Python Programming - 2301CS404

Lab - 1

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

In [ ]:
```
Name
```

01) WAP to print “Hello World”

In [2]:
```
print("Hello World")
```

```
Hello World
```

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

In [4]:
```
a=int(input("Enter a number : "))
b=int(input("Enter a number : "))
print(f"Sum(With input) = {a+b}")
c=10
d=20
print("Sum(Without input) = ",c+d)
```

```
Sum(With input) = 6
Sum(Without input) =  30
```

03) WAP to check the type of the variable.

In [6]:
```
a=input("Enter anything : ")
print(type(a))
b=int(input("Enter number : "))
print(type(b))
```

```
<class 'str'>
<class 'int'>
```

04) WAP to calculate simple interest.

In [8]:
```
p=float(input("Enter principle : "))
r=float(input("Enter rate of interest : "))
t=float(input("Enter time : "))
si=(p*r*t)/100
print("Simple Interest = ",si)
```

```
Simple Interest =  1.02
```

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

In [10]:
```
import math
radius = float(input("Enter radius of the circle : "))
area = math.pi*radius*radius
print("Area = ",area)
```

```
Area =  28.274333882308138
```

06) WAP to calculate area of a triangle.

In [12]:
```
base=int(input("Enter base of triangle : "))
height=int(input("Enter height of triangle : "))
area = height*base/2
print("Area = ",area)
```

```
Area =  3.0
```

07) WAP to compute quotient and remainder.

In [14]:
```
a=int(input("Enter first number "))
b=int(input("Enter second number "))
quotient=int(a/b)
remainder=a%b
print(f"Quotient = {quotient} \n Remainder={remainder}")
```

```
Quotient = 0
 Remainder=2
```

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

In [16]:
```
celsius=int(input("Enter celsius : "))
print(f"In Fahrenheit = ",9*celsius/5+32)
fahrenheit=int(input("Enter fahrenheit : "))
print(f"In Celsius = ",5*(fahrenheit-32)/9)
```

```
In Fahrenheit =  91.4
In Celsius =  11.11111111111111
```

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

In [18]:
```
import math
x1=int(input("Enter x1 : "))
y1=int(input("Enter y1 : "))
x2=int(input("Enter x2 : "))
y2=int(input("Enter y2 : "))
distance = math.sqrt(pow((x2-x1),2)+pow((y2-y1),2))
print("Distance = ",distance)
```

```
Distance =  5.656854249492381
```

10) WAP to print sum of n natural numbers.

In [20]:
```
n=int(input("Enter value of n : "))
sum=0
for i in range(n+1):
    sum+=i
print("Sum = ",sum)
```

```
Sum =  6
```

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

In [22]:
```
n=int(input("Enter value of n : "))
sum=0
for i in range(n+1):
    sum+=pow(i,2)
print("Sum = ",sum)
```

```
Sum =  14
```

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

In [24]:
```
first_name=input("Enter first name : ")
last_name=input("Enter last name : ")
full_name=first_name+" "+last_name
print("Full Name = ",full_name)
```

```
Full Name =  Jadeja Rudrarajsinh
```

13) WAP to swap two numbers.

In [26]:
```
a=int(input("Enter value of a : "))
b=int(input("Enter value of b : "))
temp=a
a=b
b=temp
print(f"Value of a = {a} \n Value of b = {b}")
```

```
Value of a = 4
 Value of b = 2
```

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

In [28]:
```
kilometer=int(input("Enter distance in kilometer : "))
meter=1000*kilometer
feet=3280.8399*kilometer
inch=39370.0787*kilometer
centimeter=100000*kilometer
print(f"Meter={meter}\nFeet={feet}\nInch={inch}\nCentimeter={centimeter}")
```

```
Meter=30000
Feet=98425.197
Inch=1181102.361
Centimeter=3000000
```

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

In [30]:
```
day=int(input("Enter day : "))
month=int(input("Enter month : "))
year=int(input("Enter year : "))
print(f"Date = {day}-{month}-{year}")
```

```
Date = 23-11-2024
```

Python Programming - 2301CS404

Lab - 2

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

if..else..

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

In [2]:
```
n = int(input("Enter a number : "))
if n>0:
    print("Number is positive")
elif n==0:
    print("Number is neither positive nor negative")
else:
    print("Number is negative")
```

```
Number is positive
```

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

In [4]:
```
n=int(input("Enter a number : "))
if n%2==0:
    print("Number is Even")
else:
    print("Number is Odd")
```

```
Number is Odd
```

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

In [6]:
```
a=int(input("Enter value of a : "))
b=int(input("Enter value of b : "))
if a>b:
    print("Maximum is a = ",a)
elif a==b:
    print("Both are equal")
else:
    print("Maximum is b = ",b)

c = print("Maximum is a (ternary) = ",a) if(a>b) else print("Maximum is b (terna
```

```
Maximum is b =  5
Maximum is b (ternary) =  5
```

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

In [8]:
```
a = int(input("Enter value of a : "))
b = int(input("Enter value of b : "))
c = int(input("Enter value of c : "))
if a>b:
    if a>c:
        print("Maximum is a = ",a)
    else:
        print("Maximum is c = ",c)
else:
    if b>c:
        print("Maximum is b = ",b)
    else:
        print("Maximum is c = ",c)
```

```
Maximum is c =  6
```

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]

In [10]:
```
n = int(input("Enter a year : "))
if (n%4==0 and n%100!=0) or (n%400==0):
    print("Is a Leap Year")
else:
    print("Not a Leap Year")
```

```
Is a Leap Year
```

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

In [12]:
```
n = int(input("Enter a number according to day of the week : "))
if n==1 :
```

```
    print("Monday")
elif n==2:
    print("Tuesday")
elif n==3:
    print("Wednesday")
elif n==4:
    print("Thursday")
elif n==5:
    print("Friday")
elif n==6:
    print("Saturday")
elif n==7:
    print("Sunday")
else:
    print("Enter valid Number")
```

```
Saturday
```

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

In [14]:
```
a = int(input("Enter value of a : "))
b = int(input("Enter value of b : "))
print("Enter 1 for addition")
print("Enter 2 for subtraction")
print("Enter 3 for multiplication")
print("Enter 4 for division")
choice = int(input("Enter your choice : "))
match choice:
    case 1:
        sum = a+b
        print("Sum = ",sum)
    case 2:
        sub = a-b
        print("Difference = ",sub)
    case 3:
        mul = a*b
        print("Product = ",mul)
    case 4:
        div = a/b;
        print("Quotient = ",div)
    case _:
        print("Invalid choice")
```

```
Enter 1 for addition
Enter 2 for subtraction
Enter 3 for multiplication
Enter 4 for division
Quotient =  0.6666666666666666
```

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

In [16]:
```
maths = int(input("Enter marks of Maths : "))
physics = int(input("Enter marks of Physics : "))
```

```
chemistry = int(input("Enter marks of chemistry : "))
english = int(input("Enter marks of English : "))
computer = int(input("Enter marks of Computer : "))
n = (maths+physics+chemistry+english+computer)/5.0
if n<0 or n>100:
  print("Invalid marks")
elif n<35:
    print("Fail with ",n,end=" percentage")
elif n<45:
    print("Pass Class with ",n,end=" percentage")
elif n<60:
    print("Second Class with ",n,end=" percentage")
elif n<70:
    print("First Class with ",n,end=" percentage")
elif n>70:
    print("Distinction with ",n,end=" percentage")
```

```
First Class with  69.2 percentage
```

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.

In [18]:
```
a = int(input("Enter length of side1 : "))
b = int(input("Enter length of side2 : "))
c = int(input("Enter length of side3 : "))
if a>b and a>c:
    if pow(a,2) == (pow(b,2)+pow(c,2)):
        print("Right-angled",end=" ")
if b>a and b>c:
    if pow(b,2) == (pow(a,2)+pow(c,2)):
        print("Right-angled",end=" ")
if c>a and c>b:
    if pow(c,2) == (pow(a,2)+pow(b,2)):
        print("Right-angled",end=" ")
if a==b and b==c and a==c:
    print("Equilateral triangle")
elif a==b or b==c or a==c:
    print("Iscosceles Triangle")
else:
    print("Scalene Triangle")
```

```
Right-angled Scalene Triangle
```

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

In [20]:
```
a = int(input("Enter a number : "))
b = int(input("Enter a number : "))
c = int(input("Enter a number : "))
if (a>b and b>c) or (c>b and b>a):
    print("Second largest = ",b)
if (b>a and a>c) or (c>a and a>b):
    print("Second largest = ",a)
if (b>c and c>a) or (a>c and c>b):
    print("Second largest = ",c)
```

```
Second largest =  5
```

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

In [23]:
```
unit = int(input("Enter total units : "))
if unit<=50:
    cost = unit*2.6
elif unit<=100:
    cost = 50*2.6 + (unit-50)*3.25
elif unit<=200:
    cost = 50*2.6 + 50*3.25 + (unit-100)*5.26
else:
    cost = 50*2.6 + 50*3.25 + 100*5.26 + (unit-200)*8.45
print("Total Cost = ",cost)
```

```
Total Cost =  146.25
```

Python Programming - 2301CS404

Lab - 3

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

for and while loop

01) WAP to print 1 to 10.

In [4]:
```
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.

In [6]:
```
n = int(input("Enter Number:"))
for i in range(1,n+1):
 print(i)
```

```
1
2
3
4
```

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

In [10]:
```
n = int(input("Enter Number:"))
for i in range(1,n+1):
 if i%2!=0:
     print(i)
```

```
1
3
5
7
9
```

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

In [12]:
```
n1 = int(input("Enter Number 1:"))
n2 = int(input("Enter Number 2:"))
for i in range(n1,n2+1):
 if i%2==0 and i%3!=0:
     print(i)
```

```
20
22
26
28
```

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

In [22]:
```
n = int(input("Enter Number:"))
sum = 0
for i in range(1,n+1):
    sum += i
print(sum)
```

```
3
```

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

In [26]:
```
n = int(input("Enter Number:"))
sum = 0
for i in range(1,n+1):
    sum += i**2
print(sum)
```

```
14
```

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

In [30]:
```
n = int(input("Enter Number:"))
sum = 0
for i in range(1,n+1):
 if(i%2==0):
     sum-=i
 else:
     sum+=i
print(sum)
```

```
2
```

08) WAP to print multiplication table of given number.

In [32]:
```
n = int(input("Enter Number:"))
for i in range(1,11):
    print(f"{n} x {i} = {n*i}")
```

```
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

09) WAP to find factorial of the given number.

In [34]:
```
n = int(input("Enter Number:"))
ans = 1
for i in range(1,n+1):
    ans *= i
print(ans)
```

```
120
```

10) WAP to find factors of the given number.

In [36]:
```
n = int(input("Enter Number:"))
for i in range(1,n+1):
 if(n%i==0):
     print(f"Factor:{i}")
```

```
Factor:1
Factor:2
Factor:3
Factor:6
```

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

In [42]:
```
n = int(input("Enter Number:"))
for i in range(2,n):
    if(n%i==0):
        print("number is not prime")
        break
```

```
else:
    print("number is prime")
```

```
number is prime
```

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

In [47]:
```
n = int(input("Enter Number:"))
sum = 0;
while n > 0:
    d = n % 10
    sum += d
    n = n // 10
print(sum)
```

```
4
```

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

In [51]:
```
n = int(input("Enter Number:"))
temp = n
ans = 0
while n > 0:
 d = n % 10
 ans = ans * 10 + d
 n = n //  10
if temp==ans:
 print("number is Palindrome")
else:
 print("Not Palindrome number")
```

```
number is Palindrome
```

14) WAP to print GCD of given two numbers.

In [55]:
```
n1 = int(input("Enter Number 1:"))
n2 = int(input("Enter Number 2:"))
gcd = 1
for i in range(1,min(n1,n2)):
    if n1 % i == 0 and n2 % i == 0:
        gcd = i
print ("GCD of", n1, "and", n2, "is", gcd)
```

```
GCD of 1 and 2 is 1
```

In [ ]:

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 4

Name: Jadeja Rudrarajsinh

Enrollment No:23010101411

Roll No:487

String

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

In [2]:
```
str = input("Enter String:")
rev = str[::-1]
if(rev==str):
 print("String Are Palindrome")
else:
 print("String Are Not Palindrome")
```

```
String Are Not Palindrome
```

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

In [10]:
```
str = input("Enter String:")
rev = str[::-1]
print(rev)
```

```
dad
```

03) WAP to remove ith character from given string.

In [14]:
```
str = input("Enter String:")
i = int(input("Enter Number That You Want To Remove:"))
ans = str[0:i:] + str[i+1::]
print(ans)
```

```
dad
```

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

In [16]:
```
str = input("Enter String:")
ans = 0
for i in str:
 ans+=1
print(ans)
```

```
5
```

05) WAP to print even length word in string.

In [28]:
```
str = input("Enter String:")
l = str.split(" ")
print(l)
for i in l:
    if (len(i)%2==0):
        print (i)
```

```
['hhhhhh', 'hhhh']
hhhhhh
hhhh
```

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

In [36]:
```
str = input("Enter String:")
ans = 0
str = str.lower()
for i in str:
 if(i=='a' or i=='e' or i=='i' or i=='o' or i=='u'):
    ans+=1
print(ans)
```

```
3
```

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

In [48]:
```
str = input("Enter String:")
ans = ""
final = ""
s = str.split(" ")
for i in s:
 ans = i[0:1:].upper() + i[1:len(i)-1:] + i[len(i)-1::].upper()
 final += ans + " "
print(final)
```

```
RudrarajsinH
```

08) WAP to convert given array to string.

In [40]:
```
arr = ["Hello","My","Name","is","Rudrarajsinh"]
str = ""
for i in arr:
 str+=i+" "
print(str)
```

```
Hello My Name is Rudrarajsinh
```

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

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

In [56]:
```
pass1 = input("Enter passsword: ")
pass2 = input("Confirm password: ")
if pass1 == pass2:
    print("correct")
elif pass1.lower() == pass2.lower():
    print("password Must be casesensetive.")
else:
    print("Incorrect Password.")
```

```
Incorrect Password.
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

In [64]:
```
Number = input("Enter card number: ")

Number_card = "**** **** **** " + Number[-4:]

print(Number_card)
```

```
**** **** **** 5863
```

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

s1 = decimal and s2 = medical are Anagram

In [80]:
```
s1 = input("Enter s1: ")
s2 = input("Enter s2: ")

s1 = s1.replace(" ", "").lower()
s2 = s2.replace(" ", "").lower()

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

```
"bablo" and "pablo" are not Anagrams.
```

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

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

In [84]:
```
input_string = input("Enter s1: ")

lowercase = "".join([char for char in input_string if char.islower()])
uppercase = "".join([char for char in input_string if char.isupper()])

output_string = lowercase + uppercase

print(output_string)
```

```
hdfsjhdbdJGKBJGKEJ
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 5

List

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

In [2]:
```
list = [10,20,30,40,50]
sum = 0
for i in list:
 sum+=i
else:
 print(f"Sum:{sum}")
```

```
Sum:150
```

02) WAP to find largest element in a List.

In [4]:
```
list = [10,50,40,35,60,24]
m=max(list)
print(m)
```

```
60
```

03) WAP to find the length of a List.

In [8]:
```
list = [10, 20, 30, 40, 50]

length = len(list)

print("The length of the list is:", length)
```

```
The length of the list is: 5
```

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

In [12]:
```
l1 = [1,4,6,3,7]

l1[0],l1[4] = l1[4],l1[0]
```

```
print(l1)
```

```
[7, 4, 6, 3, 1]
```

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

In [14]:
```
list = [10, 20, 30, 40, 50]

mid = len(list) // 2

first = list[:mid]
second = list[mid:]

result = second + first

print("Original list:", list)
print("Modified list:", result)
```

```
Original list: [10, 20, 30, 40, 50]
Modified list: [30, 40, 50, 10, 20]
```

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

In [22]:
```
list = [10, 20, 30, 40, 50]

print("Original list:", list)

pos1 = int(input("Enter the first position (1-based index): ")) - 1
pos2 = int(input("Enter the second position (1-based index): ")) - 1

if pos1 >= 0 and pos1 < len(list) and pos2 >= 0 and pos2<len(list):

    list[pos1], list[pos2] = list[pos2], list[pos1]
    print("Modified list:", list)
else:
    print("Invalid index")
```

```
Original list: [10, 20, 30, 40, 50]
Invalid index
```

07) WAP to reverse the list entered by user.

In [26]:
```
l1 = input("Enter space seperated string : ").split()
l1 = [int(i) for i in l1]
print(l1)
l1.reverse()
print(l1)
```

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

08) WAP to print even numbers in a list.

In [43]:
```
l1 = input("Enter a string: ").split()
l1 = [int(i) for i in l1]

l = [l1[i]for i in range (len(l1)) if l1[i]%2==0]
print(l)
```

```
[]
```

09) WAP to count unique items in a list.

In [2]:
```
n = input("Enter space separated values for list :")
li = [int(i) for i in n.split()]
count = len(set(li))
print("Count =",count)
```

```
Count = 5
```

10) WAP to copy a list.

In [4]:
```
n = input("Enter space separated values for list : ")
l1 = [i for i in n.split()]
l2 = l1.copy()
print("Copy = ",l2)
```

```
Copy =  ['1', '2', '3', '4', '5', '6']
```

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

In [6]:
```
lower = int(input("Enter lower range : "))
upper = int(input("Enter upper range : "))
li = [i for i in range(lower,upper+1) if i%2==1]
print(li)
```

```
[3, 5]
```

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

In [8]:
```
n = input("Enter space separated values : ")
li = [i for i in n.split()]
element = input("Enter element that you want to count : ")
count = li.count(element)
print(f"Count of {element} = ",count)
```

```
Count of 2 3 4 =  0
```

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

In [10]:
```
n = input("Enter space separated values : ")
li = [int(i) for i in n.split()]
if len(li)==li.count(li[0]):
    print("Second Maximum not found")
else:
    maximum = max(li)
    number = 0
    for i in li:
        if number<i and i<maximum:
```

```
            number=i
    print("Second Maximum = ",number)
```

```
Second Maximum =  4
```

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

In [12]:
```
n = input("Enter space separated values :")
li = [i for i in n.split()]
k = int(input("Enter value of k : "))
ans = set([i for i in li if li.count(i)>k])
print(list(ans))
```

```
['2']
```

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

In [14]:
```
l1=[]
for i in range(0,10):
    l1.append(i**2)
print("Without List Comprehension",li)
#With List Comprehension
l2 = [i**2 for i in range(0,10)]
print("With list comprehension : ",l2)
```

```
Without List Comprehension ['4', '2', '3', '5', '2']
With list comprehension :  [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

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

In [16]:
```
n = input("Enter space separated fruits : ")
li = [i for i in n.split()]
fruits = [i for i in li if i.startswith("b")]
print(fruits)
```

```
['banana']
```

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

In [18]:
```
n1 = input("Enter space separated values : ")
n2 = input("Enter space separated values : ")
l1 = [i for i in n1.split()]
l2 = [i for i in n2.split()]
common = [x for x in l1 if x in l2]
print(common)
```

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

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 6

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

Tuple

01) WAP to find sum of tuple elements.

In [17]:
```
tuple = (1,2,3,4,5)
total = sum(tuple)

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

```
sum of tuple:  15
```

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

In [71]:
```
tuple = (1,2,6,5,7,4)
k = int (input("Enter value of k: "))
t1 = sorted (set(tuple))
print(t1 [k:])
print(t1 [-k:])
```

```
[6, 7]
[4, 5, 6, 7]
```

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

In [51]:
```
listoftuple = [(2,4),(6,7,8,9,0)]
k = int(input("Enter K:"))
ans = []
for i in listoftuple:
    for j in i:
        if(j%k!=0):
            break;
        else:
            ans.append(i)
print(ans)
```

```
[(6, 7, 8, 9, 0)]
```

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

In [75]:
```
numbers = [1, 2, 3, 4, 5]

result = [(n, n**3) for n in numbers]

print(result)
```

```
[(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.

In [66]:
```
listoftup = [(2,4,-1,-2,),(6,7,8,9,0),(-1,0),(2,5,3)]
ans = []
for i in listoftup:
    for j in i:
        if(j<0):
            break;
    else:
       ans.append(i)
print(ans)
```

```
[(6, 7, 8, 9, 0), (2, 5, 3)]
```

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

In [113…
```
t1 = (1,2,3)
l1 = [4,5,6]

l1.extend(t1)
print(l1)

t1 += tuple(t1)
print(t1)
```

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

07) WAP to remove tuples of length K.

In [103…
```
list_of_tuple = [(1, 2), (3, 4, 5), (6,), (7, 8, 9), (10, 11)]
```

```
k = 2

result = [t for t in list_of_tuple if len(t) != k]

print("List after removing tuples of length", k, ":", result)
```

```
List after removing tuples of length 2 : [(3, 4, 5), (6,), (7, 8, 9)]
```

08) WAP to remove duplicates from tuple.

In [77]:
```
tuple = (1,2,2,3,3,3,4,4,4,4)
print(set(tuple))
```

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

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

In [115…
```
l1 = [1,2,3,4,5]

ans = tuple(l1[i] * l1[i+1] for i in range(0,len(l1)-1))
print(ans)
```

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

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

In [95]:
```
tuple = (1, 2, 3, 4, 3)

distinct = len(tuple) == len(set(tuple))

if distinct:
    print("distinct elements.")
else:
    print("not distinct elements.")
```

```
not distinct elements.
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 7

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

Set & Dictionary

01) WAP to iterate over a set.

In [2]:
```
n = input("Enter space separated values : ")
s = set(map(int,n.split()))
for i in s:
    print(i)
```

```
1
2
6
```

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

In [4]:
```
n = input("Enter space separated values : ")
s = set(map(int,n.split()))
li = list(s)
st = str(s)
t = tuple(s)
print("List =",li)
print("String =",st)
print("Tuple =",t)
```

```
List = [1, 2, 4, 6]
String = {1, 2, 4, 6}
Tuple = (1, 2, 4, 6)
```

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

In [6]:
```
n = input("Enter space separated values : ")
s = set(map(int,n.split()))
print("Maximum = ",max(s))
print("Minimum = ",min(s))
```

```
Maximum =  6
Minimum =  2
```

04) WAP to perform union of two sets.

In [8]:
```
n1 = input("Enter space separated values : ")
n2 = input("Enter space separated values : ")
s1 = set(map(int,n1.split()))
s2 = set(map(int,n2.split()))
print("Union using union() = ",s1.union(s2))
print("Union using | = ",s1|s2)
```

```
Union using union() =  {1, 2, 3, 4, 5, 6, 7, 9}
Union using | =  {1, 2, 3, 4, 5, 6, 7, 9}
```

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

In [12]:
```
l1 = input("Enter space separated values : ").split()
l2 = input("Enter space.separated values : ").split()
s1 = set(map(int,l1))
s2 = set(map(int,l2))
if s1.intersection(s2):
    print("Atleast one element is common")
else:
    print("No element is common")
```

```
Atleast one element is common
```

06) WAP to remove duplicates from list.

In [14]:
```
li = input("Enter space separated values : ").split()
li = map(int,li)
s = set(li)
li = list(s)
print("List = ",li)
```

```
List =  [1, 3, 5, 7]
```

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

In [16]:
```
words = input("Enter sentence : ").split()
word = set(i for i in words if words.count(i)==1)
print("Unique words = ",word)
```

```
Unique words =  {'Rudrarajsinh', 'Jadeja'}
```

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

In [18]:
```
n1 = input("Enter space separated values : ")
n2 = input("Enter space separated values : ")
a = set(map(int,n1.split()))
b = set(map(int,n2.split()))
a.symmetric_difference_update(b)
print("a =",a)
```

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

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

In [20]:
```
s1 = input("Enter a string ")
s2 = input("Enter a string ")
if len(s1)!=len(s2):
    print("Not Anagrams")
else:
    count1 = {}
    count2 = {}
    for i in s1:
        count1[i] = s1.count(i)
    for i in s2:
        count2[i] = s2.count(i)
    if count1==count2:
        print("Both are anagrams")
    else:
        print("Not Anagrams")
```

```
Both are anagrams
```

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

In [22]:
```
n1 = input("Enter space separated values : ")
n2 = input("Enter space separated values : ")
n3 = input("Enter space separated values : ")
l1 = map(int,n1.split())
l2 = map(int,n2.split())
l3 = map(int,n3.split())
s1 = set(l1)
s2 = set(l2)
s3 = set(l3)
print("Common elements = ",list(s1&s2&s3))
```

```
Common elements =  []
```

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

In [26]:
```
n = input("Enter a string : ")
s = set("AEIOUaeiou")
count=0
for i in n:
    if i in s:
```

```
        count+=1
print("Total number of vowels = ",count)
```

```
Total number of vowels =  3
```

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

In [32]:
```
n = input("Enter a string:")
s = set("01")
for i in n:
    if i not in s:
        print("Not a Binary String")
        break
else:
    print("Is a Binary String")
```

```
Not a Binary String
```

13) WAP to sort dictionary by key or value.

In [34]:
```
key = input("Enter keys in space-separated form: ").split()
key = map(int, key)
value = input("Enter values in space-separated form: ").split()
value = map(int, value)
d = {i: j for (i, j) in zip(key, value)}
sorted_by_key = {k: v for k, v in sorted(d.items())}
sorted_by_value = {k: v for k, v in sorted(d.items(), key=lambda item: item[1])}
print("Sort by Key =", sorted_by_key)
print("Sort by Value =", sorted_by_value)
```

```
Sort by Key = {2: 1, 4: 3, 6: 5, 8: 7}
Sort by Value = {2: 1, 4: 3, 6: 5, 8: 7}
```

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

In [36]:
```
key = input("Enter keys in space-separated form: ").split()
key = map(int, key)
value = input("Enter values in space-separated form: ").split()
value = map(int, value)
d = {i: j for (i, j) in zip(key, value)}
print("Sum of values = ",sum(d.values()))
```

```
Sum of values =  22
```

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.

In [40]:
```
key = input("Enter keys in space-separated form: ").split()
value = input("Enter values in space-separated form: ").split()
value = map(int, value)
dict1 = {i:j for (i,j) in zip(key,value)}
n = input("Enter the key : ")
```

```
if n not in dict1.keys():
    print("Key Not Found")
else:
    print("Value = ",dict1[n])
```

```
Value =  3
```

In [ ]:

Python Programming - 2301CS404

Lab - 8

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

User Defined Function

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

In [2]:
```
def calc_bmi(mass,height):
    bmi = mass/(height**2)
    return bmi

mass = float(input("Enter your mass(weight):"))
height = float(input("Enter yuor weight:"))
bmi = calc_bmi(mass,height)
print(f"your bmi is: {bmi}")
```

```
your bmi is: 0.0015833333333333333
```

02) Write a function that add first n numbers.

In [4]:
```
def add_numbers(n):
    total = (n*(n+1)/2)
    return total

n =10
result = add_numbers(n)
print(f"addtion of numbers is : {result}")
```

```
addtion of numbers is : 55.0
```

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

In [8]:
```
def is_prime(num):
    if num <= 1:
        return 0
    for i in range (2,int(num**0.5)+1):
        if num % i == 0:
            return 0
    return 1

num = int(input("Enter a number: "))
result = is_prime(num)
print(f"{num} is prime number: {result} ")
```

```
7 is prime number: 1
```

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

In [10]:
```
def primes_in_range(start, end):
    def is_prime(number):
        if number <= 1:
            return False
        for i in range(2, int(number ** 0.5) + 1):
            if number % i == 0:
                return False
        return True

    prime_numbers = [num for num in range(start, end + 1) if is_prime(num)]
    return prime_numbers

start = 10
end = 50
prime_list = primes_in_range(start, end)
print(f"Prime numbers between {start} and {end}: {prime_list}")
```

```
Prime numbers between 10 and 50: [11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

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

In [2]:
```
s = input("Enter a string : ")
def isPalindrome(s):
    return s==s[::-1]
print(isPalindrome(s))
```

```
False
```

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

In [4]:
```
li = map(int,input("Enter space separated values : ").split())
def total(li):
    sum=0
```

```
    for i in li:
        sum += i
    return sum
print("Sum = ",total(li))
```

```
Sum =  10
```

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

In [29]:
```
def sum_of_first_element(tuple_list):
    return sum(t[0] for t in tuple_list)
tuple_list = [(1,2),(3,4),(5,6)]
result = sum_of_first_element(tuple_list)
print(f"The sum of the first elements is: {result}")
```

```
The sum of the first elements is: 9
```

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

In [31]:
```
def fibonacci(n):
    if n<=0:
        return 0
    elif n==1:
        return 1
    else:
        return fibonacci(n-1)+fibonacci(n-2)
n = int(input("Enter the term position (n): "))
result = fibonacci(n)
print(f"The {n}th term of the Fibonacci Series is: {result}")
```

```
The 2th term of the Fibonacci Series is: 1
```

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

In [33]:
```
def get_student_name(rollno, student_dict):
    return student_dict.get(rollno, "Roll number not found")
dict1 = {101: 'Ajay', 102: 'Rahul', 103: 'Jay', 104: 'Pooja'}
rollno = int(input("Enter the roll number: "))
name = get_student_name(rollno, dict1)
print(f"The name of the student with roll number {rollno} is: {name}")
```

```
The name of the student with roll number 21 is: Roll number not found
```

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

In [35]:
```
def sum_of_scores_ending_with_zero(scores):
    return sum(score for score in scores if score % 10 == 0)
scores = [200, 456, 300, 100, 234, 678]
result = sum_of_scores_ending_with_zero(scores)
print(f"The sum of scores ending with zero is: {result}")
```

```
The sum of scores ending with zero is: 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'}

In [37]:
```
def invert_dictionary(original_dict):
    return {value: key for key, value in original_dict.items()}
original_dict = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
inverted_dict = invert_dictionary(original_dict)
print("Original Dictionary:", original_dict)
print("Inverted Dictionary:", inverted_dict)
```

```
Original Dictionary: {'a': 10, 'b': 20, 'c': 30, 'd': 40}
Inverted Dictionary: {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.

In [49]:
```
def is_pangram(s):
    s = s.lower()
    alphabet_set = set("abcdefghijklmnopqrstuvwxyz")
    return alphabet_set.issubset(set(s))
input_string = input("Enter a string: ")
if is_pangram(input_string):
    print("The string is a pangram.")
else:
    print("The string is not a pangram.")
```

```
The string is not a pangram.
```

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

In [47]:
```
def count_upper_lower(s):
    no_upper = 0
    no_lower = 0
    for char in s:
        if char.isupper():
            no_upper += 1
```

```
        elif char.islower():
            no_lower += 1
    return no_upper, no_lower
s1 = input('Enter a String:')
no_upper, no_lower = count_upper_lower(s1)
print(f"Number of uppercase letters: {no_upper}")
print(f"Number of lowercase letters: {no_lower}")
```

```
Number of uppercase letters: 0
Number of lowercase letters: 3
```

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

In [51]:
```
smallest = lambda x, y: x if x < y else y
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
result = smallest(num1, num2)
print(f"The smallest number is: {result}")
```

```
The smallest number is: 1.0
```

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

In [53]:
```
students = ["Alice", "Bob", "Alexander", "Catherine", "David", "Elizabeth"]
def has_more_than_7_chars(name):
    return len(name) > 7
filtered_names = list(filter(has_more_than_7_chars, students))
print("Names with more than 7 characters:", filtered_names)
```

```
Names with more than 7 characters: ['Alexander', 'Catherine', 'Elizabeth']
```

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

In [56]:
```
students = ["alice", "bob", "alexander", "catherine", "david", "elizabeth"]
def capitalize_first_letter(name):
    return name.capitalize()
capitalized_names = list(map(capitalize_first_letter, students))
print("Names with first letter capitalized:", capitalized_names)
```

```
Names with first letter capitalized: ['Alice', 'Bob', 'Alexander', 'Catherine',
'David', 'Elizabeth']
```

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

In [60]:
```
def positional_args(a, b):
    return a + b
```

```
def keyword_args(a, b):
    return a - b

def default_args(a, b=5):
    return a * b


def variable_length_positional(*args):
    return sum(args)


def variable_length_keyword(**kwargs):
    return kwargs


def keyword_only_args(*, a, b):
    return a + b


def positional_only_args(a, b, /):
    return a * b

print("Positional Arguments Result:", positional_args(10, 20))
print("Keyword Arguments Result:", keyword_args(b=20, a=10))
print("Default Arguments Result:", default_args(10))
print("Variable-Length Positional Arguments Result:", variable_length_positional
print("Variable-Length Keyword Arguments Result:", variable_length_keyword(name=
print("Keyword-Only Arguments Result:", keyword_only_args(a=10, b=20))
print("Positional-Only Arguments Result:", positional_only_args(10, 20))
```

```
Positional Arguments Result: 30
Keyword Arguments Result: -10
Default Arguments Result: 50
Variable-Length Positional Arguments Result: 15
Variable-Length Keyword Arguments Result: {'name': 'Alice', 'age': 25, 'city': 'N
ew York'}
Keyword-Only Arguments Result: 30
Positional-Only Arguments Result: 200
```

Python Programming - 2301CS404

Lab - 9

Name: Jadeja Rudrarajsinh

Enrollment No:23010101411

Roll No:487

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

In [12]:
```
fp = open("file1.txt","r")
print(fp.read())
print(fp.readline())
print(fp.readlines())
fp.close()
```

```
Jay mataji from Rudrarajsinh Jadeja

[]
```

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

In [46]:
```
fp = open("new.txt","r")
print(fp.read())
fp.close()
```

```
Khamma ghani this is Rudrarajsinh Jadeja
```

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

In [54]:
```
fp = open("field.txt","r")
for i in range(5):
    print(fp.readline().strip())
fp.close()
```

```
Jadeja
jbdfkswf
fklkwnfls
lkndlgled
klndl
```

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

In [61]:
```
fp = open("field.txt","r")
x = max(fp.read().split(),key = len)
print(x)
```

```
Rudrarajsinh
```

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

In [67]:
```
fp = open("field.txt","r")
print(len(fp.read()))
print(len(fp.read().split()))
```

```
45
0
```

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

In [79]:
```
fp = open("field.txt","r")
x1=fp.read()
fp1 = open("field2.txt","w")
fp1.write(x1)
print(x1)
fp1.close()
```

```
Jadeja
Rudrarajsinh
fklkwnfls
lkndlgled
klndl
```

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

In [81]:
```
import os
fp = open("field.txt","r")
```

```
file_size = os.path.getsize('field.txt')
print("File Size is :", file_size, "bytes")
```

```
File Size is : 45 bytes
```

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

In [2]:
```
def frequency(FileName, word):
    with open(FileName, "r") as fp:
        lines = fp.readlines()
        words = [w for line in lines for w in line.strip().split()]  # Flatten t
    print(f"Occurrence of '{word}' = {words.count(word)}")

FileName = input("Enter File Name: ")
word = input("Enter word: ")
frequency(FileName, word)
```

```
Occurrence of 'Jay' = 1
```

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.

In [92]:
```
fp = open("field.txt","w+")

for i in range(5):
    h1 = input("enter the 1 subject mark:-")
    fp.write(h1+" ")
fp.close()
fp = open("field.txt","r+")
# print(fp.read())
a=fp.read()
b=a.split()
print(a)
print(max(b))
```

```
1 2 3 4 5
5
```

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

(Note: each number should be in new line)

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

def write_primes_to_file(filename, count):
    primes = []
    num = 2
    while len(primes) < count:
```

```
        if is_prime(num):
            primes.append(num)
        num += 1

    with open(filename, "w") as fp:
        fp.writelines([str(prime) + "\n" for prime in primes])

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

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

In [85]:
```
fp = open("field.txt","r")
fp1 = open("field2.txt","r")
str1 = fp.read()
str1 = str1 + "\n" + fp1.read()
fp2 = open("field3.txt","w")
fp2.write(str1)
print(str1)
fp2.close()
```

```
Jadeja
Rudrarajsinh
fklkwnfls
lkndlgled
klndl
Jadeja
Rudrarajsinh
fklkwnfls
lkndlgled
klndl
```

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

In [87]:
```
fp = open("field.txt","r")
x1=fp.read()
print(x1)
a=x1.replace("Rudrarajsinh","Jadeja")
fp1 =open("field2.txt","w")
fp1.write(a)
fp1.close()
```

```
Jadeja
Rudrarajsinh
fklkwnfls
lkndlgled
klndl
```

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

In [89]:
```
fp = open("field.txt","r")
print("Before reading pointer is at:", fp.tell())
fp.read(2)
print("After reading 2 characters pointer is at: ",fp.tell())
fp.read(3)
```

```
print("After reading 5 characters pointer is at: ",fp.tell())
fp.read()
print("after going to the end of the file, pointer is at:", fp.tell())
fp.close()

fp = open("field.txt","r")
print("Before reading pointer is at:", fp.seek(0,2))
fp.read(2)
print("After reading 2 characters pointer is at: ",fp.seek(0,2))
fp.read(3)
print("After reading 5 characters pointer is at: ",fp.seek(0,2))
fp.read()
print("after going to the end of the file, pointer is at:", fp.seek(0,2))
fp.close()
```

```
Before reading pointer is at: 0
After reading 2 characters pointer is at:  2
After reading 5 characters pointer is at:  5
after going to the end of the file, pointer is at: 45
Before reading pointer is at: 45
After reading 2 characters pointer is at:  45
After reading 5 characters pointer is at:  45
after going to the end of the file, pointer is at: 45
```

Python Programming - 2301CS404

Lab - 10

Name: Jadeja Rudrarajsinh

Enrollment No:23010101411

Roll No:487

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.

In [2]:
```
try:
    x=int(input("enter no : "))
    y=int(input("enter no : "))

    result=x/y

    print("result",result)
except ZeroDivisionError:
    print("cannot be divide wd 0")
except ValueError:
    print("invalid input")
except TypeError:
    print("type mistake")
```

```
result 1.0909090909090908
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [12]:
```
list=[1,2,3]
dic={"name":"JP","roll":7}
try:
    print(list[2])
    print(dic["no"])

except IndexError:
    print("index error")

except KeyError:
    print("key error")
```

```
3
key error
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [16]:
```
try:
 fp = open("abc1.txt","r")
except FileNotFoundError as err:
 print(err)
else:
 print(fp.read())
 fp.close()

try:
    import non_existent_module
except ModuleNotFoundError as a:
    print("enter : ",str(a))
```

```
Jay mataji from Jadeja Rudrarajsinh!!!!!!
enter :  No module named 'non_existent_module'
```

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

In [18]:
```
try:
    a=int(input("1st no :"))
    b=int(input("2nd no :"))
    c=a/b
    print("result :",c)
except Exception as e:
    print(f"Error:{e}")
```

```
result : 0.5
```

05) WAP to demonstrate else and finally block.

In [22]:
```
try:
    a=int(input("1st no :"))
    b=int(input("2nd no :"))
    c=a/b
except Exception as e:
    print("Error:{e}")
else:
    print("result:",result)
finally:
    print("done")
```

```
result: 1.0909090909090908
done
```

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.

In [36]:
```
grade_input = input("enter grade")
try:
    grades=[int(grade) for grade in grade_input.split(',')]
    print("grades",grades)
except ValueError:
    print("error: plz enter valid grades separated by commas,with no non-numeric
```

```
grades [2]
```

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

In [24]:
```
def divide(a,b):
    try:
        result = a/b
        return result
    except ZeroDivisionError:
        print("error: cannot divide by 0")
        return None
print(divide(10,2))
print(divide(10,0))
```

```
5.0
error: cannot divide by 0
None
```

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.

In [26]:
```
try:
    age = int(input("Enter your age: "))
    if age < 18:
        raise ValueError("Enter Valid Age")
    print("Your age is:", age)
except ValueError as e:
    print("Error:", e)
```

```
Your age is: 20
```

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.

In [30]:
```
try:
    un = input("Enter your username: ")

    if len(un) < 5 or len(un) > 15:
        raise Exception("Username must be between 5 and 15 characters long")
    print("Your username is:", un)
except Exception as e:
    print("Error:", e)
```

```
Your username is: rudrarajsinh_9
```

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.

In [32]:
```
import math
try:
    num = float(input("Enter a number: "))
    if num < 0:
        raise Exception("Cannot calculate the square root of a negative number")
    print("Square root:", math.sqrt(num))
except Exception as e:
    print("Error:", e)
except ValueError:
    print("Error: Please enter a valid number")
```

```
Square root: 3.0
```

Python Programming - 2301CS404

Lab - 11

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

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.

In [1]:
```
import calc

x=int(input("enter x : "))
y=int(input("enter y : "))

print(f"{x} + {y} = {calc.add(x, y)}")
print(f"{x} - {y} = {calc.sub(x, y)}")
print(f"{x} * {y} = {calc.mul(x, y)}")
print(f"{x} / {y} = {calc.div(x, y)}")
```

```
2 + 4 = 6
2 - 4 = -2
2 * 4 = 8
2 / 4 = 0.5
```

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

In [13]:
```
import random

my_string = "Hello,World!"

random_character = random.choice(my_string)

print(random_character)
```

,

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

In [31]:
```
import random

my_list=[1,2,3,4,5,6,7,8,9,11,12,45]

random_element = random.choice(my_list)

print(random_element)
```

9

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

In [35]:
```
import random
dice_roll = random.randint(1, 6)

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


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

random.seed(2)
print(random.randint(1, 6))
```

1
1
1

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

In [57]:
```
import random

divisible_by_5 = [num for num in range(100, 999) if num % 5 == 0]


random_integers = random.sample(divisible_by_5,k= 3)

print(random_integers)
```

[505, 585, 640]

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

In [74]:
```
import random
ticket=random.sample(range(100,999),100)
print(ticket)
for i in range(1,3):
    r=random.choice(ticket)
    print('your winner',r)
```

```
[674, 337, 635, 384, 162, 214, 993, 791, 938, 906, 488, 977, 473, 318, 426, 464,
179, 442, 568, 471, 270, 609, 552, 398, 572, 237, 835, 976, 754, 321, 379, 434, 2
62, 201, 343, 580, 294, 870, 794, 482, 289, 983, 243, 914, 238, 338, 375, 927, 66
3, 748, 487, 509, 951, 943, 867, 450, 387, 837, 709, 614, 694, 806, 849, 428, 86
0, 947, 869, 830, 821, 747, 871, 833, 975, 644, 737, 752, 786, 174, 476, 415, 50
4, 595, 278, 364, 462, 551, 588, 189, 290, 422, 988, 230, 128, 206, 459, 271, 46
7, 178, 936, 879]
your winner 364
your winner 450
```

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

In [59]:
```
from datetime import datetime
datetime.now()
```

Out[59]:
```
datetime.datetime(2025, 2, 21, 10, 55, 37, 871946)
```

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

In [78]:
```
from datetime import datetime, timedelta

given_date = datetime(2025, 2, 21)
new_date = given_date - timedelta(days=7)
print("New Date after subtracting a week:", new_date)
```

```
New Date after subtracting a week: 2025-02-14 00:00:00
```

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

In [90]:
```
from datetime import datetime

date1 = datetime(2025, 2, 21)
date2 = datetime(2025, 3, 31)

difference = date2 - date1
print("Number of days between dates:", difference.days)
```

```
Number of days between dates: 38
```

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

In [94]:
```
from datetime import datetime

given_date = datetime(2025, 2, 21)
day_of_week = given_date.strftime("%A")
print("Day of the week:", day_of_week)
```

```
Day of the week: Friday
```

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

In [98]:
```
from datetime import datetime

now = datetime.now()
print("Current Date and Time:", now)

formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")
print("Formatted Date and Time:", formatted_date)
```

```
Current Date and Time: 2025-02-21 11:20:28.752426
Formatted Date and Time: 2025-02-21 11:20:28
```

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

In [96]:
```
import math

print("Pi:", math.pi)
print("Square Root of 16:", math.sqrt(16))
print("Factorial of 5:", math.factorial(5))
print("Cosine of 0 degrees:", math.cos(0))
```

```
Pi: 3.141592653589793
Square Root of 16: 4.0
Factorial of 5: 120
Cosine of 0 degrees: 1.0
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 12

Name: Jadeja Rudrarajsinh

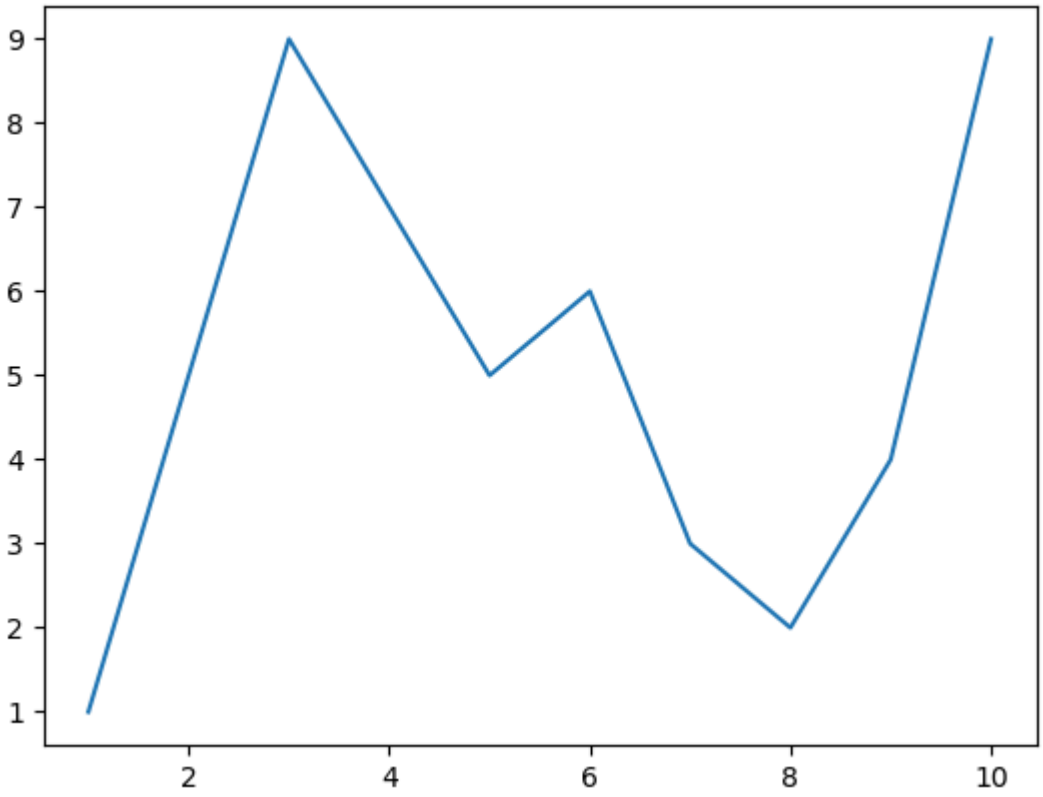
Enrollment No: 23010101411

Roll No: 487

In [3]:
```
#import matplotlib below
import matplotlib.pyplot as plt
```

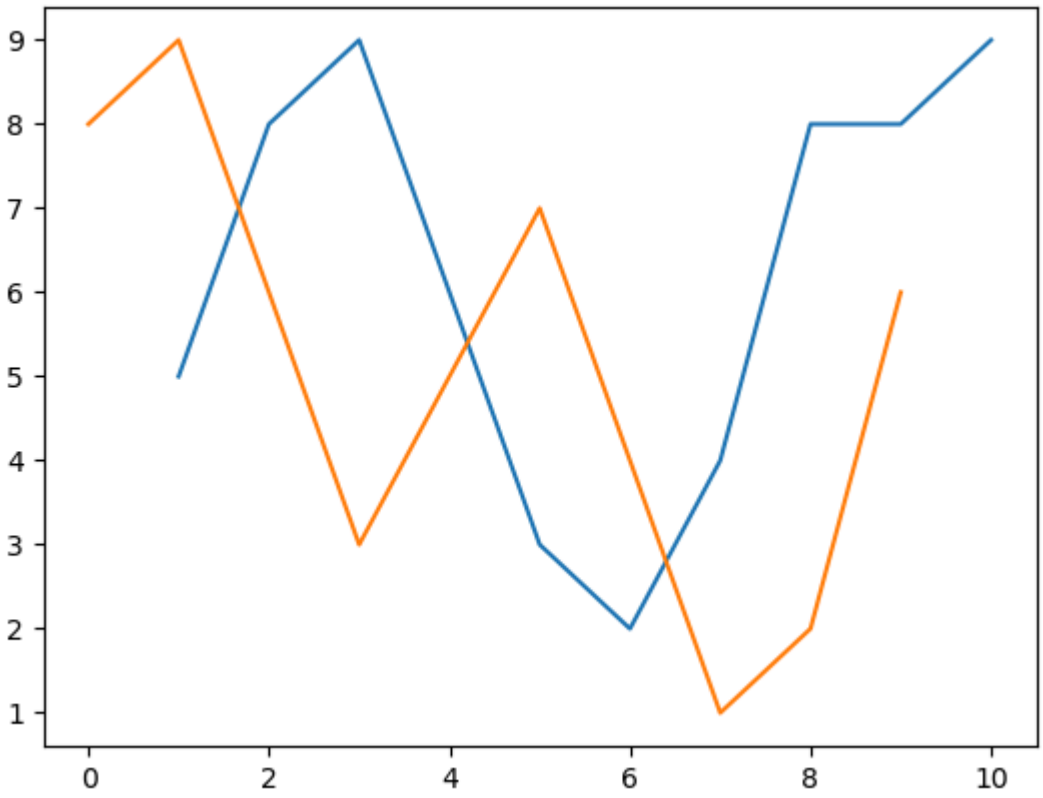
In [5]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]

# write a code to display the line chart of above x & y
plt.plot(x,y)
plt.show()
```



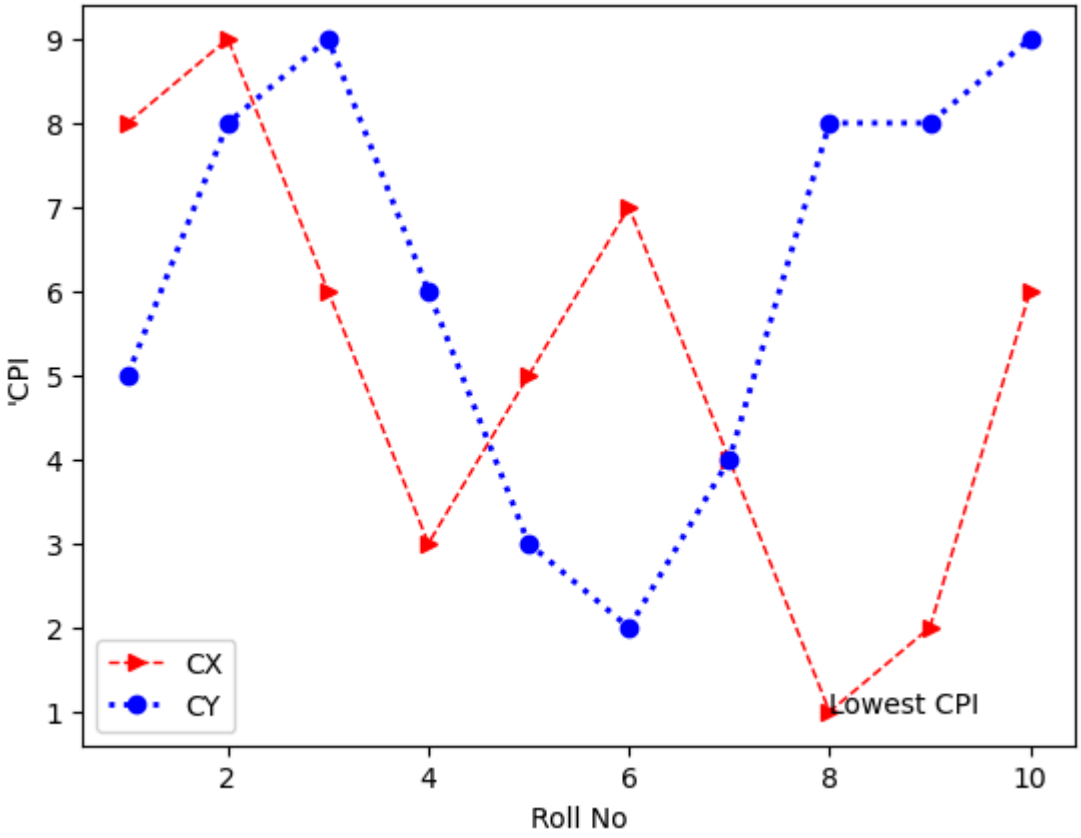
In [7]:
```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarks = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]

# write a code to display two lines in a line chart (data given above)
plt.plot(x, cxMarks, cyMarks)
plt.show()
```



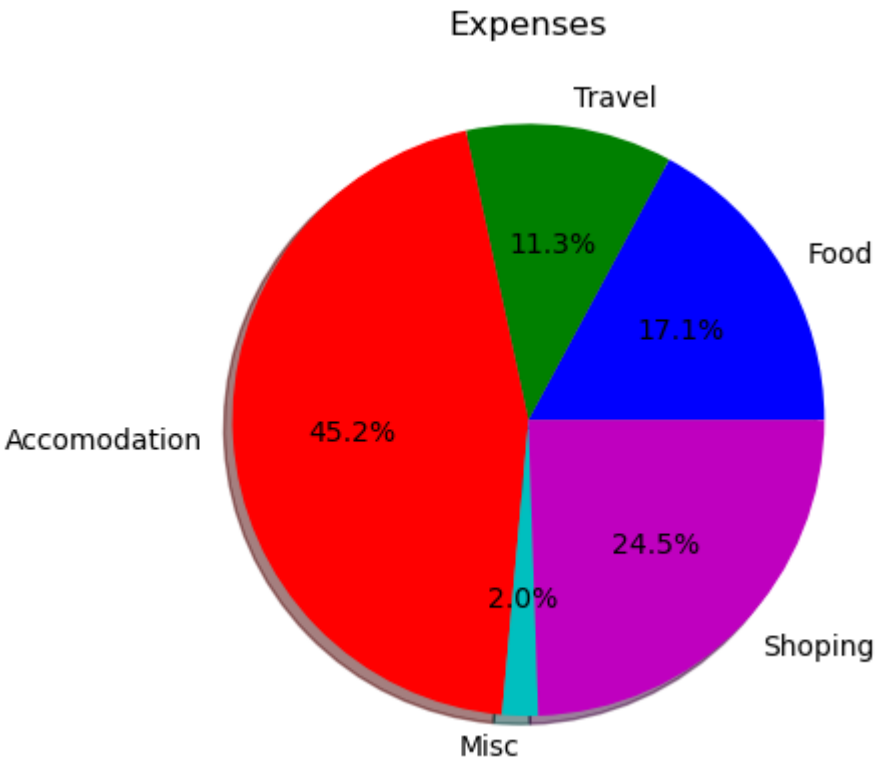
In [13]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]


# write a code to generate below graph
```



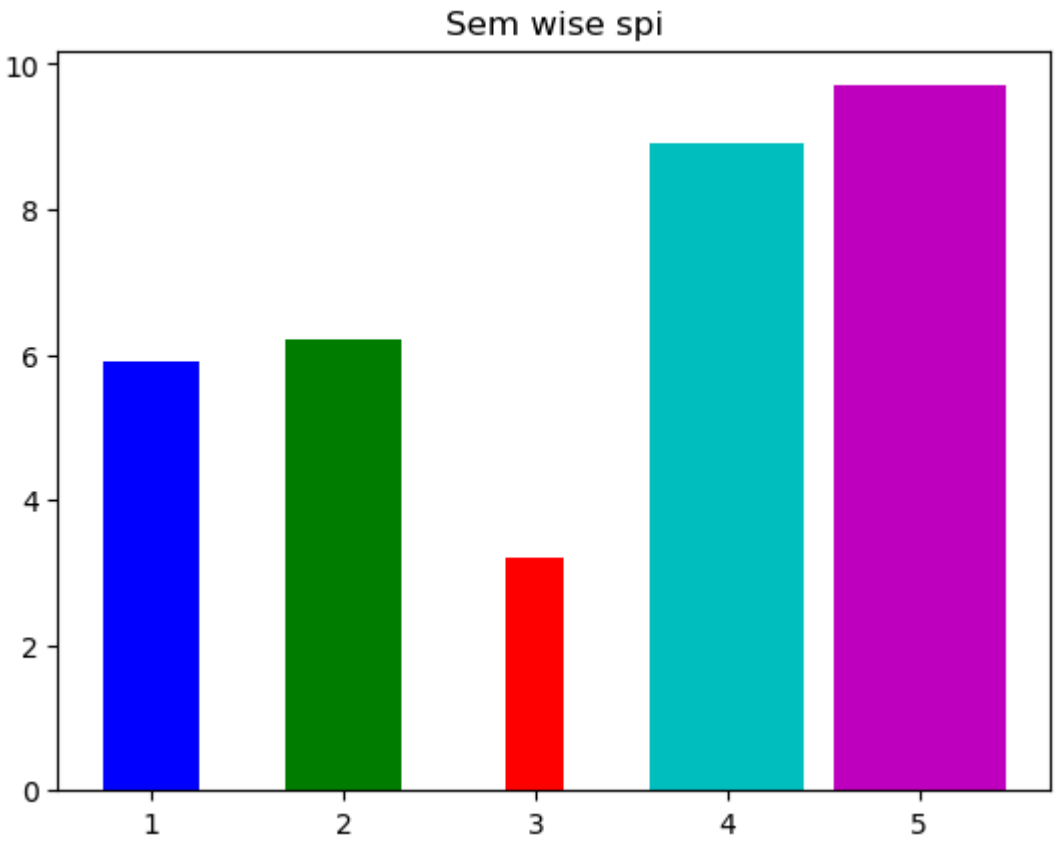
04) WAP to demonstrate the use of Pie chart.

In [9]:
```
values = [305,201,805,35,436]
l = ["Food", "Travel", "Accomodation", "Misc", "Shoping"]
c = ["b", "g", "r", "c", "m"]
plt.pie(values,colors=c,labels=l,shadow=True,autopct="%1.1f%%")
plt.title("Expenses")
plt.show()
```



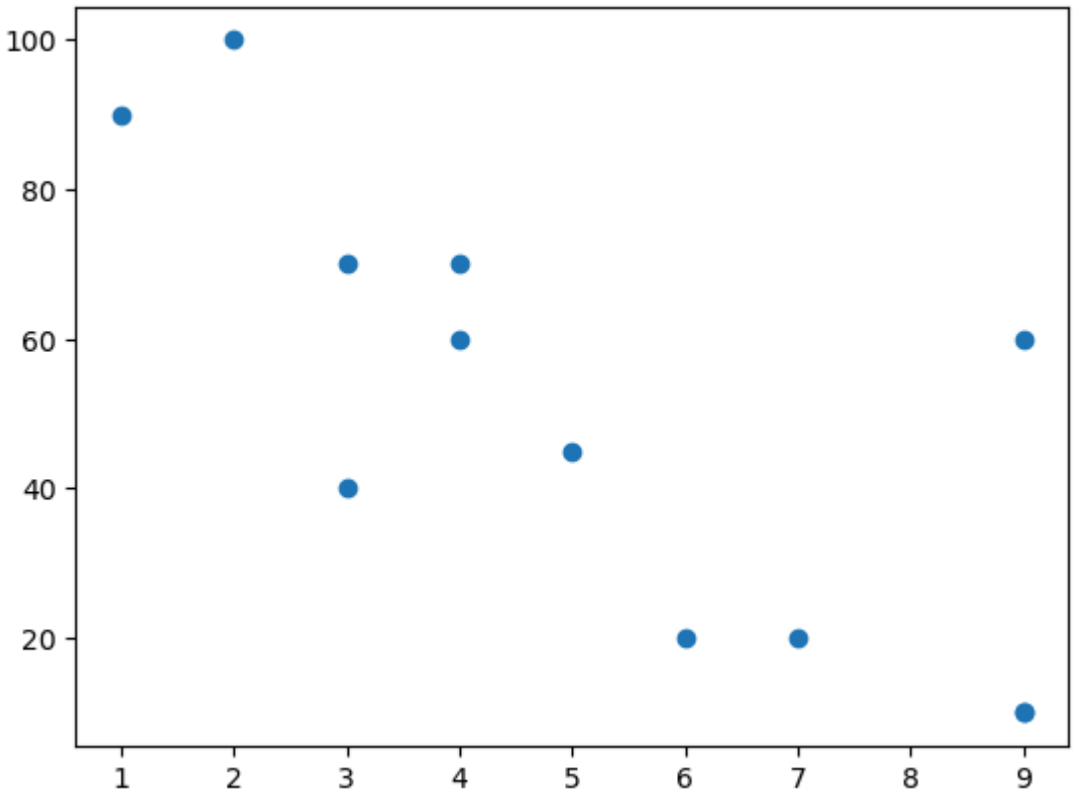
05) WAP to demonstrate the use of Bar chart.

In [11]:
```
x = [1,2,3,4,5]
y = [5.9,6.2,3.2,8.9,9.7]
l = ["1st", "2nd", "3rd", "4th", "5th"]
c = ["b", "g", "r", "c", "m"]
w = [0.5, 0.6, 0.3, 0.8, 0.9]
plt.title("Sem wise spi")
plt.bar(x,y,color=c,label=l,width=w)
plt.show()
```



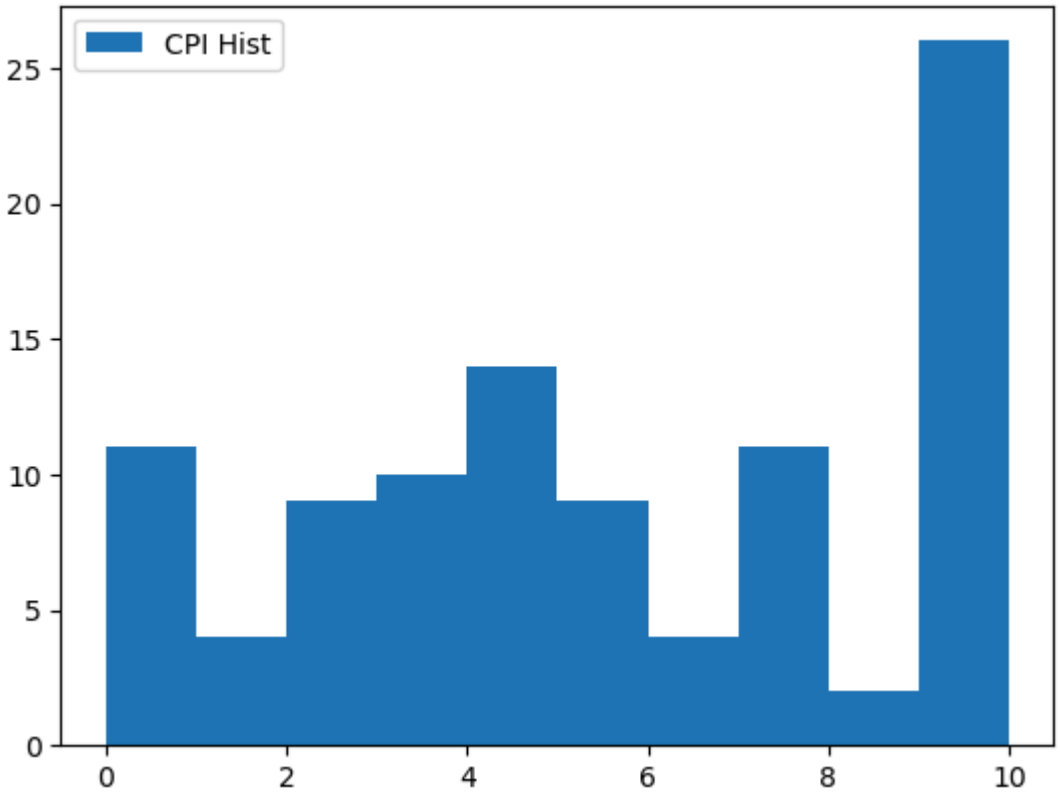
06) WAP to demonstrate the use of Scatter Plot.

In [13]:
```
carAge = [2,5,7,9,4,3,1,9,4,3,6,9]
carspeed = [100,45,20,10,60,70,90,60,70,40,20,10]
plt.scatter(carAge, carspeed)
plt.show()
```



07) WAP to demonstrate the use of Histogram.

In [17]:
```
import random
cpis = [random.randint(0, 10) for _ in range(100)]
%matplotlib inline
plt.hist(cpis, bins=10, histtype="stepfilled", align="mid", label="CPI Hist")
plt.legend()
plt.show()
```



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

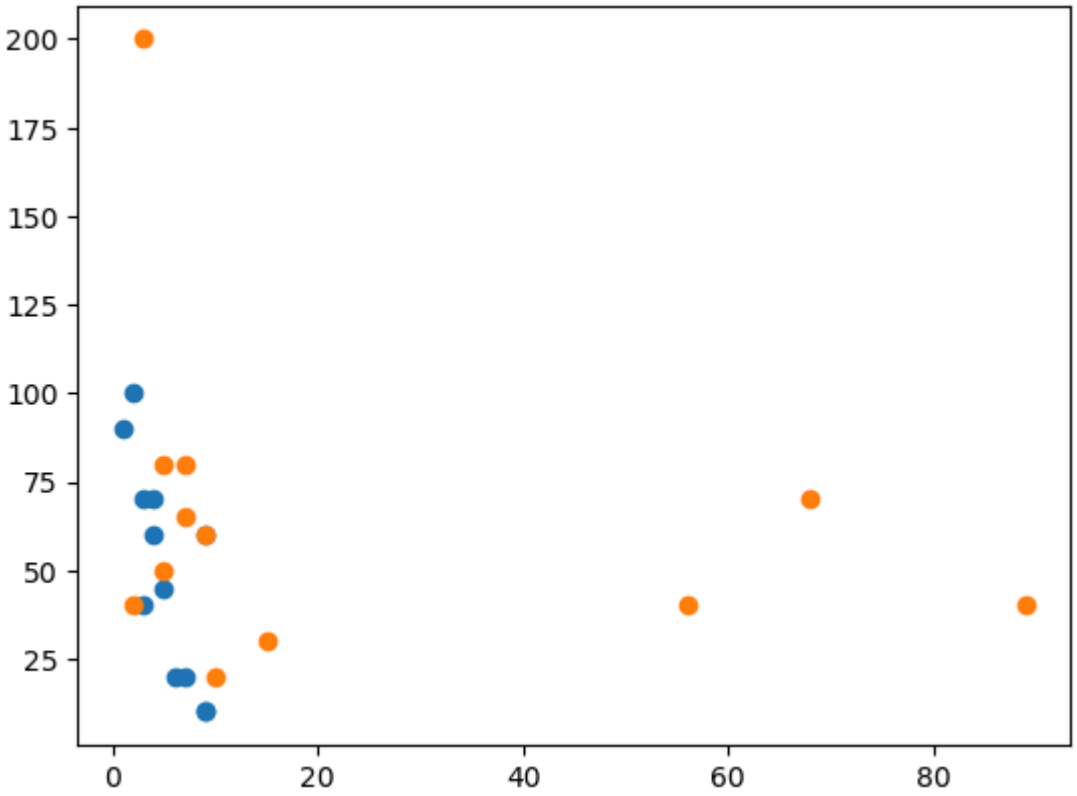
In [22]:
```
Import matplotlib.pyplot as plt
products = ['Product A', 'Product B', 'Product C', 'Product D', 'Product E']
sales = [100, 150, 120, 90, 200]
plt.bar(products, sales, color='b')
for i in range(len(products)):
    plt.text(i, sales[i]+1,str(sales[i]))
plt.title('Sales of Products')
plt.xlabel('Products')
plt.ylabel('Sales')
plt.show()
```

```
  Cell In[22], line 1
    Import matplotlib.pyplot as plt
           ^
SyntaxError: invalid syntax
```

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

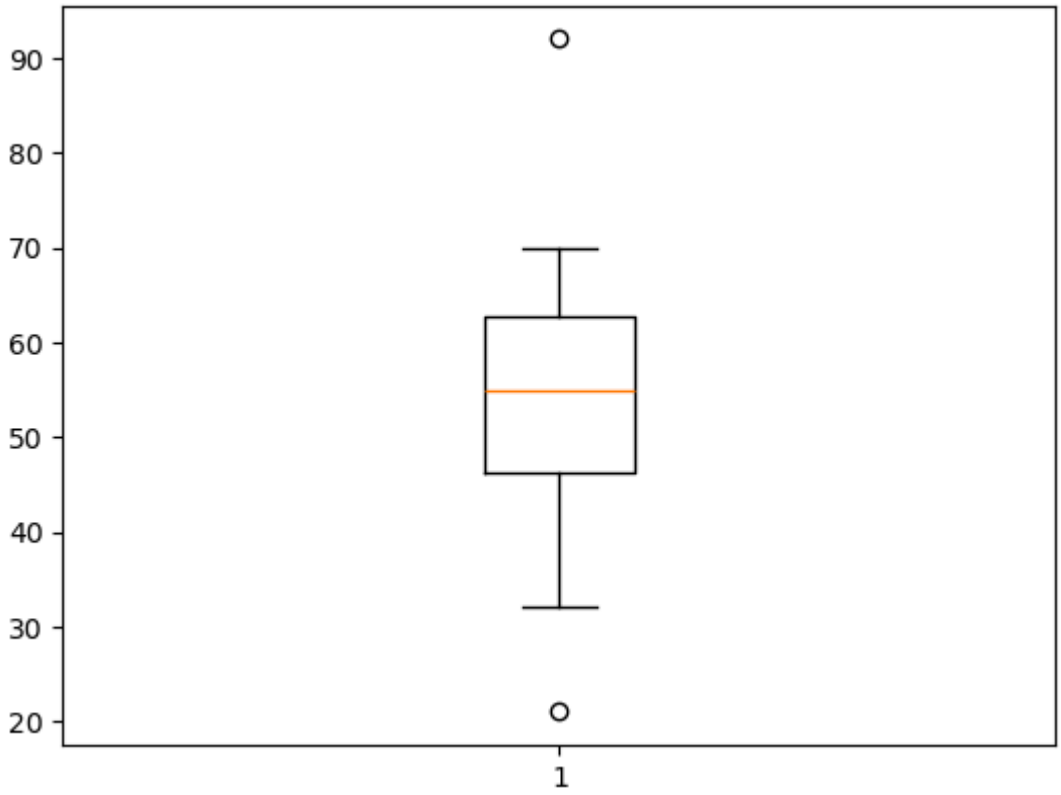
In [19]:
```
carage = [2,5,7,9,4,3,1,9,4,3,6,9]
carspeed = [100,45,20,10,60,70,90,60,70,40,20,10]
carage1 = [3,7,7,10,15,56,5,68,89,9,5,2]
carspeed1 = [200,65,80,20,30,40,50,70,40,60,80,40]
plt.scatter(carage,carspeed)
plt.scatter(carage1,carspeed1)

plt.show()
```



10) WAP to create a Box Plot.

In [21]:
```
plt.boxplot([50,45,52,63,70,21,56,68,54,57,35,62,92,32])
plt.show()
```


90
80
70
60
50
40
30
20
1

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 13

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

OOP

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

In [2]:
```
class Students:
    def __init__(self,name,age,grade):
        self.name=name
        self.age=age
        self.grade=grade
name = input("Enter name of Student : ")
age = int(input("Enter age of Student : "))
grade = input("Enter grade of Student : ")
s = Students(name,age,grade)
print(f"Name = {s.name},Age = {s.age},Grade = {s.grade}")
```

```
Name = Jadeja Rudrarajsinh,Age = 20,Grade = 8.7
```

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.

In [4]:
```
class Bank_Account:
    def __init__(self,Account_No,User_Name,Email,Account_Type,Account_Balance):
        self.Account_No=Account_No
        self.User_Name=User_Name
        self.Email=Email
        self.Account_Type=Account_Type
        self.Account_Balance=Account_Balance
    def GetAccountDetails(self):
        self.Account_No = int(input("Enter Account Number"))
        self.User_Name = input("Enter UserName")
        self.Email = input("Enter Email")
        self.Account_Type = input("Enter Account Type")
        self.Account_Balance = float(input("Enter Account Balance"))

    def DisplayAccountDetails(self):
        print(f"Account_No:{self.Account_No}")
        print(f"User_Name:{self.User_Name}")
        print(f"Email:{self.Email}")
        print(f"Account_Type:{self.Account_Type}")
        print(f"Account_Balance:{self.Account_Balance}")
def main():
    account = Bank_Account("","","","",0.0)
    account.GetAccountDetails()
    account.DisplayAccountDetails()
if __name__ == "__main__":
    main()
```

```
Account_No:23010101411
User_Name:Rudrarajsinh
Email:rudra@gmail.com
Account_Type:savings
Account_Balance:9999999999.0
```

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

In [6]:
```
import math
class Circle:
    def __init__(self,r):
        self.r=r
    def area(self):
        return math.pi*self.r*self.r
    def perimeter(self):
        return 2*math.pi*self.r
r = float(input("Enter radius of circle : "))
c = Circle(r)
print(f"Area = {c.area()}")
print(f"Perimeter = {c.perimeter()}")
```

```
Area = 1661.9025137490005
Perimeter = 144.51326206513048
```

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

In [8]:
```
class Employee:
    def __init__(self,name,age,salary):
```

```
        self.name=name
        self.age=age
        self.salary=salary
    def update(self):
        self.name=input("Enter name : ")
        self.age=int(input("Enter age : "))
        self.salary=float(input("Enter salary : "))
    def display(self):
        print(f"Name = {self.name}")
        print(f"Age = {self.age}")
        print(f"Salary = {self.salary}")
name = input("Enter name : ")
age = int(input("Enter age : "))
salary = float(input("Enter salary"))
e = Employee(name,age,salary)
n = int(input("Enter 1 to update and 0 to not update"))
if n==1:
    e.update()
    e.display()
else:
    e.display()
```

```
Name = Rudrarajsinh Jadeja
Age = 19
Salary = 99999.0
```

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

In [10]:

```
class BankAccount:
    account=0
    def __init__(self,balance):
        self.balance=balance
    def deposit(self,amount):
        self.balance+=amount
    def withdraw(self,amount):
        if balance>=amount:
            self.balance-=amount
        else:
            print("Insufficient Balance")
    def checkBalance(self):
        print("Current Balance =",self.balance)
balance = int(input("Enter balance"))
b = BankAccount(balance)
choice=0
while choice!=-1:
    print("Enter 1 to deposit")
    print("Enter 2 to withdraw")
    print("Enter 3 to display Balance")
    print("Enter -1 to exit")
    choice = int(input())
    match(choice):
        case 1:
            amount = int(input("Enter amount to deposit : "))
            b.deposit(amount)

        case 2:
            amount = int(input("Enter amount to withdraw : "))
            b.withdraw(amount)
```

```
        case 3:
            b.checkBalance()

        case -1:
            break
```

```
Enter 1 to deposit
Enter 2 to withdraw
Enter 3 to display Balance
Enter -1 to exit
Enter 1 to deposit
Enter 2 to withdraw
Enter 3 to display Balance
Enter -1 to exit
```

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

In [12]:
```
class Inventory:
    def __init__(self):
        # Initialize an empty list to hold inventory items
        self.items = []

    def add_item(self, item_name, price, quantity):
        # Add a new item or update the quantity if it already exists
        for item in self.items:
            if item[0] == item_name:  # Check if item already exists
                item[2] += quantity  # Update quantity
                print(f"Added {quantity} more of '{item_name}' to the inventory.
                return
        self.items.append([item_name, price, quantity])  # Add new item
        print(f"Added '{item_name}' to the inventory.")

    def remove_item(self, item_name, quantity):
        # Remove a specific quantity of an item or the item entirely
        for item in self.items:
            if item[0] == item_name:
                if quantity >= item[2]:
                    self.items.remove(item)  # Remove the entire item
                    print(f"Removed all of '{item_name}' from the inventory.")
                else:
                    item[2] -= quantity  # Update quantity
                    print(f"Removed {quantity} of '{item_name}' from the invento
                return
        print(f"'{item_name}' is not in the inventory.")

    def update_item(self, item_name, price=None, quantity=None):
        # Update the price and/or quantity of an item
        for item in self.items:
            if item[0] == item_name:
                if price is not None:
                    item[1] = price  # Update price
                if quantity is not None:
                    item[2] = quantity  # Update quantity
                print(f"Updated '{item_name}' in the inventory.")
                return
        print(f"'{item_name}' is not in the inventory.")
```

```
    def display_inventory(self):
        # Display the entire inventory with details
        if not self.items:
            print("The inventory is empty.")
        else:
            print("Current Inventory:")
            for item in self.items:
                print(f"{item[0]} - Price: {item[1]}, Quantity: {item[2]}")
inventory = Inventory()

# Add items
inventory.add_item("Apple", 10, 50)
inventory.add_item("Banana", 5, 100)
inventory.add_item("Orange", 8, 30)

# Display the inventory
inventory.display_inventory()

# Remove some items
inventory.remove_item("Apple", 20)

# Update the price of an item
inventory.update_item("Banana", price=6)

# Update the quantity of an item
inventory.update_item("Orange", quantity=50)

# Display the updated inventory
inventory.display_inventory()
```

```
Added 'Apple' to the inventory.
Added 'Banana' to the inventory.
Added 'Orange' to the inventory.
Current Inventory:
Apple - Price: 10, Quantity: 50
Banana - Price: 5, Quantity: 100
Orange - Price: 8, Quantity: 30
Removed 20 of 'Apple' from the inventory.
Updated 'Banana' in the inventory.
Updated 'Orange' in the inventory.
Current Inventory:
Apple - Price: 10, Quantity: 30
Banana - Price: 6, Quantity: 100
Orange - Price: 8, Quantity: 50
```

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

In [16]:

```
class Book:
    def __init__(self, title, author, year_published, genre):
        # Instance attributes
        self.title = title
        self.author = author
        self.year_published = year_published
        self.genre = genre

    def display_info(self):
        # Method to display book details
        print(f"'{self.title}' by {self.author}, published in {self.year_publish
```

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.

In [20]:
```
class student_kit:
    principal_name="Mr ABC"
    def __init__(self,student_name):
        self.student_name = student_name
        self.attendance_days=0
    def attendance(self,days_present):
        self.attendance_days+=days_present
        print(f"Marked {days_present} days of attendance for {self.student_name}
    def generate_certificate(self):
        print(f"\n--- Certificate of Attendance ---")
        print(f"Principal: {student_kit.principal_name}")
        print(f"Student Name: {self.student_name}")
        print(f"Days Present: {self.attendance_days}")
        print("----------------------------------\n")
s = student_kit("Rudrarajsinh Jadeja")
s.attendance(10)
s.generate_certificate()
```

```
Marked 10 days of attendance for Rudrarajsinh Jadeja

--- Certificate of Attendance ---
Principal: Mr ABC
Student Name: Rudrarajsinh Jadeja
Days Present: 10
----------------------------------
```

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

In [22]:
```
class Time:
    def __init__(self,hour,minute):
        self.hour=hour
        self.minute=minute
    def add(self,t):
        total_minute = self.minute+t.minute
        total_hour = self.hour+t.hour+(total_minute//60)
        total_minute%=60
        return Time(total_hour,total_minute)
    def display_time(self):
        print(f"{self.hour}:{self.minute}")
h1 = int(input("Enter hour1 : "))
m1 = int(input("Enter minute1 : "))
h2 = int(input("Enter hour1 : "))
```

```
m2 = int(input("Enter minute1 : "))
t1 = Time(h1,m1)
t2 = Time(h2,m2)
result = t1.add(t2)
result.display_time()
```

```
19:6
```

In [ ]:

Python Programming - 2301CS404

Lab - 13

Name: Jadeja Rudrarajsinh

Enrollment No: 23010101411

Roll No: 487

Continued..

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

In [2]:
```
class Rectangle:
    def __init__(self,l,b):
        self.l=l
        self.b=b
    def area(self,r):
        return r.l*r.b
l = int(input("Enter length of rectangle"))
b = int(input("Enter breadth of rectangle"))

rect = Rectangle(l,b)
print("Area =",rect.area(rect))
```

```
Area = 81
```

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().

In [6]:
```
class Square:
    def __init__(self,l):
        self.l=l
    def area(self):
        area_val = self.l*self.l
        self.output(area_val)
    def output(self,area_val):
        print(f"The area of the square =",area_val)

length = int(input("Enter the length of square : "))
square = Square(length)
square.area()
```

```
The area of the square = 64
```

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.

In [10]:
```
class Rectangle:
    def __init__(self,l,b):
        self.l=l
        self.b=b
    def area(self):
        area_val=self.l*self.b
        self.output(area_val)
    def output(self,area_val):
        print("Area =",area_val)
    @classmethod
    def compare_sides(cls,length,breadth):
        if length!=breadth:
            return True
        else:
            print("THIS IS SQUARE")
            return False
l = int(input("Enter length of rectangle"))
b = int(input("Enter breadth of rectangle"))
rect = Rectangle(l,b)
if rect.compare_sides(l,b):
    o.area()
```

```
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.

In [12]:
```
class Square:
    __side=0
    def get_side(self):
        return self.__side
    def set_side(self,side):
        self.__side=side
s = int(input("Enter side of square"))
square = Square()
square.set_side(s)
print("Side of square = ",square.get_side())
```

```
Side of square =  4
```

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().

In [14]:
```
class Profit:
    def getProfit(self, profit):
        self.profit = profit

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

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

    def getBalance(self):
        return self.balance

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

balance = int(input("Enter current balance: "))
o = BalanceSheet(balance)

choice = 0
while choice != -1:
    print("\nEnter 1 for profit")
    print("Enter 2 for loss")
    print("Enter 3 for print balance")
    print("Enter -1 to exit")
    choice = int(input("Enter your choice: "))
    match(choice):
        case 1:
            profit = float(input("Enter profit amount: "))
            o.getProfit(profit)
            o.balance += profit
        case 2:
```

```
            loss = float(input("Enter loss amount: "))
            o.getLoss(loss)
            o.balance -= loss
        case 3:
            o.printBalance()
        case -1:
            print("Exiting the program. Goodbye!")
        case _:
            print("Invalid choice. Please try again.")
```

```
Enter 1 for profit
Enter 2 for loss
Enter 3 for print balance
Enter -1 to exit
Enter 1 for profit
Enter 2 for loss
Enter 3 for print balance
Enter -1 to exit
Enter 1 for profit
Enter 2 for loss
Enter 3 for print balance
Enter -1 to exit
Exiting the program. Goodbye!
```

15) WAP to demonstrate all types of inheritance.

In [16]:
```
# Single Inheritance
class ParentSingle:
    def func1(self):
        print("This is a parent class for Single Inheritance.")

class ChildSingle(ParentSingle):
    def func2(self):
        print("This is a child class inheriting from ParentSingle.")

# Multiple Inheritance
class ClassA:
    def funcA(self):
        print("This is ClassA.")

class ClassB:
    def funcB(self):
        print("This is ClassB.")

class ClassC(ClassA, ClassB):
    def funcC(self):
        print("This is ClassC inheriting from ClassA and ClassB.")

# Multilevel Inheritance
class GrandParentMulti:
    def funcGP(self):
        print("This is the grandparent class for Multilevel Inheritance.")

class ParentMulti(GrandParentMulti):
    def funcP(self):
        print("This is the parent class inheriting from GrandParentMulti.")

class ChildMulti(ParentMulti):
    def funcC(self):
```

```
        print("This is the child class inheriting from ParentMulti.")

# Hierarchical Inheritance
class ParentHierarchical:
    def funcP(self):
        print("This is the parent class for Hierarchical Inheritance.")

class Child1(ParentHierarchical):
    def funcC1(self):
        print("This is Child1 inheriting from ParentHierarchical.")

class Child2(ParentHierarchical):
    def funcC2(self):
        print("This is Child2 inheriting from ParentHierarchical.")

# Hybrid Inheritance
class Base:
    def funcBase(self):
        print("This is the base class.")

class Derived1(Base):
    def funcD1(self):
        print("This is Derived1 inheriting from Base.")

class Derived2(Base):
    def funcD2(self):
        print("This is Derived2 inheriting from Base.")

class Hybrid(Derived1, Derived2):
    def funcHybrid(self):
        print("This is Hybrid inheriting from Derived1 and Derived2.")

# Demonstrate all types of inheritance
print("Single Inheritance:")
single_child = ChildSingle()
single_child.func1()
single_child.func2()

print("\nMultiple Inheritance:")
multi_obj = ClassC()
multi_obj.funcA()
multi_obj.funcB()
multi_obj.funcC()

print("\nMultilevel Inheritance:")
multi_lvl_child = ChildMulti()
multi_lvl_child.funcGP()
multi_lvl_child.funcP()
multi_lvl_child.funcC()

print("\nHierarchical Inheritance:")
hier_child1 = Child1()
hier_child2 = Child2()
hier_child1.funcP()
hier_child1.funcC1()
hier_child2.funcP()
hier_child2.funcC2()

print("\nHybrid Inheritance:")
hybrid_obj = Hybrid()
```

```
hybrid_obj.funcBase()
hybrid_obj.funcD1()
hybrid_obj.funcD2()
hybrid_obj.funcHybrid()
```

```
Single Inheritance:
This is a parent class for Single Inheritance.
This is a child class inheriting from ParentSingle.

Multiple Inheritance:
This is ClassA.
This is ClassB.
This is ClassC inheriting from ClassA and ClassB.

Multilevel Inheritance:
This is the grandparent class for Multilevel Inheritance.
This is the parent class inheriting from GrandParentMulti.
This is the child class inheriting from ParentMulti.

Hierarchical Inheritance:
This is the parent class for Hierarchical Inheritance.
This is Child1 inheriting from ParentHierarchical.
This is the parent class for Hierarchical Inheritance.
This is Child2 inheriting from ParentHierarchical.

Hybrid Inheritance:
This is the base class.
This is Derived1 inheriting from Base.
This is Derived2 inheriting from Base.
This is Hybrid inheriting from Derived1 and Derived2.
```

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.

In [18]:
```
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_details(self):
        print(f"Name:{self.name}")
        print(f"Age:{self.age}")
        print(f"Salary:{self.salary}")
name=input("Enter name")
age=int(input("Enter age"))
salary=int(input("Enter salary"))
```

```
e = Employee(name,age,salary)
e.display_details()
```

```
Name:Jadeja Rudrarajsinh
Age:20
Salary:9999
```

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.

In [20]:
```
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 O")
class Triangle(Shape):
    def draw(self):
        print("Drawing a Triangle Δ")
shapes = [Rectangle(),Circle(),Triangle()]
for shape in shapes:
    shape.draw()
```

```
Drawing a Rectangle []
Drawing a Circle O
Drawing a Triangle Δ
```