Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 1

23010101161 - Smit Maru - 260

01) WAP to print “Hello World”

In [12]:
```
print('Hello World')
```

```
Hello World
```

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

In [3]:
```
#without input
a,b=4,6
print(a+b)

#with input
a = int(input('Enter first number'))
b = int(input('Enter second number'))
print(a+b)
```

```
10
Enter first number5
Enter second number5
10
```

03) WAP to check the type of the variable.

In [11]:
```
print(type('abc'))
print(type(3))
print(type(3.745))
```

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

04) WAP to calculate simple interest.

In [18]:
```
p = 500
r = 0.04
t = 2

ans = (p*r*t)/100
print(ans)
```

```
0.4
```

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

In [17]:
```
r = float(input('Enter Radius Of Circle'))
print(3.14*r*r)
print(2*3.14*r)
```

```
Enter Radius Of Circle1
3.14
6.28
```

06) WAP to calculate area of a triangle.

In [16]:
```
b = float(input('Enter value of base'))
h = float(input('Enter value of height'))
print(0.5*b*h)
```

```
Enter value of base3
Enter value of height2
3.0
```

07) WAP to compute quotient and remainder.

In [1]:
```
a = int(input('Enter the number'))
print(a/10)
print(a%10)
```

```
1
0.1
```

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

In [14]:
```
fah=0;
cel = int(input("Enter celcius:"))
fah = (9/5)*cel +32;
print("Fahrenheit is:",fah)

fah = int(input("Enter Fahrenheit:"))
```

```
cel = (fah-32)*(5/9)
print("Celcius is:",cel)
```

```
Fahrenheit is: -0.3999999999999986
Celcius is: -1.1111111111111112
```

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

In [1]:
```
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}")
```

```
the distance is 1.4142135623730951
```

10) WAP to print sum of n natural numbers.

In [2]:
```
num = int(input("Enter Range:"))
total=0;
for i in range(1,num+1):
    total=total+i
print(total)
```

```
15
```

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

In [3]:
```
num = int(input("Enter Range:"))
total=0;
for i in range(1,num+1):
    total=total+(i*i)
print(total)
```

```
14
```

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

In [19]:
```
firstname = input('Enter First name')
lastname = input('Enter Last name')
print(firstname+lastname)
```

```
Enter First namerutvik
Enter Last namebhagiya
rutvikbhagiya
```

13) WAP to swap two numbers.

In [15]:
```
a = input('Enter First Number')
b = input('Enter second Number')
print('before swapped',a,b)
```

```
temp = a
a = b
b = temp
print('After swapped',a,b)
```

```
Enter First Number5
Enter second Number2
before swapped 5 2
After swapped 2 5
```

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

In [3]:
```
kilometer = float(input("Enter Kilometer:"))
meter = kilometer * 1000
feet = kilometer * 3280.84
inch = kilometer * 39370.1
centimeter = kilometer * 100000

print("Meter: ",meter)
print("Feet: ",feet)
print("Inch: ",inch)
print("Centimeter: ",centimeter)
```

```
Meter:  10000.0
Feet:  32808.4
Inch:  393701.0
Centimeter:  1000000.0
```

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

In [6]:
```
day = input("Enter Day:")
month = input("Enter Month:")
year = input("Enter Year:")

print(day,"-",month,"-",year)
```

```
10 - 10 - 10
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 2

23010101161 - Smit Maru - 260

if..else..

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

In [5]:
```
a = int(input("Enter the number"))
print("possitive") if a > 0 else ( print("Negative") if a < 0 else print("Number is
```

```
possitive
```

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

In [6]:
```
a = int(input("Enter the number"))
print("Odd") if (a%2 != 0) else print("Even")
```

```
Even
```

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

In [8]:
```
a = int(input("Enter the number 1"))
b = int(input("Enter the number 2"))
print(f"{a} is large") if a > b else (print(f"{b} is large") if b > a else print("b
```

```
both are same
```

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

In [12]:
```
a = int(input("Enter the number 1"))
b = int(input("Enter the number 2"))
c = int(input("Enter the number 3"))
if (a > b):
    if (a > c) :
        print(f"{a} is large")
    else:
        print(f"{c} is large")
else:
    if (b > c) :
        print(f"{b} is large")
    else:
        print(f"{c} is large")
```

```
14 is large
```

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 [15]:
```
a = int(input("Enter the year"))
print("given year is leep year") if (a%4==0 or a%100==0 and a%400 != 0) else print(
```

```
given year is leep year
```

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

In [27]:
```
arr = ["S","M","T","W","Th","F","Sa"]
print("Consider Sunday is 1")
a = int(input("Enter the number and get the name of the day"))
print("Ivalid number") if (a>7 or a<=0) else print(arr[a-1])


# match :
#     case :
#     case :
#     .
#     .
#     .
#     case_:
#         contant
```

```
Consider Sunday is 1
Ivalid number
```

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

In [28]:
```
a = int(input("Enter the number 1"))
b = int(input("Enter the number 2"))
```

```
print("if you went to add so click 1")
print("if you went to sub so click 2")
print("if you went to mul so click 3")
print("if you went to div so click 4")
c = int(input("Enter the number"))
if c == 1 :
    print(a+b)
elif c == 2:
    print(a-b)
elif c == 3:
    print(a*b)
elif c==4:
    print(a/b)
else:
    print("Invalid")
```

```
if you went to add so click 1
if you went to sub so click 2
if you went to mul so click 3
if you went to div so click 4
144
```

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 [29]:
```
a = int(input("Enter the number 1"))
b = int(input("Enter the number 2"))
c = int(input("Enter the number 3"))
d = int(input("Enter the number 4"))
e = int(input("Enter the number 5"))
p = ((a+b+c+d+e)*100)/500
print(f"percentage is : {p}")
if p < 35:
    print("Fail")
elif a >= 35 and a < 45:
    print("Pass Class between")
elif a >= 45 and a<60 :
    print("Second Class")
elif a>=60 and a<70 :
    print("First Class")
else:
    print("Distinction")
```

```
percentage is : 32.0
Fail
```

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 [35]:
```
a = int(input("Enter the side 1"))
b = int(input("Enter the side 2"))
b = int(input("Enter the side 3"))
if a == b == c :
    print("equilateral")
elif ((a*a) + (b*b) == (c*c)) or ((b*b) + (c*c))==(a*a) or ((a*a) + (c*c) == (b*b))
    print("right-angled triangle")
elif a != b != c:
    print("Scalene")
else:
    print("isosceles")
```

```
Scalene
```

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

In [1]:
```
a = int(input("Enter 1st side : "))
b = int(input("Enter 2nd side : "))
c = int(input("Enter 3rd size : "))


if ((a > b) and (a < c)) or ((a > c) and (a < b)):
    print("A is second Largest")
elif ((b > a) and (b < c)) or ((b > c) and (b < a)):
    print("B is second Largest")
else:
    print("C is second Largest")
```

```
B 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 [2]:
```
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)
```

```
8782.199999999999
```

In [ ]:

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 3

23010101161 - Smit Maru - 260

for and while loop

01) WAP to print 1 to 10.

In [2]:
```
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 [4]:
```
n = int(input("Enter range number"))
for i in range(1,n+1):
    print(i)
```

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

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

In [5]:
```
n = int(input("Enter the range n"))
for i in range(1,n+1,2):
    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]:
```
a = int(input("Enter the n1"))
b = int(input("Enter the n2"))
for a in range(a,b+1,1):
    if a%2==0 and a%3!=0:
        print(a)
```

```
2
4
8
10
```

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

In [14]:
```
n = int(input("Enter the n1"))
ans = 0
for n in range(1,n+1):
    ans += n
print(ans)
```

```
55
```

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

In [15]:
```
n = int(input("Enter the n1"))
ans = 0
for n in range(1,n+1):
    ans += n*n
```

```
print(ans)
```

```
30
```

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

In [22]:
```
# n = int(input("Enter the n1"))
# ans = 0
# for n in range(1,n+1):
#     if n%2==0:
#         ans -= n
#     else:
#         ans += n
# print(ans)
n = int(input("Enter the n1"))
sum = 0
sign = 1
for i in range(1,n+1):
    sum+=i*sign
    sign*=-1
print(sum)
```

```
6
```

08) WAP to print multiplication table of given number.

In [21]:
```
n = int(input("Enter the n1"))
i = 1
for i in range(1,n+1):
    print(f"{n} * {i} = {n*i}")
    i+=1
```

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

09) WAP to find factorial of the given number.

In [23]:
```
n = int(input("Enter the n1"))
ans = 1
for i in range(n,0,-1):
    ans *= i
print(ans)
```

```
120
```

10) WAP to find factors of the given number.

In [24]:
```
n = int(input("Enter the n1"))
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 [37]:
```
n = int(input("Enter the n1"))
count = 0
for i in range(2,n):
    if n%i!=0:
        count+=1
        break
if count==1:
    print("not")
else:
    print("prime")
```

```
not
```

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

In [30]:
```
n = input()
ans = 0
for i in range(len(n)):
    ans += int(n[i])
print(ans)
```

```
6
```

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

In [32]:
```
n = input("Enter the number")
c = n[::-1]
if c == n:
    print("palindrome")
else:
    print("Not")
```

```
Not
```

14) WAP to print GCD of given two numbers.

In [42]:
```
n1 = int(input("Enter n1"))
n2 = int(input("Enter n2"))
ans = 0
for i in range(1,(n1 if n1>n2 else n2),1):
    if n1%i==0 and n2%i==0:
        ans = i
print(ans)
```

```
9
```

Python Programming - 2301CS404

Lab - 4

23010101161 - Smit Maru - 260

String

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

In [6]:
```
s = input("Enter String")
sr = s[::-1]
if s == sr:
    print("Palindrome")
else:
    print("Not palindrome")
```

```
Palindrome
```

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

In [11]:
```
s = input("Enter String")
sr = s[::-1]
print(sr)
```

```
tims
```

03) WAP to remove ith character from given string.

In [31]:
```
s = input("Enter String")
n = int(input("Enter INT"))
print(s[:n:] + s[n+1::])
```

```
smta
```

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

In [16]:
```
s = input("Enter String")
count = 0
for char in s:
    count += 1
print(count)
```

```
4
```

05) WAP to print even length word in string.

In [22]:
```
s = input("Enter String")
a = s.split()
for i in range(0,len(a),1):
    if len(a[i]) % 2 == 0:
        print(a[i])
```

```
am
smit
```

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

In [30]:
```
s = input("Enter String")
a = ['a','e','i','o','u','A','E','I','O','U']
count = 0
for i in range(0,len(s)):
    if s[i] in a:
        count += 1
print(count)
```

```
3
```

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

In [1]:
```
s = input("Enter String")
a = s.split()
ans = ""
for i in range(0,len(a)):
    if len(a[i]) == 1:
        ans += a[i].upper()
    else:
        k = 0
        for char in a[i]:
            if k==0 or k==(len(a[i])-1):
                ans += char.upper()
                k += 1
            else:
                ans += char
                k += 1
```

```
    ans += " "
print(ans)
```

```
I AM SmiT MarU
```

08) WAP to convert given array to string.

In [15]:
```
s = input("Enter words separated by Coma(,) : ")
arr = s.split(',')
result = " ".join(arr)
# print(type(result))
print("Converted string:", result)
```

```
<class 'str'>
Converted string: wasd;mlmswedf wesfd
```

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

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

In [4]:
```
s = input("Enter String")
t = input("Enter String")
print("case mistek") if t.lower()==s else print("Not")
```

```
case mistek
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

In [2]:
```
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: **** **** **** 1234
```

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

s1 = decimal and s2 = medical are Anagram

In [2]:
```
a = input("Enter first string: ")
b = input("Enter second string: ")

print("Anagram") if sorted(a) == sorted(b) else print("Not Anagram")
```

```
Not Anagram
```

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

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

In [5]:
```
s = input("Enter the input")
a , b = "",""
for char in s:
    if(char.islower()):
        a += char
    else:
        b += char
print(a+b)
```

```
lsarwiwhtwEHMV
```

In [ ]:
```

```

Python Programming - 2301CS404

Lab - 5

23010101161 - Smit Maru - 260

List

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

In [14]:
```
a = []
while True:
    number = int(input("Enter the number and if you enter 0 then out from the list"
    if number != 0:
        a.append(number)
    elif(number == 0):
        break
print(a)
sum = 0
for i in a:
    sum += i
print(sum)
```

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

02) WAP to find largest element in a List.

In [22]:
```
a = []
while True:
    number = int(input("Enter the number and if you enter 0 then out from the list"
    if number != 0:
        a.append(number)
    elif(number == 0):
        break
large = a[0]
```

```
for i in a:
    if i >= large:
        large = i
print(large)
```

```
32
```

03) WAP to find the length of a List.

In [23]:
```
a = []
while True:
    number = int(input("Enter the number and if you enter 0 then out from the list"
    if number != 0:
        a.append(number)
    elif(number == 0):
        break
print(len(a))
```

```
3
```

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

In [28]:
```
a = []
while True:
    number = int(input("Enter the number and if you enter 0 then out from the list"
    if number != 0:
        a.append(number)
    elif(number == 0):
        break
temp = a[0]
a[0] = a[-1]
a[-1] = temp
print(a)
```

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

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

In [52]:
```
a = [1,2,3,4,5]
b=[]
b = a[(len(a)//2):]+a[:(len(a)//2) ]
print(b)
```

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

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

In [53]:
```
a = [1,2,3,4,5]
n1 = int(input("Enter the number 1"))
n2 = int(input("Enter the number 2"))
temp = a[n1]
```

```
a[n1] = a[n2]
a[n2] = temp
print(a)
```

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

07) WAP to reverse the list entered by user.

In [54]:
```
l = [1,2,3,4,5,6]
print(l[::-1])
```

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

08) WAP to print even numbers in a list.

In [55]:
```
l = [1,2,3,4,5,6]
for i in l:
    if i%2 == 0:
        print(i)
```

```
2
4
6
```

09) WAP to count unique items in a list.

In [65]:
```
l = [1,2,5,3,4,5,6,6]
a = {i for i in l}
print(len(a))
```

```
6
```

10) WAP to copy a list.

In [66]:
```
l = [1,2,3,4,5,6]
l2 = l.copy()
print(l2)
```

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

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

In [68]:
```
n = int(input("Enter tye range number"))
for i in range(0,n+1):
    if i%2 != 0:
        print(i)
```

```
1
3
5
```

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

In [69]:
```
l = [1,2,5,3,4,5,6,6]
print(l.count(6))
```

```
2
```

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

In [70]:
```
l = [1,2,3,4,5,6]
l.sort()
print(l[-2])
```

```
5
```

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

In [1]:
```
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)
```

```
[]
```

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

In [72]:
```
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 [8]:
```
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)
```

```
['baana', 'ba']
```

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

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

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

```
{3, 4}
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 6

23010101161 - Smit Maru - 260

Tuple

01) WAP to find sum of tuple elements.

In [17]:
```
t1 = []
n = int(input("Enter trhe size of tuple"))
for i in range(n):
    t1.append(int(input("Enter the element")))
t1 = tuple(t1)
print(sum((t1)))
```

```
10
```

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

In [19]:
```
t1 = []
n = int(input("Enter trhe size of tuple"))
for i in range(n):
    t1.append(int(input("Enter the element")))
t1 = tuple(t1)
t1 = sorted(t1)
k = int(input("Enter k"))
print(tuple(t1[:k:1]))
print(tuple(t1[:k:-1]))
```

```
(1, 2)
(5, 4)
```

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

In [22]:
```
t1 = []
n = int(input("Enter trhe size of tuple"))
for i in range(n):
    t1.append(int(input("Enter the element")))
t1 = tuple(t1)
count = 0
k = int(input("Enter k"))
for i in t1:
    if i%k == 0:
        print(i)
        count += 1
if count == 0:
    print(f"No element is devisable by {k}")
```

```
No element is devisable by 6
```

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

In [42]:
```
t1 = []
n = int(input("Enter trhe size of tuple"))
for i in range(n):
    t1.append(int(input("Enter the element")))
t1 = tuple(t1)
ans = [(t1[i],t1[i]**3) for i in range(len(t1))]
ans
########################################################################## OR ###############
# ans = []
# for i in range(0,len(t1)):
#     temp = (t1[i],t1[i]**3)
#     ans.append(temp)
# ans
```

Out[42]:
```
[(2, 8), (3, 27)]
```

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

In [47]:
```
t1 = [(1, 2, 3), (5, 6, -7), (18, 45, 93)]
ans = []
for i in range(len(t1)):
    for j in range(len(t1[i])):
        if t1[i][j] >= 0:
            ans.append(t1[i][j])
print(tuple(ans))
```

```
(1, 2, 3, 5, 6, 18, 45, 93)
```

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

In [53]:
```
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 [1]:
```
l1 = [(1, 2), (1, 2, 3), (1, 2, 3, 4), (1, 2, 3, 4, 5)]
k = int(input("Enter the length"))
l = [i for i in l1 if len(i) != k]
print(l)
```

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

08) WAP to remove duplicates from tuple.

In [3]:
```
t1 = (1,2,2,3)
t1 = set(t1)
t1 = tuple(t1)
t1
```

Out[3]: (1, 2, 3)

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

In [4]:
```
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[4]: (2, 6, 12, 4)

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

In [6]:
```
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
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 7

23010101161 - Smit Maru - 260

Set & Dictionary

01) WAP to iterate over a set.

In [3]:
```
n = int(input("Enter the size of the set"))
s = set()
for i in range(n):
    s.add(input(f"Enter {i+1} element"))
for i in s:
    print(i)
```

```
hello
10.3
True
2
```

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

In [13]:
```
s = {10,'hello',10.2,False}
print(f"Set to List : {list(s)}")
print(f"Set to Tuple : {tuple(s)}")
print(f"Set to Str : {''.join(map(str,s))}")
```

```
Set to List : [False, 10.2, 10, 'hello']
Set to Tuple : (False, 10.2, 10, 'hello')
Set to Str : False10.210hello
```

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

In [15]:
```
s = {65,10,21,10.2,98}
print(f"Max : {max(s)}")
print(f"Min : {min(s)}")
```

```
Max : 98
Min : 10
```

04) WAP to perform union of two sets.

In [22]:
```
s1 = {1,'hello',2,3.3,True}
s2 = {4.4,5,6,8}
print(f"Union : {s1.union(s2)}")
```

```
Union : {1, 2, 3.3, 'hello', 4.4, 5, 6, 8}
```

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

In [25]:
```
s1 = {1,2,3.3}
s2 = {4.4,5,6,8}
True if len(s1&s2)>=1 else False
```

Out[25]:
```
False
```

06) WAP to remove duplicates from list.

In [28]:
```
s1 = [1,2,3.3,3.3,4,4]
print(list(set(s1)))
```

```
[1, 2, 3.3, 4]
```

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

In [35]:
```
s1 = "Smit Maru Smit"
l = set(s1.split())
for i in l:
    if(s1.count(i) == 1):
        print(i)
```

```
Maru
```

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

In [39]:
```
s1 = {1,2,3.3,4.4}
s2 = {4.4,5,6,8}
print(s1.symmetric_difference(s2))
```

```
{1, 2, 3.3, 5, 6, 8}
```

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

In [41]:
```
s1 = 'Smit'
s2 = 'tmiS'
True if sorted(s1) == sorted(s2) else False
```

Out[41]: True

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

In [43]:
```
s1 = {1,2,3.3,4.4}
s2 = {4.4,5,6,8}
s3 = {4.4,9,10}
print((s1&s2)&s3)
```

```
{4.4}
```

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

In [45]:
```
s1 = 'aeiouAEIOU'
s2 = 'Haril Hadvani'
temp = set(s1) & set(s2)
print(len(temp))
```

```
2
```

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

In [1]:
```
s = (input("Enter String"))
temp = set(s)
a = {'0','1'}
if temp.issubset(a):
    print("Binary")
else:
    print("Not")
```

```
Binary
```

13) WAP to sort dictionary by key or value.

In [2]:
```
data = {"apple": 3, "banana": 1, "cherry": 2}

# Sort by keys
by_keys = dict(sorted(data.items(), key=lambda x: x[0]))
print("Sorted by keys:", by_keys)

# Sort by values
by_values = dict(sorted(data.items(), key=lambda x: x[1]))
print("Sorted by values:", by_values)
```

```
Sorted by keys: {'apple': 3, 'banana': 1, 'cherry': 2}
Sorted by values: {'banana': 1, 'cherry': 2, 'apple': 3}
```

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

In [4]:
```
user_input = input("Enter a dictionary : ")


user_dict = {}
for item in user_input.split(','):
    key, value = item.split(':')
    user_dict[key] = float(value)
total_sum = sum(user_dict.values())
print(f"The sum of all values in the dictionary is: {total_sum}")
```

```
The sum of all values in the dictionary is: 11.0
```

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 [7]:
```
# Given dictionary
dict1 = {'a': 5, 'c': 8, 'e': 2}

# Key to search
find = 'A'

# Check if the key exists in the dictionary
if find in dict1:
    print(f"The value of '{find}' is:", dict1[find])
else:
    print(f"Key '{find}' Not Found")
```

```
Key 'A' Not Found
```

In [ ]:
```

```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 8

23010101161 - Smit Maru - 260

User Defined Function

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

In [3]:
```
def calculate_bmi(mass, height):
    bmi = mass / (height ** 2)
    return bmi
mass = float(input("Enter your mass in kilograms: "))
height = float(input("Enter your height in meters: "))
bmi = calculate_bmi(mass, height)
print(f"Your BMI is: {bmi:.2f}")
```

```
Your BMI is: 23.88
```

02) Write a function that add first n numbers.

In [4]:
```
def sum_of_first_n_numbers(n):
    return n * (n + 1) // 2
n = int(input("Enter a positive integer (n): "))
result = sum_of_first_n_numbers(n)
print(f"The sum of the first {n} natural numbers is: {result}")
```

```
The sum of the first 25 natural numbers is: 325
```

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

In [7]:
```
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
number = int(input("Enter a number: "))
result = is_prime(number)
print(f"The number {number} is prime: {result}")
```

```
The number 13 is prime: 1
```

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

In [8]:
```
def is_prime(num):

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

def primes_between(start, end):
    prime_list = []
    for num in range(start, end + 1):
        if is_prime(num):
            prime_list.append(num)
    return prime_list
start = int(input("Enter the starting number: "))
end = int(input("Enter the ending number: "))

prime_numbers = primes_between(start, end)
print(f"Prime numbers between {start} and {end}: {prime_numbers}")
```

```
Prime numbers between 25 and 56: [29, 31, 37, 41, 43, 47, 53]
```

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

In [10]:
```
def is_palindrome(s):
    s = s.replace(" ", "").lower()
    # Compare the string with its reverse
    return s == s[::-1]
input_string = input("Enter a string: ")
result = is_palindrome(input_string)
print(f"Is the string a palindrome? {result}")
```

```
Is the string a palindrome? True
```

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

In [11]:
```
def sum_of_list_elements(lst):
    return sum(lst)
numbers = list(map(float, input("Enter elements of the list separated by spaces: ")
result = sum_of_list_elements(numbers)
print(f"The sum of all elements in the list is: {result}")
```

```
The sum of all elements in the list is: 16.0
```

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

In [12]:
```
def sum_of_first_elements(tuples_list):
    return sum(t[0] for t in tuples_list)
tuples_list = [(1, 2), (3, 4), (5, 6)]
result = sum_of_first_elements(tuples_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 [23]:
```
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 11th term of the Fibonacci Series is: 89
```

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 [24]:
```
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 102 is: Rahul
```

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 [25]:
```
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 [26]:
```
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 [30]:
```
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 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 [32]:
```
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: 6
Number of lowercase letters: 6
```

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

In [33]:
```
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: 25.0
```

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

In [34]:
```
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 [35]:
```
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', 'Dav
id', '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 [1]:
```
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(1,
print("Variable-Length Keyword Arguments Result:", variable_length_keyword(name="Al
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': 'New
York'}
Keyword-Only Arguments Result: 30
Positional-Only Arguments Result: 200
```

In [ ]:

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 9

23010101161 - Smit Maru - 260

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('text.txt')
print(fp.read())
fp.close()
```

```
in the form of a string
linelinelineline by line
in the form of a list
in the form of a string
line by line
in the form of a list
```

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

In [3]:
```
fp = open('abc.txt','x')
print(fp.write("File is Create"))
fp.close()
```

14

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

In [4]:
```
fp = open('text.txt')
data = fp.readlines()
for i in range(0,5):
    print(data[i][:len(data[i])-1])
fp.close()
```

```
in the form of a string
linelinelineline by line
in the form of a list
in the form of a string
line by line
```

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

In [5]:
```
fp = open('text.txt')
data = fp.readlines()
long = ''

for line in data:
    words = line.split()
    for word in words:
        if len(word) > len(long):
            long = word
print(long)
fp.close()
```

```
linelinelineline
```

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

In [35]:
```
fp = open('text.txt')
data = fp.readlines()
print(f"No. of lines = {len(data)}")

word_len = 0
char_Count = 0
for line in data:
    word_len += len(line.split())
    char_Count += len(line)
print(word_len)
print(char_Count)
fp.close()
```

```
No. of lines = 6
30
129
```

06) WAP t ntent of a file to the another file.

In [36]:
```
fp = open('text.txt')
fp2 = open('text-copy.txt','x')
fp2.write(fp.read())
fp.close()
fp2.close()
```

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

In [40]:
```
fp = open('text.txt')
fp.seek(0,2)
print(fp.tell())
```

```
129
```

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

In [ ]:
```
def occurances_of_char():
    fp = open('text.txt')
    data = fp.readlines()
    x =
    char_count = 0
    for line in data:
        word_len = line.split()
        for j in word_len:
            if(x = j):
                char_Count += 1
    fp.close()
    print()
```

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 [6]:
```
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 [2]:
```
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 [7]:
```
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 [1]:
```
def replace_word(input_file, output_file, word1, word2):
    """Replace all occurrences of word1 with word2 in input_file and save to output
    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 [12]:
```
fp = open("text.txt","rb")
fp.read(5)
print(fp.tell())
fp.seek(0,2)
fp.seek(-4,1)
print(fp.tell())
fp.close()
```

```
5
125
```

In [ ]:
```
```

Python Programming - 2301CS404

Lab - 10

23010101161 - Smit Maru - 260

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 = 11
    b = 0
    c = '0'
    print(a/b)
except ZeroDivisionError:
    print("Number not Divisiable by zero")
try:
    d = int(input("Enter number"))
except ValueError:
    print("Enter integer number")
try:
    print(a+c)
except TypeError:
    print("Number is not concat with string/char")
```

```
Number not Divisiable by zero
```

```
Enter integer number
Number is not concat with string/char
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [7]:
```python
l1 = [10,20,30,40]
try:
    print(l1[4])
except IndexError:
    print("Index is out of range")
d1 = {1:'a',2:'b'}
try:
    print(d1[3])
except:
    print("Key not found in d1")
```

```
Index is out of range
Key not found in d1
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [12]:
```python
try:
    fp = open("abc.txt",'r')
    print(fp.read())
except FileNotFoundError:
    print("File not found")
try:
    import aaaa
except ModuleNotFoundError:
    print("Module not found")
```

```
File not found
Module not found
```

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

In [18]:
```python
try:
    a = 11
    b = 0
    c = '0'
    d = int(input("Enter number"))
    print(a+c)
    print(a/b)
except Exception as e:
    print(type(e).__name__+" : "+str(e))
```

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

05) WAP to demonstrate else and finally block.

In [19]:
```
try:
    print(11/0)
except ZeroDivisionError:
    print("Number can not devide by zero")
finally:
    print("Finally")
try:
    print(11/2)
except ZeroDivisionError:
    print("Number can not devide by zero")
else:
    print("Else")
```

```
11.0
Finally
5.5
Else
```

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 [35]:
```
try:
    s = input("Enter comma-separated numbers: ")
    a = s.split(',')
    numbers = [int(i) for i in a]
    print("Converted numbers:", numbers)
except ValueError:
    print("values could not be converted to an integer.")
```

```
Converted numbers: [1, 2, 3, 4]
```

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

In [32]:
```
class zero(Exception):
    pass
try:
    a = 1
    b = 0
    if b != 0:
        print(a/b)
    else:
```

```
        raise zero
except zero:
    print("Number not devide by zero")
```

```
Number not devide 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 [22]:
```
class ageError(Exception):
    def __init(self,msg):
        msg = self.msg
try:
    age = int(input("Enter the age"))
    if age > 18:
        print(age)
    else:
        raise ageError("Age grate then 18")
except ageError as e:
    print(e)
```

```
19
```

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 [25]:
```
class InvalidUsernameError(Exception):
    def __init(self,msg):
        msg = self.msg
try:
    s = input("Enter the name")
    if (len(s) >= 5) and (len(s) <= 15):
        print(s)
    else:
        raise InvalidUsernameError("Inavlid User Name")
except InvalidUsernameError as e:
    print(e)
```

```
smitmaru
```

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 [29]:
```
class NegativeNumberError(Exception):
    def __init(self,msg):
        msg = self.msg
try:
    a = int(input("Enter the age"))
    if a > 0:
        print(a)
    else:
        raise NegativeNumberError("a must be grate then 0")
except NegativeNumberError as e:
    print(e)
```

```
a must be grate then 0
```

Python Programming - 2301CS404

Lab - 11

23010101161 - Smit Maru - 260

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 [6]:
```
import Cal
print(Cal.Add(10,20))
print(Cal.Sub(10,20))
print(Cal.Mul(10,20))
print(Cal.Div(12,0))
print(Cal.Div(12,1))
```

```
30
-10
200
Error: Division by zero is not allowed
12.0
```

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

In [8]:
```
import random
s1 = input("Enter String")
print(s1[random.randint(0,len(s1)-1)])
```

```
d
```

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

In [9]:
```
import random
s1 = [10,20,3.0,40,50,60,70,80,980]
print(s1[random.randint(0,len(s1)-1)])
```

```
60
```

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

In [15]:
```
random.seed(10)
print(random.randint(1,6))
```

```
5
```

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

In [16]:
```
a = random.randrange(100,999,5)
b = random.randrange(100,999,5)
c = random.randrange(100,999,5)
print(a,b,c)
```

```
140 645 715
```

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 [20]:
```
l1 = []
while len(l1)!=100:
    n = random.randint(1,1000)
    if n not in l1:
        l1.append(n)
print(l1)

winner = random.choice(l1)
print('Winner : ',winner)
l1.remove(winner)
runnerup = random.choice(l1)
print('Runners up : ',runnerup)
```

```
[444, 994, 425, 457, 252, 698, 282, 147, 633, 535, 183, 886, 123, 274, 467, 310, 16
9, 677, 665, 966, 841, 179, 797, 181, 492, 792, 356, 336, 446, 230, 6, 558, 730, 45,
340, 919, 328, 249, 82, 269, 459, 415, 597, 163, 400, 943, 891, 508, 690, 915, 248,
754, 538, 933, 279, 533, 495, 617, 512, 65, 171, 502, 691, 473, 920, 411, 140, 431,
555, 599, 358, 552, 398, 719, 170, 573, 452, 102, 992, 998, 426, 849, 37, 893, 3, 45
0, 549, 67, 52, 359, 95, 156, 112, 623, 465, 854, 819, 497, 143, 470]
Winner :  533
Runners up :  411
```

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

In [18]:
```
import datetime
print(datetime.datetime.now())
```

```
2025-02-10 13:14:26.431616
```

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

In [27]:
```
a = datetime.datetime.now()
b = datetime.timedelta(days = 7)
print(a-b)
```

```
2025-02-03 13:22:17.575423
```

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

In [33]:
```
a = datetime.datetime(2025,2,10)
b = datetime.datetime(2025,3,10)
print(abs(a-b).days)
```

```
28
```

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

In [37]:
```
a = datetime.datetime(2006,2,1)
print(a.strftime('%A'))
```

```
Wednesday
```

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

In [39]:
```
a = datetime.datetime(2006,2,1)
print(a.strftime('%c'))
```

```
Wed Feb  1 00:00:00 2006
```

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

In [42]:
```
import math
p = [3]
q = [1]
print (math.dist(p,q))
```

```
2.0
```

In [ ]:

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

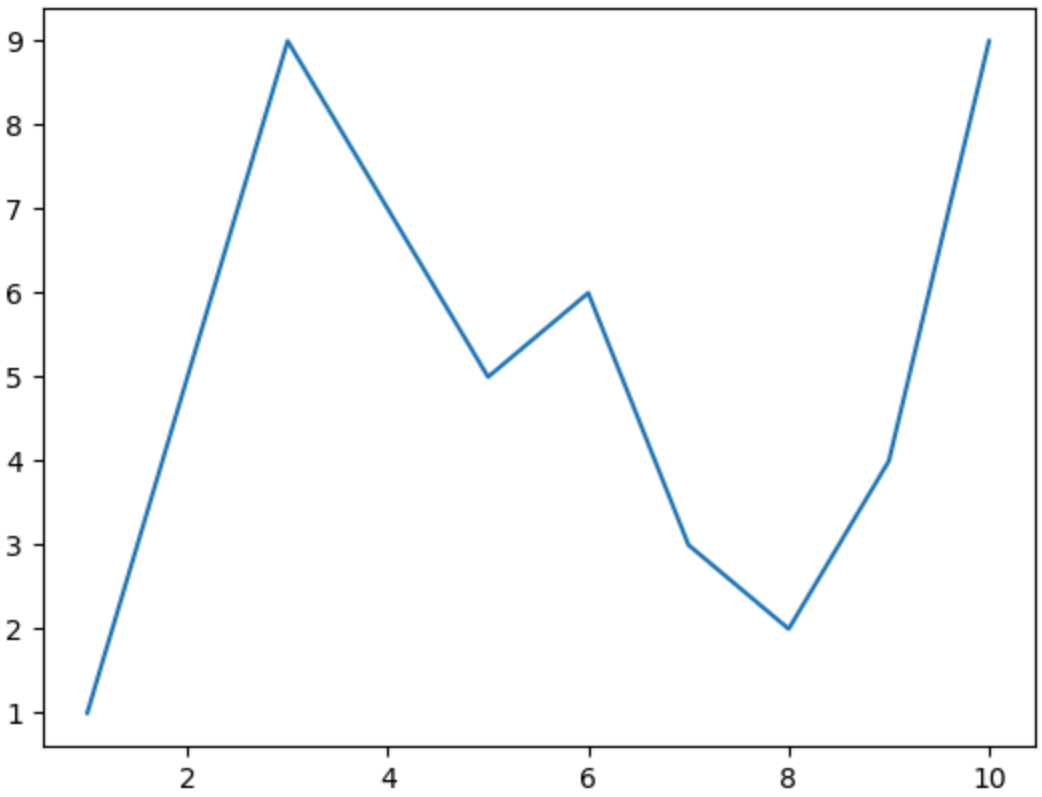
Python Programming - 2301CS404

Lab - 12

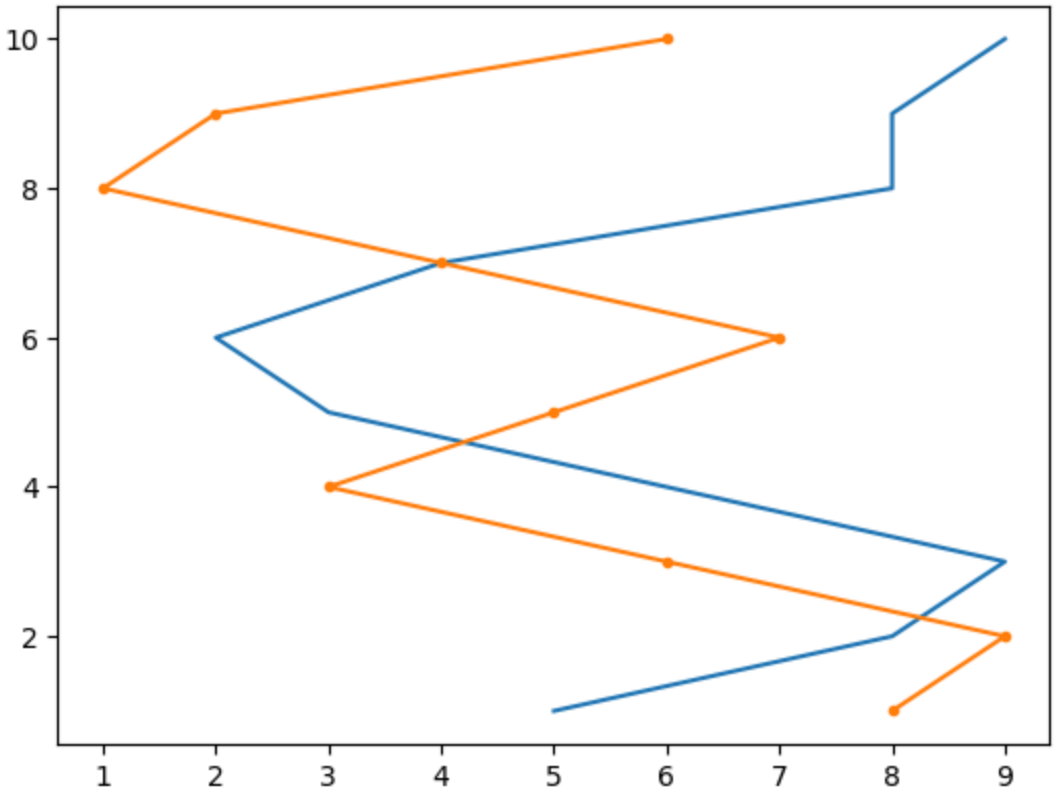
Smit Maru - 23010101161 - 260

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

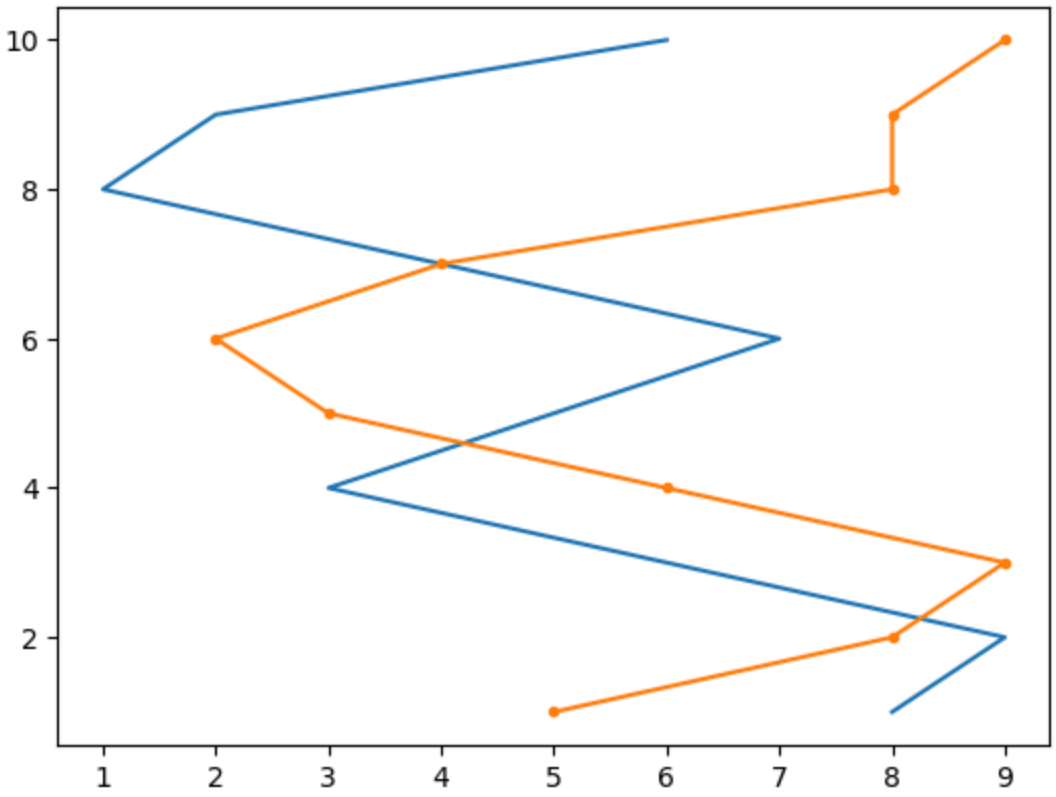
In [8]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]
plt.plot(x,y)
plt.show(x,y)
# write a code to display the line chart of above x & y
```



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)
```

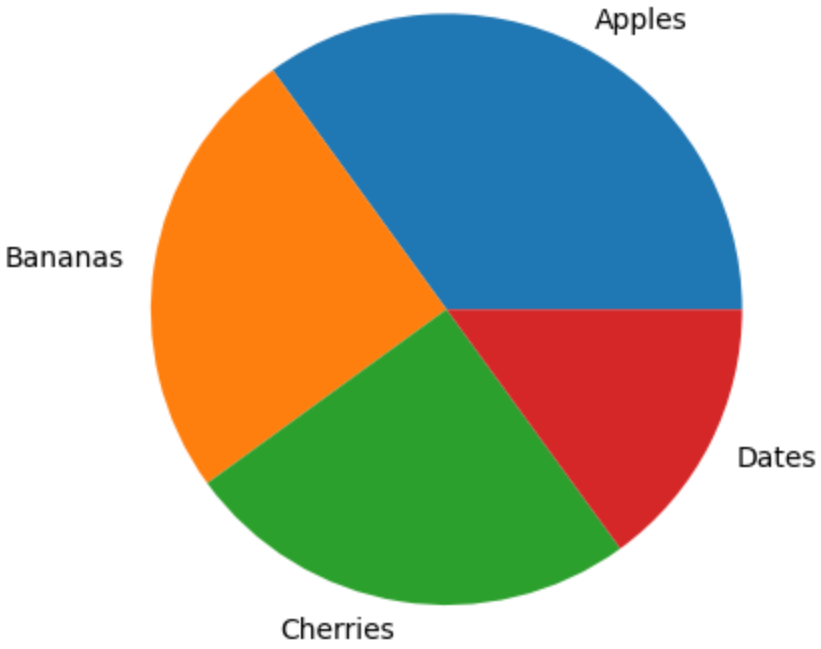


In [22]:
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]

# write a code to generate below graph
```



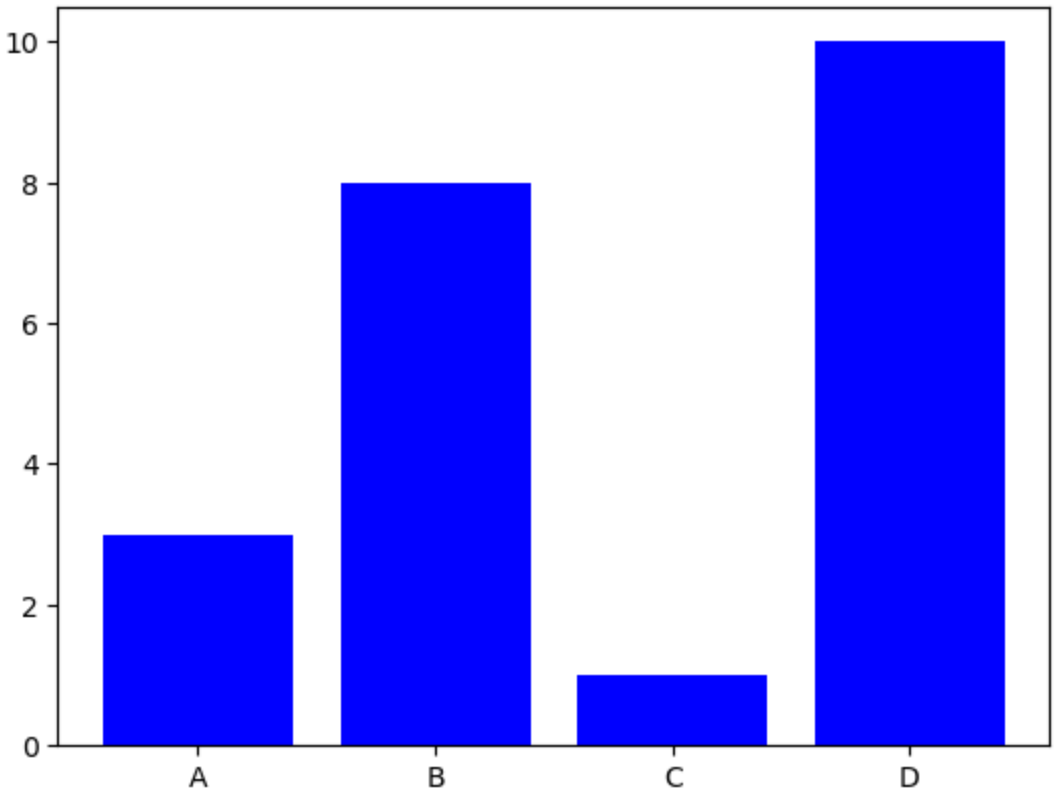
04) WAP to demonstrate the use of Pie chart.

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

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



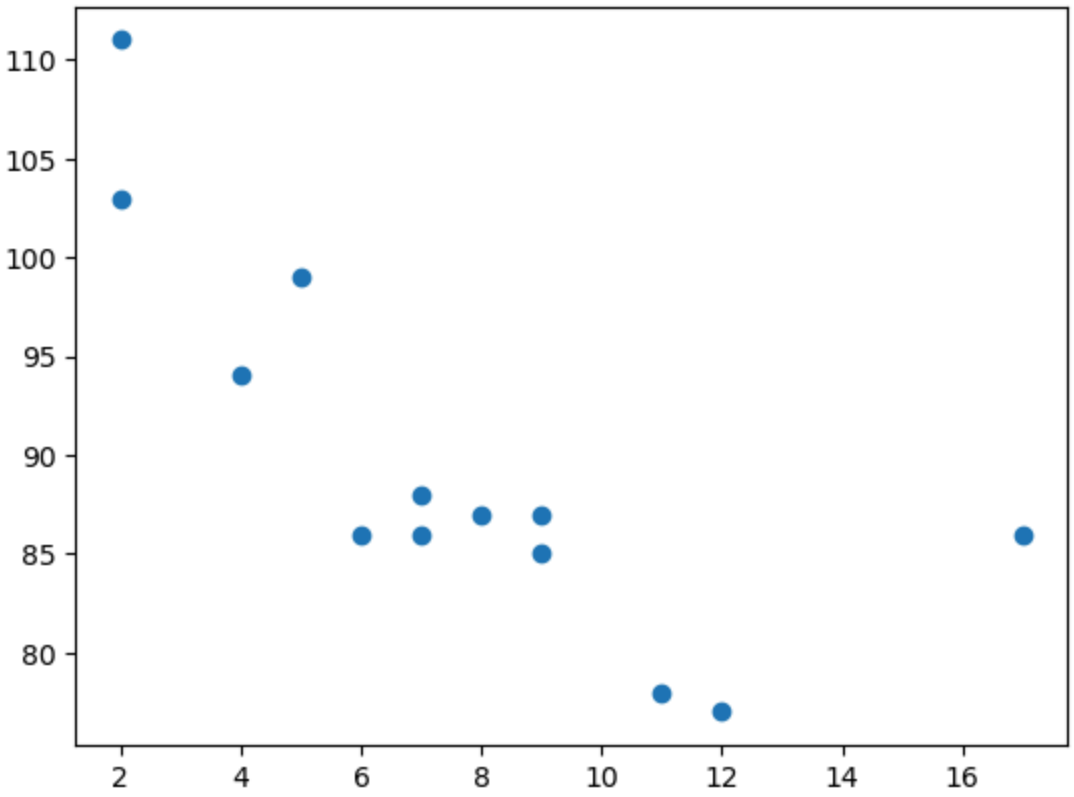
05) WAP to demonstrate the use of Bar chart.

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

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



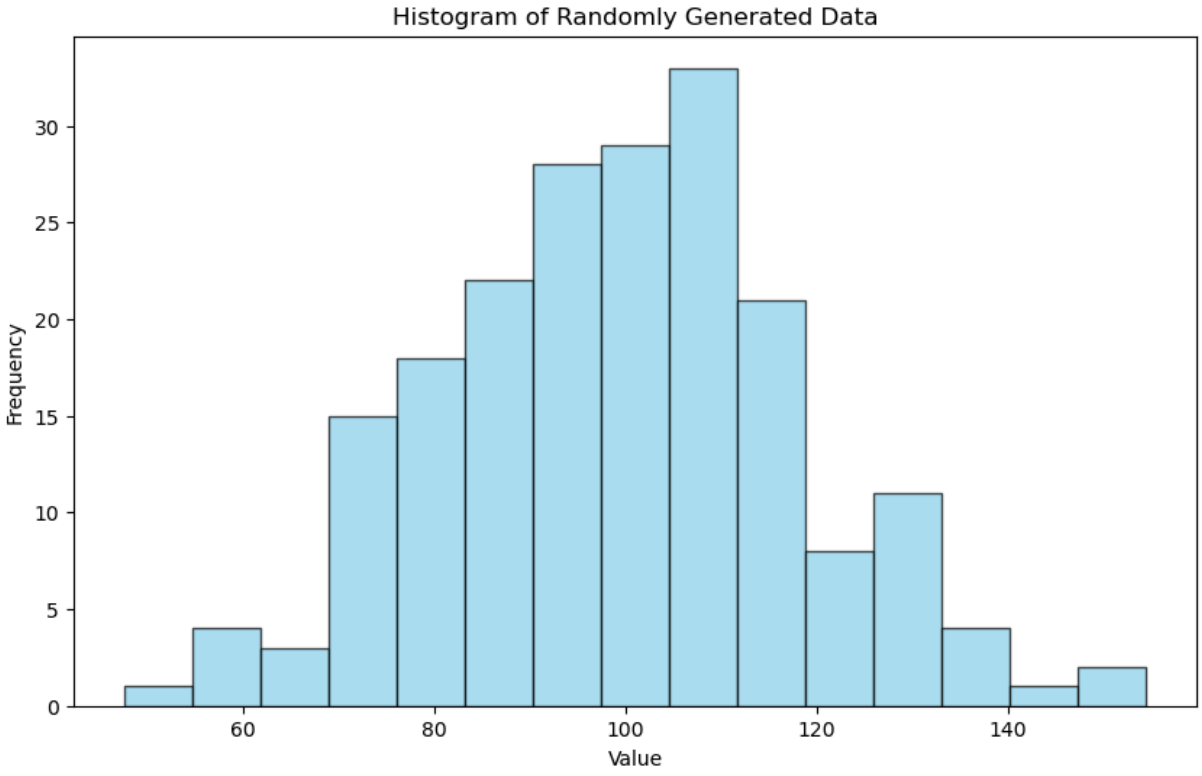
06) WAP to demonstrate the use of Scatter Plot.

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()
```



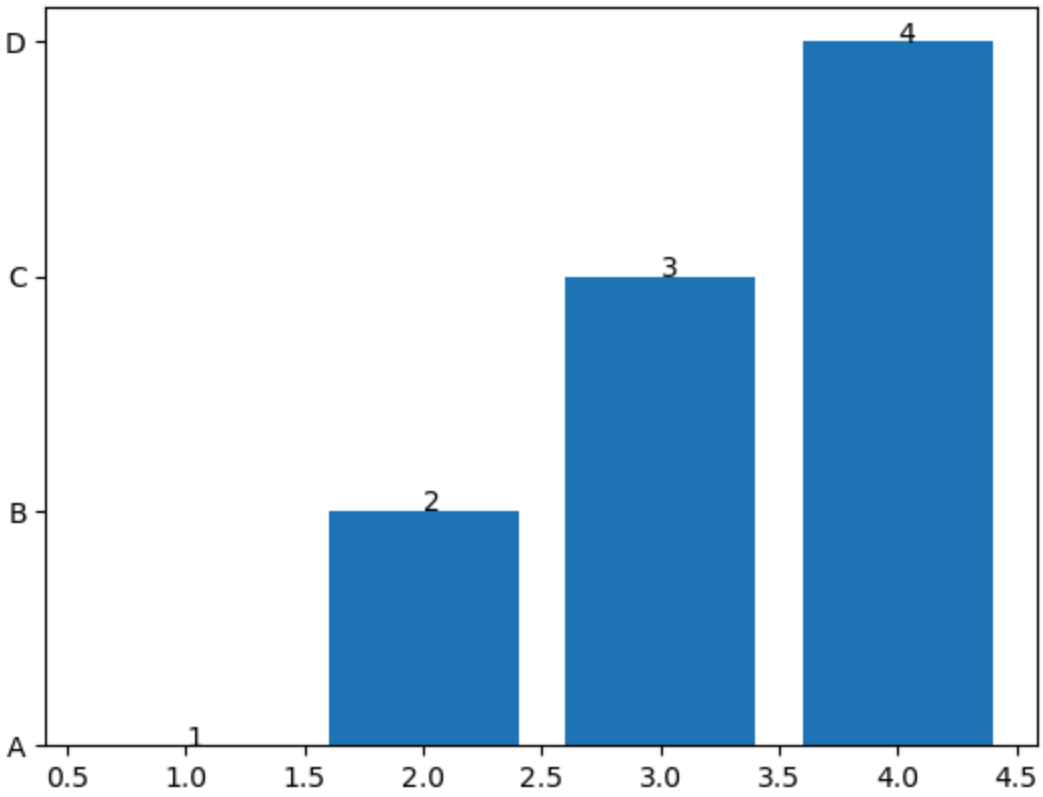
07) WAP to demonstrate the use of Histogram.

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()
```



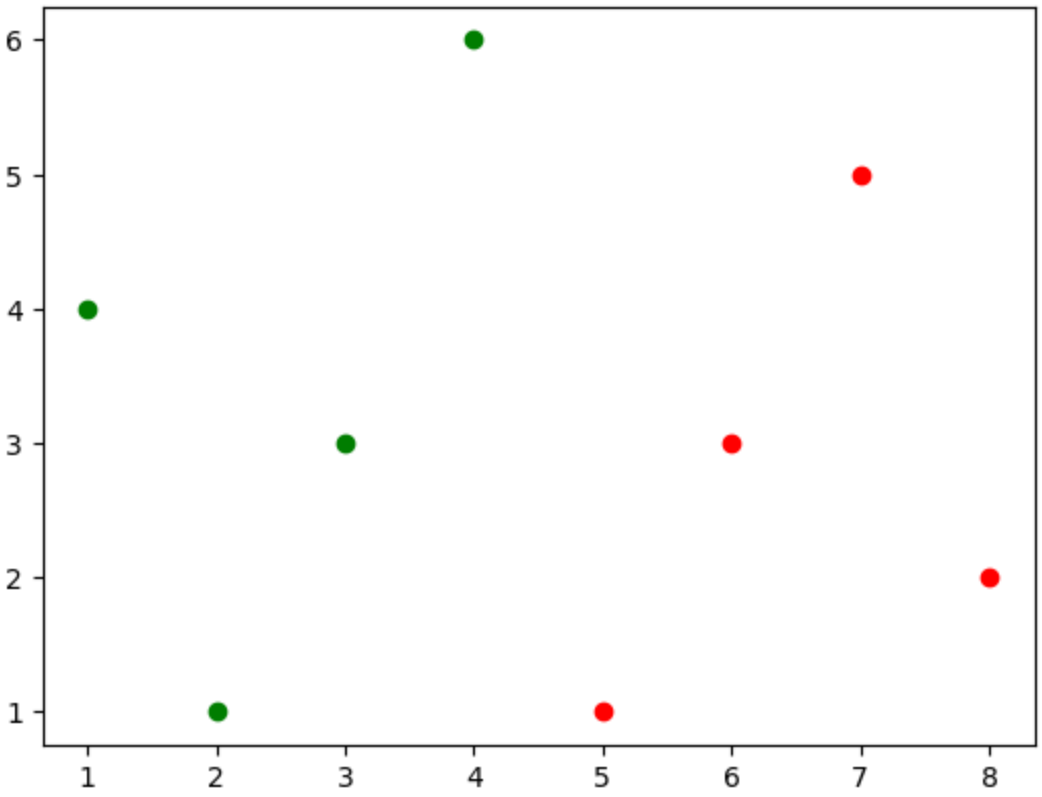
08) WAP to display the value of each bar in a bar chart using Matplotlib.

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()
```



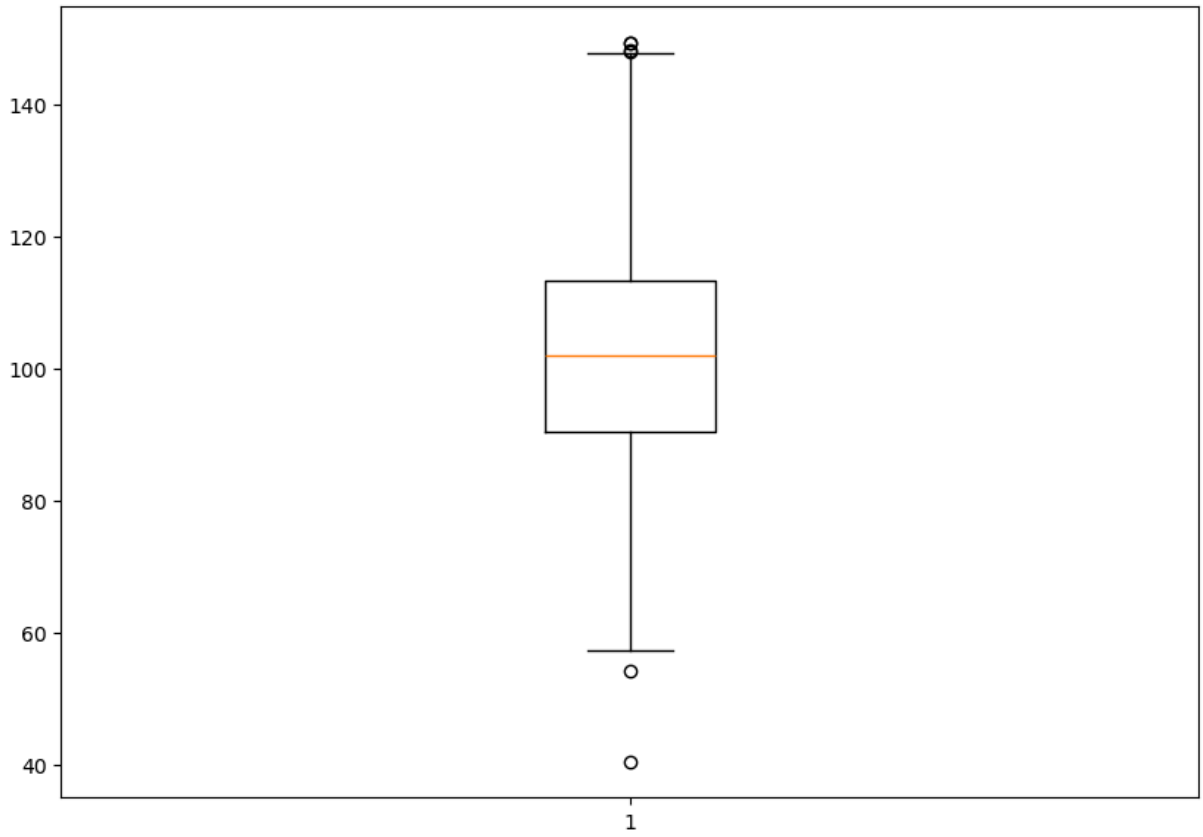
09) WAP create a Scatter Plot with several colors in Matplotlib?

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 [ ]:

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 13

Smit Maru - 23010101161 - 260

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 [7]:
```
class Students:
    def __init__(self,name,age,grade):
        self.name = name
        self.age = age
        self.grade = grade

stu = Students("Smit",20,"A+")

print("Name =",stu.name)
print("Age =",stu.age)
print("Grade =",stu.grade)
```

```
Name = Smit
Age = 20
Grade = A+
```

02) Create a class named Bank_Account with Account_No, User_Name, Email,Account_Type and Account_Balance data members. Also create a method GetAccountDetails() and DisplayAccountDetails(). Create main method to demonstrate the Bank_Account class.

In [11]:
```
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):
        return {
            "Account_No" : self.Account_No,
            "User_Name" : self.User_Name,
            "Email" : self.Email,
            "Account_Type" : self.Account_Type,
            "Account_Balance" : self.Account_Balance
        }

    def DisplayAccountDetails(self):
        Account_Details = self.GetAccountDetails()
        for key, value in Account_Details.items():
            print(f"{key} : {value}")

Account = Bank_Account(123456789, "Smit", "smit@gmail.com", "Current",100000000)
Account.DisplayAccountDetails()
```

```
Account_No : 123456789
User_Name : Smit
Email : smit@gmail.com
Account_Type : Current
Account_Balance : 100000000
```

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

In [ ]:
```
import math

class circle:
    def __init__(self,radius):
        self.radius = radius

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

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

circle = Circle(1)
print("Area :",circle.area())
print("Perimeter :",circle.perimeter())
```

```
Area : 3.141592653589793
Perimeter : 6.283185307179586
```

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

In [23]:
```
class Employee:
    def __init__(self, name, age, salary):
        self.name = name
        self.age = age
        self.salary = salary

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

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

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

employee = Employee("Smit",20, 100000)
employee.display_information()

employee.update_salary(150000)
employee.update_age(22)
employee.display_information()
```

```
Name = Smit
Age = 20
Salary = 100000
Name = Smit
Age = 22
Salary = 150000
```

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

In [26]:
```
class BankAccount:
    def __init__(self, initial_balance=0):
        self.balance = initial_balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited {amount}. New balance is {self.balance}.")
        else:
            print("Enter Positive Deposit Amount.")

    def withdraw(self, amount):
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew {amount}. New balance is {self.balance}.")
```

```
        else:
            print("Invalid withdrawal amount or insufficient balance.")

    def check_balance(self):
        print(f"Your current balance is {self.balance}.")
```

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 [ ]:
```
class InventoryItem:
    def __init__(self, name, price, quantity):
        self.name = name
        self.price = price
        self.quantity = quantity

    def add(self, amount):
        self.quantity += amount

    def remove(self, amount):
        if amount <= self.quantity:
            self.quantity -= amount
        else:
            print("Not enough items to remove.")

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

    def __str__(self):
        return f"Item: {self.name}, Price: {self.price}, Quantity: {self.quantity}"
```

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

In [ ]:
```
class Dog:
    def __init__(self, name, breed, age):
        self.name = name
        self.breed = breed
        self.age = age

my_dog = Dog("Lucky", "Golden Retriever", 3)

print(my_dog.name)
print(my_dog.breed)
print(my_dog.age)
```

```
Lucky
Golden Retriever
3
```

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 [ ]:
```
class student_kit:
    principal_name = "Mr ABC"

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

    def attendance(self, days):
        self.days_present = days

    def create_certificate(self):
        return f"Certificate of {self.student_name}: Present for {self.days_present
```

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

In [33]:
```
class Time:
    def __init__(self, hour, minute):
        self.hour = hour
        self.minute = minute

    def add(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)


time1 = Time(1, 30)
time2 = Time(2, 45)
result = time1.add(time2)
print(f"Total Time: {result.hour} hours and {result.minute} minutes")
```

```
Total Time: 4 hours and 15 minutes
```

Python Programming - 2301CS404

Lab - 13

Smit Maru - 23010101161 - 260

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 [23]:
```
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("namra", 30, 50000)

emp.display()
```

```
Name: namra, 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 [ ]: