01) WAP to print “Hello World”

```
print('Hello World')

Hello World
```

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

```
a,b = 1,2
print(a+b)

c=int(input('Enter First Number c: '))
d=int(input('Enter Second Number d: '))
print(c+d)

3
Enter First Number c: 3
Enter Second Number d: 4
7
```

03) WAP to check the type of the variable.

```
a=5
print(type(a))

b=input('Enter input:')
print(type(b))

<class 'int'>
Enter input:Riya
<class 'str'>
```

04) WAP to calculate simple interest.

```
P=float(input('Enter P:'))
R=float(input('Enter R:'))
N=float(input('Enter N:'))

I=(P*R*N)/100
print(I)

Enter P:10
Enter R:10
Enter N:10
10.0
```

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

```
R=float(input('Enter radius of a circle:'))

Area = 3.14*R*R
print('Area =',Area)

Perimeter = 2*3.14*R
print('Perimeter =',Perimeter)

Enter radius of a circle:3
Area = 28.259999999999998
Perimeter = 18.84
```

06) WAP to calculate area of a triangle.

```
H = float(input('Enter Height : '))
B = float(input('Enter Base : '))

Area = 0.5*H*B
print('Area = ',Area)

Enter Height : 2
Enter Base : 4
Area =  4.0
```

07) WAP to compute quotient and remainder.

```
a = 20
b = 3

quotient = a // b
remainder = a % b

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

Quotient: 6
Remainder: 2
```

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

```
temp = float(input("Enter temperature: "))
choice = input("Convert from (C)elsius to Fahrenheit or (F)ahrenheit
to Celsius? (C/F): ").upper()

if choice == "C":
    fahrenheit = (temp * 9/5) + 32
    print(f"{temp} Celsius is equal to {fahrenheit} Fahrenheit.")
elif choice == "F":
    celsius = (temp - 32) * 5/9
    print(f"{temp} Fahrenheit is equal to {celsius} Celsius.")
else:
    print("Invalid choice! Please enter either 'C' or 'F'.")

Enter temperature:  355
Convert from (C)elsius to Fahrenheit or (F)ahrenheit to Celsius?
(C/F):  F

355.0 Fahrenheit is equal to 179.44444444444446 Celsius.
```

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

```
import math
x1, y1 = map(float, input("Enter coordinates of the first point (x1,
y1): ").split())
x2, y2 = map(float, input("Enter coordinates of the second point (x2,
y2): ").split())

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

print("The distance between the points is:", distance)

Enter coordinates of the first point (x1, y1):  1 2
Enter coordinates of the second point (x2, y2):  3 4

The distance between the points is: 2.8284271247461903
```

10) WAP to print sum of n natural numbers.

```
n = int(input("Enter a number: "))

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

print("Sum of the first", n, "natural numbers is:", sumOfn)

Enter a number:  5

Sum of the first 5 natural numbers is: 15
```

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

```
n = int(input("Enter a number: "))

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

print("Sum of the first", n, "natural numbers is:", sumOfsquareOfn)

Enter a number:  3

Sum of the first 3 natural numbers is: 14
```

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

```
FirstName = (input("Enter a First Name: "))
LastName = (input("Enter a Last Name: "))
print("First Name is",FirstName,"and Last Name is",LastName)

Enter a First Name:  Riya
Enter a Last Name:  Bhimani

First Name is Riya and Last Name is Bhimani
```

13) WAP to swap two numbers.

```
a = int(input("Enter a number a : "))
b = int(input("Enter a number b : "))
temp=0
temp=a
a=b
b=temp
print("a = ",a)
print("b = ",b)

Enter a number a :  2
Enter a number b :  3

a =  3
b =  2
```

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

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

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


print(kilometers,"kilometers is equal to:")
print(meters,"meters")
print(feet,"feet")
print(inches,"inches")
print(centimeters,"centimeters")

Enter the distance in kilometers:  5

5.0 kilometers is equal to:
5000.0 meters
16404.2 feet
```

```
196850.5 inches
500000.0 centimeters
```

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

```
day = int(input("Enter a day : "))
month = int(input("Enter a month : "))
year = int(input("Enter a year : "))
print(day,"-",month,"-",year)
```

```
Enter a day :  23
Enter a month :  11
Enter a year :  2024

23 - 11 - 2024
```

if..else..

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

```
a = int (input('Enter a number : '))

if(a>0):
    print(a ,'is positive.')
else:
    print(a ,'is negative')

Enter a number :  2

2 is positive.
```

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

```
a = int (input('Enter a number : '))

if(a%2 == 0):
    print(a ,'is Even.')
else:
    print(a ,'is Odd')

Enter a number :  3

3 is Odd
```

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

```
# Simple If

a = int (input('Enter a number : '))
b = int (input('Enter a number : '))

if(a>b):
    print(a ,'is Largest number.')
else:
    print(b ,'is Largest number.')

Enter a number :  2
Enter a number :  4

4 is Largest number.
```

```
# Ternary Operator

a = int (input('Enter a number : '))
b = int (input('Enter a number : '))

print(a , 'is largest number.') if(a>b) else print(b , 'is largest
number.')

Enter a number :  2
Enter a number :  4

4 is largest number.
```

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

```
# Simple If_elseif

a = int (input('Enter a first number a : '))
b = int (input('Enter a second number b : '))
c = int (input('Enter a third number c : '))

if(a>b):
    if(a>c):
        print(a ,'is Largest number.')
    else:
        print(c ,'is Largest number.')
else:
    if(b>c):
        print(b ,'is Largest number')
    else:
        print(c ,'is Largest number.')

Enter a first number a :  1
Enter a second number b :  2
Enter a third number c :  3

3 is Largest number.

# Ternary Operator

a = int (input('Enter a first number a : '))
b = int (input('Enter a second number b : '))
c = int (input('Enter a third number c : '))

largest = a if (a > b and a > c) else (b if b > c else c)

print("The largest number is:", largest)

Enter a first number a :  1
Enter a second number b :  2
Enter a third number c :  3
```

```
The largest number is: 3
```

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]

```
a = int (input('Enter a Year : '))

if(a%4==0):
    if(a%100!=0):
        print(a ,'is Leap Year.')
    else:
        print(a ,'is Leap not Year.')
elif(a%400==0):
    print(a ,'is Leap Year.')
else:
    print(a ,'is not Leap Year.')
```

```
Enter a Year :  2023

2023 is not Leap Year.
```

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

```
d = int(input('Enter a number between 0 to 6 : '))

if(d==0):
    print('Sunday')
elif(d==1):
    print('Monday')
elif(d==2):
    print('Tuesday')
elif(d==3):
    print('Wednesday')
elif(d==4):
    print('Thursday')
elif(d==5):
    print('Friday')
elif(d==6):
    print('Saturday')
else:
    print('Please Enter A valid Number')
```

```
Enter a number between 0 to 6 :  3

Wednesday
```

```
d = int(input('Enter a number between 0 to 6 : '))

match d:
    case 0:
        print('Sunday')
    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 _:
        print('Please Enter A valid Number')
```

```
Enter a number between 0 to 6 :  4

Thursday
```

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

```
a = int (input('Enter a first number a : '))
b = int (input('Enter a second number b : '))
c = int (input('Enter 1 for Addition , Enter 2 for Subtraction , Enter
3 for Multiplication , Enter 4 for Division'))

if(c==1):
    print(a,'+',b,'=', a+b)
elif(c==2):
    print(a,'-',b,'=', a-b)
elif(c==3):
    print(a,'*',b,'=', a*b)
elif(c==4):
    print(a,'/',b,'=', a/b)
else:
    print('Please enter valid operator')
```

```
Enter a first number a :  3
Enter a second number b :  4
Enter 1 for Addition , Enter 2 for Subtraction , Enter 3 for
Multiplication , Enter 4 for Division 1

3 + 4 = 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

```
a = int (input('Enter a first number a : '))
b = int (input('Enter a second number b : '))
c = int (input('Enter a third number c : '))
d = int (input('Enter a fourth number d : '))
e = int (input('Enter a fifth number e : '))
Total = a+b+c+d+e
print ('Total marks = ' , Total)
Percentage = (Total*100)/500
print ('Percentage = ' , Percentage)

if(Percentage>70):
    print('Distinction')
elif(Percentage>=60):
    if(Percentage<=70):
        print('First Class')
elif(Percentage>=45):
    if(Percentage<60):
        print('Second Class')
elif(Percentage>=35):
    if(Percentage<45):
        print('Pass')
if(Percentage<35):
    print('Fail')

Enter a first number a :  89
Enter a second number b :  58
Enter a third number c :  90
Enter a fourth number d :  99
Enter a fifth number e :  99

Total marks =  435
Percentage =  87.0
Distinction
```

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.

```
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 b + c > a and a + c > b:
    # Check for equilateral triangle
    if a == b == c:
        print("The triangle is Equilateral.")
    # Check for isosceles triangle
    elif a == b or b == c or a == c:
        print("The triangle is Isosceles.")
    elif a**2 + b**2 == c**2 or b**2 + c**2 == a**2 or a**2 + c**2 ==
b**2 and a!=b!=c:
        print("The triangle is Right-Angled and Scalene.")
    # Check for right-angled triangle using Pythagoras theorem
    elif a**2 + b**2 == c**2 or b**2 + c**2 == a**2 or a**2 + c**2 ==
b**2:
        print("The triangle is Right-Angled.")
    # If none of the above, it's a scalene triangle
    #else:
        #print("The triangle is Scalene.")
else:
    print("The sides do not form a valid triangle.")

Enter the first side:  3
Enter the second side:  4
Enter the third side:  5

The triangle is Right-Angled and Scalene.
```

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

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

if (num1 >= num2 and num1 <= num3) or (num1 <= num2 and num1 >= num3):
    second_largest = num1
elif (num2 >= num1 and num2 <= num3) or (num2 <= num1 and num2 >=
num3):
    second_largest = num2
else:
    second_largest = num3

print("The second largest number is:", second_largest)

Enter the first number:  1
Enter the second number:  2
Enter the third number:  3

The second largest number is: 2.0
```

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

```python
units = float(input("Enter the number of units consumed: "))

if units <= 50:
    bill = units * 2.60
elif units <= 100:
    bill = (50 * 2.60) + (units - 50) * 3.25
elif units <= 200:
    bill = (50 * 2.60) + (50 * 3.25) + (units - 100) * 5.26
else:
    bill = (50 * 2.60) + (50 * 3.25) + (100 * 5.26) + (units - 200) *
8.45

print("The total electricity bill is: Rs.",bill)
```

```
Enter the number of units consumed:  30

The total electricity bill is: Rs. 78.0
```

for and while loop

01) WAP to print 1 to 10.

```
for i in range (1,11) :
    print(i)
```

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

02) WAP to print 1 to n.

```
n = int ( input ("Enter n : ") )
for i in range (1,n+1) :
    print(i)
```

```
Enter n :  5

1
2
3
4
5
```

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

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

```
Enter n :  10

1
3
5
7
9
```

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

```
n1 = int ( input ("Enter number 1 : "))
n2 = int ( input ("Enter number 2 : "))
for i in range (n1,n2+1):
    if(i%2==0):
        if(i%3!=0):
            print(i)

Enter number 1 :  1
Enter number 2 :  8

2
4
8
```

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

```
n = int ( input ("Enter n : ") )
sum = 0
for i in range (1,n+1) :
    sum = sum+i
else:
    print('Answer = ' , sum)

Enter n :  4

10
```

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

```
n = int ( input ("Enter n : ") )
sum = 0
for i in range (1,n+1,1) :
    sum = sum + (i*i)
else:
    print('Answer = ' , sum)

Enter n :  3

14
```

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

```
n = int ( input ("Enter n : ") )
sum = 0
for i in range (1,n+1) :
    if (i%2==0):
        sum = sum - i
    else:
        sum = sum + i
else:
    print('Answer = ' , sum)

Enter n :  3

Answer =  2
```

08) WAP to print multiplication table of given number.

```
n = int ( input ("Enter n : ") )
ans = 1
for i in range (1,11) :
    ans = n*i
    print(n , '*' , i ,'=' ,ans)

Enter n :  5

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

09) WAP to find factorial of the given number.

```
n = int ( input ("Enter n : ") )
fact = 1
for i in range (1,n+1) :
    fact = fact * i
print('Factorial of' , n , 'is = ',fact)

Enter n :  5

Factorial of 5 is =  120
```

10) WAP to find factors of the given number.

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

Enter n :  6
```

```
1
2
3
6
```

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

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

```
Enter n :  5
```

```
5  is prime.
```

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

```
n = int ( input ("Enter n : ") )
ans=0
rem=0
sum=0

while (n>0):
    rem = n%10
    sum = sum + rem
    n = int(n/10)        # n = n//10
else:
    print('Sum = ' ,sum)
```

```
Enter n :  123
```

```
Sum =  6
```

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

```
n = int ( input ("Enter n : ") )
rev=0
rem=0
num=n

while(n>0):
    rem = n%10
    rev = (rev*10)+rem
    n = int(n/10)        # n = n//10
```

```
print(rev)
if(rev==num):
    print('Palindrome')
else:
    print('Not Palindrome')
```

```
Enter n :  153

351
Not Palindrome
```

14) WAP to print GCD of given two numbers.

```
n1 = int ( input ("Enter number 1 : "))
n2 = int ( input ("Enter number 2 : "))
min = n1;
gcd = 0;

if(n1>n2):
    min=n1
for i in range (1,min+1):
    if(n1%i==0):
        if(n2%i==0):
            gcd = i
print(gcd)
```

```
Enter number 1 :  10
Enter number 2 :  5

5
```

String

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

```
a = input("Enter a String : ")
b = a[::-1]
if(a==b):
    print(a , " is Palindrome!!")
else:
    print(a , " is not Palindrome!!")
```

```
Enter a String :  RIYA

RIYA  is not Palindrome!!
```

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

```
str = "Riya Bhimani"
str1 = str.split(' ')
str2 = str1[::-1]
str3 = ' '.join(str2)
print(str3)
```

```
Bhimani Riya
```

03) WAP to remove ith character from given string.

```
s = input("Enter the string: ")
i = int(input("Enter the position of the character to remove: "))
if 0 <= i < len(s):
    result = s[:i] + s[i+1:]
    print("String after removing the character:", result)
else:
    print("Invalid position!")
```

```
Enter the string:  riya
Enter the position of the character to remove:  2

String after removing the character: ria
```

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

```
str1 = input("Enter a String : ")
count=0
for i in str1:
    count+=1
print("Length of " , str1 , " = " , count)
```

```
Enter a String :  Riya

Length of  Riya  =  4
```

05) WAP to print even length word in string.

```
str = input("Enter a String : ")
str1 = str.split()
for i in str1:
    if len(i)%2==0:
        print(i)

Enter a String :  Bhimani Riya

Riya
```

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

```
str1 = input("Enter a String : ")
count=0
for i in str1:
    if 'a' in i or 'e' in i or 'i' in i or 'o' in i or 'u' in i:
        count+=1
print(count)

Enter a String :  Bhimani

3
```

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

```
s = input("Enter a string: ")
words = s.split()
result = []
for word in words:
    if len(word) > 1:
        word = word[0].upper() + word[1:-1] + word[-1].upper()
    else:
        word = word.upper()

    result.append(word)

final_result = " ".join(result)
print("Modified string:", final_result)

Enter a string:  bhimani

Modified string: BhimanI
```

08) WAP to convert given array to string.

```
arr = ["Hello", "World", "from", "me"]
result = " ".join(arr)
print(result)

Hello World from me
```

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

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

```
Password = input("Enter a Password:")
ComfirmPassword = input("Enter a ComfirmPassword:")
if Password == ComfirmPassword:
    print("Password Matched")
elif Password.lower() == ComfirmPassword.lower():
    print("Case Mistake")
else:
    print("Invalid Password")

Enter a Password: Riya
Enter a ComfirmPassword: Bhimani

Invalid Password
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

```
CardNo = "1234 5678 9012 3456"
Li = CardNo.split()
Output = "**** **** **** " + Li[3]
print(Output)

**** **** **** 3456

CardNo = "1234 5678 9012 3456"
Output = "**** **** **** " + CardNo[-4:]
print(Output)

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

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

s1 = decimal and s2 = medical are Anagram

```
s1 = "decimal"
s2 = "medical"
if len(s1) != len(s2):
    print("The strings are not Anagrams.")
else:
    if sorted(s1) == sorted(s2):
        print("The strings are Anagrams.")
    else:
        print("The strings are not Anagrams.")
```

```
The strings are Anagrams.
```

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

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

```
str = input("Enter a String : ")
str1=''
str2=''
for i in str:
    if i.islower():
        str1+=i
    else:
        str2+=i
str3 = str1 + str2
print(str3)
```

```
Enter a String :  EHlsarwiwhtwMV

lsarwiwhtwEHMV
```

List

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

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

```
15
```

02) WAP to find largest element in a List.

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

```
5
```

03) WAP to find the length of a List.

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

```
5
```

```
print(len(l1))
```

```
5
```

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

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

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

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

```
l = [1, 2, 3, 4, 5, 6, 7, 8]
mid = len(l) // 2
first_part = l[:mid]
second_part = l[mid:]
result = second_part + first_part
print("Modified list:", result)
```

```
Modified list: [5, 6, 7, 8, 1, 2, 3, 4]
```

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

```
l1=[]
for i in range(1,6,1):
    l1.append(int(input(("Enter Element "))))

a=int(input("Enter index 1: "))
b=int(input("Enter index 2: "))

temp=l1[a]
l1[a]=l1[b]
l1[b]=temp

print(l1)
```

```
Enter Element  1
Enter Element  2
Enter Element  3
Enter Element  4
Enter Element  5
Enter index 1:  2
Enter index 2:  3

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

07) WAP to reverse the list entered by user.

```
l1=[1,2,3,4,5]

l2=l1[::-1]
print(l2)
```

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

08) WAP to print even numbers in a list.

```
l1=[1,2,3,4,5]

for i in l1:
    if i%2==0:
        print(i)

2
4
```

09) WAP to count unique items in a list.

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

3
```

10) WAP to copy a list.

```
l1=[1,2,3,4,5]
l2=l1.copy()
print(l2)

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

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

```
n1 = int(input("Enter starting range Number: "))
n2 = int(input("Enter ending range number: "))

l1=[i for i in range(n1,(n2+1)) if i%2!=0]
print(l1)

Enter starting range Number:  1
Enter ending range number:  5

[1, 3, 5]

n1 = int(input("Enter starting range Number: "))
n2 = int(input("Enter ending range number: "))

l1=[i for i in range(n1,(n2+1),2)]
print(l1)

Enter starting range Number:  1
Enter ending range number:  5
```

```
[1, 3, 5]
```

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

```
l1=[1,2,3,4,5,4,3]
n=int(input("Enter number: "))
print(l1.count(n))

Enter number:  3

2
```

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

```
l1=[1,2,8,4,5]
l1.sort(reverse=True)
print("Second largest Number=" ,l1[1])

Second largest Number= 5
```

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

```
l1 = [1, 2, 8, 4, 5, 4, 8, 5, 4, 8, 5]
k = 2
result = []
for i in set(l1):
    if l1.count(i) > k:
        result.append(i)

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

Elements with frequency greater than 2 : [4, 5, 8]
```

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

```
l1=[1,2,3,4,5]
l2=[]
for i in l1:
    j=i**2
    l2.append(j)
print(l2)

[1, 4, 9, 16, 25]

l1=[1,2,3,4,5]
ans = [i**2 for i in l1]
print(ans)

[1, 4, 9, 16, 25]
```

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

```
fruits = input("Enter a list of fruits separated by commas: 
").split(',')
fruits_starting_with_b = [fruit.strip() for fruit in fruits if 
fruit.strip().startswith('b')]
print("Fruits starting with 'b':", fruits_starting_with_b)
```

```
Enter a list of fruits separated by commas:  banana,Banana,Mango,biii

Fruits starting with 'b': ['banana', 'biii']
```

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

```
l1=[1,2,3,4,5]
l2=[10,2,13,4,15]
common_list=[]
for i in l1:
    for j in l2:
        if i==j:
            common_list.append(i)
print(common_list)
```

```
[2, 4]
```

Tuple

01) WAP to find sum of tuple elements.

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

```
15
```

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

```
data = (10, 20, 5, 7, 30, 15, 25)
K = 3
sorted_data = sorted(data)
min_k_elements = sorted_data[:K] # [:k] This means "slice from the start of the list up to (but not including) index k."
max_k_elements = sorted_data[-K:] # [-k:] This means "slice from index -k (kth element from the end) to the end of the list."
print("Minimum K elements:", min_k_elements)
print("Maximum K elements:", max_k_elements)
```

```
Minimum K elements: [5, 7, 10]
Maximum K elements: [20, 25, 30]
```

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

```
t1 = (10,20,30)
t2 = (3,6,9)
t3 = (7,14,21)
l = [t1,t2,t3]
k = int(input("Enter K:"))
for i in l:
    count = 0
    for j in i:
        if j%k==0:
            count = count+1
        if count == len(i):
            print(i)
```

```
Enter K: 2
```

```
(10, 20, 30)
```

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

```python
Li=[1,2,3,4,5]
ans = [(i,i**3) for i in Li]
print(ans)
```

```
[(1, 1), (2, 8), (3, 27), (4, 64), (5, 125)]
```

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

```python
t1 = (10,20,30)
t2 = (3,6,-9)
t3 = (7,-14,21)
l = [t1,t2,t3]
for i in l:
    count=0
    for j in i:
        if j>0:
            count=count+1
    if count==len(i):
        print(i)
```

```
(10, 20, 30)
```

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

```python
t=(1,2,3)
l=[7,8,9,10]
l.append(t)
print(l)
```

```
[7, 8, 9, 10, (1, 2, 3)]
```

```python
t=(1,2,3)
l=[7,8,9,10]
c_t = t + (l,)
print(c_t)
```

```
(1, 2, 3, [7, 8, 9, 10])
```

07) WAP to remove tuples of length K.

```python
tuples_list = [(1, 2), (3, 4, 5), (7, 8, 9, 10), (11, 12)]
K = 2

filtered_list = []
```

```
for t in tuples_list:
    count = 0
    for _ in t:
        count += 1
    if count != K:
        filtered_list.append(t)

print("Filtered list:", filtered_list)

#[[The underscore (_) is a conventional placeholder name in Python for
a variable whose value is not used.
#It tells readers and Python itself that the loop variable is not
important in this context.]]
```

```
Filtered list: [(3, 4, 5), (7, 8, 9, 10)]
```

08) WAP to remove duplicates from tuple.

```
my_tuple = (1, 2, 3, 2, 4, 5, 3, 6, 7)
unique_tuple = tuple(set(my_tuple))
print("Tuple after removing duplicates:", unique_tuple)
```

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

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

```
my_tuple = (2, 3, 4, 5)
result = [my_tuple[i] * my_tuple[i + 1] for i in range(len(my_tuple) -
1)]
result_tuple = tuple(result)
print("Resultant tuple after multiplying adjacent elements:",
result_tuple)
```

```
Resultant tuple after multiplying adjacent elements: (6, 12, 20)
```

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

```
my_tuple = (2, 3, 4, 5, 5)
flag = True
for i in my_tuple:
    if my_tuple.count(i) != 1:
        flag = False
        break
if flag == False:
    print("Not distinct")
else:
    print("Distinct")
```

```
Not distinct
```

Set & Dictionary

01) WAP to iterate over a set.

```
s1={1,2,3,4,5}
for i in s1:
    print(i)
```

```
1
2
3
4
5
```

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

```
s1={1,2,3,4,5}
l1=list(s1)
str1=str(s1)
t1=tuple(s1)
print("Set:" ,s1)
print("List:" ,l1)
print("String:" ,str1)
print("Tuple:" ,t1)
```

```
Set: {1, 2, 3, 4, 5}
List: [1, 2, 3, 4, 5]
String: {1, 2, 3, 4, 5}
Tuple: (1, 2, 3, 4, 5)
```

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

```
s1={1,2,3,4,5}
min=i
max=i
for i in s1:
    if i>max:
        max=i
    if i<min:
        min=i
print("Maximum number is:" ,max)
print("Minimum number is:" ,min)
```

```
Maximum number is: 5
Minimum number is: 1
```

04) WAP to perform union of two sets.

```
s1={1,2,3,4,5}
s2={5,6,7,8,9}
s1.union(s2)
#{1,2,3,4,5}.union({5,6,7,8,9})
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

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

```
l1=[1,2,3,4,5]
l2=[6,7,8,9]
count=0
for i in l1:
    for j in l2:
        if(i==j):
            count+=1
if(count!=0):
    print("have common element")
else:
    print("have not common element")
```

```
have not common element
```

06) WAP to remove duplicates from list.

```
l1=[1,2,3,4,5,3,4]
s1=set(l1)
print(list(s1))

# second method
l2=[]
for i in l1:
    if i not in l2:
        l2.append(i)
print(l2)
```

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

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

```
s1="hi hello how hi are"
str2=set(s1.split())
print(str)

{'hello', 'how', 'are', 'hi'}
```

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

```
s1={1,2,3,4,5}
s2={5,6,7,8,9}
s1.difference(s2)

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

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

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

```
l1=[1,2,3]
l2=[2,3,4]
l3=[2,5,6]
s1=set(l1)
s2=set(l2)
s3=set(l3)
s1 & s2 & s3

{2}
```

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

```
str1= "Darshan University"
vowel={'a','e','i','o','u','A','E','I','O','U'}
count=0
for i in str1:
    if i in vowel:
        count+=1
print(count)

6
```

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

```
str1= input("Enter a String:")
s1={'0','1'}
for i in str1:
    if i not in s1:
        print("Not Binary")
else:
    print("Binary")

Enter a String: 010

Binary
```

13) WAP to sort dictionary by key or value.

```
d1={101:'abc',102:'xyz',100:'def'}
d2=list(d1)
d2.sort()
print(d2)
d3=list(d1.values())
d3.sort()
print(d3)

[100, 101, 102]
['abc', 'def', 'xyz']
```

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

```
dict={'abc':101,'pqr':102,'xyz':103}
sum=0
for i in dict:
    sum=sum+dict[i]
print(sum)

306
```

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.

```
dict =  {'a':100 ,'b':200 ,'g':300 ,'d':400 ,'e':500}
key = input('Enter Key :')
if key in dict:
    print(dict[key])
else:
    print('Key Not Found!')

Enter Key : f

Key Not Found!
```

User Defined Function

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

```
mass = int(input("Enter Mass:"))
h = int(input("Enter height:"))
def calBMI(mass,height):
    BMI = mass/h**2
    print(BMI)
calBMI(mass,h)

Enter Mass: 4
Enter height: 2

1.0
```

02) Write a function that add first n numbers.

```
n = int(input("Enter n:"))
def addNnumbers(n):
    sum=0
    for i in range(n+1):
        sum = sum+i
    print(sum)
addNnumbers(n)

Enter n: 5

15
```

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

```
n = int(input("Enter n:"))
def prime(n):
    count=0
    j=2
    while(j<=n/2):
        if n%j==0:
            return 0
        j+=1
    else:
        return 1
prime(n)
```

```
Enter n: 4

0
```

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

```
n1 = int(input("Enter n1:"))
n2 = int(input("Enter n2:"))
def prime(n):
    count=0
    j=2
    while(j<=n/2):
        if n%j==0:
            return 0
        j+=1
    else:
        return 1
def listPrime(n1,n2):
    list=[]
    for i in range(n1,n2+1):
        if prime(i)==1:
            print(i)
            list.append(i)
    print(list)
listPrime(n1,n2)

Enter n1: 1
Enter n2: 6

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

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

```
s1=input("Enter a string:")
def palindrome(s1):
    s2=s1[::-1]
    if(s1==s2):
        return True
    else:
        return False
palindrome(s1)

Enter a string: abccba
```

```
True
```

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

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

```
Sum =  15
```

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

```
l1=[(1,2,3),(4,5,6),(7,8,9)]
l2=[]
def sum(l1):
    sum=0
    for i in l1:
        sum=sum+i[0]
    print(sum)
sum(l1)
```

```
12
```

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

```
def fibonacci(n):
    if n <= 0:
        return "Invalid Input"
    elif n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

nthTerm = 10
print("The",nthTerm,"th term of the Fibonacci series
is",fibonacci(nthTerm))
```

```
The 10 th term of the Fibonacci series is 34
```

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

```python
def get_student_name(rollno, students):
    return students.get(rollno, "Roll number not found")

dict1 = {101: 'Ajay', 102: 'Rahul', 103: 'Jay', 104: 'Pooja'}
rollno = int(input("Enter rollNo:"))
print("The name of the student with roll
number",rollno,"is",get_student_name(rollno, dict1))
```

```
Enter rollNo: 103

The name of the student with roll number 103 is Jay
```

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

```python
scores = [200, 456, 300, 100, 234, 678]
def sumOfLast0(scores):
    sum=0
    i=0
    for i in scores:
        if(i%10==0):
            sum=sum+i
    print("Sum=",sum)
sumOfLast0(scores)
```

```
Sum= 600
```

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'}

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

def invert_dict(input_dict):
    return {value: key for key, value in input_dict.items()}

inverted_dict = invert_dict(original_dict)
```

```
print("Before: ",original_dict)
print("After: ",inverted_dict)
```

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

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.

```
str1="the quick brown fox jumps over the lazy dog"
def pangram2(str1):
    for i in 'qwertyuioplkjhgfdsazxcvbnm':
        if i not in str1:
            return False
    return True
pangram2(str1)
```

```
True
```

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

```
s1 = "AbcDEfgh"
def countAlpha(s1):
    Lcount=0
    Ucount=0
    for i in s1:
        if(i.isupper()==True):
            Ucount+=1
        else:
            Lcount+=1
    print("Lowercase characters=",Lcount)
    print("Uppercase characters=",Ucount)
countAlpha(s1)
```

```
Lowercase characters= 5
Uppercase characters= 3
```

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

```
a=int(input("Enter a:"))
b=int(input("Enter b:"))
```

```
Small = lambda a,b : a if a>b else b
Small(a,b)

Enter a: 1
Enter b: 2

2
```

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

```
students=['Riya','ShrutiPatel','RadhuPatel']
ans=list(filter(lambda x:len(x)>7,students))
ans

['ShrutiPatel', 'RadhuPatel']
```

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

```
students=['riya','shrutiPatel','radhuPatel']
capitalized_names = list(map(lambda name: name.capitalize(), 
students))
print(capitalized_names)

['Riya', 'Shrutipatel', 'Radhupatel']
```

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

```
# Positional Args
def positional_args(a, b):
    return a + b
result1 = positional_args(5, 3)
print("Addition By Positional Arguments:", result1)

Addition By Positional Arguments: 8

#Keyword Args
def keyword_args(a, b):
    return a + b
result2 = keyword_args(a=5, b=3)
print("Addition By Keyword Arguments:", result2)

Addition By Keyword Arguments: 8
```

```
#Default Args
def default_args(a, b=10):
    return a + b
result3a = default_args(15)
result3b = default_args(15, 5)
print("Addition By Default Arguments:", result3a, result3b)
```

```
Addition By Default Arguments: 25 20
```

```
#Variable Legngth Positional(*args) & variable length Keyword
Arguments (**kwargs)
def variable_args(*args, **kwargs):
    sum_args = sum(args)
    concatenated_kwargs = " ".join(f"{key}={value}" for key, value in
kwargs.items())
    return f"Sum of args: {sum_args}, Kwargs: {concatenated_kwargs}"
result4 = variable_args(1, 2, 3, name="Alice", age=25)
print("Variable Length Arguments:", result4)
```

```
Variable Length Arguments: Sum of args: 6, Kwargs: name=Alice age=25
```

```
#Keyword-Only & Positional Only Arguments
def mixed_args(a, /, b, *, c):
    return f"Positional-Only: {a}, Regular: {b}, Keyword-Only: {c}"
result5 = mixed_args(1, b=2, c=3)
print("Keyword-Only & Positional-Only Arguments:", result5)
```

```
Keyword-Only & Positional-Only Arguments: Positional-Only: 1, Regular:
2, Keyword-Only: 3
```

File I/O

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

```
file_path = "new.txt"
try:
    # Read as a single string
    with open(file_path, "r") as file:
        content = file.read()
        print("=== File Content as String ===")
        print(content)

    # Read line by line
    with open(file_path, "r") as file:
        print("\n=== File Content Line by Line ===")
        for line in file:
            print(line, end="")

    # Read into a list
    with open(file_path, "r") as file:
        content_list = file.readlines()
        print("\n\n=== File Content as List ===")
        print(content_list)

except FileNotFoundError:
    print("Error: File not found. Check the path!")
```

```
=== File Content as String ===
10
20
30
40
50


=== File Content Line by Line ===
10
20
30
```

```
40
50


=== File Content as List ===
['10\n', '20\n', '30\n', '40\n', '50\n']
```

02) WAP to create file named "new.txt" only if it doesn't exist.

```python
f = open ("new.txt" , "w")
f.close()
```

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

```python
f = open ("new.txt" , "r")
for i in range(0,5):
    print(f.readline())
f.close()
```

```
This

Is

A

Python

Subject
```

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

```python
fp = open("newcopy.txt","r")
li = fp.readlines()
max = 0
for i in li:
    nw = i.split()
    for j in nw:
        if len(j)>max:
            max = len(j)
            lw = j
print(lw)
fp.close()
```

```
Subject
```

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

```
f = open ("new.txt" , "r")
Li=f.readlines()
NoOfLines = len(Li)
NoOfWords = sum(len(i.split()) for i in Li)
NoOfCharacters = sum(len(i) for i in Li)

print("Number of Lines:" , NoOfLines)
print("Number of Words:" , NoOfWords)
print("Number of Characters:" , NoOfCharacters)

f.close()
```

```
Number of Lines: 7
Number of Words: 7
Number of Characters: 32
```

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

```
f1 = open ("new.txt" , "r")
f2 = open ("newcopy.txt" , "w")

f2.write(f1.read())
print("Coppied successfully !!")

f1.close()
f2.close()
```

```
Coppied successfully !!
```

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

```
with open("new.txt", "rb") as fp:
    file_size = fp.seek(0, 2)
print(f"The size of the file is: {file_size} bytes")
fp.close()
```

```
The size of the file is: 32 bytes
```

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

```
f = open("new.txt" , "r")
def NameFrequency(f):
    nThis = 0
    nIs=0
    for i in f:
        nThis += i.split().count('This')
        nIs += i.split().count('Is')
    print(nThis)
    print(nIs)
NameFrequency(f)
f.close()
```

```
2
2
```

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.

```
li = []
fp = open('Score.txt', "w+")
for i in range(5):
    score = int(input(f'Enter score for subject {i+1}: '))
    li.append(score)
    fp.write(str(score) + "\n")
fp.seek(0)
print("Stored scores in file:")
print(fp.read())
fp.close()
max_score = max(li)
print('Highest score over these 5 subjects is:', max_score)
```

```
Enter score for subject 1:  10
Enter score for subject 2:  20
Enter score for subject 3:  30
Enter score for subject 4:  40
Enter score for subject 5:  50

Stored scores in file:
10
20
30
40
50

Highest score over these 5 subjects is: 50
```

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

(Note: each number should be in new line)

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

with open("primenumbers.txt", "w") as file:
    count = 0
    num = 2
    while count < 100:
        if is_prime(num):
            file.write(str(num) + "\n")
            count += 1
        num += 1
```

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

```
def merge_files(file1, file2, output_file):
    with open(file1, "r") as f1, open(file2, "r") as f2, 
open(output_file, "w") as out:
        out.write(f1.read() + "\n" + f2.read())

file1 = "File.txt"
file2 = "File2.txt"
output_file = "MergedFile.txt"

try:
    merge_files(file1, file2, output_file)
    print(f"Files merged successfully into {output_file}")
except FileNotFoundError:
    print("One or both input files are missing.")
```

```
Files merged successfully into MergedFile.txt
```

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

```
w1=input("Enter the word to be updated:")
w2=input("Enter the updated word:")
fp=open("new.txt","r")
data=fp.read().split()
data1=str(data)
data2=data1.replace(w1,w2)
fp1=open("newcopy.txt","w")
fp1.write(data2)
fp.close()
fp1.close()
```

```
Enter the word to be updated: pretty
Enter the updated word: beautiful
```

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

```
with open("sample.txt", "w") as f:
    f.write("Hello, this is a sample text file.")

with open("sample.txt", "rb") as f:
    # Case 1: tell() at the beginning
    print("Initial position:", f.tell())  # Should be 0

    # Case 2: seek() from the beginning
    f.seek(7)  # Move to the 7th byte (0-based index)
    print("Position after seek(7):", f.tell())
    print("Character at new position:", f.read(5).decode())  # Read 5
characters

    # Case 3: seek() from the current position (Must be in binary
mode)
    f.seek(3, 1)  # Move 3 bytes forward from current position
    print("Position after seek(3, 1):", f.tell())
    print("Character at new position:", f.read(5).decode())  # Read 5
characters

    # Case 4: seek() from the end
    f.seek(-6, 2)  # Move 6 bytes before the end of file
    print("Position after seek(-6, 2):", f.tell())
    print("Character at new position:", f.read().decode())  # Read
till end
```

```
Initial position: 0
Position after seek(7): 7
Character at new position: this
Position after seek(3, 1): 15
Character at new position: a sam
Position after seek(-6, 2): 28
Character at new position:  file.
```

Python Programming - 2301CS404 Lab - 10
Riya Bhimani | 23010101030 | 03-02-2025

Exception Handling

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.

```
try:
    b=0
    a=int(input("Enter a:"))
    c=int(input("Enter num:"))
    print(a/b)
    print(a+'a')
except ZeroDivisionError:
    print("ZeroDivisionError Occured !")
except ValueError:
    print("ValueError Occured !")
except TypeError:
    print("TypeError Occured !")
```

```
Enter a: 10
Enter num: 10

ZeroDivisionError Occured !
```

```
try:
    b=0
    a=int(input("Enter a:"))
    c=int(input("Enter num:"))
    print(a/b)
    print(a+'a')
except (ZeroDivisionError,ValueError,TypeError) as error:
    print(error)
```

```
Enter a: 10
Enter num: str

invalid literal for int() with base 10: 'str'
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

```
li=[0,1,2,3,4,5]
try:
    print(li[10])
except IndexError as msg:
    print("Index Error :",msg)
```

```
Index Error : list index out of range
```

```
dic = {101:'abc',102:'pqr',103:'xyz'}
try:
    print(dic[5])
except KeyError as msg:
    print("Key Error Occured...:",msg)
```

```
Key Error Occured...: 5
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

```
try:
    fp=open('abc.txt',"r")
    fp.read()
except FileNotFoundError as error:
    print("FileNotFoundError:",error)
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'abc.txt'
```

```
try:
    import maths
except ModuleNotFoundError as error:
    print("ModuleNotFoundError:",error)
```

```
ModuleNotFoundError: No module named 'maths'
```

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

```
try:
    b=0
    a=int(input("Enter a:"))
    c=int(input("Enter num:"))
    print(a/b)
    print(a+'a')
    print(d)
    e='agsjefh'
    print(e[10])
```

```
except (ZeroDivisionError,ValueError,TypeError,NameError,IndexError)
as error:
    print(error)

Enter a: 10

string index out of range
```

05) WAP to demonstrate else and finally block.

```
try:
    fp=open('abcd.txt',"r")
except FileNotFoundError as error:
    print(error)
else:
    print(fp.read())
    fp.close()
finally:
    print("This block is always executed !!")

abcdefghijk
This block is always executed !!
```

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

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

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

```
try:
    gradeString = input("Enter a string by comma separated:")
    gradeList = [ int(i) for i in gradeString.split(',') ]
    print(gradeList)
except ValueError as error:
    print(error)

Enter a string by comma separated: hqwb,bwgh,shwjb,wjvd

invalid literal for int() with base 10: 'hqwb'
```

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

```
def divide(a,b):
    try:
        print(a/b)
    except ZeroDivisionError as error:
        print(error)
divide(12,0)

division by zero
```

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.

```
try:
    age=int(input("Enter age:"))
    if age<18:
        raise ValueError("Enter Valid Age")
    else:
        print(age)
except ValueError as error:
    print(error)

Enter age: 13

Enter Valid Age
```

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.

```
class InvalidUsernameError(Exception):
    pass
try:
    UserName = input("Enter UserName :")
    length = len(UserName)
    if len(UserName) < 5 or len(UserName) > 15:
        raise InvalidUsernameError ("Username must be between 5 and 15 
characters long")
    else:
        print(UserName)
except InvalidUsernameError as error:
    print(error)

Enter UserName : riya
```

```
Username must be between 5 and 15 characters long
```

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.

```
import math
class NegativeNumberError(Exception):
    def __init__(self,msg):
        self.msg = msg
try:
    n = int(input("Enter a number:"))
    if(n<0):
        raise NegativeNumberError ("Cannot calculate the square root 
of a negative number")
    else:
        print(math.sqrt(n))
except NegativeNumberError as error:
    print(error)

Enter a number: -4

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

Modules

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

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

```
import Calculator

num1 = int(input("Enter number 1 :"))
num2 = int(input("Enter number 2 :"))

print("Addition = ",Calculator.Addition(num1,num2))
print("Subtraction = ",Calculator.Subtraction(num1,num2))
print("Multiplication = ",Calculator.Multiplication(num1,num2))
print("Division = ",Calculator.Division(num1,num2))
```

```
Enter number 1 : 12
Enter number 2 : 4

Addition =  16
Subtraction =  8
Multiplication =  48
Division =  0
```

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

```
import random
s = input("Enter a String: ")
i = random.randrange(0,len(s))
print("Random Character of Given String = ",s[i])
```

```
Enter a String:  riya

Random Character of Given String =  i
```

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

```
l1=[1,2,3,4,5]
randomelement = random.choice(l1)
print("Random Element of Given List = ",randomelement)
```

```
Random Element of Given List =  5
```

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

```
import random
random.seed(3)
print("Random same number=",random.randint(1,6))

Random same number= 2
```

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

```
import random
for i in range(0,3):
    Random_Integer = random.randrange(100,999,5)
    print("Random Integer = ",Random_Integer)

Random Integer =  855
Random Integer =  795
Random Integer =  265
```

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

```
li=[]
for i in range(0,100):
    n = random.randint(0,100)
    li.append(n)
winner = random.choice(li)
runnerup = random.choice(li)
while(winner == runnerup):
    runnerup = random.choice(li)
print("Winner = ",winner)
print("Runner Up = ",runnerup)

Winner =  100
Runner Up =  60
```

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

```
import datetime
print("Current Date = ",datetime.datetime.today())

Current Date =  2025-02-10 13:21:41.753344
```

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

```
today = datetime.datetime.now()
df1 = today- datetime.timedelta(days=7)
print("current date : ",today)
print("before 7 days : ",df1)
```

```
current date :  2025-02-10 13:28:54.097787
before 7 days :  2025-02-03 13:28:54.097787
```

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

```
import datetime

date1 = input('Enter first date (DD-MM-YYYY): ')
date2 = input('Enter second date (DD-MM-YYYY): ')

d1 = datetime.datetime.strptime(date1, '%d-%m-%Y')
d2 = datetime.datetime.strptime(date2, '%d-%m-%Y')

difference = abs((d2 - d1).days)

print('Number of days between given dates is:', difference)
```

```
Enter first date (DD-MM-YYYY):  01-02-2023
Enter second date (DD-MM-YYYY):  01-03-2023

Number of days between given dates is: 28
```

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

```
date=input("Enter a date1: ")
convert_date = datetime.datetime.strptime(date,"%d/%m/%Y")

print(convert_date)
d1=convert_date.strftime("%A")
print("day of the week of a given date :",d1)
```

```
Enter a date1:  01/02/2023

2023-02-01 00:00:00
day of the week of a given date : Wednesday
```

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

```
import datetime
print(datetime.datetime.today())
```

```
2025-03-06 11:14:53.329001
```

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

```python
import math
print('Vlue Of Pi =',math.pi)
print('Floor =',math.floor(22.22))
print('Ceil =',math.ceil(22.22))
```

```
Vlue Of Pi = 3.141592653589793
Floor = 22
Ceil = 23
```

python-programming-lab-12

March 11, 2025

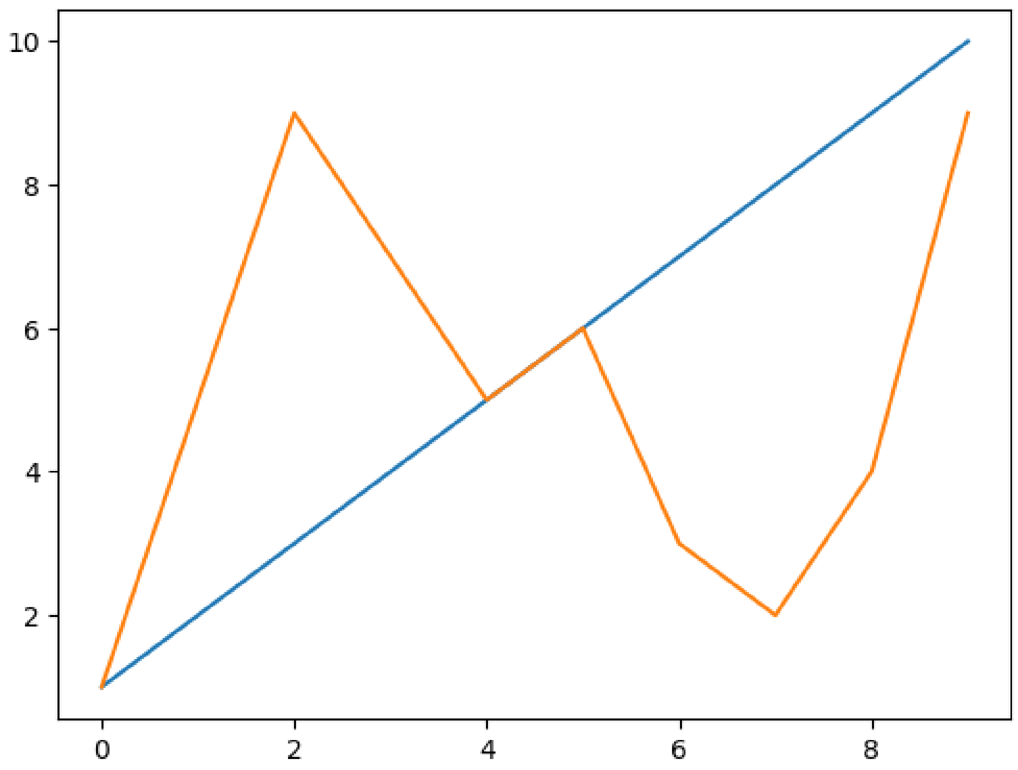
Python Programming - 2301CS404

Lab - 12

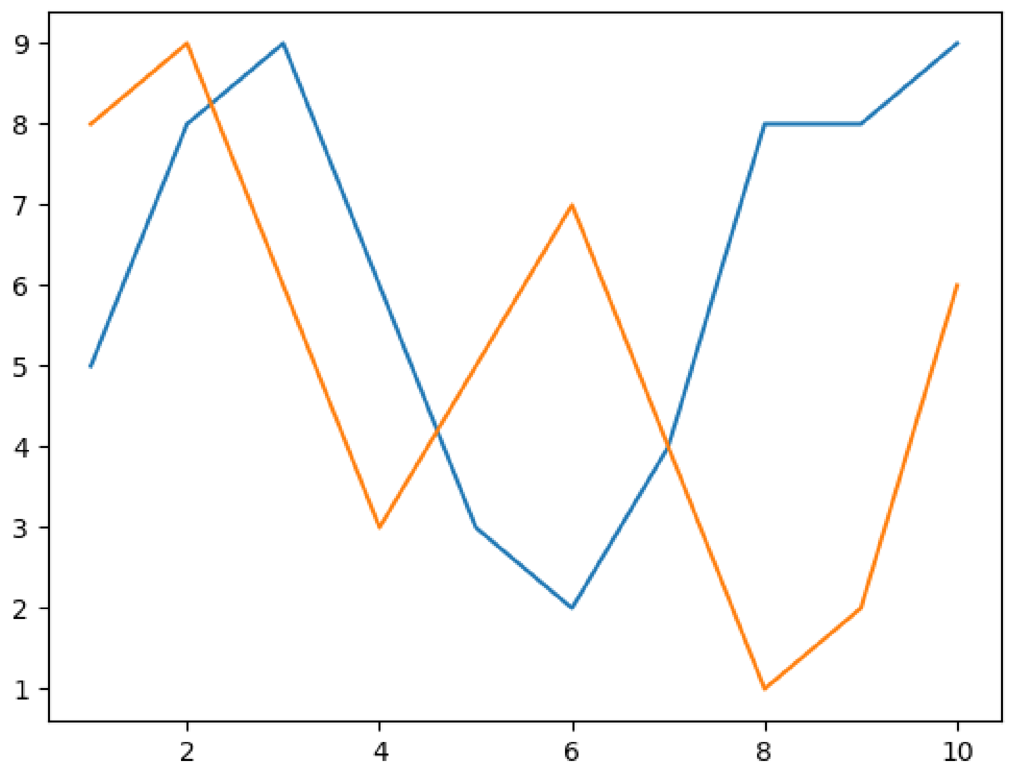
Riya Bhimani | 23010101030 | 17-02-2025

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

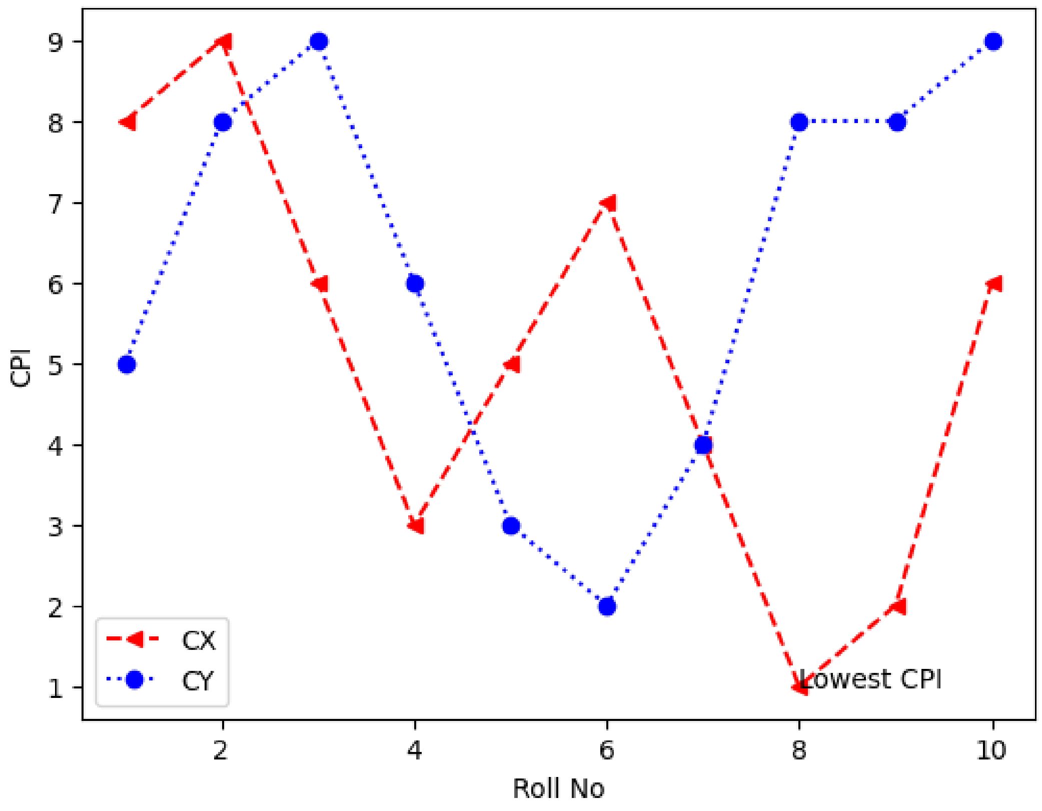
[7]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]

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

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



[9]:
```
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)
plt.show()
```

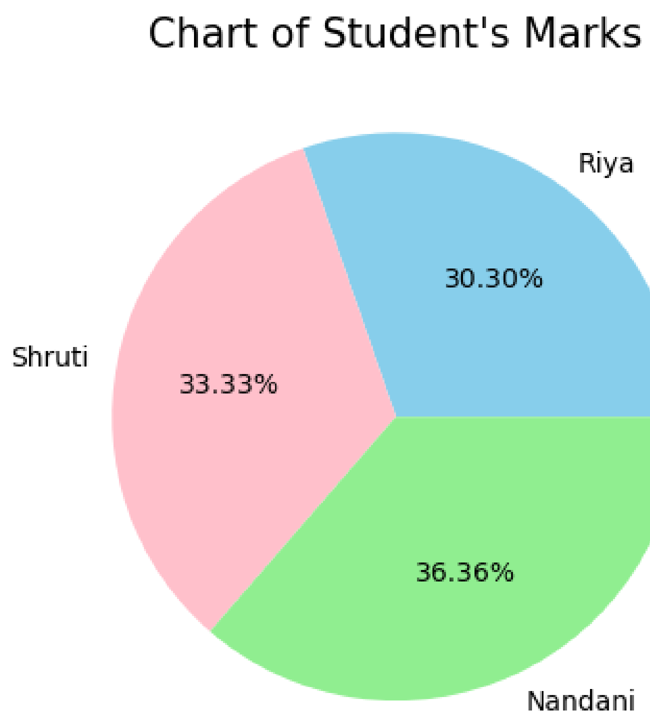


[18]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]



# write a code to generate below graph

plt.plot(x,cxMarks,color='red',linestyle='--',marker='<')
plt.plot(x,cyMarks,color='blue',linestyle=':',marker='o')
plt.xlabel("Roll No")
plt.ylabel("CPI")
plt.legend(['CX','CY'])
plt.annotate("Lowest CPI",xy=[8,1])
plt.show()
```



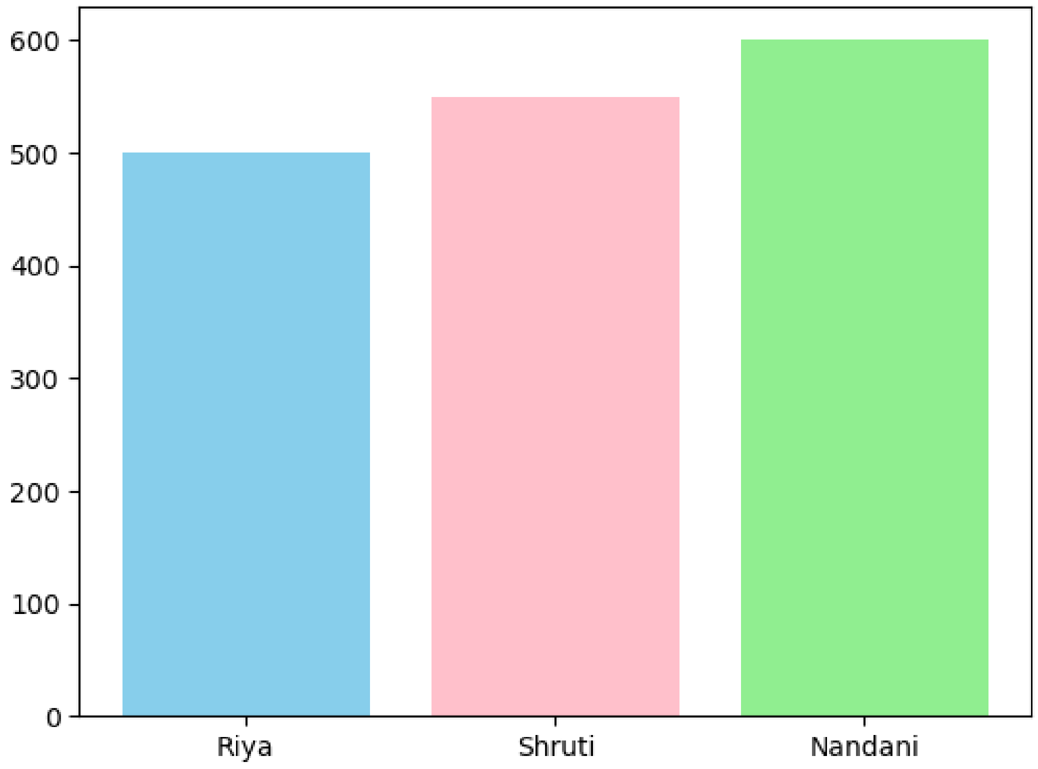
0.0.1 04) WAP to demonstrate the use of Pie chart.

[33]:
```
Students=['Riya','Shruti','Nandani']
Marks=[500,550,600]
Colours=['skyblue','pink','lightgreen']
plt.pie(Marks,autopct="%1.2f%%",colors=Colours,labels=Students)
plt.title("Chart of Student's Marks",fontsize=15)
plt.show()
```



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

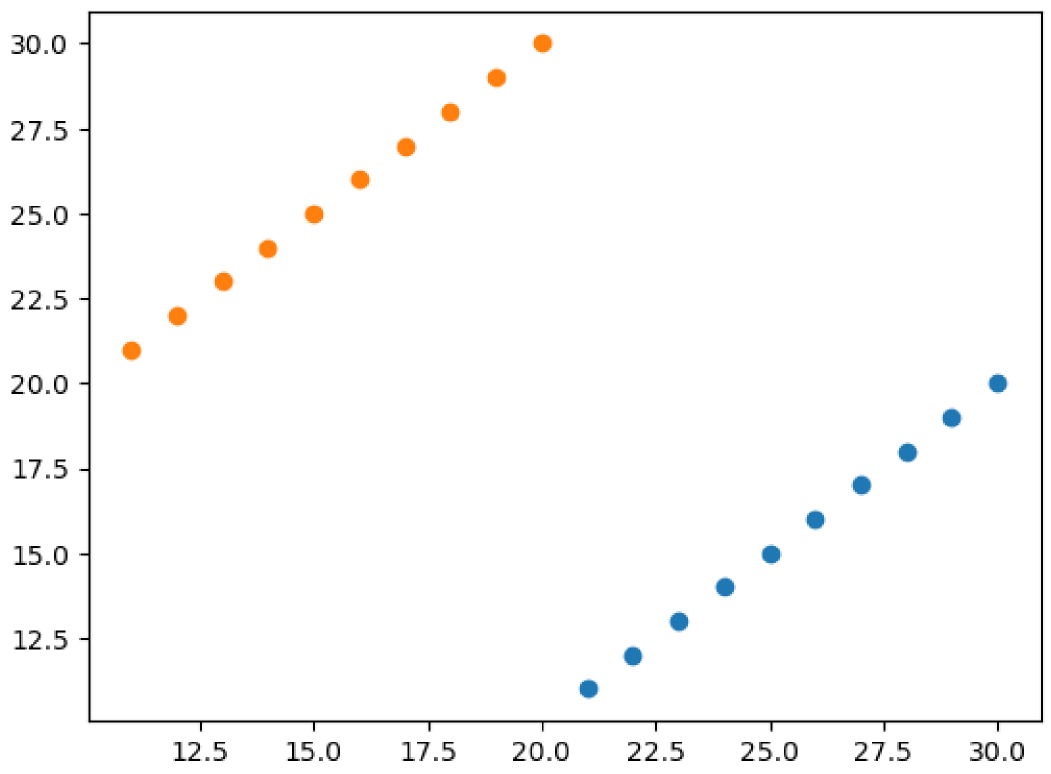
```
[35]: Students=['Riya','Shruti','Nandani']
      Marks=[500,550,600]
      Colours=['skyblue','pink','lightgreen']
      plt.bar(Students,Marks,color=Colours)
      plt.show()
```



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

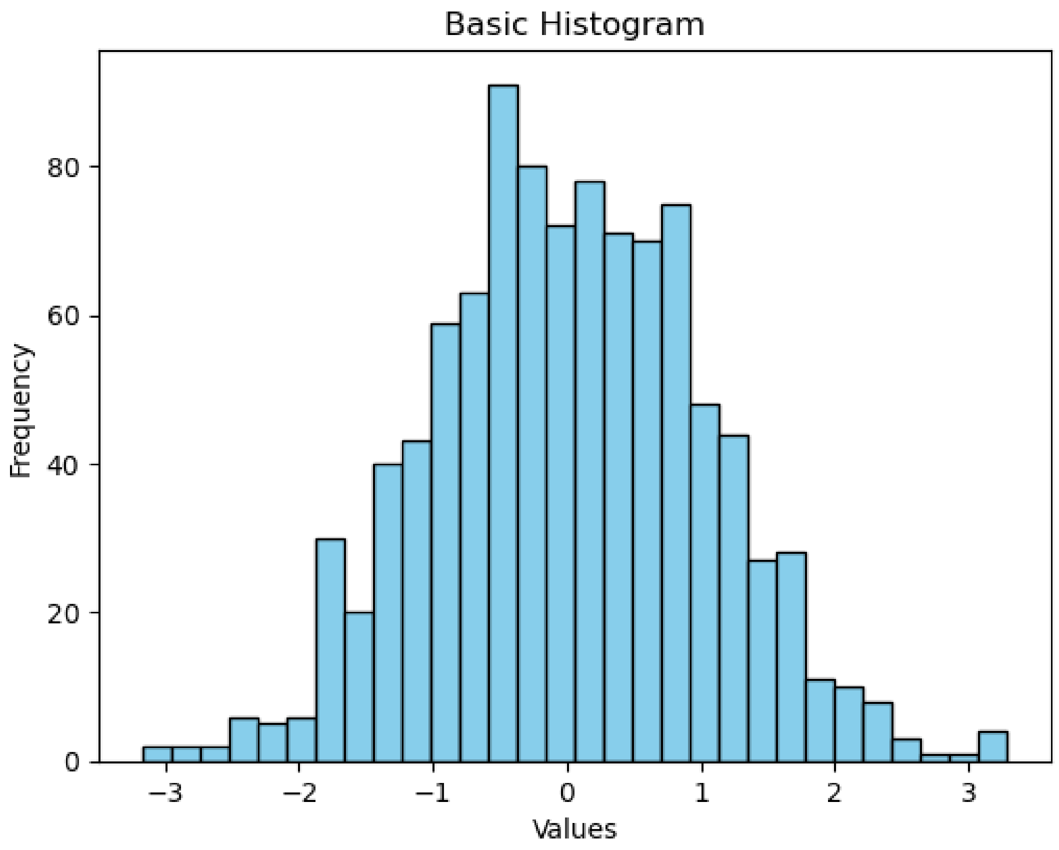
[42]:
```
import matplotlib.pyplot as plt
import numpy as np
```

[47]:
```
Mid1=[21,22,23,24,25,26,27,28,29,30]
Mid2=[11,12,13,14,15,16,17,18,19,20]
plt.scatter(Mid1,Mid2)
plt.scatter(Mid2,Mid1)
plt.show()
```



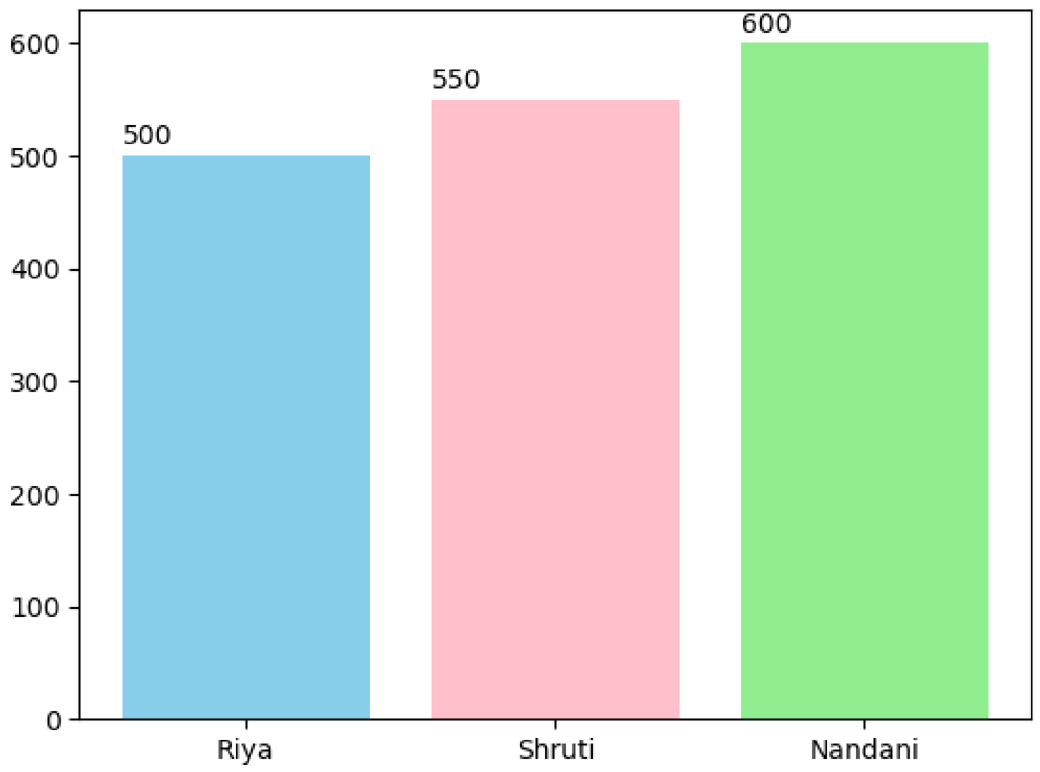
0.0.4 07) WAP to demonstrate the use of Histogram.

[59]:
```
data = np.random.randn(1000)
plt.hist(data, bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Basic Histogram')
plt.show()
```



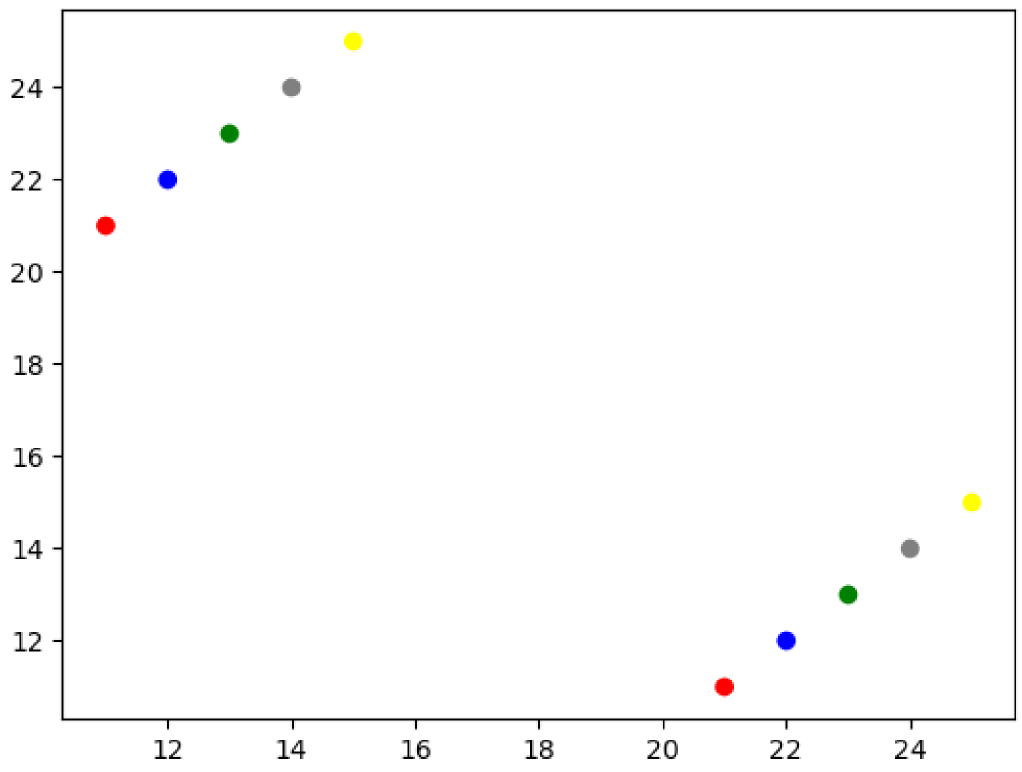
0.0.5 08) WAP to display the value of each bar in a bar chart using Matplotlib.

[54]:
```
Students=['Riya','Shruti','Nandani']
Marks=[500,550,600]
Colours=['skyblue','pink','lightgreen']
bars=plt.bar(Students,Marks,color=Colours)
for i in bars:
    yc=i.get_height()
    plt.text(i.get_x(),yc+10,f"{yc}")
plt.show()
```



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

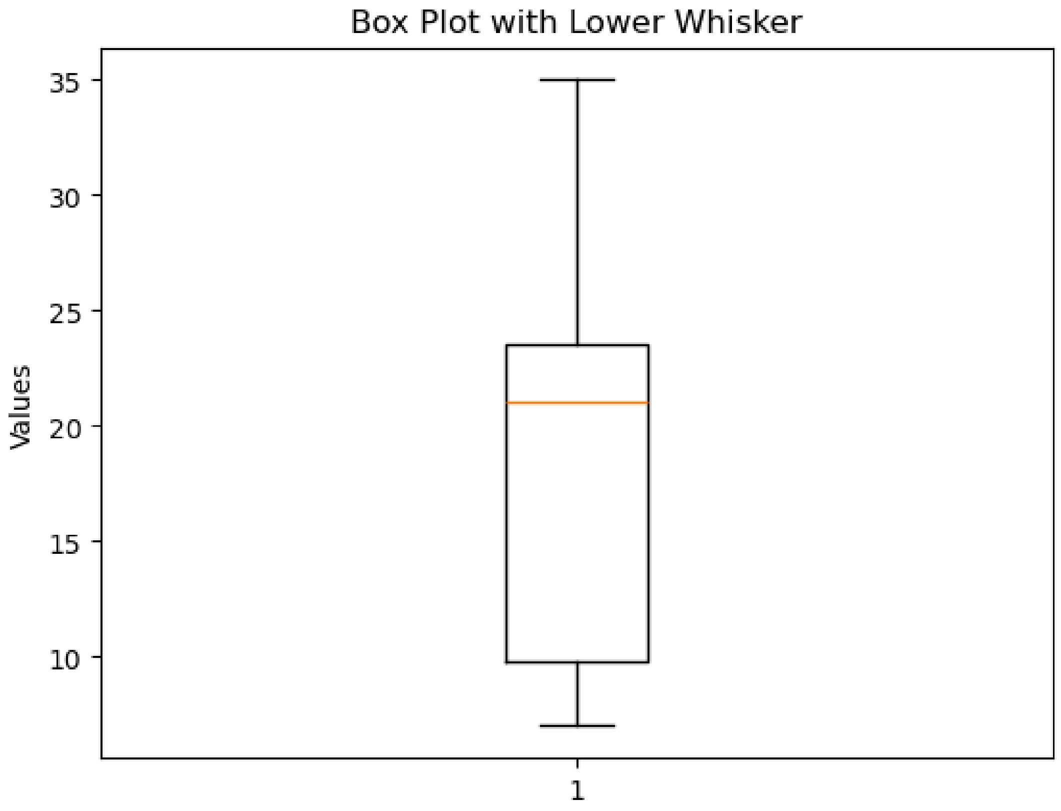
[56]:
```
Mid1=[21,22,23,24,25]
Mid2=[11,12,13,14,15]
Colours=['red','blue','green','gray','yellow']
plt.scatter(Mid1,Mid2,color=Colours)
plt.scatter(Mid2,Mid1,color=Colours)
plt.show()
```



0.0.7 10) WAP to create a Box Plot.

[64]:
```
# Sample data
data = np.array([7, 8, 9, 10, 15, 20, 22, 23, 23, 25, 30, 35])

# Calculate Q1 and Q3
Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)
IQR = Q3 - Q1

# Calculate the lower whisker
lower_whisker = max(min(data), Q1 - 1.5 * IQR)

# Print result
print("Lower Whisker (Starting Level):", lower_whisker)

# Create a box plot
plt.boxplot(data)
plt.title("Box Plot with Lower Whisker")
plt.ylabel("Values")
plt.show()
```

```
Lower Whisker (Starting Level): 7
```



```
[ ]:
```

OOP

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

```
class Students:
    def __init__(self,name,age,grade):
        self.Name = name
        self.Age = age
        self.Grade = grade

s1 = Students('Riya',18,'A++')
print(s1.Name)
print(s1.Age)
print(s1.Grade)
```

```
Riya
18
A++
```

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.

```
class Bank_Account:
    # Account_No = ''
    # User_Name = ''
    # Email = ''
    # Account_Type = ''
    # Account_Balance = 0

    def 
GetAccountDetails(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 DisplayAccountDetails(self):
        print("Account_No:",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)

ba = Bank_Account()
ba.GetAccountDetails('1234567890','Riya','riya@gmail.com','saving',100
0000)
ba.DisplayAccountDetails()

# For main method
#if __name__ == '__main__':
    #ba = Bank_Account()

#ba.GetAccountDetails('1234567890','Riya','riya@gmail.com','saving',10
00000)
    #ba.DisplayAccountDetails()

Account_No: 1234567890
User_Name: Riya
Email: riya@gmail.com
Account_Type: saving
Account_Balance: 1000000
```

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

```
class Circle:
    def __init__(self,radius):
        self.Radius= radius

    def perimeter(self):
        print("Perimeter = ",2*3.14*self.Radius)

    def area(self):
        print("Area = ",3.14*self.Radius*self.Radius)

C = Circle(4)
C.perimeter()
C.area()

Perimeter =  25.12
Area =  50.24
```

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

```
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 is not None:
            self.name = name
        if age is not None:
            self.age = age
        if salary is not None:
            self.salary = salary

    def display_info(self):
        print("Employee Name:",self.name, "\nAge:",self.age, "\
nSalary:",self.salary)

employee1 = Employee("Riya", 18, 50000)
employee1.display_info()
employee1.update_info(age=19, salary=55000)
employee1.display_info()
```

```
Employee Name: Riya
Age: 18
Salary: 50000
Employee Name: Riya
Age: 19
Salary: 55000
```

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

```
class Bank_Account:
    current_balance = 0
    def __init__(self,current_balance,money):
        self.Current_Balance = current_balance
        self.Money = money

    def deposit(self):
        self.Current_Balance += self.Money
        print("After Deposit",self.Money," Balance
is",self.Current_Balance)

    def withdraw(self):
        if (self.Current_Balance >= self.Money) :
            self.Current_Balance -= self.Money
        print("After Withdraw",self.Money," Balance 
is",self.Current_Balance)

    def checkbalance(self):
        print("Now Current Balance is",self.Current_Balance)

B_acc = Bank_Account(10000,5000)
B_acc.deposit()
B_acc.withdraw()
B_acc.checkbalance()
```

```
After Deposit 5000  Balance is 15000
After Withdraw 5000  Balance is 10000
Now Current Balance is 10000
```

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

```
class Managing_inventory:
    def _init_ (self):
        pass

    def AddItems(self):
        self.name = input("Enter your items name: ")
        self.price = float(input("Enter item price :"))
        self.quantity = int(input("Enter a quantity of your item:"))

    def DisplayItem(self):
        print("Item name is :",self.name)
        print("Item price is :",self.price)
        print("Item quantity is : ",self.quantity)

    def  UpdateItem(self,name=None,price=None,quantity=None):
        if name:
            self.name = name
        if price:
            self.price = price
        if quantity:
            self.quantity = quantity

    def DeleteItem(self):
        self.name = None
        self.price = None
        self.quantity = None

items = Managing_inventory()
```

```
items.AddItems()

print("\n after add data...\n")
items.DisplayItem()

print("\n after update Items........\n")
items.UpdateItem(name='laptop' , price =1)
items.DisplayItem()

print("\n after delete items Items........\n")
items.DeleteItem()
items.DisplayItem()

Enter your items name:  Clothes
Enter item price : 2000
Enter a quantity of your item: 20


 after add data...

Item name is : Clothes
Item price is : 2000.0
Item quantity is :  20

 after update Items........

Item name is : laptop
Item price is : 1
Item quantity is :  20

 after delete items Items........

Item name is : None
Item price is : None
Item quantity is :  None
```

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

```
class Student:
    def __init__(self,name,spi):
        self.Name = name
        self.SPI = spi

s1 = Student('Riya','8.63')
print("Name =",s1.Name)
print("SPI =",s1.SPI)

Name = Riya
SPI = 8.63
```

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.

```
class Student_Kit:
    principal_name = "Mr ABC"

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

    def mark_attendance(self, days):
        self.attendance_days = days

    def generate_certificate(self):
        print("Certificate of Attendance\
nStudent:" ,self.student_name,"\nPrincipal:
",Student_Kit.principal_name,"\nDays Present: ",self.attendance_days)

student1 = Student_Kit("Riya")
days_present = int(input("Enter number of days present: "))
student1.mark_attendance(days_present)
student1.generate_certificate()

Enter number of days present:  30

Certificate of Attendance
Student: Riya
Principal:  Mr ABC
Days Present:  30
```

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

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

    def add(self, other):
        total_minutes = self.minute + other.minute
        extra_hours = total_minutes // 60
        minutes = total_minutes % 60
        hours = self.hour + other.hour + extra_hours
        return Time(hours, minutes)

    def displayTime(self):
        print(self.hour , "hours" , self.minute , "minutes")

time1 = Time(2, 45)
time2 = Time(1, 30)
result = time1.add(time2)
result.displayTime()
```

```
4 hours 15 minutes
```

Continued..

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

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

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

def calculate_area(rect):
    return rect.area()

rect1 = Rectangle(20, 2)
print("Area of rectangle:", calculate_area(rect1))
```

```
Area of rectangle: 40
```

11) Calculate the area of a square.

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

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

    def area(self):
        return self.output(self.side * self.side)

    def output(self, area):
        print("Area of the square: ",area)
side = int(input("Enter side:"))
sq = Square(side)
sq.area()
```

```
Enter side: 7
```

```
Area of the square:  49
```

12) Calculate the area of a rectangle.

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

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):
        if length == width:
            print("THIS IS SQUARE.")
        else:
            self.length = length
            self.width = width

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

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

    @classmethod
    def validate(cls, length, width):
        if length == width:
            print("THIS IS SQUARE.")
            return None
        return cls(length, width)

side1 = int(input("Enter side1:"))
side2 = int(input("Enter side2:"))
rect1 = Rectangle.validate(side1, side2)
if rect1:
    rect1.area()

side1 = int(input("Enter side1:"))
side2 = int(input("Enter side2:"))
rect2 = Rectangle.validate(side1, side2)
if rect2:
    rect2.area()
```

```
Enter side1: 10
Enter side2: 11

Area of the square:  110

Enter side1: 10
Enter side2: 10
```

```
THIS IS SQUARE.
```

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

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

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

    def get_side(self):
        return self.__side

    def set_side(self, side):
        if side > 0:
            self.__side = side
        else:
            print("Side length must be positive.")

side = int(input("Enter side:"))
sq = Square(side)
print("Side:", sq.get_side())

newside = int(input("Enter newside:"))
sq.set_side(newside)
print("Updated Side:", sq.get_side())

negativeside = int(input("Enter negativeside:"))
sq.set_side(negativeside)
```

```
Enter side: 10

Side: 10

Enter newside: 12

Updated Side: 12

Enter negativeside: -3

Side length must be positive.
```

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

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

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

```
class Profit:
    def getProfit(self):
        self.profit = float(input("Enter profit: "))

class Loss:
    def getLoss(self):
        self.loss = float(input("Enter loss: "))

class BalanceSheet(Profit, Loss):
    def getBalance(self):
        return self.profit - self.loss

    def printBalance(self):
        print("Net Balance:", self.getBalance())

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

```
Enter profit:  500
Enter loss:  150

Net Balance: 350.0
```

15) WAP to demonstrate all types of inheritance.

```
# 1. Single Inheritance
class Parent:
    def show(self):
        print("Single Inheritance: Parent class")

class Child(Parent):
    pass

# 2. Multiple Inheritance
class Father:
    def fatherFeature(self):
        print("Multiple Inheritance: Feature from Father")
```

```python
class Mother:
    def motherFeature(self):
        print("Multiple Inheritance: Feature from Mother")

class Child2(Father, Mother):
    pass

# 3. Multilevel Inheritance
class Grandparent:
    def grandFeature(self):
        print("Multilevel Inheritance: Feature from Grandparent")

class Parent2(Grandparent):
    pass

class Child3(Parent2):
    pass

# 4. Hierarchical Inheritance
class Base:
    def baseFeature(self):
        print("Hierarchical Inheritance: Base class feature")

class Derived1(Base):
    pass

class Derived2(Base):
    pass

# 5. Hybrid Inheritance (Combination of multiple types)
class A:
    def featureA(self):
        print("Hybrid Inheritance: Feature A")

class B(A):
    def featureB(self):
        print("Hybrid Inheritance: Feature B")

class C(A):
    def featureC(self):
        print("Hybrid Inheritance: Feature C")

class D(B, C):
    pass

print("\n--- Single Inheritance ---")
c1 = Child()
c1.show()

print("\n--- Multiple Inheritance ---")
```

```python
c2 = Child2()
c2.fatherFeature()
c2.motherFeature()

print("\n--- Multilevel Inheritance ---")
c3 = Child3()
c3.grandFeature()

print("\n--- Hierarchical Inheritance ---")
d1 = Derived1()
d2 = Derived2()
d1.baseFeature()
d2.baseFeature()

print("\n--- Hybrid Inheritance ---")
d = D()
d.featureA()
d.featureB()
d.featureC()
```

```
--- Single Inheritance ---
Single Inheritance: Parent class

--- Multiple Inheritance ---
Multiple Inheritance: Feature from Father
Multiple Inheritance: Feature from Mother

--- Multilevel Inheritance ---
Multilevel Inheritance: Feature from Grandparent

--- Hierarchical Inheritance ---
Hierarchical Inheritance: Base class feature
Hierarchical Inheritance: Base class feature

--- Hybrid Inheritance ---
Hybrid Inheritance: Feature A
Hybrid Inheritance: Feature B
Hybrid Inheritance: Feature C
```

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

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

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

```
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 display(self):
        print("Name:",self.name, "\nAge:", self.age, "\nSalary:",
self.salary)

emp = Employee("Riya", 18, 500000)
emp.display()

Name: Riya
Age: 18
Salary: 500000
```

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

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

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.

```
from abc import ABC, abstractmethod

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

class Rectangle(Shape):
```

```python
    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
```