s1))
print(tuple(s1))
print(str(s1))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
(1, 2, 3, 4, 5, 6, 7, 8, 9)
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

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

[20]:
```
s1={1,2,3,4,5,6,7,8,9}
print(f'Maximum number:{max(s1)}')
print(f'Minimum number:{min(s1)}')
```

```
Maximum number:9
Minimum number:1
```

1.0.4 04) WAP to perform union of two sets.

[22]:
```
s1={1,2,3,4,5,6,7,8,9}
s2={1,2,3,44,55,66,77,88,99}
print(f'Union is:{s1|s2}')
```

```
Union is:{1, 2, 3, 4, 5, 6, 7, 8, 9, 99, 66, 44, 77, 55, 88}
```

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

[40]:
```
l1=[1,2,3,4,5,6,7,8,9]
l2=[1,2,3,11,22,33,44,55,66]
common=[x for x in l1 if x in l2]
print(set(common))
```

```
{1, 2, 3}
```

1.0.6 06) WAP to remove duplicates from list.

[24]:
```
li=[1,1,2,2,2,3,3,4,4,5,6,7]
print(f'Removed duplicates:{set(li)}')
```

```
Removed duplicates:{1, 2, 3, 4, 5, 6, 7}
```

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

[44]:
```
li=('mango','apple','banana','cherry','apple')
print(f'Unique words:{set(li)}')
```

```
Unique words:{'cherry', 'banana', 'mango', 'apple'}
```

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

[50]:
```
A={1,2,3,3,4,5,6,7}
B={1,2,3,4,5}
print(f'Removed duplicates:{set(A^B)}')
```

```
Removed duplicates:{6, 7}
```

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

[96]:
```
l1=('state')
l2=('taste')
l=set(l1.lower())
m=set(l2.lower())
if l == m:
    print("Given string is anagram")
else:
    print("Not anagram")
```

```
Given string is anagram
```

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

[104]:
```
l1=[1,2,3,4,5,6,7,8,9]
l2=[2,3,4,5,6]
l3=[2,5,6,7,8,9]
print(f'Common elements are:{set(l1)&set(l2)&set(l3)}')
```

```
Common elements are:{2, 5, 6}
```

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

[116]:
```
s1 = 'Hello World'
count = 0
for i in s1:
    if i.lower() in {'a', 'e', 'i', 'o', 'u'}:
        count += 1
if count > 0:
    print(f"Number of vowels: {count}")
else:
    print("No vowels")
```

```
Number of vowels: 3
```

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

[5]:
```
# Input string
string = input("Enter a string: ")
is_binary = all(char in '01' for char in string)
if is_binary:
    print(f'"{string}" is a binary string.')
else:
    print(f'"{string}" is not a binary string.')
```

```
Enter a string:  101010101010
"101010101010" is a binary string.
```

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

[1]:
```
my_dict = {'banana': 3, 'apple': 5, 'cherry': 2, 'date': 4}

# Sorting by keys
sorted_by_keys = {k: my_dict[k] for k in sorted(my_dict)}
print("Dictionary sorted by keys:")
print(sorted_by_keys)

# Sorting by values
sorted_by_values = {k: v for k, v in sorted(my_dict.items(), key=lambda item:␣
 ↪item[1])}
print("\nDictionary sorted by values:")
print(sorted_by_values)
```

```
Dictionary sorted by keys:
{'apple': 5, 'banana': 3, 'cherry': 2, 'date': 4}

Dictionary sorted by values:
{'cherry': 2, 'banana': 3, 'date': 4, 'apple': 5}
```

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

[9]:
```
user_input = input("Enter a dictionary (e.g., {'a': 10, 'b': 20}): ")
user_dict = eval(user_input)
if all(isinstance(value, (int, float)) for value in user_dict.values()):
    total_sum = sum(user_dict.values())
    print(f"The sum of all values in the dictionary is: {total_sum}")
else:
    print("All values in the dictionary must be numeric.")
```

```
Enter a dictionary (e.g., {'a': 10, 'b': 20}):  {'a':123,'b':234}
The sum of all values in the dictionary is: 357
```

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.

[11]:
```
dict1 = {'a': 5, 'c': 8, 'e': 2}
key = input("Enter the key to search in the dictionary: ")
if key in dict1:
    print(f"The value of '{key}' is: {dict1[key]}")
else:
    print("Key Not Found")
```

```
Enter the key to search in the dictionary:  c
The value of 'c' is: 8
```

hdkxnb1rk

March 10, 2025

Python Programming - 2301CS404

Lab - 8

Raj Dedakiya | 23010101056 | 325

1 User Defined Function

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

[31]:
```
def BMI(weight,height):
    bmi=weight/(height**2);
    return bmi

BMI(56,1.80)
```

[31]:
```
17.28395061728395
```

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

[41]:
```
def add(n):
    sum=0
    for i in range(1,n+1):
        sum=sum+i
    return sum

add(5)
```

[41]:
```
15
```

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

[61]:
```
def prime(a):
    if(a==0 or a==1):
        return 0;
    elif(a>1):
        for i in range(2,a):
```

```
            if(a%i==0):
                return 0
        else:
            return 1

prime(43)
```

[61]: 1

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

[68]:
```
li=[]
def Primeno(a,b):
    for i in range(a,b):
        if(prime(i)):
            li.append(i)
Primeno(1,10)
print(li)
```

```
[2, 3, 5, 7]
```

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

[91]:
```
st=input("Enter a string:")
def palindrome(st):
    if(st[::-1]==st):
        return True
    else:
        return False
palindrome(st)
```

```
Enter a string: lol
```

[91]: True

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

[93]:
```
li=[1,2,3,4,5,6,7,8,9]
def ans():
    Sum=sum(li)
    return Sum
ans()
```

[93]: 45

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

[107]:
```
li=[(0,1,2),(4,5,6),(7,8,9)]
def listsum():
    sum=0
    for i,j,k in li:
            sum=sum+i
    print(sum)
listsum()
```

```
11
```

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

[157]:
```
def fibonacci(a,b,n):
    if n==1:
        return a
    return fibonacci(b,a+b,n-1)

print(fibonacci(0,1,12))
```

```
89
```

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

[31]:
```
dict1 = {101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'}
def nameofstudent(n):
    a=dict1.get(n)
    return a

result=nameofstudent(103)
print(result)
```

```
Jay
```

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

[17]:
```
scores = [200, 456, 300, 100, 234, 678]
def sumofScores(scores):
    total=0
    for i in scores:
        if i%10==0:
```

```
            total+=i
    return total

result=sumofScores(scores)
print(result)
```

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'}

[39]:
```
ini_dict={'a': 10, 'b':20, 'c':30, 'd':40}
def invertDictonary(ini_dict):
    inv_dict = dict(zip(ini_dict.values(), ini_dict.keys()))
    return inv_dict

invertDictonary(ini_dict)
```

[39]:
```
{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.

[53]:
```
string="the quick brown fox jumps over the lazy dog"

def pangram(string):
    string=string.replace(" ","")
    string=string.lower()
    x=list(set(string))
    x.sort()
    x="".join(x)
    alphabets="abcdefghijklmnopqrstuvwxyz"
    if(x==alphabets):
        print("The string is a pangram")
    else:
        print("The string is not a pangram")

pangram(string)
```

```
The string is a pangram
```

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

[65]:
```
s1 = "AbcDEfgh"
def countUPLW(s1):
    lower=0
    upper=0
    for i in s1:
        if i.islower():
            lower+=1
        else:
            upper+=1
    print(f"Lower case are:{lower}")
    print(f"Upper case are:{upper}")

countUPLW(s1)
```

```
Lower case are:5
Upper case are:3
```

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

[67]:
```
min_number = lambda a, b : min(a,b)

print(min_number(5, 8))
```

```
5
```

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

[3]:
```
students=['Alexander','Benjamin','Jonathan','Malay','Meet','Dhairya']
def myFunc(x):
  if len(x) > 7:
    return True
  else:
    return False

names = filter(myFunc, students)

for x in names:
  print(x)
```

```
Alexander
Benjamin
Jonathan
```

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

[62]:
```
students = ['alexander', 'benjamin', 'jonathan', 'malay', 'meet', 'dhairya']

def uppercaseusingMap(names):
    return list(map(str.capitalize, names))

result = uppercaseusingMap(students)
print(result)
```

```
['Alexander', 'Benjamin', 'Jonathan', 'Malay', 'Meet', 'Dhairya']
```

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

[92]:
```
# Positional Arguments
print("Positional Arguments::")
def nameAge(name, age):
    print("Hi, I am", name)
    print("My age is ", age)

nameAge(name="Prince", age=20)
print()

# Keyword Arguments
print("Keyword Arguments::")
def my_function(child3, child2, child1):
  print("The youngest child is " + child3)

my_function(child1 = "alexander", child2 = "benjamin", child3 = "Jonathan")
print()

# Default Arguments
print("Default Arguments::")
def my_function(country = "India"):
  print("I am from " + country)

my_function()
my_function("Australia")
print()
```

```
# Variable Legngth Positional(*args) & variable length Keyword Arguments␣
  ↪(**kwargs)
print("Variable Legngth Positional(*args) & variable length Keyword Arguments␣
  ↪(**kwargs)::")
# *args
def my_function(*kids):
  print("*args:The youngest child is " + kids[2])

my_function("alexander", "benjamin", "Jonathan")
#**kwargs
def my_function(**kid):
  print("**kargs:His last name is " + kid["lname"])

my_function(fname = "alexander", lname = "benjamin")
print()

# Keyword-Only & Positional Only Arguments
print("Keyword-Only & Positional Only Arguments::")
```

```
Positional Arguments::
Hi, I am Prince
My age is  20

Keyword Arguments::
The youngest child is Jonathan

Default Arguments::
I am from India
I am from Australia

Variable Legngth Positional(*args) & variable length Keyword Arguments
(**kwargs)::
*args:The youngest child is Jonathan
**kargs:His last name is benjamin

Keyword-Only & Positional Only Arguments::
```

2w8mepgch

March 10, 2025

Python Programming - 2301CS404

Lab - 9

Raj Dedakiya | 23010101056 | 325

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

[11]:
```
f = open("newFile.txt", "r")
print(f.read())
f.close()

f = open("newFile.txt", "r")
print(f.readline())
f.close()

f = open("newFile.txt", "r")
print(f.readlines())
f.close()
```

```
Hello World!!!!!!!!!!
Good Mornings…
Hello World!!!!!!!!!!

['Hello World!!!!!!!!!!\n', 'Good Mornings…']
```

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

[49]:
```
f = open("new.txt","w")
f.write("Hi Hello from python")
f.close()
```

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

[31]:
```
f = open("newFile.txt", "r")
for i in range(5):
    print(f.readline())
f.close()
```

```
Hello World!!!!!!!!!!

Good Mornings…

1

2

3
```

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

[55]:
```
f = open("newFile.txt", "r")
words = f.read().split()
print (max(words, key=len))
f.close()
```

```
Mornings…
```

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

[51]:
```
#No.of lines
f = open("newFile.txt","r")
lines=len(f.readlines())
print(lines)
f.close()

#No. of words
f = open("newFile.txt","r")
lines=len(f.readline())
print(lines)
f.close()
```

```
#No. of characters
f = open("newFile.txt","r")
lines=len(f.read())
print(lines)
f.close()
```

```
9
22
57
```

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

[59]:
```
f1 = open("newFile.txt","r")
f2 = open("newFile2.text","w")
for i in f1:
    f2.write(i)
f1.close()
f2.close()
```

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

[67]:
```
import os
sz = os.path.getsize("newFile.txt")
print(sz)
```

```
57
```

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

[71]:
```
def frequency(file_content,specific_word):
    # Method-1
    count = file_content.count(specific_word)
    # Method-2
    # count = 0
    # for i in file_content:
    #     if i == specific_word:
    #         count += 1
    return count

fp = open("newFile.txt","r")
data = fp.read()
print(frequency(data.split(),'Hello'))
fp.close()
```

```
1
```

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.

[5]:
```
fp = open("newFile.txt", "w+")
for i in range(1, 6):
    mark = input(f"Enter marks of subject {i}: ")
    fp.write(mark + "\n")
fp.seek(0)
l1 = [int(mark.strip()) for mark in fp.readlines()]
print(f"The highest score is: {max(l1)}")
fp.close()
```

```
Enter marks of subject 1:  5
Enter marks of subject 2:  10
Enter marks of subject 3:  15
Enter marks of subject 4:  20
Enter marks of subject 5:  25
The highest score is: 25
```

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)

[3]:
```
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def prime_numbers(n, filename):
    count = 0
    num = 2
    with open(filename, 'w') as fp:
        while count < n:
            if is_prime(num):
                fp.write(f"{num}\n")
                count += 1
            num += 1

prime_numbers(100, "primenumbers.txt")
```

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

[13]:
```
f1 = open("newFile.txt", "r")
f2 = open("new.txt", "r")
fp = open("mergedFile.txt", "w")
fp.write(f1.read())
fp.write(f2.read())
f1.close()
f2.close()
fp.close()

print("Files 'newFile.txt' and 'newFile2.txt' have been merged into 'mergedFile.
 ↪txt'")
```

```
Files 'newFile.txt' and 'newFile2.txt' have been merged into 'mergedFile.txt'
```

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

[15]:
```
f1 = open("mergedFile.txt", "r")
content = f1.read()
updated_content = content.replace('Hi', 'bye')
new_file = open("updated_content.txt", "w")
new_file.write(updated_content)
f1.close()
new_file.close()

print("The word replacement has been done and saved to 'updated_content.txt'")
```

```
The word replacement has been done and saved to 'updated_content.txt'
```

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.

[3]:
```
with open('newFile.txt', 'w') as fp:
    fp.write("Hello, this is a sample file for demonstrating tell() and seek().
 ↪")

with open('newFile.txt', 'r') as fp:
    # Case 1: Tell the current position from the beginning
    print("Case 1: Current position from beginning (before reading):", fp.
 ↪tell())
    print("Reading first 5 characters:")
    print(fp.read(5))  # Read first 5 characters
    print("Position after reading 5 characters:", fp.tell())

    # Case 2: Seek from the beginning (SEEK_SET)
    fp.seek(0, 0)  # Move the pointer to the beginning
```

```
    print("\nCase 2: Seek from the beginning to position 0:", fp.tell())
```

```
Case 1: Current position from beginning (before reading): 0
Reading first 5 characters:
Hello
Position after reading 5 characters: 5

Case 2: Seek from the beginning to position 0: 0
```

[ ]:

6o3reeq0s

March 10, 2025

Python Programming - 2301CS404

Lab - 10

Raj Dedakiya | 23010101056 | 325

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.

[14]:
```
try:
    # ZeroDivisionError
    result = 10 / 0

    # ValueError
    value = int("not_a_number")

    # TypeError
    result = "string" + 10

#handling using seperate except block
except ZeroDivisionError:
    print("Error: Cannot divide by zero.")
except ValueError:
    print("Error: Invalid value provided.")
except TypeError:
    print("Error: Type mismatch encountered.")

try:
    # ZeroDivisionError
    result = 10 / 0

    # ValueError
    value = int("not_a_number")
```

```
    # TypeError
    result = "string" + 10

#handling using single except block
except (ZeroDivisionError, ValueError, TypeError) as e:
    print(f"Error: {str(e)}")
```

```
Error: Cannot divide by zero.
Error: division by zero
```

1.0.2 02) WAP to handle following exceptions:

1. IndexError
2. KeyError

[42]:
```
#Index Error
try:
    l1=[1,2,3,4,5,6]
    print(l1[6])
except IndexError as e:
    print(f'Index Error:{str(e)}')

#KeyError
try:
    d1={'a':1,'b':2,'c':3}
    print(d1['d'])
except KeyError as e:
    print(f'KeyError: {str(e)}')
```

```
Index Error:list index out of range
KeyError: 'd'
```

1.0.3 03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

[50]:
```
#FileNotFoundError
try:
    fp=open("a.txt", "r")
    fp.read()
except FileNotFoundError as e:
    print(f'FileNotFoundError:{str(e)}')

#ModuleNotFoundError
try:
    import a
```

```
except ModuleNotFoundError as e:
    print(f'ModuleNotFoundError:{str(e)}')
```

```
FileNotFoundError:[Errno 2] No such file or directory: 'a.txt'
ModuleNotFoundError:No module named 'a'
```

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

[63]:
```
try:
    # ZeroDivisionError
    result = 10 / 0

    # IndexError
    # my_list = [1, 2, 3]
    # print(my_list[5])

    # ValueError
    # value = int("not_a_number")

    # KeyError
    # my_dict = {'a': 1}
    # print(my_dict['b'])

    # FileNotFoundError
    # with open("non_existent_file.txt", "r") as file:
    #     content = file.read()

    # ModuleNotFoundError
    # import non_existent_module

except Exception as e:
    print(f"An error occurred: {str(e)}")
```

```
An error occurred: division by zero
```

1.0.5 05) WAP to demonstrate else and finally block.

[57]:
```
try:
    result = 10 / 2
    print(result)

except ZeroDivisionError as e:
    print(f"An error occurred: {str(e)}")

else:
    print("he try block executed successfully.")
```

```
finally:
    print("This block is always executed")
```

```
5.0
he try block executed successfully.
This block is always executed
```

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.

[75]:
```
grades_input = input("Enter a list of grades separated by commas: ")

try:
    grades = [int(grade.strip()) for grade in grades_input.split(',')]
    print("Grades:", grades)

except ValueError:
    print("Error: Some of the values you entered cannot be converted to␣
 ↪integers. Please enter valid numbers.")
```

```
Enter a list of grades separated by commas:  10,20.3.50,60
Error: Some of the values you entered cannot be converted to integers. Please
enter valid numbers.
```

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

[65]:
```
def divide(a, b):
    try:
        result = a / b

    except ZeroDivisionError:
        return "Error: Cannot divide by zero."

    else:
        return result

a = float(input("Enter the a: "))
b = float(input("Enter the b: "))

result = divide(a, b)
print(f"Result: {result}")
```

```
Enter the a:  5
Enter the b:  0
Result: Error: Cannot divide 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.

[71]:
```
def ageInvalid():
    try:
        age = int(input("Enter your age: "))
        if age < 18:
            raise ValueError("Enter Valid Age")
        else:
            print(f"Your age is {age}")
    except ValueError as e:
        print(f"Error: {str(e)}")

ageInvalid()
```

```
Enter your age:  5
Error: 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.

[81]:
```
class InvalidUsername(Exception):
    pass
def custom_exception():
    try:
        name=input("Enter your name::")
        if(len(name)<5 or len(name)>15):
            raise InvalidUsername("Username must be between 5 and 15 characters␣
 ↪long")
    except InvalidUsername as e:
        print(f"Error:{str(e)}")
custom_exception()
```

```
Enter your name:: raj
Error: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.

[79]:
```
import math
class NegativeNumberError(Exception):
    pass
def calculate_square_root():
    try:
        number = float(input("Enter a number to calculate its square root: "))
        if number < 0:
            raise NegativeNumberError("Cannot calculate the square root of a␣
 ↪negative number")
        sqrt_result = math.sqrt(number)
        print(f"The square root of {number} is: {sqrt_result}")
    except NegativeNumberError as e:
        print(f"Error: {str(e)}")
    except ValueError:
        print("Error: Please enter a valid number.")
calculate_square_root()
```

```
Enter a number to calculate its square root:  25
The square root of 25.0 is: 5.0
```

uyovr1bnm

March 10, 2025

Python Programming - 2301CS404

Lab - 11

Raj Dedakiya | 23010101056 | 325

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]:
```
from another import add,multiply,divide,subtract
print(add(5,6))
print(subtract(10,5))
print(multiply(5,6))
print(divide(4,2))
```

```
11
5
30
2.0
```

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

[164]:
```
import random
a='Hello World'
print(random.choice(a))
```

```
o
```

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

[32]:
```
li=[1,2,3,4,5,6,7,8,9]
print(random.choice(li))
```

```
5
```

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

[90]:
```
random.seed(2)

def roll_dice():
    return random.randint(1,6)

print(roll_dice())
```

```
1
```

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

[100]:
```
def get_random_integer():
    count = 0
    l1 = []
    while count != 3:
        random_number = random.randint(100, 999)
        if random_number % 5 == 0:
            l1.append(random_number)
            count += 1
    return l1

print(get_random_integer())
```

```
[470, 265, 770]
```

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.

[114]:
```
import random

tickets = [''.join(random.choices('0123456789', k = 6)) for _ in range(100)]

winner = random.choice(tickets)
tickets.remove(winner)
runner_up = random.choice(tickets)

print(f"Winner: Ticket {winner}")
print(f"Runner-up: Ticket {runner_up}")
```

```
Winner: Ticket 751582
Runner-up: Ticket 968558
```

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

[118]:
```
import datetime
current_time = datetime.datetime.now()
print(current_time)
```

```
2025-02-12 12:58:30.892162
```

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

[129]:
```
new_date = datetime.datetime.now() - datetime.timedelta(weeks = 1)
print(f"Date after subtracting a week: {new_date}")
```

```
Date after subtracting a week: 2025-02-05 13:01:37.878858
```

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

[133]:
```
start_date = datetime.date(2025, 2, 4)
end_date = datetime.date(2025, 2, 12)
no_of_days = abs(start_date - end_date)
print(no_of_days)
```

```
8 days, 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.)

[143]:
```
date1=datetime.date(2025,2,12)
dayname=date1.strftime("%A")
print(dayname)
```

```
Wednesday
```

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

[147]:
```
print(datetime.datetime.now())
```

```
2025-02-12 13:06:45.029571
```

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

[155]:
```
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
```

as8qpniwe

March 10, 2025

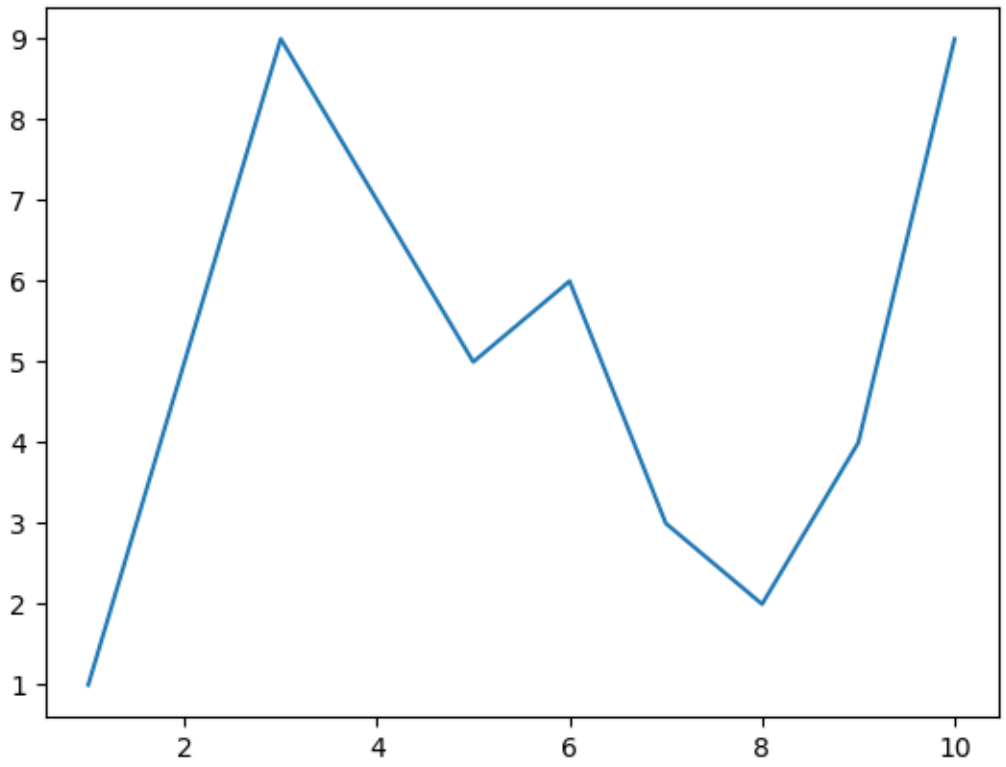
Python Programming - 2301CS404

Lab - 12

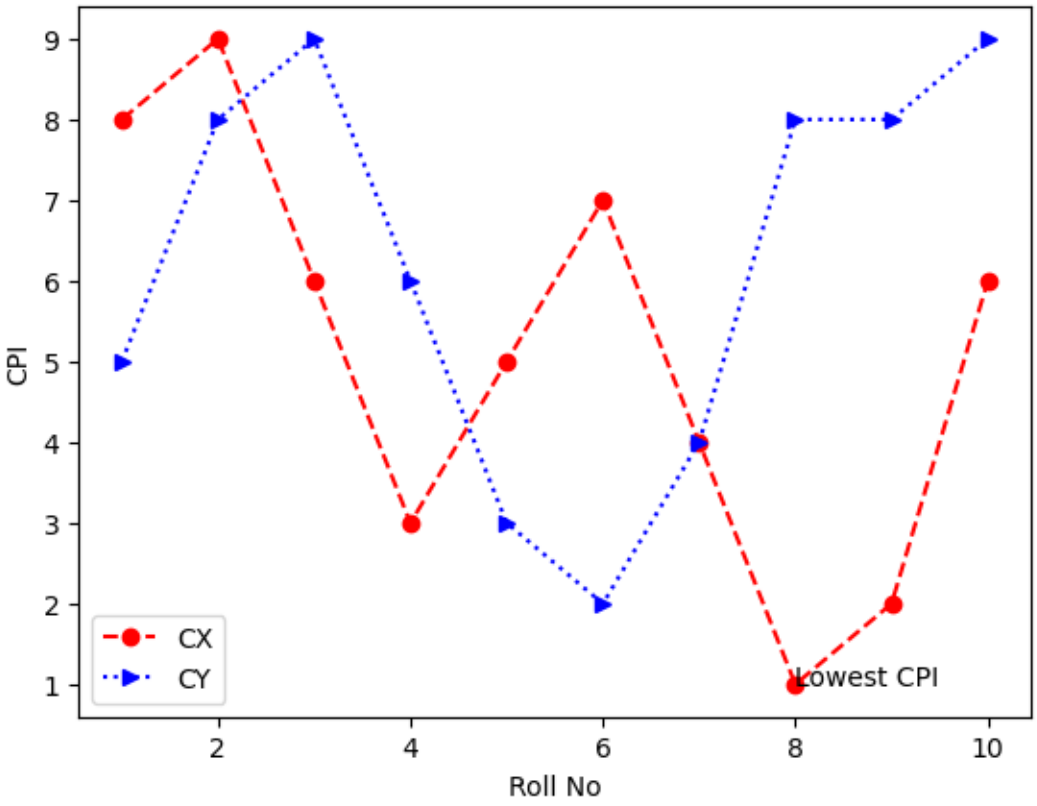
Raj Dedakiya | 23010101056 | 325

[ ]:
```
#import matplotlib below
```

[4]:
```
import matplotlib.pyplot as plt
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]
plt.plot(x,y)
plt.show()
# write a code to display the line chart of above x & y
```



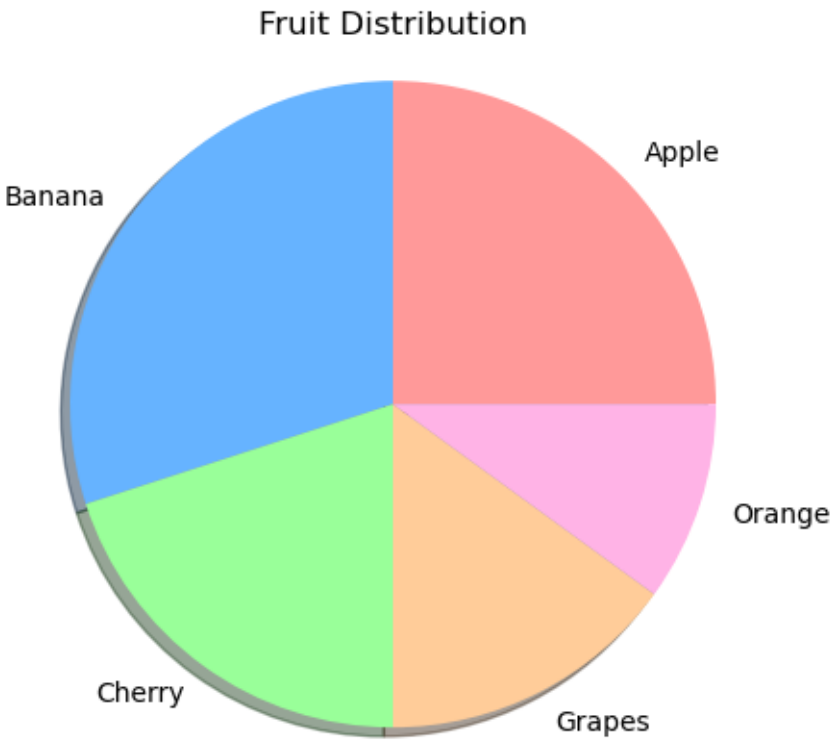
[19]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]
plt.plot(x,cxMarks,label="CX",color="r",marker="o",linestyle="--")
plt.plot(x,cyMarks,label="CY",color="b",marker=">",linestyle="dotted")
plt.xlabel("Roll No")
plt.ylabel("CPI")
plt.legend()
plt.annotate('Lowest CPI',xy=[8,1])
plt.show()
```



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

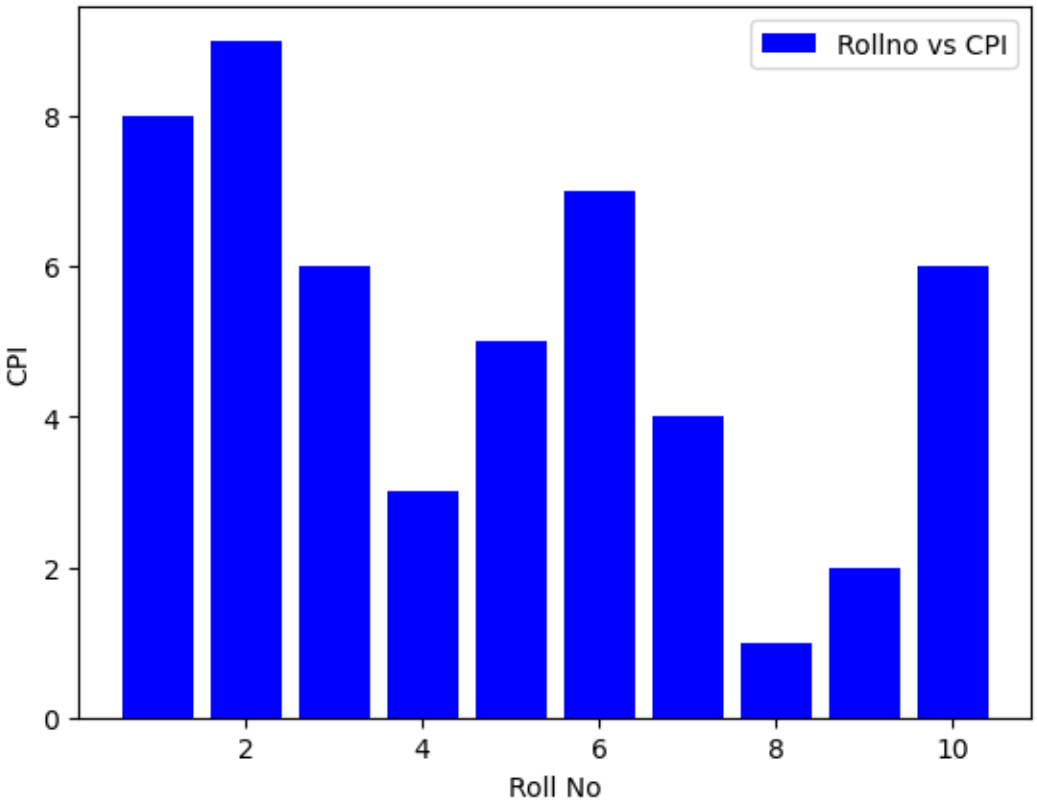
[114]:
```
labels = ['Apple', 'Banana', 'Cherry', 'Grapes', 'Orange']
sizes = [25, 30, 20, 15, 10]
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#ffb3e6']

plt.pie(sizes, labels=labels, colors=colors, shadow=True)
plt.axis('equal')
plt.title('Fruit Distribution')
plt.show()
```



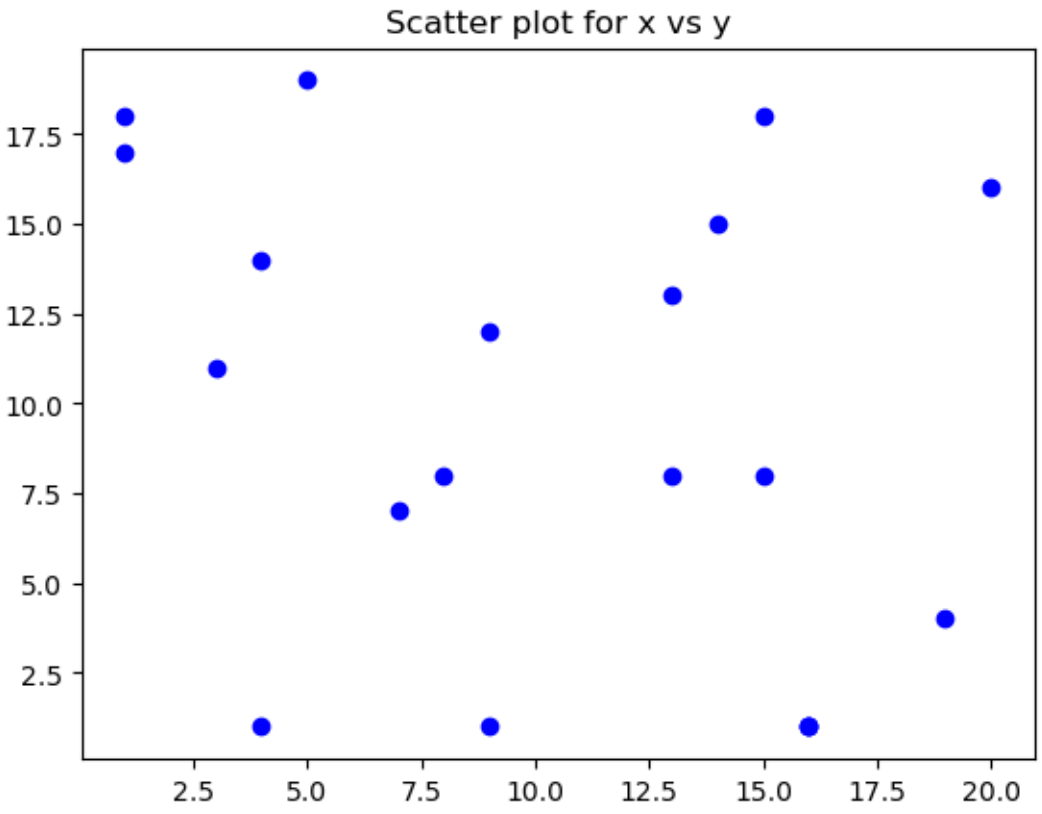
0.0.2 05) WAP to demonstrate the use of Bar chart.

[58]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
plt.bar(x,cxMarks,label="Rollno vs CPI",color="b")
plt.xlabel("Roll No")
plt.ylabel("CPI")
plt.legend()
plt.show()
```



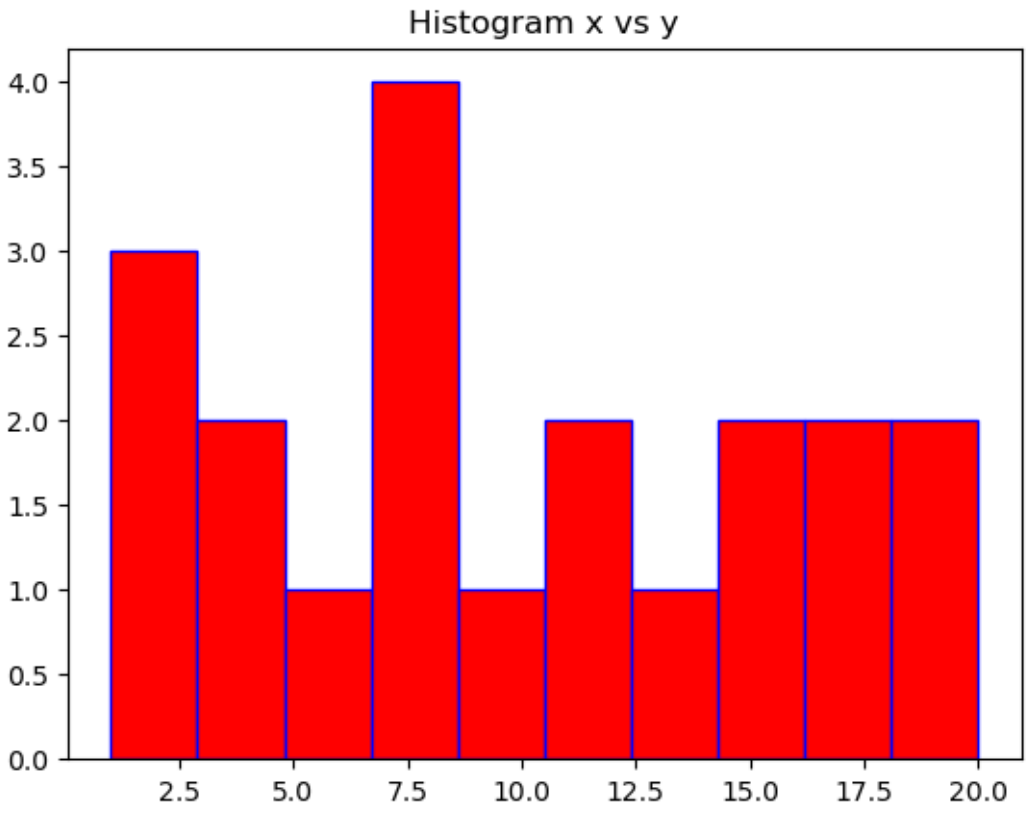
0.0.3 06) WAP to demonstrate the use of Scatter Plot.

[82]:
```
import random as r
r.seed(1)
x = [r.randint(1,20) for i in range(20)]
y = [r.randint(1,20) for i in range(20)]
plt.scatter(x,y,color="b")
plt.title("Scatter plot for x vs y")
plt.show()
```



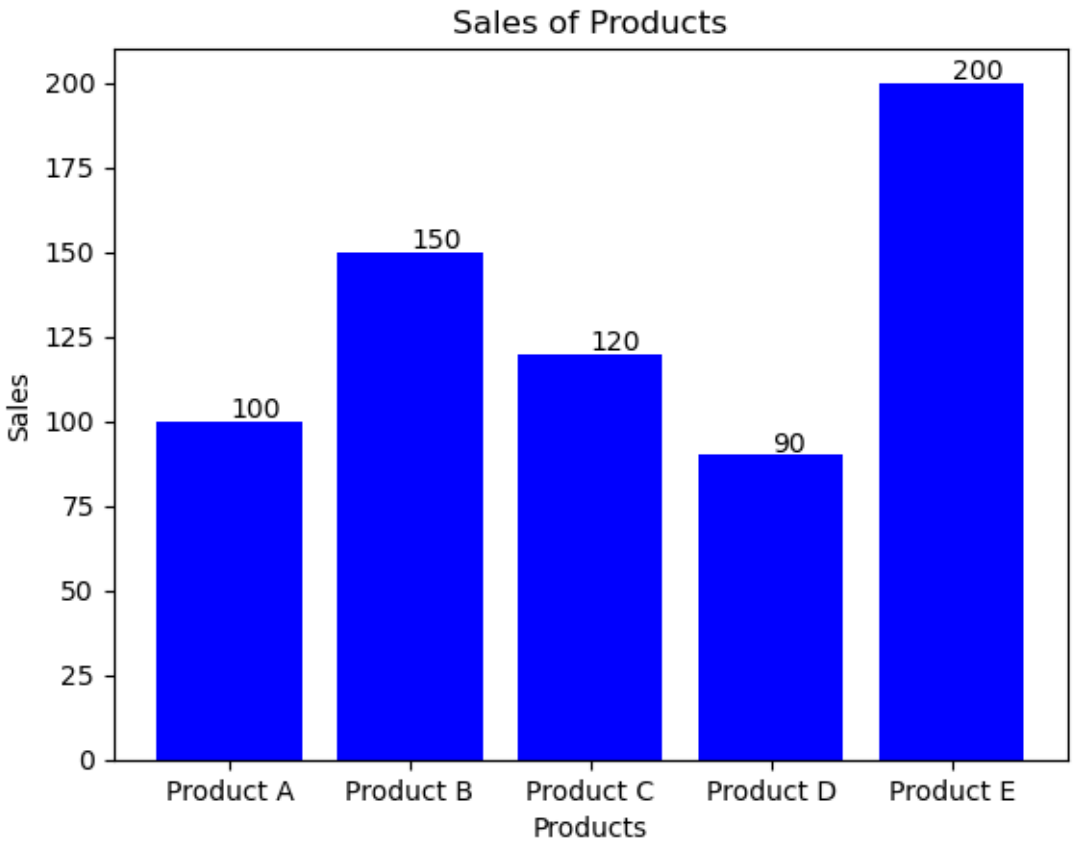
0.0.4 07) WAP to demonstrate the use of Histogram.

[108]:
```
r.seed(5)
data = [r.randint(1,20) for i in range(20)]
plt.hist(data,edgecolor="b",color="r")
plt.title("Histogram x vs y")
plt.show()
```



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

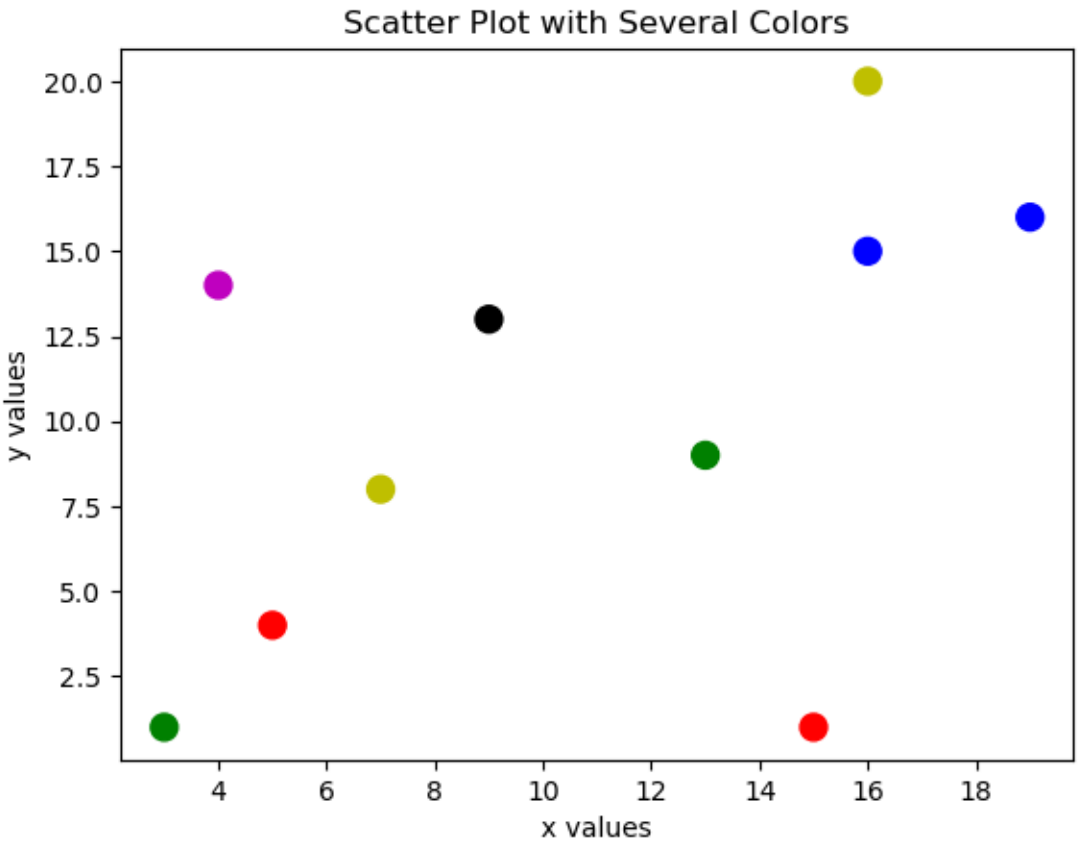
[128]:
```
products = ['Product A', 'Product B', 'Product C', 'Product D', 'Product E']
sales = [100, 150, 120, 90, 200]
plt.bar(products, sales, color='b')
for i in range(len(products)):
    plt.text(i, sales[i]+1,str(sales[i]))
plt.title('Sales of Products')
plt.xlabel('Products')
plt.ylabel('Sales')
plt.show()
```



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

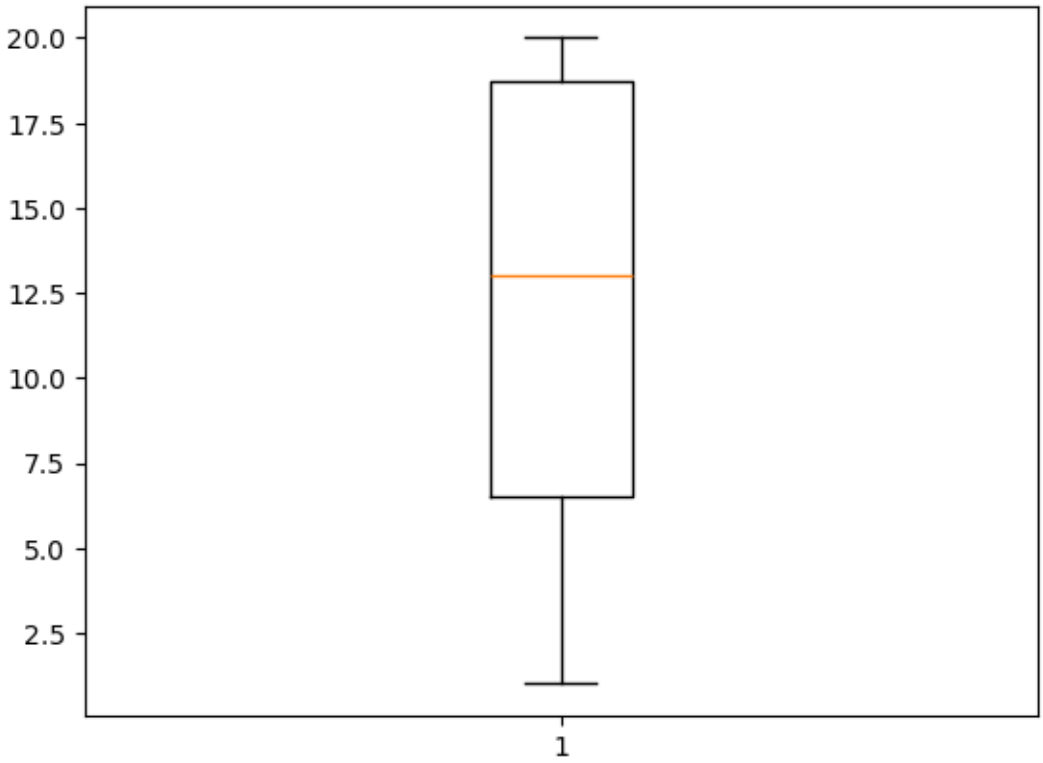
[134]:
```
r.seed(1)
x = [r.randint(1,20) for i in range(10)]
y = [r.randint(1,20) for i in range(10)]
colors = ['r','b','g','k','m','y','r','b','g','y']

plt.scatter(x, y, color=colors, s=100)
plt.title('Scatter Plot with Several Colors')
plt.xlabel('x values')
plt.ylabel('y values')
plt.show()
```



0.0.7 10) WAP to create a Box Plot.

[158]:
```
data=[r.randint(1,20) for i in range(10)]
plt.boxplot(data)
plt.show()
```


20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
1

22m0qghu4

March 10, 2025

Python Programming - 2301CS404

Lab - 13

Raj Dedakiya | 23010101056 | 325

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.

[1]:
```
class Student:
    def __init__(self,name,age,grade):
        self.name=name
        self.age=age
        self.grade=grade

obj=Student("abc",19,"A")

print(obj.name)
print(obj.age)
print(obj.grade)
```

```
abc
19
A
```

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.

[2]:
```
class Bank_Account:
    def __init__(self):
        pass

    def GetAccountDetails(self):
        self.Account_no=int(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=int(input("Enter Account_Balance:"))

    def DisplayAccountDetails(self):
        print(self.Account_no)
        print(self.User_Name)
        print(self.Email)
        print(self.Account_Type)
        print(self.Account_Balance)

obj=Bank_Account()
obj.GetAccountDetails()
obj.DisplayAccountDetails()
```

```
Enter Account_no:123
Enter User_Name:jgf
Enter Email:jfedfjh
Enter Account_Type:hds
Enter Account_Balance:545
123
jgf
jfedfjh
hds
545
```

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

[36]:
```
import math
class Circle:
    def area(self,r):
        print(f"Area:{math.pi*r*r}")

    def perimeter(self,r):
        print(f"Perimeter:{2*math.pi*r}")

obj=Circle()
obj.area(7)
obj.perimeter(7)
```

```
Area:153.93804002589985
Perimeter:43.982297150257104
```

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.

[3]:
```
class Employee:
    def __init__(self,name,age,salary):
        self.name=name
        self.age=age
        self.salary=salary

    def updatedetails(self,name,age,salary):
        if name:
            self.name=name
        if age:
            self.age=age
        if salary:
            self.salary=salary

    def displaydetails(self):
        print("Employee Information:")
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Salary: {self.salary}")

obj=Employee("abc",20,555500000)
obj.displaydetails()
obj.updatedetails("abc",19,1000000)
obj.displaydetails()
```

```
Employee Information:
Name: abc
Age: 20
Salary: 555500000
Employee Information:
Name: abc
Age: 19
Salary: 1000000
```

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

[28]:
```
class Bank_Account:
    def __init__(self,acc_no,balance):
        self.acc_no=acc_no
        self.balance=balance

    def deposit(self,amount):
        if amount > 0:
            self.balance += amount
```

```
            print(f"Deposited Rs.{amount}. Current balance: Rs.{self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self,amount):
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrew Rs.{amount}. Current balance: Rs.{self.
 ↪balance}")
            else:
                print("Insufficient funds.")
        else:
            print("Withdrawal amount must be positive.")

    def check_balance(self):
        print(f"Balance:Rs.{self.balance}")

obj=Bank_Account(151322,10000000)
obj.deposit(1000)
obj.withdraw(123)
obj.check_balance()
```

```
Deposited Rs.1000. Current balance: Rs.10001000
Withdrew Rs.123. Current balance: Rs.10000877
Balance:Rs.10000877
```

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.

[38]:
```
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, item_name, price, quantity):
        if item_name in self.items:
            self.items[item_name]['quantity'] += quantity
        else:
            self.items[item_name] = {'price': price, 'quantity': quantity}
        print(f"Added {quantity} {item_name}(s) to inventory.")

    def remove_item(self, item_name, quantity):
        if item_name in self.items:
            if self.items[item_name]['quantity'] >= quantity:
                self.items[item_name]['quantity'] -= quantity
                print(f"Removed {quantity} {item_name}(s) from inventory.")
            else:
```

```
                print(f"Not enough {item_name} in inventory to remove.")
        else:
            print(f"{item_name} not found in inventory.")

    def update_price(self, item_name, new_price):
        if item_name in self.items:
            self.items[item_name]['price'] = new_price
            print(f"Updated price of {item_name} to ${new_price}.")
        else:
            print(f"{item_name} not found in inventory.")

    def display_inventory(self):
        if not self.items:
            print("Inventory is empty.")
        else:
            print("Inventory Details:")
            for item_name, details in self.items.items():
                print(f"{item_name}: Price - ${details['price']}, Quantity -␣
↪{details['quantity']}")

inventory = Inventory()
inventory.add_item("Laptop", 1200, 10)
inventory.add_item("Phone", 800, 15)
inventory.display_inventory()
inventory.remove_item("Laptop", 5)
inventory.update_price("Phone", 850)
inventory.display_inventory()
inventory.remove_item("Laptop", 6)
inventory.update_price("Tablet", 300)
```

```
Added 10 Laptop(s) to inventory.
Added 15 Phone(s) to inventory.
Inventory Details:
Laptop: Price - $1200, Quantity - 10
Phone: Price - $800, Quantity - 15
Removed 5 Laptop(s) from inventory.
Updated price of Phone to $850.
Inventory Details:
Laptop: Price - $1200, Quantity - 5
Phone: Price - $850, Quantity - 15
Not enough Laptop in inventory to remove.
Tablet not found in inventory.
```

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

[36]:
```
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.

[4]:
```
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: ukfg
Enter the number of days present in class: 5
Attendance recorded: 5 days
Certificate of Attendance

Principal: Mr ABC
Student: ukfg
Days Present: 5
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.

[42]:
```
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)
```

[ ]:

8hdxiznt2

March 10, 2025

Python Programming - 2301CS404

Lab - 13

Raj Dedakiya | 23010101056 | 325

0.1 Continued..

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

[2]:
```
class ractangle:
    def __init__(self,length,width):
        self.length=length
        self.width=width

    def area(self,ractangle):
        return ractangle.length*ractangle.width

a=int(input("Enter length : "))
b=int(input("Enter width : "))
rect=ractangle(a,b)
print(f"Area of ractangle : {rect.area(rect)}")
```

```
Enter length : 20
Enter width : 20
Area of ractangle : 400
```

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().

[7]:
```
class square:
    def __init__(self,a):
        self.a=a

    def area(self):
        ans=self.a*self.a
        self.output(ans)
    def output(self,b):
```

```
        print(b)
d=int(input("Enter value : "))
f=square(d)
f.area()
```

```
Enter value : 10
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.

[12]:
```
class Rectangle:
    def __init__(self, length, width):
        if length == width:
            print("THIS IS SQUARE")
        else:
            self.length = length
            self.width = width

    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 compare_sides(self):
        if self.length == self.width:
            print("This is a square.")
        else:
            print("This is a rectangle.")

length=int(input("Enter length : "))
width=int(input("Enter width : "))
rect = Rectangle(length,width)
rect.area()
rect.compare_sides()
```

```
Enter length : 10
Enter width : 5
The area of the rectangle is: 50
This is a rectangle.
```

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.

[14]:
```
class Square:
    def __init__(self, side):
        self.__side = side

    def get_side(self):
        return self.__side

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


square = Square(5)
print(square.get_side())
square.set_side(7)
print(square.get_side())
```

```
5
7
```

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().

[15]:
```
class Profit:
    def __init__(self):
        self.profit = 0

    def getProfit(self, profit):
        self.profit = profit

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

    def getLoss(self, loss):
        self.loss = loss

class BalanceSheet(Profit, Loss):
    def __init__(self):
```

```
        Profit.__init__(self)
        Loss.__init__(self)

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

    def printBalance(self):
        print(f"Balance: {self.getBalance()}")


bs = BalanceSheet()
bs.getProfit(5000)
bs.getLoss(2000)
bs.printBalance()
```

```
Balance: 3000
```

0.1.12 15) WAP to demonstrate all types of inheritance.

[13]:
```
class Animal:
    def speak(self):
        return "Animal speaks"

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

class Flyer:
    def fly(self):
        return "Can fly"

class Bird(Animal, Flyer):
    def chirp(self):
        return "Bird chirps"

class Puppy(Dog):
    def play(self):
        return "Puppy plays"

class Cat(Animal):
    def meow(self):
        return "Cat meows"

class Bat(Bird, Dog):
    def hang(self):
        return "Bat hangs upside down"
```

```
dog = Dog()
print(dog.speak())
print(dog.bark())


bird = Bird()
print(bird.speak())
print(bird.fly())
print(bird.chirp())


puppy = Puppy()
print(puppy.speak())
print(puppy.bark())
print(puppy.play())

cat = Cat()
print(cat.speak())
print(cat.meow())


bat = Bat()
print(bat.speak())
print(bat.fly())
print(bat.bark())
```

```
Animal speaks
Dog barks
Animal speaks
Can fly
Bird chirps
Animal speaks
Dog barks
Puppy plays
Animal speaks
Cat meows
Animal speaks
Can fly
Dog barks
```

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


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

        super().__init__(name, age)
        self.salary = salary

    def __str__(self):
        return f"Employee(Name: {self.name}, Age: {self.age}, Salary: {self.
 ↪salary})"

emp = Employee("xyz", 30, 50000)
print(emp)
```

```
Employee(Name: xyz, 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]:
```
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
```

[ ]: