Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 1

name:thummar jainil

enrollment number:23010101272 - 385

01) WAP to print “Hello World”

In [1]:
```
print("Hello World")
```

```
Hello World
```

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

In [3]:
```
a=12
b=15
print("the ans is:",(a+b))
```

```
the ans is: 27
```

03) WAP to check the type of the variable.

In [9]:
```
a=10
print(type(a))
```

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

04) WAP to calculate simple interest.

In [23]:
```
principal=float(input("enter prinacipal:"))
rate=float(input("enter rate:"))
time=float(input("enter time:"))
interest=(principal*rate*time)/100
print("simple interest is:",interest)
```

```
enter prinacipal: 5
enter rate: 6
enter time: 7
simple interest is: 2.1
```

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

In [15]:
```
radius=float(input("enter radius"))
area=3.14*radius*radius
perimeter=2*3.14*radius
print(area)
print(perimeter)
```

```
enter radius 5
78.5
31.400000000000002
```

06) WAP to calculate area of a triangle.

In [ ]:
```
base=float(input("enter base:"))
height=float(input("enter height:"))
area=1/2*base*height
print(area)
```

```
enter base: 5
enter height: 2
5.0
```

07) WAP to compute quotient and remainder.

In [7]:
```
divident=float(input("please enter divident"))
divisor=float(input("please enter dicisor"))
quotient=divident/divisor
reminder=divident%divisor
print(f"the quotient is {quotient} and reminder is {reminder}")
```

```
please enter divident 4
please enter dicisor 2
the quotient is 2.0 and reminder is 0.0
```

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

In [17]:
```
degree=float(input("enter temprature in degree"))
fahrenheit=float(input("enter temprature in fahrenheit"))
dtof=(9/5*degree)+32
ftod=5/9*(fahrenheit-32)
print(f"the dtof is {dtof} and ftod is {ftod}")
```

```
enter temprature in degree 20
enter temprature in fahrenheit 68
the dtof is 68.0 and ftod is 20.0
```

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

In [31]:
```
import math
x1=float(input("enter x1"))
```

```
y1=float(input("enter y1"))
x2=float(input("enter x2"))
y2=float(input("enter y2"))
distance=math.sqrt(pow((x2-x1),2)+pow((y2-y1),2))
print(f"the distance is {distance}")
```

```
enter x1 3
enter y1 4
enter x2 7
enter y2 1
the distance is 5.0
```

10) WAP to print sum of n natural numbers.

In [12]:
```
num=int(input("enter number"))
sum=0
for i in range (1,num+1):
    sum=sum+i
print(sum)
```

```
15
```

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

In [14]:
```
import math
num=int(input("enter number"))
sum=0
for i in range (1,num+1):
    sum=sum+pow(i,2)
print(sum)
```

```
55
```

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

In [18]:
```
firstname=input("enter your first name")
lastname=input("enter your last name")
concate=firstname+" "+lastname
print(concate)
```

```
thummar jainil
```

13) WAP to swap two numbers.

In [25]:
```
a=int(input("enter number 1:"))
b=int(input("enter number 2:"))
temp=a
a=b
b=temp
print(a,b)
```

```
enter number 1: 5
enter number 2: 6
6 5
```

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

In [22]:
```
distance=float(input("enter distence in kilometer"))
ktom=1000*distance
ktof=3280.84*distance
ktoi=39370.08*distance
kmtocm=100000*distance
print(f"your distance in meter is {ktom}")
print(f"your distance in feet is {ktof}")
print(f"your distance in inches is {ktoi}")
print(f"your distance in cm is {kmtocm}")
```

```
your distance in meter is 5000.0
your distance in feet is 16404.2
your distance in inches is 196850.40000000002
your distance in cm is 500000.0
```

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

In [43]:
```
from word2number import w2n
months = {
    "january": 1, "february": 2, "march": 3, "april": 4,
    "may": 5, "june": 6, "july": 7, "august": 8,
    "september": 9, "october": 10, "november": 11, "december": 12
}

day = input("Enter the day (e.g., twenty-three): ").strip().lower()
month = input("Enter the month (e.g., November): ").strip().lower()
year = input("Enter the year (e.g., two thousand twenty-four): ").strip().lower(

day_number = w2n.word_to_num(day)
year_number = w2n.word_to_num(year)

if month in months:
        month_number = months[month]
else:
        print("Invalid month entered.")
        exit()

print(f"{day_number}-{month_number}-{year_number}")
```

```
2-7-2000
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 2

name:thummar jainil

enrollment number:23010101272 - 385

if..else..

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

In [3]:
```
number=int(input("enter a number"))
if number>=0:
    print("the number is positive")
else:
    print("the number is negitive")
```

```
the number is negitive
```

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

In [10]:
```
number=int(input("enter a number"))
if number%2==0:
    print("the number is even")
else:
    print("the number is odd")
```

```
the number is odd
```

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

In [14]:
```
number1=int(input("enter nubmer"))
number2=int(input("enter nubmer"))
print("number2 is large") if number1<number2 else print("number1 is large")
```

```
number2 is large
```

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

In [22]:
```
num1=int(input("enter 1 nubmer"))
num2=int(input("enter 2 nubmer"))
num3=int(input("enter 3 nubmer"))
if num1>num2:
    if num1>num3:
        print(f"{num1} is largest")
    else:
        print(f"{num3} is largest")
else:
    if num2>num3:
        print(f"{num2} is largest")
    else:
        print(f"{num3} is largest")
```

```
3 is largest
```

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 [30]:
```
year=int(input("etner year"))
if (year%4==0 and year%100!=0) or (year%400==0):
    print("leap year")
else:
    print("not")
```

```
leap year
```

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

In [44]:
```
day=int(input("etner day number"))
match day:
    case 1:
        print("sunday")
    case 2:
        print("monday")
    case 3:
        print("tuesday")
    case 4:
        print("wednesday")
    case 5:
        print("thursday")
    case 6:
        print("friday")
    case 7:
        print("saturday")
    case _:
        print("enter number between 1 to 7")
```

```
monday
```

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

In [46]:
```
day=int(input("etner number for operation,1:sum,2:sub,3:mul,4:div:-"))
num1=int(input("enter 1 nubmer"))
num2=int(input("enter 2 nubmer"))
match day:
    case 1:
        print(num1+num2)
    case 2:
        print(num1-num2)
    case 3:
        print(num1*num2)
    case 4:
        print(num1/num2)
    case _:
        print("enter number between 1 to 4")
```

```
2.0
```

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 [54]:
```
sub1=int(input("enter sub1's marks"))
sub2=int(input("enter sub2's marks"))
sub3=int(input("enter sub3's marks"))
sub4=int(input("enter sub4's marks"))
sub5=int(input("enter sub5's marks"))
avg=(sub1+sub2+sub3+sub4+sub5)/5
if avg>70:
    print("Distinction")
elif avg>60:
    print("first class")
if avg>45:
    print("second class")
else:
    print("fail")
print(f"your percentage is {avg}%")
```

```
second class
your percentage is 56.8%
```

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

In [76]:
```
size1=int(input("enter sub1's marks"))
size2=int(input("enter sub2's marks"))
size3=int(input("enter sub3's marks"))
if size1==size2==size3:
    print("equilateral")
elif size1==size2!=size3 or size1!=size2==size3:
```

```
    print("isosceles")
elif (size1*size1)==(size2*size2)+(size3*size3) or (size2*size2)==(size1*size1)+
    print("right-angled and scalene")
else:
    print("scalene")
```

```
right-angled and scalene
```

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

In [70]:
```
num1=int(input("enter 1 nubmer"))
num2=int(input("enter 2 nubmer"))
num3=int(input("enter 3 nubmer"))
if num1>num2:
    if num1>num3:
        print(f"{num3} is second largest")
    else:
        print(f"{num1} is second largest")
else:
    if num2>num3:
        print(f"{num3} is second largest")
    else:
        print(f"{num2} is second largest")
```

```
2 is second largest
```

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 [1]:
```
u = int(input("Enter Unit  : "))

if(u<=50) :
                r=u*2.60;
elif(u<=150) :
                r=50*260+(u-50)*3.25
elif(u<=250) :
                r=50*2.60+100*3.25+(u-150)*5.26
elif (u>250) :
                r=50*2.60+100*3.25+100*5.26+(u-250)*8.45

i=r*0.20
r=r+i

print(r)
```

```
3712.2
```

In [ ]:
```

```

Python Programming - 2301CS404

Lab - 3

name:thummar jainil

enrollment number:23010101272 - 385

for and while loop

01) WAP to print 1 to 10.

In [5]:
```
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 [11]:
```
n=int(input("enter a number"))
for i in range(1,n+1):
      print(i)
```

```
1
2
3
4
5
```

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

In [13]:
```
n=int(input("enter a number"))
for i in range(1,n+1):
    if(i%2==1):
      print(i)
```

```
1
3
5
7
```

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

In [38]:
```
a=int(input("enter a"))
n=int(input("enter a number"))
for i in range(a,n+1):
    if(i%2==0 and i%3!=0):
      print(i)
```

```
2
4
8
```

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

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

```
15
```

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

In [36]:
```
n=int(input("enter a number"))
sum=0
for i in range(1,n+1):
    sum=sum+(i**2)
print(sum)
```

```
55
```

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

In [44]:
```
n=int(input("enter a number"))
sum=0
for i in range(1,n+1):
    if(i%2==0):
        sum-=i
    else:
```

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

```
3
```

08) WAP to print multiplication table of given number.

In [48]:
```
n=int(input("enter the number"))
#print in reverce order
for i in range(1,11):
    print(f"{n} X {i} = {n*i}")
```

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

09) WAP to find factorial of the given number.

In [51]:
```
n=int(input("enter a number"))
sum=1
for i in range(1,n+1):
    sum=sum*i
print(sum)
```

```
120
```

10) WAP to find factors of the given number.

In [55]:
```
n=int(input("enter a number"))
for i in range(1,n+1):
    if(n%i==0):
        print(i)
```

```
1
2
5
10
```

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

In [61]:
```
n=int(input("enter a number"))
count=0
for i in range(1,n+1):
    if(n%i==0):
        count+=1
if(count==2):
    print("the number is prime")
else:
    print("the number is not prme")
```

```
the number is prime
```

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

In [104…
```
a=int(input("enter a number a"))
b=int(input("enter a number b"))
while(a>0 and b>0):
    if(a>b):
        a=a%b
    else:
        b=b%a
if(a==0):
    print(b)
else:
    print(a)
```

```
5
```

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

In [98]:
```
n=int(input("enter a number"))
sum=n
rev=0
while(n>0):
    rev=(rev*10+int(n%10))
    n=int(n/10)
if(sum==rev):
    print("palindrome")
else:
    print("not")
```

```
palindrome
```

14) WAP to print GCD of given two numbers.

In [102…
```
a=int(input("enter a number a"))
b=int(input("enter a number b"))
while(a>0 and b>0):
    if(a>b):
        a=a%b
    else:
        b=b%a
if(a==0):
    print(b)
else:
    print(a)
```

```
5
```

In [ ]:
```
password=input("enter password")
```

Python Programming - 2301CS404

Lab - 4

name:thummar jainil

enrollment number:23010101272 - 385

String

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

In [8]:
```
s=input("enter a string")
a=s[::-1]
if(a==s):
    print("palindrome")
else:
    print("not")
```

```
palindrome
```

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

In [10]:
```
s=input("enter a string")
a=s[::-1]
print(a)
```

```
sdsa
```

03) WAP to remove ith character from given string.

In [20]:
```
s='hello'
n=int(input("enter place"))
for i in range(len(s)+1):
    if(i==n):
```

```
        s=s[:i]+s[i+1:]
print(s)
```

```
hllo
```

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

In [22]:
```
s='hello'
n=0
for char in s:
    n+=1
print(n)
```

```
5
```

05) WAP to print even length word in string.

In [121…
```
s='hello'
n=1
for char in s:
    if(n%2==0):
        print(char)
    n+=1
```

```
e
l
```

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

In [ ]:
```
s='hello'
s.lower()
c=0
for char in string:
    if(char=='a' or char=='e' or char=='i' or char=='o' or char=='u'):
        c+=1
print(c)
```

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

In [35]:
```
s='hello world'
a=s.split()
b=[]
for i in a:
    b.append(i.capitalize())
a=b
a1=""
for i in a:
    i=i[::-1]
    i=i.replace(i[0],i[0].upper())
    a1+=i[::-1]+" "
print(a1)
```

```
HellO WorlD
```

08) WAP to convert given array to string.

In [45]:
```
a=['sdas','sads','wqee','ghr']
s=(" ".join(a))
print(s)
```

```
sdas sads wqee ghr
```

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

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

In [47]:
```
p1=input("enter password")
p2=input("enter confirm password")
if(p1!=p2):
    print("please enter same password")
else:
    print("your password is same")
```

```
your password is same
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

In [1]:
```
card_number = input("Enter your credit card number: ")
card_number = card_number.replace(" ", "")
masked_card_number = "**** **** **** " + card_number[-4:]
print("Display as:", masked_card_number)
```

```
Display as: **** **** **** 5
```

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

s1 = decimal and s2 = medical are Anagram

In [3]:
```
s1=input("enter string 1")
s2=input("enter string 2")
a=[]
b=[]
flag=True
if(len(s1)!=len(s2)):
    print('not')
else:
    for i in range(len(s1)):
        a.append(s1[i])
        b.append(s2[i])
    a.sort()
    b.sort()
for i in range(len(s1)):
    if(a[i]==b[i]):
        flag=True
    else:
        print("not")
```

```
        break
if(flag):
    print("yes")
```

```
yes
```

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

input : EHIsarwiwhtwMV

output : IsarwiwhtwEHMV

In [3]:
```
s = input("Enter the input")
a , b = "",""
for char in s:
    if(char.islower()):
        a += char
    else:
        b += char
print(a+b)
```

```
dsfsdfEHIMV
```

In [ ]:
```

```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 5

name:thummar jainil

enrollment number:23010101272 - 385

List

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

In [2]:
```
l1=[1,2,3,5,6]
sum=0
for i in l1:
    sum+=sum+i
print(sum)
```

```
60
```

02) WAP to find largest element in a List.

In [4]:
```
l1=[45,34,7,2,4,356,42]
max=0
for i in l1:
    if(max<i):
        max=i
print(max)
```

```
356
```

03) WAP to find the length of a List.

In [8]:
```
l1=[45,34,7,2,4,356,42]
count=0
for i in l1:
    count+=1
```

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

```
7
7
```

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

In [10]:
```
l1=[45,34,7,2,4,356,42]
temp=l1[-1]
l1[-1]=l1[0]
l1[0]=temp
print(l1)
```

```
[42, 34, 7, 2, 4, 356, 45]
```

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

In [28]:
```
l1=[45,34,7,2,4,356,42]
l2=l1[:len(l1)//2]
l3=l1[len(l1)//2:]
print(l2)
print(l3)
l1.clear()
for i in l3:
    l1.append(i)
for i in l2:
    l1.append(i)
print(l1)
```

```
[45, 34, 7]
[2, 4, 356, 42]
[2, 4, 356, 42, 45, 34, 7]
```

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

In [30]:
```
l1=[45,34,7,2,4,356,42]
i1=int(input("enter first index"))
i2=int(input("enter second index"))
if(i1>len(l1) or i2>len(l1)):
    print("enter proper index")
else:
    temp=l1[i1]
    l1[i1]=l1[i2]
    l1[i2]=temp
print(l1)
```

```
[45, 34, 356, 2, 4, 7, 42]
```

07) WAP to reverse the list entered by user.

In [42]:
```
l1=[]
size=int(input("enter the size of list"))
for i in range(size):
    a=int(input())
```

```
    l1.append(a)
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 [44]:
```
l1=[45,34,7,2,4,356,42]
l2=[]
for i in l1:
    if(i%2==0):
        l2.append(i)
print(l2)
```

```
[34, 2, 4, 356, 42]
```

09) WAP to count unique items in a list.

In [52]:
```
l1=[45,34,7,34,1,45,10,2,4,356,42]
l1=list(set(l1))
print(l1)
```

```
[1, 34, 2, 4, 356, 7, 10, 42, 45]
```

10) WAP to copy a list.

In [56]:
```
l1=[45,34,7,2,4,356,42]
l2=l1.copy()
print(l2)
```

```
[45, 34, 7, 2, 4, 356, 42]
```

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

In [60]:
```
n=int(input("enter a range"))
for i in range(n):
    if not i%2==0:
        print(i)
```

```
1
3
5
7
9
```

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

In [66]:
```
l1=[45,34,7,34,1,45,10,2,4,356,42]
print(l1.count(34))
```

```
2
```

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

In [1]:
```
l = [1,2,3,4,5,6]
l.sort()
print(l[-2])
```

```
5
```

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

In [2]:
```
from collections import Counter
l1 = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5]
k = int(input("Enter the frequency: "))
freq_counter = Counter(l1) #create a dictionary
l = [i for i, count in freq_counter.items() if count >= k]
print(l)
```

```
[2, 3, 4]
```

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

In [3]:
```
a = []
for i in range(0,10):
    a.append(i*i)
print(a)
```

```
[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 [6]:
```
a = []
ans = []
while True:
    s = (input("Enter the number and if you enter 0 then out from the list"))
    if s != '0':
        a.append(s)
    elif(s == '0'):
        break
for i in a:
    if i[0] == 'b' or i[0] == 'B':
        ans.append(i)
print(ans)
```

```
['badf', 'banana']
```

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

In [7]:
```
list1 = [1, 2, 3, 4]
list2 = [3, 4, 5, 6]

intersection = set(list1).intersection(list2)
print(intersection)
```

```
{3, 4}
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 6

name:thummar jainil

enrollment number:23010101272 - 385

Tuple

01) WAP to find sum of tuple elements.

In [2]:
```
t1=(2,4,23,54,6,23,64)
sum=0
for i in t1:
    sum+=i
print(sum)
```

```
176
```

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

In [60]:
```
t1=(2,4,23,54,6,23,64,2,4)
t1=sorted(tuple(set(t1)))
n=int(input("enter a number"))
print(t1[:k])
print(t1[len(t1)-k:])
```

```
[2, 4, 6]
[23, 54, 64]
```

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

In [30]:
```
l1=[(2,4,6),(3,6,9),(4,8,12),(5,10,15),(6,12,18)]
n=int(input("enter a number"))
for i in l1:
```

```
    count=0
    for j in i:
        if (j%n==0):
           continue
        else:
            count+=1
    if(count==0):
        print(i)
```

```
(3, 6, 9)
(6, 12, 18)
```

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

In [ ]:
```
l1=[(2,4,6),(3,6,9),(4,8,12),(5,10,15),(6,12,18)]
```

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

In [64]:
```
l1=[(2,4,6),(3,6,9),(4,8,-12),(-5,10,15),(6,12,18)]
for i in l1:
    count=0
    for j in i:
        if (j>0):
           continue
        else:
            count+=1
    if(count==0):
        print(i)
```

```
(2, 4, 6)
(3, 6, 9)
(6, 12, 18)
```

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

In [1]:
```
t1 = []
a = [1,2,3]
b = [1,2,3]
c = [1,2,3]
t1.append(a)
t1.append(b)
t1.append(c)
print(tuple(t1))
########################################################## vice - versa ############
t2 = []
a = [1,2,3]
b = [1,2,3]
c = [1,2,3]
t2.append(tuple(a))
t2.append(tuple(b))
t2.append(tuple(c))
print(t2)
```

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

07) WAP to remove tuples of length K.

In [114…
```
l1=[(2,4,6,3),(23,54,3,6,9),(8,12),(15,),(6,12,3)]
n=int(input("enter length k"))
count=0
for i in l1:
    if(len(i)==n):
        l1.remove(i)
    else:
        continue
print(l1)
```

```
[(2, 4, 6, 3), (23, 54, 3, 6, 9), (8, 12), (15,)]
```

08) WAP to remove duplicates from tuple.

In [116…
```
l1=(1,3,5,78,4,3,12,1,1,3,4,9)
l1=tuple(set(l1))
print(l1)
```

```
(1, 3, 4, 5, 9, 12, 78)
```

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

In [2]:
```
t1 = (1,2,3,4)
t1 = list(t1)
for i in range(len(t1)-1):
    t1[i] = t1[i] * t1[i+1]
t1 = tuple(t1)
t1
```

Out[2]: (2, 6, 12, 4)

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

In [3]:
```
t1 = (1,2,3,4,5)
flag = True
temp = t1[0]
for i in range(i,len(t1)):
    for j in range(i+1,len(t1)):
        if t1[i] == t1[j]:
            flag = False
            break
if flag:
    print("distinct")
else:
    print("not distinct")
```

```
distinct
```

In [ ]:

Python Programming - 2301CS404

Lab - 7

name:thummar jainil

enrollment number:23010101272 - 385

Set & Dictionary

01) WAP to iterate over a set.

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

```
1
2
3
4
5
```

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

In [50]:
```
s={246,3,2,76,9,64,5}
l=list(s)
print(type(l))
string=str(s)
print(type(string))
t=tuple(s)
print(type(t))
```

```
<class 'list'>
<class 'str'>
<class 'tuple'>
```

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

In [10]:
```
s={246,3,2,76,9,64,5}
print(max(s))
print(min(s))
```

```
246
2
```

04) WAP to perform union of two sets.

In [56]:
```
s1={246,3,2,76,9,64,5}
s2={3,2,34,56,64,32}
m=s1|s2
print(m)
```

```
{64, 32, 2, 3, 34, 5, 9, 76, 246, 56}
```

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

In [58]:
```
list1 = [1, 2, 3, 4, 5]
list2 = [5, 6, 7, 8, 9]

if(set(list1) & set(list2)):
    print('available')
else:
    print('not')
```

```
available
```

06) WAP to remove duplicates from list.

In [63]:
```
list1 = [1, 2, 3, 4, 5, 6, 5, 3, 4]
list1=list(set(list1))
print(list1)
```

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

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

In [73]:
```
string='hello my name is jainil hello nice my self is jainil what is your name'
l=string.split()
l=list(set(l))
print(l)
```

```
['self', 'your', 'what', 'my', 'nice', 'hello', 'name', 'is', 'jainil']
```

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

In [75]:
```
set_a = {1, 2, 3, 4, 5}
set_b = {4, 5, 6, 7, 8}
set_a -= set_a & set_b
print(set_a)
```

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

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

In [93]:
```
s1='decimal'
s2='medical'
se1=set(s1)
se2=set(s2)
d1=dict()
d2=dict()
for i in se1:
    d1[i] = s1.count(i)
for i in se2:
    d2[i]=s2.count(i)
print(d1)
print(d2)
if(d1==d2):
    print('anagram')
else:
    print('not')
```

```
{'i': 1, 'e': 1, 'm': 1, 'c': 1, 'l': 1, 'a': 1, 'd': 1}
{'i': 1, 'e': 1, 'm': 1, 'c': 1, 'l': 1, 'a': 1, 'd': 1}
anagram
```

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

In [119…
```
l1 = [1,24,45,6,7,68,9]
l2 = [2,4,5,24,6,7,10]
l3 = [5,7,8,24,6,7,45]

l1 = set(l1)
l2 = set(l2)
l3 = set(l3)



if(l1 & l2 & l3):
    print("available common element : ",(l1 & l2 & l3))
```

```
available common element :  {24, 6, 7}
```

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

In [95]:
```
s = "ayush vasoya"
s1 = set(s)
print(s1)

count = 0
for i in s1:
    if i=='a' or i=='e' or i=='i' or i=='u' or i=='o':
        count+=1
print(count)
```

```
{' ', 'v', 's', 'u', 'y', 'o', 'h', 'a'}
3
```

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

In [113…
```
s = input("Enter a string: ")
for char in s:
        if char not in '01':
            ans = False
        else:
            ans = True
if (ans):
    print("The string is a binary string.")
else:
    print("The string is not a binary string.")
```

```
The string is a binary string.
```

13) WAP to sort dictionary by key or value.

In [107…
```
dict1 = { 5 : 'o' ,  3 : 'l' ,2 : 'e', 4 : 'l', 1 : 'h'  }

s1 = dict(sorted(dict1.items()))
print(f"sorted by keys : {s1}")

s2 =  dict(sorted(dict1.items(),key = lambda item : item[1]))
print(f"sorted by values : {s2}")
```

```
sorted by keys : {1: 'h', 2: 'e', 3: 'l', 4: 'l', 5: 'o'}
sorted by values : {2: 'e', 1: 'h', 3: 'l', 4: 'l', 5: 'o'}
```

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

In [111…
```
d={'a':1,'b':2,'c':3}
sum=0
for i in d.values():
    sum+=i
print(sum)
```

```
6
```

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 [121…
```
dict1={'a':5,'c':8,'e':2}
k=input("Enter Key : ")
if(k not in dict1.keys()):
    print("Key not found")
else:
    print(dict1[k])
```

```
5
```

Python Programming - 2301CS404

Lab - 8

name:thummar jainil

enrollment number:23010101272 - 385

User Defined Function

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

In [3]:
```
def bgmi(m,h):
    return (m/h**2)
print(bgmi(50,6))
```

```
1.3888888888888888
```

02) Write a function that add first n numbers.

In [4]:
```
def add(n):
    sum=0
    for i in range(n+1):
        sum+=i
    return sum
print(add(3))
```

```
6
```

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

In [8]:
```
def prime(n):
    count=0
    for i in range(1,n):
        if(n%i==0):
            count+=1
```

```
    if(count==1):return 1
    else:return 0
print(prime(5))
```

```
1
```

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

In [11]:
```
def listPrime(n1,n2):
    l1=[]
    for i in range(n1,n2+1):
        count=0
        for j in range(1,i):
            if(i%j==0):
                count+=1
        if(count==1):
            l1.append(i)
    return l1
print(listPrime(1,100))
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

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

In [15]:
```
def palindrome(s1):
    i=0
    j=len(s1)-1
    for i in range(int(len(s1)/2)):
        if(s1[i]==s1[j]):
            i+=1
            j-=1
        else:
            return False
    return True
print(palindrome('madam'))
```

```
True
```

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

In [18]:
```
def sumList(l1):
    return sum(l1)
print(sumList([1,2,3,4,5]))
sumList([1,2,3,4,5])
```

```
15
```

Out[18]: 15

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

In [26]:
```
l1 = [(6, 2, 3),(4, 7),(5,2 ,5 ,3)]
def sum1(l):
    ans  = 0
    for i in l:
        ans = ans + i[0]
    return ans
answer  =  sum1(l1)
answer
```

Out[26]: 15

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

In [28]:
```
def fib(n):
    if (n==0 or n==1): return n
    else: return (fib(n-1)+fib(n-2))
fib(5)
```

Out[28]: 5

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 [30]:
```
dict1 = {101:'Ajay',102:'Rahul',103:'Pooja'}
rollNo = int(input("Enter Roll No : "))
dict1.get(rollNo)
```

Out[30]: 'Pooja'

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 [31]:
```
scores = [200, 456, 300, 100, 234, 678]
def sumofscore(scores):
    ans = 0
    for i in scores:
        if i % 10 == 0:
            ans = ans +i
    return ans
answer = sumofscore(scores)
answer
```

Out[31]: 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 [35]:
```
dic = {'a': 10, 'b':20, 'c':30, 'd':40}
def swap(d):
    return {v : k for k ,v in d.items()}
ans = swap(dic)
ans
```

Out[35]:
```
{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 [34]:
```
string ="the quick brown fox jumps over the lazy dog"
set1 = set(string.lower())
set1.remove(' ')
def panagram(set1):
    if(len(set1)==26):
        print("Pangram string")
    else:
        print("not Pangram string")
panagram(set1)
```

```
Pangram string
```

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 [39]:
```
def upperlower(s1):
    no_upper =0
    no_lower =0
    for i in s1:
        if i.isupper():
            no_upper = no_upper+1

        elif i.islower():
            no_lower = no_lower +1
    return list([no_upper,no_lower])
upperlower("AbcDEfgh")
```

Out[39]:
```
[3, 5]
```

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

In [40]:
```
get_min = lambda a, b : min(a,b)

print(get_min(5, 8))
```

```
5
```

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

In [2]:
```
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 [3]:
```
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 [4]:
```
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
```

In [ ]:

Python Programming - 2301CS404

Lab - 9

name:thummar jainil

enrollment number:23010101272 - 385

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 [1]:
```
fp=open("file.txt","r")
print(fp.read())
fp.close()
```

```
hello
my
name is
how are you
```

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

In [1]:
```
fp=open("new.txt","x")
fp.write("hello")
fp.close()
```

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

In [5]:
```
fp=open("file1.txt","r")
for i in range(5):
    print(fp.readline())
fp.close()
```

```
hello

how

are

you

my
```

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

In [6]:
```
fp = open("file1.txt","r")
wl = fp.read().split()
w_len = list(map(len,wl))
maxi = max(w_len)
ans = [i for i in wl if len(i) == maxi]
print(ans)
```

```
['demonstrate']
```

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

In [7]:
```
fp=open("file1.txt","r")
l1 =fp.readlines()
print("lines: ",len(l1))
fp.seek(0,0)
l2=fp.read().split()
print("words: ",len(l2))
fp.seek(0,0)
s= fp.read()
print("char :",len(s))

fp.close()
```

```
lines:  7
words:  10
char : 51
```

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

In [2]:
```
with open('file1.txt', 'r') as src:
    content = src.read()

with open('file2.txt', 'w') as dest:
    dest.write(content)

print("File copied successfully.")
```

```
File copied successfully.
```

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

In [3]:
```
with open('file.txt', 'rb') as file:
    contents = file.read()
    filesize = len(contents)

print(f"Size of the file: {filesize}")
```

```
Size of the file: 28
```

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

In [16]:
```
word = input("Enter words:")
fp = open("file1.txt","r")
def frequency(f,w):
    c = 0
    for i in f:
        for j in i.split():
            if(j == w):
                c += 1
    return c
print(frequency(fp,word))
fp.close()
```

```
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 [ ]:
```
fp = open("score.txt", "w")
for i in range(1, 6):
    scores = input(f"Enter the score for subject {i}: ")
    fp.write(scores)
fp.close()

fp = open("score.txt", "r")
fp.readlines()
scores = [int(score.strip()) for score in scores]
highest_score = max(scores)
print(highest_score)
fp.close()
```

```
5
```

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

(Note: each number should be in new line)

In [15]:
```
l1=[]
count = 0
while count < 100:
    num = 0
```

```
    for j in range(1,count):
        if(count%j==0):
            num+=1
    if(num==1):
        l1.append(count)
    count+=1
l2=[str(i)+"\n" for i in l1]
with open("primenumbers.txt", "w") as file:
    file.writelines(l2)
print(l1)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
79, 83, 89, 97]
```

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

In [ ]:
```
def merge_files(new, file2, output_file):
    """Merge contents of file1 and file2 into output_file."""
    with open(output_file, "w") as outfile:
        for file in [new, file2]:
            with open(new, "r") as infile:
                outfile.write(infile.read() + "\n")

# File names
new = "new.txt"
file2 = "file2.txt"
output_file = "merged.txt"

# Merge the files
merge_files(new, file2, output_file)

print(f"Files '{new}' and '{file2}' have been merged into '{output_file}'.")
```

```
Files 'new.txt' and 'file2.txt' have been merged into 'merged.txt'.
```

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

In [ ]:
```
def replace_word(input_file, output_file, word1, word2):
    """Replace all occurrences of word1 with word2 in input_file and save to out
    with open(input_file, "r", encoding="utf-8") as infile:
        data = infile.read()

    updated_data = data.replace(word1, word2)

    with open(output_file, "w", encoding="utf-8") as outfile:
        outfile.write(updated_data)

    print(f"Replaced '{word1}' with '{word2}' and saved to '{output_file}'.")

input_file = "new.txt"
output_file = "updated.txt"
word1 = "oldword"
word2 = "newword"

replace_word(input_file, output_file, word1, word2)
```

Replaced 'oldword' with 'newword' and saved to 'updated.txt'.

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

In [18]:
```
fp = open("file.txt","rb")
fp.read(5)
print(fp.tell())
fp.seek(0,2)
fp.seek(-4,1)
print(fp.tell())
fp.close()
```

```
5
24
```

Python Programming - 2301CS404

Lab - 10

name:thummar jainil

enrollment number:23010101272 - 385

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 [8]:
```
try:
    a=int(input('eter a number'))
    b=int(input('enter a second number'))
    ans=a/b
    print(ans)
except (ZeroDivisionError,ValueError,TypeError) as err:
    print(type(err))
```

```
<class 'ValueError'>
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [1]:
```
li = [1,2,3,4,5,6]
try:
    print(li[10])
```

```
except IndexError as msg:
    print(msg)
```

```
list index out of range
```

In [2]:
```
li = {1 :'thummar',2 :'jainil'}
try:
    print(li[3])
except KeyError :
    print("key not found")
```

```
key not found
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [9]:
```
try:
    fp=open("abc.txt","r")
    fp.read()
    fp.close()
except FileNotFoundError as err:
    print(type(err))
```

```
<class 'FileNotFoundError'>
```

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

In [4]:
```
try:
    ans = 10 + 'a'
    print(ans)
except (ZeroDivisionError,ValueError,TypeError) as err:
    print(type(err))
```

```
<class 'TypeError'>
```

05) WAP to demonstrate else and finally block.

In [14]:
```
try:
    fp = open("file1.txt","r")
except FileNotFoundError as err:
    print(err)
else:
    print(fp.read())
    fp.close()
finally:
    print("This block will always be executed.")
```

```
12345
This block will always be 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.

In [11]:
```
li = list(input("Enter Grades coma saperated : ").split(","))
try:
    li2 = list(int(i for i in li))
except TypeError as err :
    print(err)
else:
    print(li)
```

```
int() argument must be a string, a bytes-like object or a real number, not 'gener
ator'
```

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

In [12]:
```
class ZeroDivisionError(Exception):
    def _init_(self,msg):
        self.msg = msg

try:
    a = int(input("enter first number : "))
    b = int(input("Enter second number : "))
    if b !=  0:
        print(a // b)
    else:
        raise ZeroDivisionError("can not divide by zero.")

except ZeroDivisionError as err:
    print(err)
```

```
can not divide 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.

In [13]:
```
class ValueError(Exception):
    def __init__(self,msg):
        self.msg = msg
try:
    a = int(input("enter first number : "))

    if a > 18:
        print(a)
    else:
        raise ValueError("Enter Valid Age")
except ValueError as err:
    print(err)
```

```
23
```

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 [16]:
```
class InvalidUsernameError(Exception):
    def _init_(self,msg):
        self.msg = msg

try:
    a = (input("enter a name : "))
    ans=len(a)
    if ans>=5 and ans<=15 :
        print(a)
    else:
        raise InvalidUsernameError("Username must be between 5 and 15 characters

except InvalidUsernameError as err:
    print(err)
```

```
jainil
```

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 [23]:
```
import math
class NegativeNumberError(Exception):
    def _init_(self,msg):
        self.msg = msg
try:
    a = int(input("enter first number : "))
    if  a>=0:
        print(math.sqrt(a))
    else:
        raise NegativeNumberError("Cannot calculate the square root of a negativ
except NegativeNumberError as err:
    print(err)
```

```
2.23606797749979
```

In [ ]:
```

```

Python Programming - 2301CS404

Lab - 11

name:thummar jainil

enrollment number:23010101272 - 385

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 [2]:
```
import calculator as cs
print(cs.add(3,5))
print(cs.sub(3,5))
print(cs.mul(3,5))
print(cs.div(3,5))
```

```
8
-2
15
0.6
```

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

In [4]:
```
import random
s="thummar jaini"
print(random.choice(s))
```

```
a
```

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

In [6]:
```
import random
s=[1,2,3,4,5,6,7,8]
print(random.choice(s))
```

```
8
```

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

In [126…
```
import random
random.seed(1)
print(random.randint(1,6))
```

```
2
```

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

In [151…
```
import random
i=0
while(i<3):
    num=random.randrange(100,999)
    if(num%5==0):
        print(num)
        i+=1
```

```
930
790
660
```

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 [175…
```
import random
lottery_tickets = random.sample(range(100000, 999999),100)
winner, runner_up = random.sample(lottery_tickets, 2)
print("Lottery Tickets Generated:", lottery_tickets)
print("Winner Ticket:", winner)
print("Runner-up Ticket:", runner_up)
```

```
Lottery Tickets Generated: [180852, 634008, 797562, 281657, 288289, 913914, 25682
8, 248413, 961457, 435317, 420468, 212069, 843780, 639343, 975235, 731130, 40774
6, 232434, 316783, 248562, 671990, 857730, 133302, 917620, 431422, 960912, 75381
6, 942904, 804851, 679811, 981557, 882431, 823092, 315413, 286808, 413447, 55365
3, 663601, 265566, 150917, 849547, 800216, 359309, 364856, 915557, 167543, 81520
8, 568393, 947514, 551067, 675951, 362374, 667675, 560743, 992645, 664196, 57532
9, 111394, 514932, 976914, 455120, 279849, 370500, 609380, 125594, 931591, 77784
0, 536924, 698321, 119829, 165348, 825303, 472185, 708248, 245001, 722378, 23120
7, 245223, 371699, 969200, 390365, 517120, 691472, 520565, 280537, 742195, 19358
2, 344873, 609604, 107840, 286204, 654383, 432651, 625231, 780357, 559608, 81986
1, 770156, 866951, 336695]
Winner Ticket: 981557
Runner-up Ticket: 849547
```

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

In [1]:
```
from datetime import datetime
y=datetime.now()
print(y)
```

```
2025-02-26 18:32:04.787495
```

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

In [177…
```
from datetime import timedelta,datetime
y=datetime.now()
y=y+timedelta(days=-7)
print(y)
```

```
2025-02-06 13:30:13.268311
```

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

In [178…
```
from datetime import datetime
d1=datetime(2005,11,14)
d2=datetime(2006,11,14)
y=(d2-d1).days
print(y)
```

```
365
```

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

In [183…
```
import datetime

day1 = datetime.datetime(2025, 2, 13, 12,40,10)
print(day1.strftime("%A"))
```

```
Thursday
```

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

In [181…
```
import datetime as d
#datetime object
day1 = d.datetime(2011,11,9,10,30,30)
print(day1.year)
print(day1.strftime("%A"))
print(day1.strftime("%a"))
print(day1.strftime("%H"))
print(day1.strftime("%B"))
print(day1.strftime("%p"))

print(day1 - d.timedelta(weeks=1))
```

```
2011
Wednesday
Wed
10
November
AM
2011-11-02 10:30:30
```

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

In [174…
```
import math as m

print("PI : ",m.pi)
print("Factorial of 5 :",m.factorial(5))
print("sqrt : ",m.sqrt(9))
print("square : ",m.pow(3,2))
```

```
PI :  3.141592653589793
Factorial of 5 : 120
sqrt :  3.0
square :  9.0
```

In [ ]:
```

```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Python Programming - 2301CS404

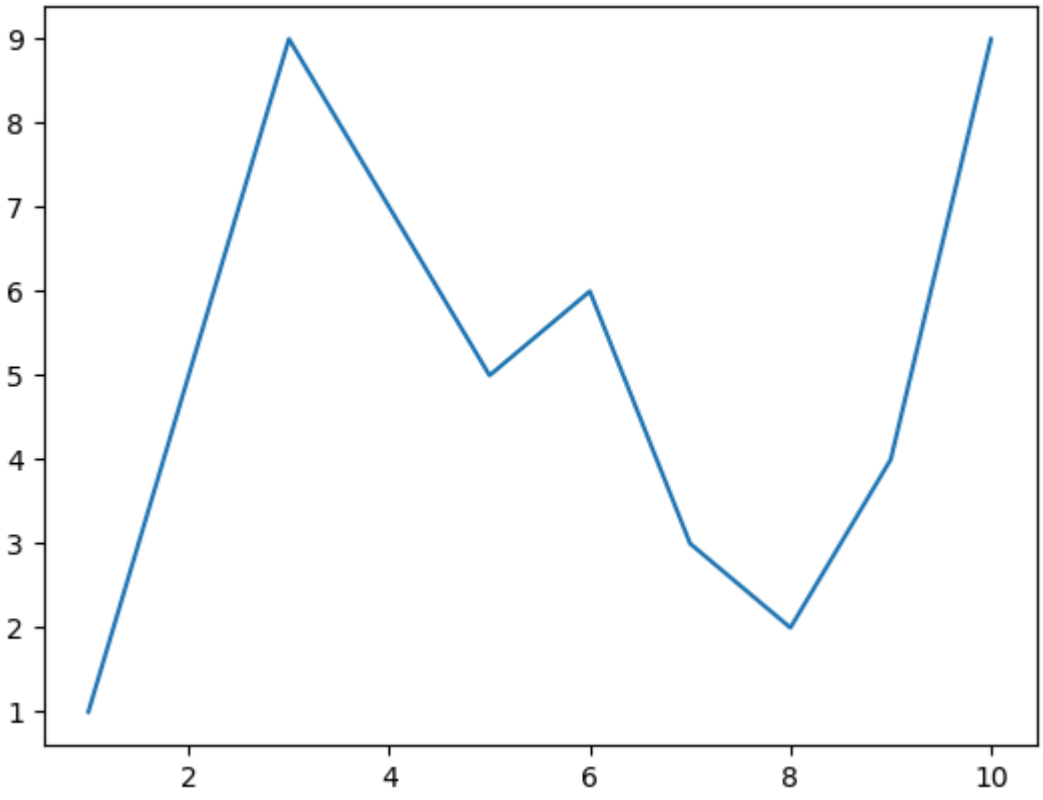
Lab - 12

name:thummar jainil

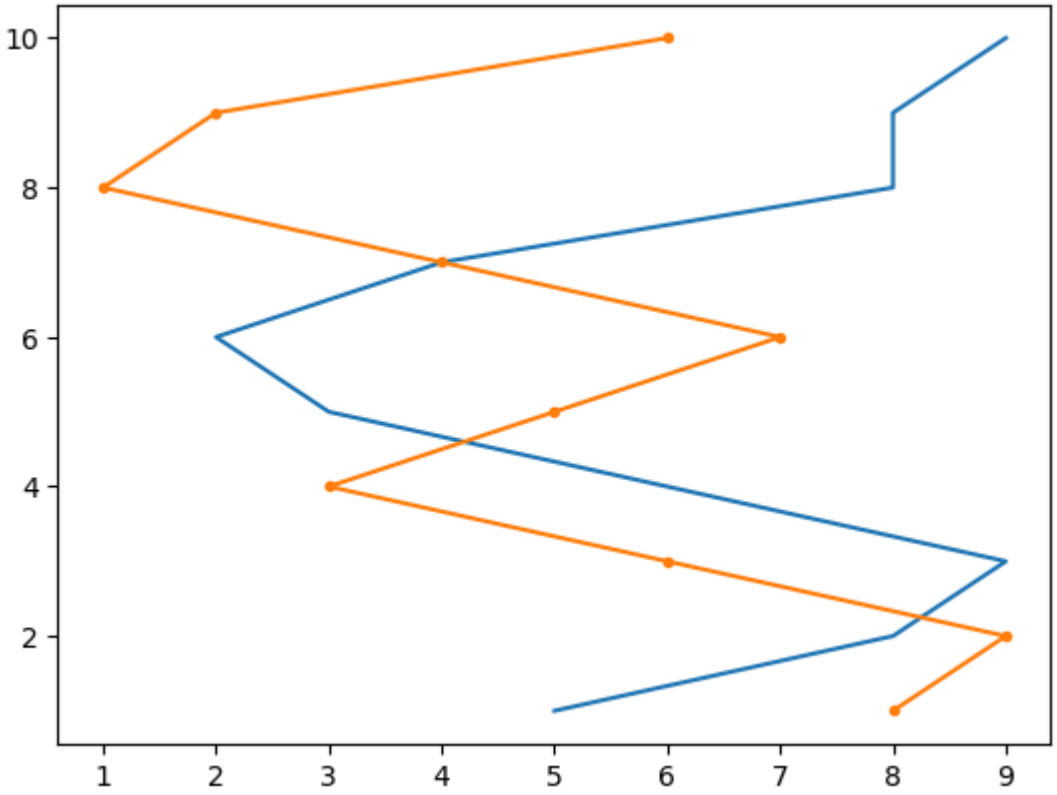
enrollment number:23010101272 - 385

In [7]:
```
import matplotlib.pyplot as plt
```

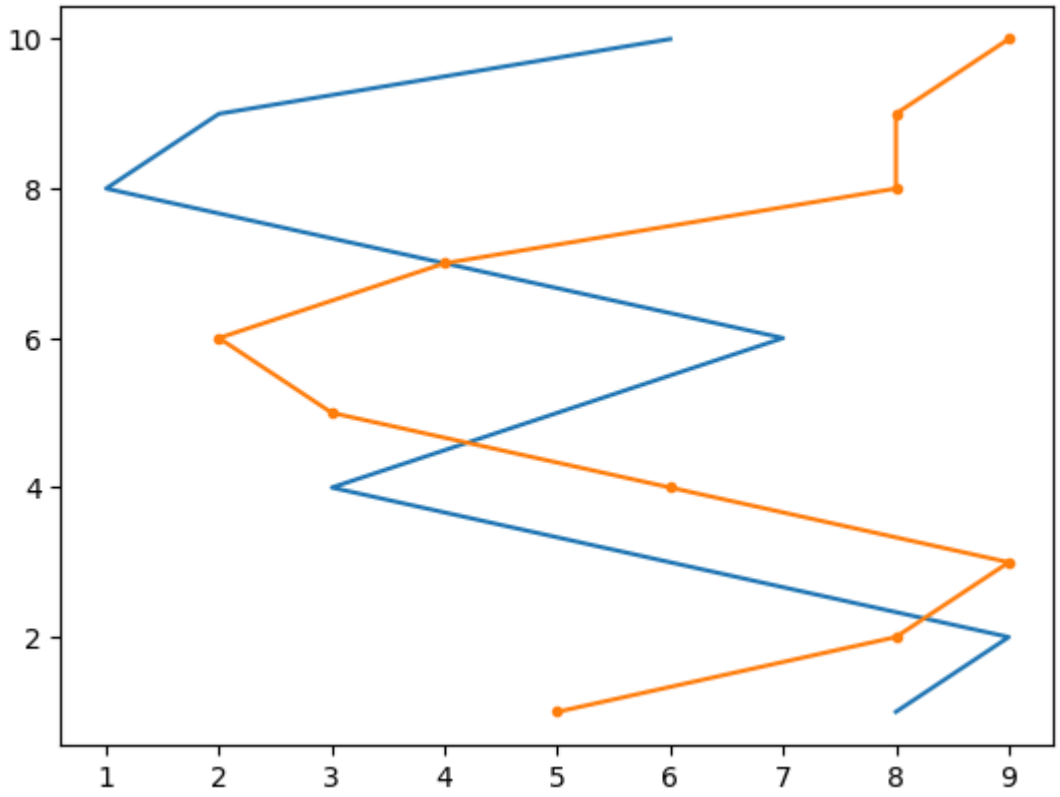
In [8]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]
plt.plot(x,y)
plt.show(x,y)
# write a code to display the line chart of above x & y
```



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

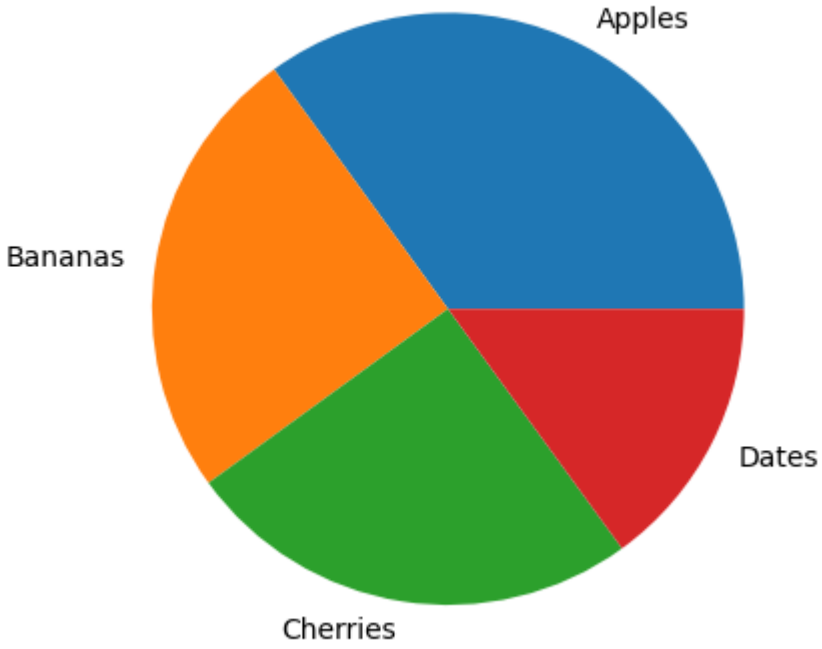


In [22]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]

# write a code to generate below graph
```



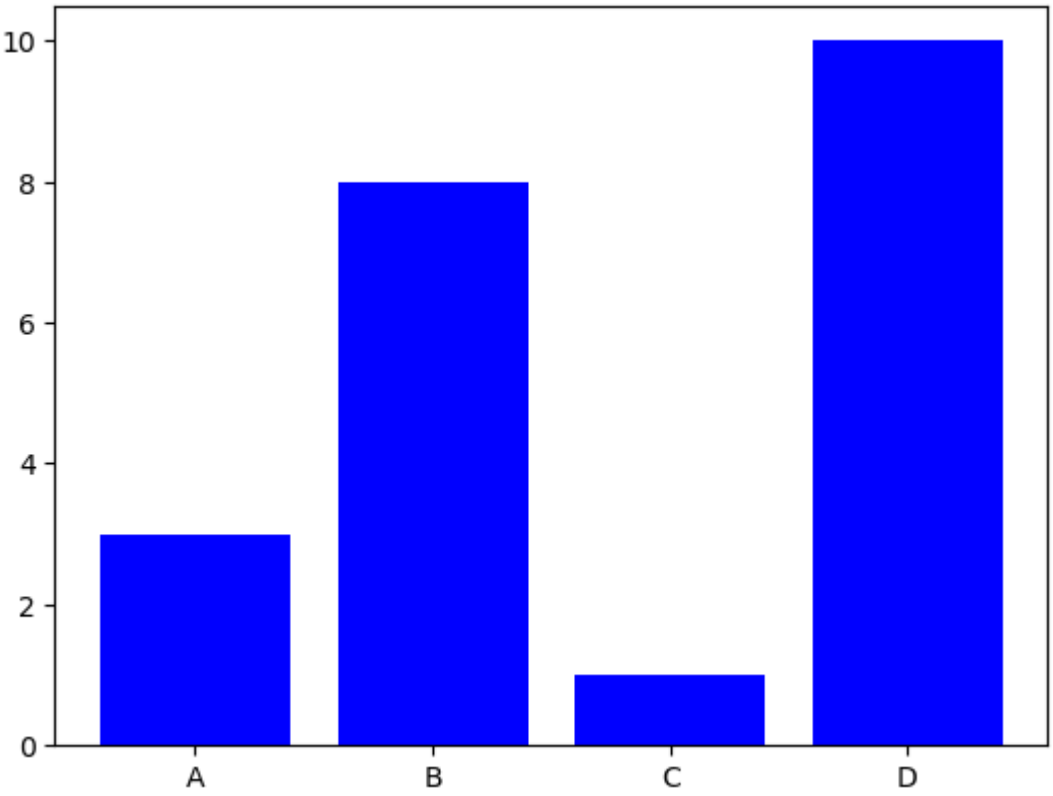
04) WAP to demonstrate the use of Pie chart.

In [43]:
```
y = [35, 25, 25, 15]
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.show()
```



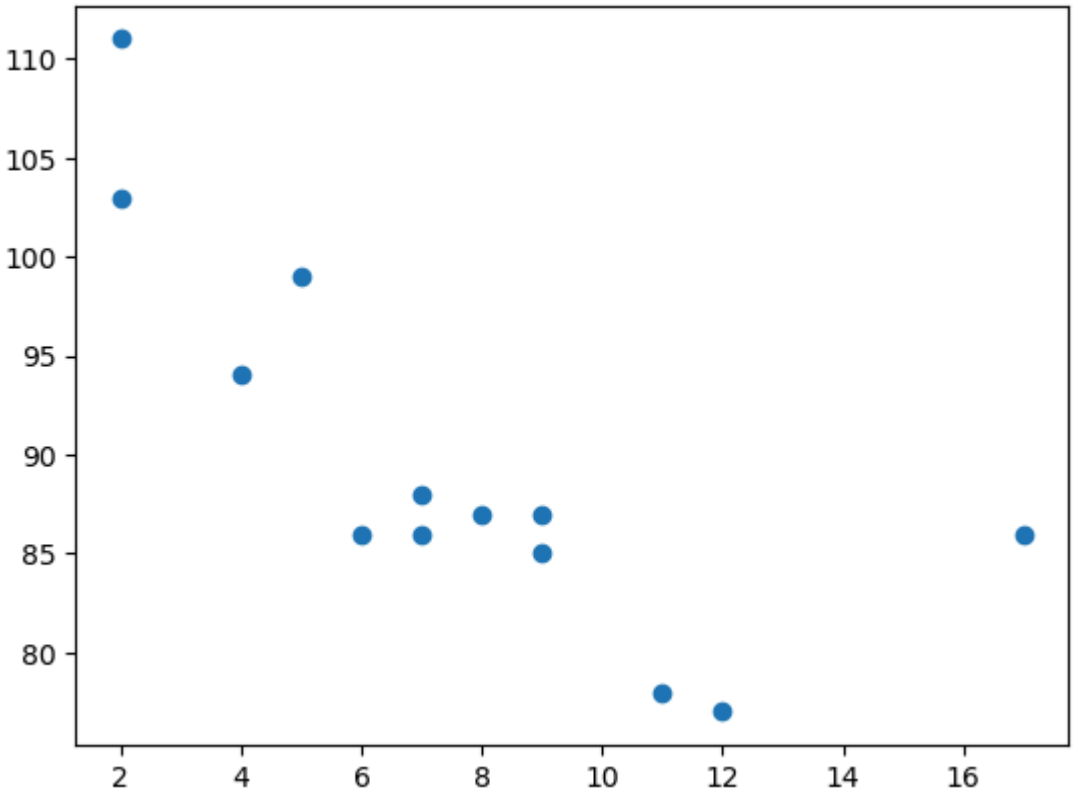
05) WAP to demonstrate the use of Bar chart.

In [27]:
```
x = ["A", "B", "C", "D"]
y = [3, 8, 1, 10]

plt.bar(x, y, color = "b")
plt.show()
```



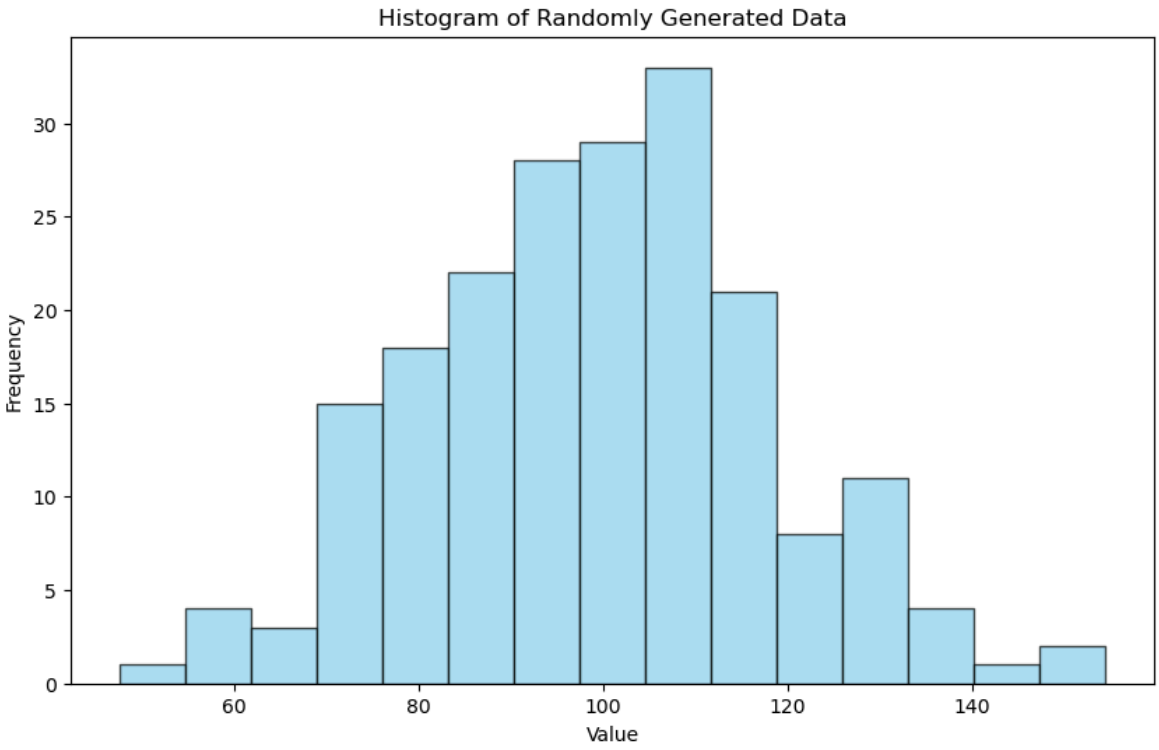
06) WAP to demonstrate the use of Scatter Plot.

In [26]:
```
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

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



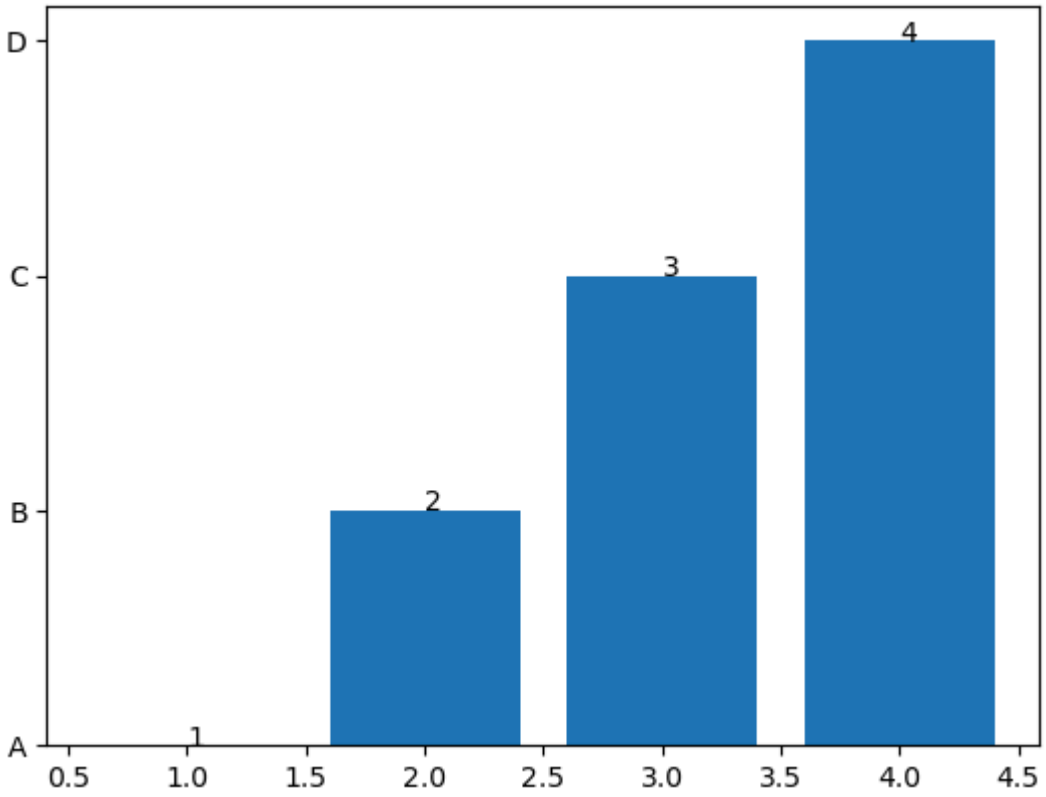
07) WAP to demonstrate the use of Histogram.

In [29]:
```
import matplotlib.pyplot as plt
import numpy as np

# Generate random data
np.random.seed(42)
data = np.random.normal(100, 20, 200)

plt.figure(figsize=(10, 6))
plt.hist(data, bins=15, color='skyblue', edgecolor='black', alpha=0.7)

plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram of Randomly Generated Data')
plt.show()
```



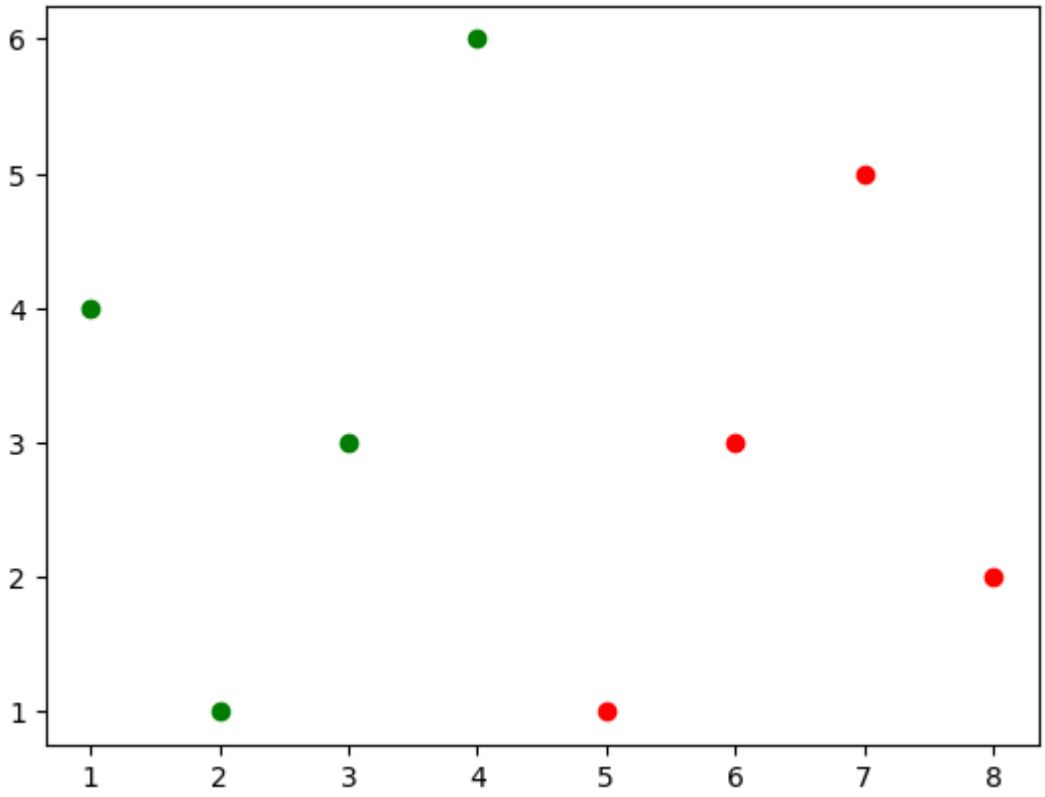
08) WAP to display the value of each bar in a bar chart using Matplotlib.

In [25]:
```
x = ["A", "B", "C", "D"]
y = [1, 2, 3, 4]
plt.bar(y,x)

for index, value in enumerate(y):
    plt.text(value, index,str(value))

plt.show()
```



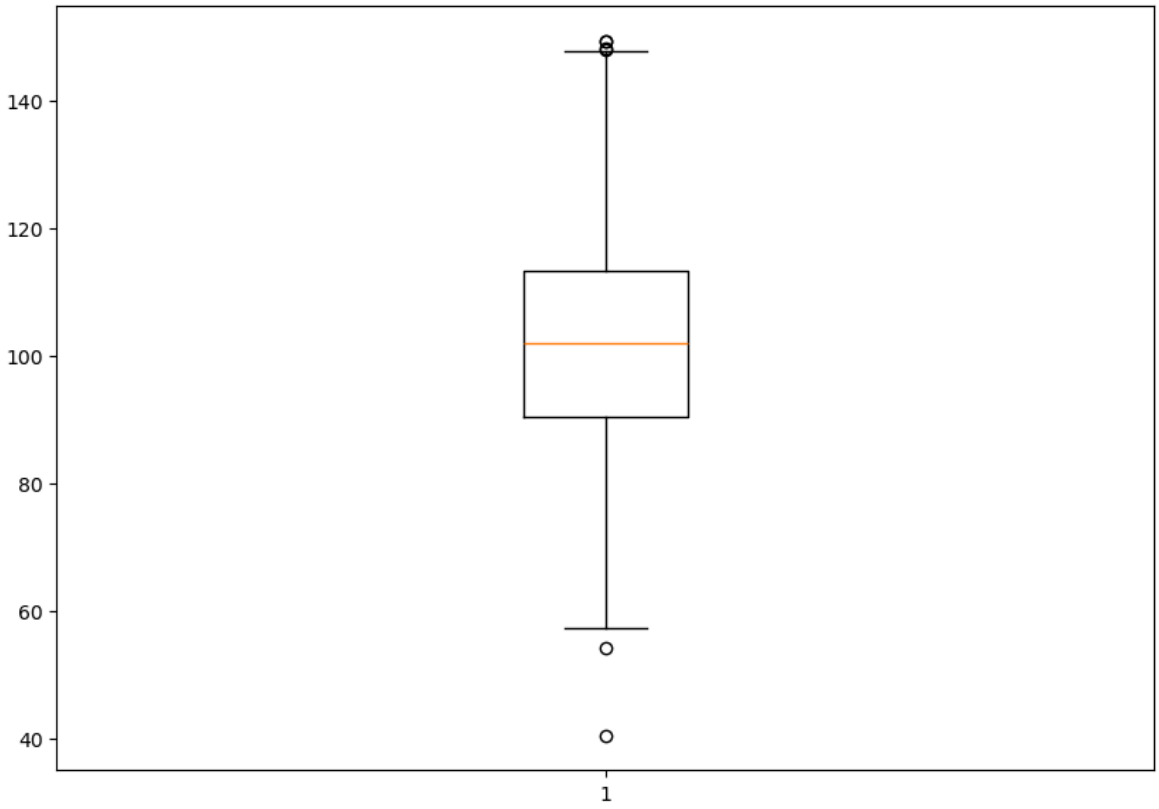
09) WAP create a Scatter Plot with several colors in Matplotlib?

In [22]:
```
x = [1, 2, 3, 4]
y = [4, 1, 3, 6]

plt.scatter(x, y, c='green')

x = [5, 6, 7, 8]
y = [1, 3, 5, 2]

plt.scatter(x, y, c='red')
plt.show()
```



10) WAP to create a Box Plot.

In [20]:
```
import matplotlib.pyplot as plt
import numpy as np

# Creating dataset
np.random.seed(10)
data = np.random.normal(100, 20, 200)

fig = plt.figure(figsize =(10, 7))
# Creating plot
plt.boxplot(data)

# show plot
plt.show()
```



In [ ]:

Python Programming - 2301CS404

Lab - 13

name:thummar jainil

enrollment number:23010101272 - 385

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

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 [2]:
```
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 = input("enter account no.")
        self.user_name = input("enter username")
```

```
        self.email = input("enter email")
        self.account_type = input("enter account type")
        self.account_balance = float(input("enter balance"))

    def DisplayAccountDetails(self):
        print("\n----- Account Details -----")
        print(f"Account Number: {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:.2f}")

userdetails = Bank_Account(123,'jainil','jainil@gmail.com','current','50000')
userdetails.DisplayAccountDetails

def main():

    account = Bank_Account("", "", "", "", 0.0)


    account.GetAccountDetails()


    account.DisplayAccountDetails()

if __name__ == "__main__":
    main()
```

```
----- Account Details -----
Account Number: 1
User Name: jainil
Email: jainilthummar@123
Account Type: business
Account Balance: $50000.00
```

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

In [3]:

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

    def findarea(self):
        return 3.14* self.r * self.r

    def perimeter(self):
        return 2*3.14*self.r
r = int(input("Enter radius="))
c = Circle(r)

area = c.findarea()
perimeter=c.perimeter()
print("Area = ",area)
print("Perimeter = ",perimeter)
```

```
Area =  78.5
Perimeter =  31.400000000000002
```

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

In [6]:
```
class Employee:
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary


    def update_name(self, new_name):
        self.name = new_name


    def update_age(self, new_age):
        self.age = new_age


    def update_salary(self, new_salary):
        self.salary = new_salary


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


employee1 = Employee("jainil", 21, 50000)
employee1.display_info()


employee1.update_name("jainil thummar")
employee1.update_age(31)
employee1.update_salary(55000)


employee1.display_info()
```

```
Employee Name: jainil
Employee Age: 21
Employee Salary: 50000
Employee Name: jainil thummar
Employee Age: 31
Employee Salary: 55000
```

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

In [8]:
```
class BankAccount:
    def __init__(self, account_holder, balance=0.0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
```

```
            self.balance += amount
            print(f"Deposited ${amount:.2f}. New balance: ${self.balance:.2f}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if amount <= 0:
            print("Withdrawal amount must be positive.")
        elif amount > self.balance:
            print("Insufficient balance!")
        else:
            self.balance -= amount
            print(f"Withdrew ${amount:.2f}. New balance: ${self.balance:.2f}")

    def check_balance(self):
        print(f"Account balance: ${self.balance:.2f}")


account = BankAccount("Utsav", 1000)
account.check_balance()

account.deposit(500)
account.withdraw(200)
account.withdraw(2000)
account.check_balance()
```

```
Account balance: $1000.00
Deposited $500.00. New balance: $1500.00
Withdrew $200.00. New balance: $1300.00
Insufficient balance!
Account balance: $1300.00
```

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 [9]:

```
class managing_inventory:
    def __init__(self, name, price,quantity):
        self.name = name
        self.price = price
        self.quantity=quantity

    def add(self):
        self.name = input("enter name.")
        self.price = input("enter price")
        self.quantity = input("enter quantity")
    def update_name(self, new_name):
        self.name = new_name

    def update_price(self, new_price):
        self.price = new_price

    def update_quantity(self, new_quantity):
        self.quantity = new_quantity

    def display_info(self):
        print(f"Name: {self.name}")
        print(f"price: {self.price}")
        print(f"quantity: {self.quantity}")
```

```
    def remove(self):

        self.name = None
        self.price = None
        self.quantity = None
        print("Item removed from inventory.")

b1=managing_inventory("LCD",50000,50)
b1.display_info()
b1.update_price(55000)
b1.display_info()
b1.remove()
b1.display_info()
```

```
Name: LCD
price: 50000
quantity: 50
Name: LCD
price: 55000
quantity: 50
Item removed from inventory.
Name: None
price: None
quantity: None
```

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

In [10]:
```
class Car:
    def __init__(self, make, model, year, color):

        self.make = make
        self.model = model
        self.year = year
        self.color = color
    def display_details(self):

        print(f"Car Details:")
        print(f"Make: {self.make}")
        print(f"Model: {self.model}")
        print(f"Year: {self.year}")
        print(f"Color: {self.color}")
my_car = Car(make="Tesla", model="Model S", year=2023, color="Red")
my_car.display_details()
```

```
Car Details:
Make: Tesla
Model: Model S
Year: 2023
Color: Red
```

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 [11]:
```
class StudentKit:

    principal_name = "Mr. ABC"
    def __init__(self, student_name):
        self.student_name = student_name
        self.attendance_days = 0
    def take_attendance(self, days_present):

        self.attendance_days = days_present
    def generate_certificate(self):

        print(f"Certificate of Attendance")
        print(f"Student Name: {self.student_name}")
        print(f"Principal: {StudentKit.principal_name}")
        print(f"Days Present: {self.attendance_days}")
        print(f"Total Days of School: 180")
        print("Certificate awarded for attendance.\n")
student_name = input("Enter student name: ")
student = StudentKit(student_name)
days_present = int(input(f"Enter number of days {student_name} was present: "))
student.take_attendance(days_present)
student.generate_certificate()
```

```
Certificate of Attendance
Student Name: jainil
Principal: Mr. ABC
Days Present: 50
Total Days of School: 180
Certificate awarded for attendance.
```

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

In [12]:
```
class Time:
    def __init__(self, hour, minute):
        self.hour = hour
        self.minute = minute
    def add_time(self, other):

        total_minutes = self.minute + other.minute
        total_hours = self.hour + other.hour + total_minutes // 60
        total_minutes = total_minutes % 60


        return Time(total_hours, total_minutes)

    def __str__(self):
        return f"{self.hour} hours, {self.minute} minutes"
time1 = Time(2, 45)
time2 = Time(3, 30)
result = time1.add_time(time2)
print(f"Resulting time: {result}")
```

```
Resulting time: 6 hours, 15 minutes
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 13

name:thummar jainil

enrollment number:23010101272 - 385

Continued..

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

In [3]:
```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

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

def display_area(rect):
    print(f"Area of the rectangle: {rect.calculate_area()}")

my_rect = Rectangle(10, 5)

display_area(my_rect)
```

```
Area of the rectangle: 50
```

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 [5]:
```
class Square:
    def __init__(self, side):
```

```
        self.side = side

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

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

square = Square(6)

square.area()
```

```
Area of the square: 36
```

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 [9]:
```
class Rectangle:
    def __init__(self, length, width):
        if self.is_square(length, width):
            print("THIS IS SQUARE")
            self.length = self.width = None
        else:
            self.length = length
            self.width = width

    def area(self):
        if self.length is not None and self.width is not None:
            area = self.length * self.width
            self.output(area)
            return area

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

    @staticmethod
    def is_square(length, width):
        return length == width


rect1 = Rectangle(5, 10)
if rect1.length is not None:
    rect1.area()

rect2 = Rectangle(8, 8)
```

```
The area of the rectangle is: 50
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 [11]:
```
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.")

sq = Square(5)
print("Side:", sq.get_side())
sq.set_side(10)
print("Updated Side:", sq.get_side())

sq.set_side(-3)
```

```
Side: 5
Updated Side: 10
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().

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


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

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


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

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


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


    def printBalance(self):
        print(f"The balance is: {self.balance}")


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

```
The balance is: 950000.0
```

15) WAP to demonstrate all types of inheritance.

In [13]:

```
class Fruit:
    def taste(self):
        print("Fruits have different tastes.")

class Mango(Fruit):
    def color(self):
        print("Mango is yellow.")


class Alphonso(Mango):
    def size(self):
        print("Alphonso mango is small.")


class Citrus:
    def vitamin_c(self):
        print("Citrus fruits are rich in Vitamin C.")

class Orange(Fruit, Citrus):
    def peel(self):
        print("Orange has a thick peel.")

class Apple(Fruit):
    def shape(self):
        print("Apple is round.")


class Banana(Fruit, Citrus):
    def energy(self):
        print("Banana gives instant energy.")


mango = Mango()
mango.taste()
```

```
mango.color()

alphonso = Alphonso()
alphonso.taste()
alphonso.color()
alphonso.size()

orange = Orange()
orange.taste()
orange.vitamin_c()
orange.peel()

apple = Apple()
apple.taste()
apple.shape()

banana = Banana()
banana.taste()
banana.vitamin_c()
banana.energy()
```

```
Fruits have different tastes.
Mango is yellow.
Fruits have different tastes.
Mango is yellow.
Alphonso mango is small.
Fruits have different tastes.
Citrus fruits are rich in Vitamin C.
Orange has a thick peel.
Fruits have different tastes.
Apple is round.
Fruits have different tastes.
Citrus fruits are rich in Vitamin C.
Banana gives instant energy.
```

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

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

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

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

emp = Employee("jainil", 30, 50000)

emp.display()
```

```
Name: jainil, Age: 30
Salary: 50000
```

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 [19]:
```
from abc import ABC, abstractmethod

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

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

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

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

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

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

```
Drawing a Rectangle
Drawing a Circle
Drawing a Triangle
```

In [ ]:
```

```