Python Programming - 2301CS404 Lab - 1 Dadhaniya Parth 24010101621

01) WAP to print “Hello World”

```
print("Hello World")
```

```
Hello World
```

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

```
# without input function
n1 = 30
n2 = 20
ans = n1 + n2
print("Addition is:",ans)
```

```
Addition is: 50
```

```
# with input function
n1 = int(input("Enter 1st Number:"))
n2 = int(input("Enter 2nd Number:"))
ans = n1 + n2
print("Addition is:",ans)
```

```
Enter 1st Number: 12
Enter 2nd Number: 12

Addition is: 24
```

03) WAP to check the type of the variable.

```
a = "Parth"
b = 563
print(type(a))
print(a)
print(type(b))
print(b)
```

```
<class 'str'>
Parth
<class 'int'>
563
```

04) WAP to calculate simple interest.

```
p = int(input("Enter principle:"))
t = int(input("Enter time period:"))
r = int(input("Enter rate of interest:"))
print('The principal is', p)
print('The time period is', t)
print('The rate of interest is',r)
si = (p * t * r)/100
print("Simple interest is : ",si)

Enter principle: 10000
Enter time period: 5
Enter rate of interest: 5

The principal is 10000
The time period is 5
The rate of interest is 5
Simple interest is :  2500.0
```

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

```
radius = float(input("Enter the radius of the circle:"))
area = 3.14 * radius * radius
print("Area of circle is:",area)

Enter the radius of the circle: 23

Area of circle is: 1661.06
```

06) WAP to calculate area of a triangle.

```
s1 = int(input("Enter 1st side:"))
s2 = int(input("Enter 2nd side:"))
s3 = int(input("Enter 3rd side:"))
s = (s1+s2+s3) / 2
area = ((s*(s-s1)*(s-s2)*(s-s3))**0.5)
print("The area of the triangle is: ",area)

Enter 1st side: 7
Enter 2nd side: 8
Enter 3rd side: 9

The area of the triangle is:  26.832815729997478
```

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

```
celsius = float(input("Enter degree in celsius:"))
fahrenheit = (celsius * 1.8) + 32
print("Fahrenheit is:",fahrenheit)

Enter degree in celsius: 40

Fahrenheit is: 104.0
```

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

```
x1 = float(input("Enter x-coordinate of the first point: "))
y1 = float(input("Enter y-coordinate of the first point: "))

x2 = float(input("Enter x-coordinate of the second point: "))
y2 = float(input("Enter y-coordinate of the second point: "))

distance = ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5

print(f"The distance between the two points is: {distance}")
```

```
Enter x-coordinate of the first point:  3
Enter y-coordinate of the first point:  4
Enter x-coordinate of the second point:  7
Enter y-coordinate of the second point:  1

The distance between the two points is: 5.0
```

10) WAP to print sum of n natural numbers.

```
n = int(input("Enter a positive integer: "))

if n < 1:
    print("Please enter a positive integer greater than 0.")
else:
    total = n * (n + 1) // 2

print(f"The sum of the first {n} natural numbers is: {total}")
```

```
Enter a positive integer:  6

The sum of the first 6 natural numbers is: 21
```

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

```
n = int(input("Enter a positive integer: "))

if n < 1:
    print("Please enter a positive integer greater than 0.")
else:
    sum_of_squares = n * (n + 1) * (2 * n + 1) // 6
print(f"The sum of squares of the first {n} natural numbers is:
{sum_of_squares}")
```

```
Enter a positive integer:  5

The sum of squares of the first 5 natural numbers is: 55
```

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

```
fname = input("Enter First Name:")
lname = input("Enter Second Name:")
print("Full Name is:",fname + lname)

Enter First Name: Parth
Enter Second Name: Patel

Full Name is: ParthPatel
```

13) WAP to swap two numbers.

```
num1 = int(input("Enter the first number (a): "))
num2 = int(input("Enter the second number (b): "))

print(f"Before swapping: num1 = {num1}, num2 = {num2}")

temp = num1
num1 = num2
num2 = temp

print(f"After swapping: num1 = {num1}, num2 = {num2}")

Enter the first number (a):  63
Enter the second number (b):  89

Before swapping: num1 = 63, num2 = 89
After swapping: num1 = 89, num2 = 63
```

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

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

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

print(f"The distance in meters is: {meters:.2f} m")
print(f"The distance in feet is: {feet:.2f} ft")
print(f"The distance in inches is: {inches:.2f} in")
print(f"The distance in centimeters is: {centimeters:.2f} cm")

Enter the distance in kilometers:  1.5

The distance in meters is: 1500.00 m
The distance in feet is: 4921.26 ft
The distance in inches is: 59055.15 in
The distance in centimeters is: 150000.00 cm
```

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

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

```
Enter day: 16
Enter month: 11
Enter year: 2005

16-11-2005
```

if..else..

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

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

if(n>0):
    print("Number is positive")
elif(n<0):
    print("Number is negative")
else:
    print("Number is Zero")
```

```
Enter number: 8

Number is positive
```

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

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

if(n%2==0):
    print("Number is Even")
else:
    print("Number is Odd")
```

```
Enter number: 5

Number is Odd
```

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

```
# using simple if

n1 = int(input("Enter 1st Number:"))
n2 = int(input("Enter 2nd Number:"))

if(n1>n2):
    print(f"{n1} is Largest Number")
else:
    print(f"{n2} is Largest Number")
```

```
Enter 1st Number: 20
Enter 2nd Number: 15

20 is Largest Number
```

```python
#using ternary operator

n1 = int(input("Enter 1st Number:"))
n2 = int(input("Enter 2nd Number:"))

large = f"{n1} is Largest Number" if(n1>n2) else f"{n2} is Largest 
Number"
print(large)
```

```
Enter 1st Number: 27
Enter 2nd Number: 15

27 is Largest Number
```

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

```python
n1 = int(input("Enter 1st Number:"))
n2 = int(input("Enter 2nd Number:"))
n3 = int(input("Enter 3rd Number:"))

largest = n1 if (n1 > n2 and n1 > n3) else (n2 if n2 > n3 else n3)

print(f"The largest number is: {largest}")
```

```
Enter 1st Number: 12
Enter 2nd Number: 45
Enter 3rd Number: 32

The largest number is: 45
```

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]

```python
year = int(input("Enter Year:"))

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

```
Enter Year: 2024

2024 is Leap
```

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

```
day = int(input("Enter Number between 1-7:"))

match day:
    case 1:
        print("Sunday")
    case 2:
        print("Monday")
    case 3:
        print("Tuesday")
    case 4:
        print("Wednesday")
    case 5:
        print("Thrusday")
    case 6:
        print("Friday")
    case 7:
        print("Saturday")
    case _:
        print("Invalid Input")
```

```
Enter Number between 1-7: 8

Invalid Input
```

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

```
n1 = int(input("Enter 1st Number: "))
n2 = int(input("Enter 2nd Number:"))

cal = input("Enter +,-,* or /:")

match cal:
    case '+':
        print(f"{n1}+{n2} = {n1+n2}")
    case '-':
        print(f"{n1}-{n2} = {n1-n2}")
    case '*':
        print(f"{n1}*{n2} = {n1*n2}")
    case '/':
        print(f"{n1}/{n2} = {n1/n2}")
    case _:
        print("Invalid Choice")
```

```
Enter 1st Number:  12
Enter 2nd Number: 12
Enter +,-,* or /: +
12+12 = 24
```

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

```
s1 = int(input("Enter 1st Subject Marks:"))
s2 = int(input("Enter 2nd Subject Marks:"))
s3 = int(input("Enter 3rd Subject Marks:"))
s4 = int(input("Enter 4th Subject Marks:"))
s5 = int(input("Enter 5th Subject Marks:"))
total = s1+s2+s3+s4+s5
print("Total Marks:",total)
percentage = float(total)*(100/500)
print("Percentage is:",percentage)
if(percentage<35):
    print("Fail")
elif(percentage>=35 and percentage<45):
    print("Pass Class")
elif(percentage>=45 and percentage<60):
    print("Second Class")
elif(percentage>=60 and percentage<70):
    print("First Class")
else:
    print("Distinction")
```

```
Enter 1st Subject Marks: 45
Enter 2nd Subject Marks: 89
Enter 3rd Subject Marks: 65
Enter 4th Subject Marks: 98
Enter 5th Subject Marks: 56

Total Marks: 353
Percentage is: 70.60000000000001
Distinction
```

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

```
side1 = float(input("Enter 1st Side:"))
side2 = float(input("Enter 2nd Side:"))
side3 = float(input("Enter 3rd Side:"))

if (side1==side2==side3):
    print("Euilateral Triangle")
```

```
elif(side1==side2 or side1==side3 or side2==side3):
    print("Isosceles Triangle")
else:
    print("Scaler Triangle")
```

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

```
num1 = int(input("Enter 1st Number:"))
num2 = int(input("Enter 2nd Number:"))
num3 = int(input("Enter 3rd Number:"))

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

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

```
Enter 1st Number: 12
Enter 2nd Number: 7
Enter 3rd Number: 9

The second largest number is: 9
```

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

```
unit=int(input("Enter unit:"))
if unit>=1 and unit<=50:
      print(unit*2.60)
elif unit>50 and unit<=100:
      a=50*2.60
      b=unit-50
      c=b*3.25
      print(a+c)
elif unit>100 and unit<=200:
      a=50*2.60
      b=50*3.25
```

```
     c=unit-100
     d=c*5.26
     print(a+b+d)
elif unit>200:
     a=50*2.60
     b=50*3.25
     c=100*5.26
     d=unit-200
     e=d*8.45
     print(a+b+c+e)
```

```
Enter unit: 250
```

```
1241.0
```

for and while loop

01) WAP to print 1 to 10.

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

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

02) WAP to print 1 to n.

```
n=int(input("Enter Number:"))

for i in range(1,n+1):
    print(i)
```

```
Enter Number: 5

1
2
3
4
5
```

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

```
n = int(input("Enter Number:"))

for i in range(1,n+1):
    if(i % 2 !=0):
        print(i)
```

```
Enter Number: 5

1
3
5
```

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

```
n1 = int(input("Enter 1st Number:"))
n2 = int(input("Enter 2nd Number:"))

for i in range(n1, n2+1):
    if(i%2==0 and i%3!=0):
        print(i)
```

```
Enter 1st Number: 1
Enter 2nd Number: 20

2
4
8
10
14
16
20
```

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

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

```
Enter Number: 10

55
```

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

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

```
Enter number: 7

140
```

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

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

Enter number: 7

4
```

08) WAP to print multiplication table of given number.

```
n = int(input("Enter no:"))
for i in range(1,11):
     print(n,"*",i,"=",n*i)

Enter no: 2

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

09) WAP to find factorial of the given number.

```
n = int(input("Enter Number:"))
fact = 1

for i in range(1,n+1):
    fact = fact * i
print(fact)

Enter Number: 5

120
```

10) WAP to find factors of the given number.

```
n = int(input("Enter Number:"))

for i in range(1,n+1):
    if n%i==0:
        print(i)

Enter Number: 6
```

```
1
2
3
6
```

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

```
n = int(input("Enter Number:"))

for count in range(2,n):
    if n%count==0:
        print("Number is not prime")
        break
    else:
        print("Number is Prime")
        break
```

```
Enter Number: 6

Number is not prime
```

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

```
no = int(input("Enter Number:"))
sum = 0
while(no>0):
    i = no % 10
    sum = sum+i
    no = no//10
print(sum)
```

```
Enter Number: 154

10
```

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

```
n = int(input("Enter Number:"))
a = n
sum = 0
while(n>0):
    i = n%10
    sum = sum*10+i
    n = n//10
if(a == sum):
    print("Number is Palindrome")
else:
    print("Number is Not Palindrome")
```

```
Enter Number: 151
```

```
Number is Palindrome
```

14) WAP to print GCD of given two numbers.

```
n1 = int(input("Enter 1st Number:"))
n2 = int(input("Enter 2nd Number:"))
min = 0
if(n1<n2):
    min = n1
else:
    min = n2
for i in range(1,min+1):
    if n1%i==0 and n2%i==0:
        gcd = i
print("gcd of:",n1,"and",n2,"is:",gcd)
```

```
Enter 1st Number: 60
Enter 2nd Number: 36

gcd of: 60 and 36 is: 12
```

String

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

```
str1 = input("Enter string here:")
str2 = str1[::-1]
if(str1 == str2):
      print("pallindrome")
else:
      print("not pallindrome")
```

```
Enter string here: aba

pallindrome
```

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

```
str1 = input("Enter string here:")
words = str1.split()
reverse_word = words[::-1]
reverse_string = ' '.join(reverse_word)
print(reverse_string)
```

```
Enter string here: Parth Patel

Patel Parth
```

03) WAP to remove ith character from given string.

```
str1 = input("Enter string here:")
i = int(input("Enter index for remove a character:"))
str2 = str1[:i]+str1[i+1:]
print(str2)
```

```
Enter string here: kaya
Enter index for remove a character: 1

kya
```

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

```
str1 = input("Enter string here:")
length = 0
for i in str1:
    length += 1
print(length)
```

```
Enter string here: Parth Dadhaniya

15
```

05) WAP to print even length word in string.

```
str1 = input("Enter string here:")
str2 = str1.split(' ')
print(str2)
for i in str2:
    if len(i) % 2 == 0:
        print(i)

Enter string here: Parth Dadhaniya once

['Parth', 'Dadhaniya', 'once']
once
```

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

```
str1 = input("Enter string here:")
count = 0
vowels = "aeiouAEIOU"
for i in str1:
    if i in vowels:
        count += 1
print(count)

Enter string here: Parth

1
```

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

```
str1 = input("Enter string here:")
words = str1.split()
result = []

for i in words:
    if len(i) > 1:
        i = i[0].upper() + i[1:-1] + i[-1].upper()
    else:
        i = i.upper()
    result.append(i)

cap = ' '.join(result)
print(cap)

Enter string here: parth dadhaniya
```

```
PartH DadhaniyA
```

08) WAP to convert given array to string.

```
array = ['I','am','Parth','Student','of','B-tech']
s = ' '.join(array)
print("Array is:",array)
print("String is:",s)

Array is: ['I', 'am', 'Parth', 'Student', 'of', 'B-tech']
String is: I am Parth Student of B-tech
```

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

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

```
password = input("Enter Password:")
conpassword = input("Reenter Password:")
if password == conpassword:
    print("Password Matched")
elif password.lower() == conpassword:
    print("Password do not match. This issue seems to be sensitivity")
else:
    print("Password do not match")

Enter Password: Parth
Reenter Password: parth

Password do not match. This issue seems to be sensitivity
```

10) : Display credit card number.

card no. : 1234 5678 9012 3456

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

```
cno = input("Enter credit card number:")
result = '**** **** **** ' + cno[-4:]
print("Updated Number is:",result)

Enter credit card number: 1456 7869 4356 1611

Updated Number is: **** **** **** 1611
```

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

s1 = decimal and s2 = medical are Anagram

```
s1 = input("Enter 1st word:")
s2 = input("Enter 2nd word:")
```

```
print(sorted(s1))
print(sorted(s2))

if sorted(s1) == sorted(s2):
    print("Strings are Anagram")
else:
    print("Strings are not Anagram")
```

```
Enter 1st word: decimal
Enter 2nd word: medical

['a', 'c', 'd', 'e', 'i', 'l', 'm']
['a', 'c', 'd', 'e', 'i', 'l', 'm']
Strings are Anagram
```

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

input : EHlsarwiwhtwMV

output : lsarwiwhtwEHMV

```
s1 = input("Enter string here:")
lowercase = "".join(sorted([char for char in s1 if char.islower()]))
uppercase = "".join(sorted([char for char in s1 if char.isupper()]))
result = lowercase + uppercase
print(result)
```

```
Enter string here: EHlsarwiwhtwMV

ahilrstwwwEHMV
```

List

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

```
n = int(input("Enter size of list:"))
l1 = []
sum = 0
for i in range(1,n+1):
    x = int(input("Enter elements:"))
    l1.append(x)
    sum = sum+x
print(l1)
print(sum)
```

```
Enter size of list: 5
Enter elements: 1
Enter elements: 2
Enter elements: 3
Enter elements: 4
Enter elements: 5

[1, 2, 3, 4, 5]
15
```

02) WAP to find largest element in a List.

```
n = int(input("Enter size of list:"))
l2 = []
for i in range(1,n+1):
     x=int(input("Enter elements:"))
     l2.append(x)
print("List is:",l2)
print("Maximum value is:",max(l2))
```

```
Enter size of list: 4
Enter elements: 34
Enter elements: 87
Enter elements: 9076
Enter elements: 54

List is: [34, 87, 9076, 54]
Maximum value is: 9076
```

03) WAP to find the length of a List.

```python
n = int(input("Enter size of list:"))
l3 = []
for i in range(1,n+1):
     x = int(input("Enter elements:"))
     l3.append(x)
print("List is:",l3)
length = len(l3)
print('Length Of List is:',length)
```

```
Enter size of list: 3
Enter elements: 34
Enter elements: 5
Enter elements: 7

List is: [34, 5, 7]
Length Of List is: 3
```

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

```python
n = int(input("Enter size of list:"))
l4 = []
for i in range(1,n+1):
     x = int(input("Enter elements:"))
     l4.append(x)
print("List is:",l4)
if len(l4) >= 2:
    l4[0],l4[-1] = l4[-1], l4[0]
print(l4)
```

```
Enter size of list: 5
Enter elements: 1
Enter elements: 2
Enter elements: 34
Enter elements: 5
Enter elements: 6

List is: [1, 2, 34, 5, 6]
[6, 2, 34, 5, 1]
```

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

```python
n = int(input("Enter size of list:"))
a = []
b = []
print("Elements are:")
for i in range(n):
     x = int(input())
```

```
      a.append(x)
part = len(a)//2
for i in range(part,len(a)):
      b.append(a[i])
for i in range(0,part):
      b.append(a[i])
print(b)
```

```
Enter size of list: 5

Elements are:

 1
 2
 3
 4
 5

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

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

```
n = int(input("Enter size of list:"))
a = []
b = []
print("Elements are:")
for i in range(n):
      x = int(input())
      a.append(x)
print("Enter two positions for interchange:")
n = int(input())
m = int(input())
b = a[n-1]
a[n-1] = a[m-1]
a[m-1] = b
print(a)
```

```
Enter size of list: 5

Elements are:

 1
 2
 3
 4
 5

Enter two positions for interchange:
 2
 5

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

07) WAP to reverse the list entered by user.

```
n = int(input("Enter size of list:"))
l7 = []
print("Elements are:")
for i in range(n):
      x = int(input())
      l7.append(x)
print("List is:",l7)
reverse = l7[::-1]
print("Reversed List is:",reverse)
```

```
Enter size of list: 4

Elements are:

 21
 95
 63
 89

List is: [21, 95, 63, 89]
Reversed List is: [89, 63, 95, 21]
```

08) WAP to print even numbers in a list.

```
n = int(input("Enter size of list:"))
l8 = []
print("Elements are:")
for i in range(n):
      x = int(input())
      l8.append(x)
print("List is:",l8)

for i in l8:
    if i % 2 == 0:
        print(i)
```

```
Enter size of list: 7

Elements are:

 21
 34
 56
 76
 89
 56
 11

List is: [21, 34, 56, 76, 89, 56, 11]
34
56
76
56
```

09) WAP to count unique items in a list.

```
n = int(input("Enter size of list:"))
l9 = []
print("Elements are:")
for i in range(n):
     x = int(input())
     l9.append(x)
print("List is:",l9)

print(len(set(l9)))

Enter size of list: 3

Elements are:

 1
 2
 2

List is: [1, 2, 2]
2
```

10) WAP to copy a list.

```
n = int(input("Enter size of list:"))
l10 = []
print("Elements are:")
for i in range(n):
     x = int(input())
     l10.append(x)
print("List is:",l10)

copied = l10.copy()
print("Copy is List is:",copied)

Enter size of list: 5

Elements are:
 1
 2
 3
 4
 5

List is: [1, 2, 3, 4, 5]
Copy is List is: [1, 2, 3, 4, 5]
```

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

```
n1 = int(input("Enter Starting:"))
n2 = int(input("Enter Ending:"))

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

Enter Starting: 1
Enter Ending: 15

1
3
5
7
9
11
13
15
```

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

```
n = int(input("Enter size of list:"))
l12 = []
print("Elements are:")
for i in range(n):
      x = int(input())
      l12.append(x)
print("List is:",l12)
oc = int(input("Enter value for count"))
count = l12.count(oc)
print(f"Count of {oc} is: {count}")

Enter size of list: 5

Elements are:

 1
 1
 2
 4
 1

List is: [1, 1, 2, 4, 1]

Enter value for count 1

Count of 1 is: 3
```

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

```
n = int(input("Enter size of list:"))
l13 = []
print("Elements are:")
for i in range(n):
      x = int(input())
      l13.append(x)
print("List is:",l13)

sl = sorted(set(l13))[-2]
print("Second Largest Number of List is:",sl)

Enter size of list: 5

Elements are:

 1
 2
 3
 4
 5

List is: [1, 2, 3, 4, 5]
Second Largest Number of List is: 4
```

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

```
n = int(input("Enter size of list: "))
l14 = []

print("Elements are:")
for i in range(n):
    x = int(input())
    l14.append(x)
print("List is:", l14)

k = int(input("Enter key: "))

frequency = {}

for element in l14:
    frequency[element] = frequency.get(element, 0) + 1

print(f"\nElements appearing more than {k} times:")
found = False

for element, count in frequency.items():
    if count > k:
        print(f"Element {element} appears {count} times")
        found = True

if not found:
    print(f"No elements appear more than {k} times")
```

```
Enter size of list:  6

Elements are:

 1
 2
 2
 3
 2
 1

List is: [1, 2, 2, 3, 2, 1]

Enter key:  2


Elements appearing more than 2 times:
Element 2 appears 3 times
```

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

```
#
squares = []
for i in range(10):
    squares.append(i * i)
print("Squares using for loop:", squares)
```

```
Squares using for loop: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
# Method 2: Using List Comprehension
squares_comp = [i * i for i in range(10)]
print("Squares using list comprehension:", squares_comp)
```

```
Squares using list comprehension: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

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

```
n = int(input("Enter the number of fruits: "))

fruits = []

print("Enter the fruits:")
for i in range(n):
    fruit = input()
    fruits.append(fruit)

b_fruits = [i for i in fruits if i.lower().startswith('b')]
print("\nOriginal list:", fruits)
print("Fruits starting with 'b':", b_fruits)
```

```
Enter the number of fruits:  6

Enter the fruits:

 apple
 blueberry
 banana
 berry
 grape
 mango


Original list: ['apple', 'blueberry', 'banana ', 'berry', 'grape',
'mango']
Fruits starting with 'b': ['blueberry', 'banana ', 'berry']
```

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

```
n = int(input("Enter size of both list:"))

list1 = []
for i in range(1,n+1):
     x = int(input("Enter elements of First List:"))
     list1.append(x)
print("First List is:",list1)

list2 = []
for i in range(1,n+1):
     x = int(input("Enter elements of Second List:"))
     list2.append(x)
print("Second List is:",list2)

common_elements = [element for element in list1 if element in list2]

print("Common elements:", common_elements)
```

```
Enter size of both list: 6
Enter elements of First List: 1
Enter elements of First List: 2
Enter elements of First List: 3
Enter elements of First List: 4
Enter elements of First List: 5
Enter elements of First List: 6

First List is: [1, 2, 3, 4, 5, 6]

Enter elements of Second List: 6
Enter elements of Second List: 7
Enter elements of Second List: 8
Enter elements of Second List: 9
Enter elements of Second List: 0
Enter elements of Second List: 1

Second List is: [6, 7, 8, 9, 0, 1]
Common elements: [1, 6]
```

Tuple

01) WAP to find sum of tuple elements.

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

```
21
```

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

```
t2 = (99,87,23,41,56)
max = max(t2)
min = min(t2)
print("Maximum is:",max)
print("Minimum is:",min)
```

```
Maximum is: 99
Minimum is: 23
```

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

```
data = [(6, 12, 18), (4, 9, 15), (10, 20, 30), (8, 16, 24)]

K = int(input("Enter Key:"))

result = []

for tup in data:
    for elem in tup:
        if elem % K != 0:
            break
    else:
        result.append(tup)

print(result)
```

```
Enter Key: 5
```

```
[(10, 20, 30)]
```

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

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

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

print(result)
```

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

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

```
tuples_list = [(1, 2, 3), (-1, 2, 3), (4, 5, 6), (0, 7, 8)]

result = [t for t in tuples_list if all(x > 0 for x in t)]

print(result)
```

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

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

```
my_list = [1, 2, 3]
my_tuple = (4, 5)

result_list = my_list + list(my_tuple)

result_tuple = my_tuple + tuple(my_list)

print(result_list)
print(result_tuple)
```

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

07) WAP to remove tuples of length K.

```
tuples_list = [(1, 2, 3), (-1, 2, 3), (4, 5, 6), (7, 8)]

k=int(input("Enter key:"))

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

print(result)
```

```
Enter key: 2

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

08) WAP to remove duplicates from tuple.

```
t1 = (12,1,1,12,14)
s = set(t1)
t = tuple(s)
print("Result is:",t)
```

```
Result is: (1, 12, 14)
```

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

```
my_tuple = (1, 2, 3, 4, 5)
result = []

for i in range(len(my_tuple) - 1):
    result.append(my_tuple[i] * my_tuple[i + 1])

result_tuple = tuple(result)

print(result_tuple)
```

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

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

```
t1 = (1,2,3,4,5,6)
l_t = len(t1)
s = set(t1)
l_s = len(s)
if l_t == l_s:
    print("Tuple is Distinct")
else:
    print("Tuple is not Distinct")
```

```
Tuple is Distinct
```

Set & Dictionary

01) WAP to iterate over a set.

```
s1 = {'hello',45,89.23,45}
for i in s1:
    print(i)
```

```
89.23
45
hello
```

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

```
my_set = {1, 2, 3, 4, 5}

set_to_list = list(my_set)
print("Set converted to List:", set_to_list)

set_to_string = ''.join(map(str, my_set))
print("Set converted to String:", set_to_string)

set_to_tuple = tuple(my_set)
print("Set converted to Tuple:", set_to_tuple)
```

```
Set converted to List: [1, 2, 3, 4, 5]
Set converted to String: 12345
Set converted to Tuple: (1, 2, 3, 4, 5)
```

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

```
my_set = {10, 25, 5, 80, 15}

max_value = max(my_set)
min_value = min(my_set)

print("Maximum value in the set:", max_value)
print("Minimum value in the set:", min_value)
```

```
Maximum value in the set: 80
Minimum value in the set: 5
```

04) WAP to perform union of two sets.

```
s1 = {10,20,30,40,50}
s2 = {10,20,60,70,80}
```

```
uni = s1.union(s2)
print(uni)
```

```
{70, 40, 10, 80, 50, 20, 60, 30}
```

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

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

common = set(list1) & set(list2)

if common:
    print("The lists have at least one common element:", common)
else:
    print("The lists have no common elements.")
```

```
The lists have at least one common element: {5}
```

06) WAP to remove duplicates from list.

```
list1 = [12,12,24,36,48,48,60]
result = set(list1)
print(result)
```

```
{36, 12, 48, 24, 60}
```

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

```
text = "This is a test string with test words and unique words"

words = text.split()

unique_words = set(words)

print("Unique words in the string:", unique_words)
```

```
Unique words in the string: {'is', 'words', 'a', 'string', 'and', 
'This', 'with', 'test', 'unique'}
```

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

```
set1 = {1,2,3,4,5,6,7}
set2 = {1,2,3,9,7}
result = set1-set1.intersection(set2)
print(result)
```

```
{4, 5, 6}
```

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

```
str1 = "medical"
str2 = "decimal"
s1 = set(str1)
s2 = set(str2)
dict1 = {}
dict2 = {}

for i in s1:
    dict1[i] = str1.count(i)
for i in s2:
    dict2[i] = str2.count(i)

if dict1==dict2:
    print("String is anagram")
else:
    print("String is not anagram")
```

```
String is anagram
```

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

```
l1 = {1,2,3,4,5,6}
l2 = {1,2,3,4,7,8}
l3 = {1,2,3,9,10,11}
result = set(l1&l2&l3)
print(result)
```

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

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

```
input_string = input("Enter a string: ")

vowels = {'a', 'e', 'i', 'o', 'u'}

input_string = input_string.lower()

count = 0

for char in input_string:
    if char in vowels:
        count += 1

print(count)
```

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

```
input_string = input("Enter a string: ")

if all(char in {'0', '1'} for char in input_string):
    print("The string is a binary string.")
else:
    print("The string is not a binary string.")
```

```
Enter a string:  101010
```

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

13) WAP to sort dictionary by key or value.

```
dic = {'a':56,'c':78,'b':67}
result = sorted(dic.items(), key = lambda kv : (kv[1],kv[0]))
print(result)
```

```
[('a', 56), ('b', 67), ('c', 78)]
```

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

```
d1 = {'a': 5, 'b': 8, 'c': 2}
sum = 0
for i in d1.values():
    sum += i
print(sum)
```

```
15
```

15) WAP to handle missing keys in dictionaries.

Example : Given, dict1 = {'a': 5, 'c': 8, 'e': 2}

if you look for key = 'd', the message given should be 'Key Not Found', otherwise print the value of 'd' in dict1.

```
d1 = {'a': 5, 'b': 8, 'c': 2}
key = input("Enter key:")
d1.get(key,"Key Not Found")
```

```
Enter key: a
```

```
5
```

User Defined Function

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

```
h=float(input("Enter your height in M:"))
w=float(input("Enter your weight in K6:"))
def calculateBMI(h,b):
    bmi = b / (h ** 2)
    return bmi
print(f"Your BMI is: {calculateBMI(h,w)}")
```

```
Your BMI is: 19.568236333796236
```

02) Write a function that add first n numbers.

```
n = int(input("Enter N:"))
def sumOfN(n):
    sum=0
    while n>0:
        sum = sum + n
        n = n - 1
    return sum
print(f"Sum of first n number is: {sumOfN(n)}")
```

```
Sum of first n number is: 465
```

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

```
def isprime(n):
    for i in range(2,n):
        if n % i == 0:
            return 0
        else:
            return 1
isprime(17)
```

```
1
```

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

```
def listOfPrime(n,m):
    primes = []
    for i in range(n, m + 1):
        for j in range(2, i):
            if i % j == 0:
                break
            else:
                primes.append(i)
    return primes
listOfPrime(1,10)

[3, 5, 5, 5, 7, 7, 7, 7, 7, 9]
```

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

```
def ispalindrome(s):
    if s == s[::-1]:
        return True
    else:
        return False
ispalindrome("abc")

False
```

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

```
def sumOfList(l):
    sum = 0
    for i in l:
        sum = sum+i
    return sum
sumOfList([5,4,3,2,1])

15
```

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

```
def sumOfFirstElement(l):
    sum = 0
    for i in l:
        sum = sum+i[0]
    return sum

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

5
```

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

```
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)

print(fibonacci(10))
```

```
55
```

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

```
def getname(d, rollno):
    return d.get(rollno)

getname({101:'Ajay', 102:'Rahul', 103:'Jay', 104:'Pooja'},103)
```

```
'Jay'
```

10) Write a function to get the sum of the scores ending with zero.

Example : scores = [200, 456, 300, 100, 234, 678]

Ans = 200 + 300 + 100 = 600

```
def sumOfScore(l):
    sum=0
    for i in l:
        if i % 10 == 0:
            sum+=i
    return sum

sumOfScore([200,456,300,100,234,678])
```

```
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'}

```
def invertDictionary(d):
    id = {}
    for k, v in d.items():
        id.update({v:k})
    return id

print(invertDictionary({'a': 10, 'b': 20, 'c': 30, 'd' : 40}))

{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.

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

```
def countUpperAndLower(s):
    upper = 0
    lower = 0
    for i in s:
        if i.isupper():
            upper = upper+1
        elif i.islower():
            lower = lower+1
    return upper, lower

countUpperAndLower("AbcDEfgh")

(3, 5)
```

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

```
min = lambda a,b : a if a<b else b
min(53,10)

10
```

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

```
studentName = filter(lambda x : len(x) > 7,
["zxxczc","abcefghijk","lmnopwq"])
```

```
for i in studentName:
    print(i)

abcefghijk
```

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

```
studentName = map(lambda x : x.capitalize() ,
["zxxczc","abcefghijk","lmnopwq"])
for i in studentName:
    print(i)

Zxxczc
Abcefghijk
Lmnopwq
```

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

```
def positional(a,b):
    return a+b

print(positional(5,10))
#-------------------------
def keyword(a,b):
    return a+b

print(keyword(a=5,b=10))
#-------------------------
def default(a=0,b=0):
    return a+b

default()
#-------------------------
def variableLenPos(*args):
    sum=0
    for i in args:
        sum+=i
    return sum

print(variableLenPos(10,20,30))
#-------------------------
def lenKeyword(**kyword):
    return kyword
```

```
print(lenKeyword(a=10,b=20,c=30))
#-------------------------
def positionalOnly(a,b,/):
    return a+b

print(positionalOnly(10,20))
#-------------------------
def keywordOnly(*,a,b):
    return a+b

print(keywordOnly(a=10,b=20))
```

```
15
15
60
{'a': 10, 'b': 20, 'c': 30}
30
30
```

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

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

```
# same directory
fp = open("demo.txt","r")
data = fp.read()
print("File Data is:",data)
fp.close()
```

```
File Data is: dadhaniya Parth
```

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

```
# different directory
fp = open("D:\\B-Tech\\Semester-4\\PP\\demo1.txt")
print("Data is:",fp.read())
fp.close()
```

```
Data is: Dadhaniya Parth
```

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

```
# line by line
fp = open("demo.txt","r")
line1 = fp.readline()
print(line1, end="")
line2 = fp.readline()
print(line2, end="")
line3 = fp.readline()
print(line3, end="")
fp.close()
```

```
Dadhaniya Parth
Student of Darshan University
Full Stack Developer
```

```
# in the form of a list
fp = open("demo.txt","r")
li = fp.readlines()
print(li)
fp.close()
```

```
['Dadhaniya Parth\n', 'Student of Darshan University\n', 'Full Stack
Developer']
```

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

```
file = open("demo.txt", 'r')

num_lines = 0
num_words = 0
num_chars = 0

for line in file:
    num_lines += 1
    num_words += len(line.split())
    num_chars += len(line)

file.close()

print(num_lines)
print(num_words)
print(num_chars)
```

```
3
9
66
```

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

```
sf = open('demo.txt', 'r')
df = open('demo2.txt', 'w')

df.write(sf.read())

sf.close()
df.close()
```

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

```
fp = open('demo.txt', 'r')
fp.seek(0, 2)
size = fp.tell()
fp.close()
print("File size is:",size,"bytes")
```

```
File size is: 66 bytes
```

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

```python
def frequency(fp, word):
    count = 0
    with open(fp, 'r') as file:
        for line in file:
            count += line.lower().split().count(word.lower())
    return count

word = input("Enter the word to count: ")

word_count = frequency("demo.txt", word)
print("Count Of Words:",word_count)
```

```
Enter the word to count:  Parth

Count Of Words: 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.

```python
f = open("marks.txt", "w")
for i in range(5):
    f.write(input("Enter marks: ") + "\n")
f.close()

f = open("marks.txt", "r")
marks = [int(line.strip()) for line in f]
f.close()
print("Highest score:", max(marks))
```

```
Enter marks:  99
Enter marks:  867
Enter marks:  54
Enter marks:  67
Enter marks:  89

Highest score: 867
```

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

(Note: each number should be in new line)

```python
fp = open("primenumbers.txt", "w")
n, count = 2, 0
```

```
while count < 100:
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            break
    else:
        fp.write(str(n) + "\n")
        count += 1
    n += 1
fp.close()
```

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

```
f1 = open("file1.txt", "r")
f2 = open("file2.txt", "r")
f3 = open("mergedfile.txt", "w")

f3.write(f1.read())
f3.write("\n")
f3.write(f2.read())

f1.close()
f2.close()
f3.close()
```

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

```
f1 = open("demo.txt", "r")
f2 = open("updatedfile.txt", "w")

word1 = "Parth"
word2 = "Patel"

data = f1.read().replace(word1, word2)
f2.write(data)

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

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

```
f = open("example.txt", "w+")
f.write("Hello, this is an example file.")
f.seek(0)
print("Current position:", f.tell())
f.seek(7)
print("Position after seeking to 7:", f.tell())
print("Reading from position 7:", f.read(4))
f.close()
```

```
Current position: 0
Position after seeking to 7: 7
Reading from position 7: this
```

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.

```python
# Handling exceptions using separate except blocks
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    result = a / b
    print("Result:", result)

    lst = [1, 2, 3]
    index = int(input("Enter index to access: "))
    print("Element:", lst[index])

except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
except ValueError:
    print("Error: Invalid input! Please enter an integer.")
except TypeError:
    print("Error: Unsupported operation between different types.")
```

```
Enter a number:  10
Enter another number:  0

Error: Division by zero is not allowed.
```

```python
# Handling all exceptions using a single except block
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    result = a / b
    print("Result:", result)

    lst = [1, 2, 3]
    index = int(input("Enter index to access: "))
    print("Element:", lst[index])
```

```
except (ZeroDivisionError, ValueError, TypeError) as e:
    print(f"Error occurred: {e}")

Enter a number:  10
Enter another number:  0

Error occurred: division by zero
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

```
try:
    lst = [1, 2, 3]
    print(lst[5])

except IndexError:
    print("Error: List index out of range.")

try:
    d = {"a": 1, "b": 2}
    print(d["a"])

except KeyError:
    print("Error: Key not found in dictionary.")

Error: List index out of range.
1
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

```
try:
    with open("demo.txt", "r") as file:
        content = file.read()
        print("File Content is:",content)

except FileNotFoundError:
    print("Error: File not found.")

try:
    import non_existent_module

except ModuleNotFoundError:
    print("Error: Module not found.")

File Content is: Dadhaniya Parth
Student of Darshan University
```

```
Full Stack Developer
Error: Module not found.
```

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

```
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    result = a / b
    print("Ans=",result)

    lst = [1, 2, 3]
    print(lst[1])

    d = {"a": 1, "b": 2}
    print(d["a"])

    import non_existent_module

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

```
Enter a number:  10
Enter another number:  2

Ans= 5.0
2
1
An error occurred: No module named 'non_existent_module'
```

05) WAP to demonstrate else and finally block.

```
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    result = a / b
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
except ValueError:
    print("Error: Invalid input! Please enter an integer.")
else:
    print("Division successful! Result:", result)
finally:
    print("Execution completed. This will always run.")
```

```
Enter a number:  10.5

Error: Invalid input! Please enter an integer.
Execution completed. This will always run.
```

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

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

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

```
try:
    grades_input = input("Enter a list of grades separated by commas: ")
    grades = [int(grade.strip()) for grade in grades_input.split(",")]
    print("Grades:", grades)
except ValueError:
    print("Error: Please enter only numeric values separated by commas.")
```

```
Enter a list of grades separated by commas:  89,A,79

Error: Please enter only numeric values separated by commas.
```

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

```
def divide(a, b):
    try:
        result = a / b
        return result
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."

num1 = int(input("Enter numerator: "))
num2 = int(input("Enter denominator: "))

print("Result:", divide(num1, num2))
```

```
Enter numerator:  10
Enter denominator:  0

Result: Error: Division by zero is not allowed.
```

08) WAP that gets an age of a person form the user and raises ValueError with error message: "Enter Valid Age" :

If the age is less than 18.

otherwise print the age.

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

```
Enter your age:  12

Error: Enter Valid Age
```

09) WAP to raise your custom Exception named InvalidUsernameError with the error message : "Username must be between 5 and 15 characters long":

if the given name is having characters less than 5 or greater than 15.

otherwise print the given username.

```
class InvalidUsernameError(Exception):
    pass

try:
    username = input("Enter your username: ")
    if len(username) < 5 or len(username) > 15:
        raise InvalidUsernameError("Username must be between 5 and 15 
characters long")
    print("Username:", username)
except InvalidUsernameError as e:
    print("Error:", e)
```

```
Enter your username:  tree

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

10) WAP to raise your custom Exception named NegativeNumberError with the error message : "Cannot calculate the square root of a negative number" :

if the given number is negative.

otherwise print the square root of the given number.

```
import math

class NegativeNumberError(Exception):
    pass

try:
    num = float(input("Enter a number: "))
    if num < 0:
        raise NegativeNumberError("Cannot calculate the square root of a negative number")
    print("Square root:", math.sqrt(num))
except NegativeNumberError as e:
    print("Error:", e)
```

```
Enter a number:  -12

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

Modules

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

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

```
import calculator as cal

a = int(input("Enter first number: "))
b = int(input("Enter second number: "))

print("Addition:", cal.add(a, b))
print("Subtraction:", cal.sub(a, b))
print("Multiplication:", cal.mul(a, b))
print("Division:", cal.div(a, b))
```

```
Enter first number:  10
Enter second number:  2
```

```
Addition: 12
Subtraction: 8
Multiplication: 20
Division: 5.0
```

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

```
import random

string = input("Enter a string: ")
random_char = random.choice(string)

print("Random character:", random_char)
```

```
Enter a string:  Parth Dadhaniya
```

```
Random character: i
```

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

```
import random

lst = input("Enter list elements separated by spaces: ").split()
random_element = random.choice(lst)
```

```
print("Random element:", random_element)

Enter list elements separated by spaces:  apple banana mango grapes

Random element: banana
```

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

```
import random

fixed_number = int(input("Enter a fixed dice number (1 to 6): "))

def roll_dice():
    random.seed(42)
    return fixed_number

print("Rolled number:", roll_dice())

Enter a fixed dice number (1 to 6):  2

Rolled number: 2
```

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

```
import random

random_numbers = [random.randrange(100, 1000, 5) for _ in range(3)]

print("Random numbers divisible by 5:", random_numbers)

Random numbers divisible by 5: [915, 240, 130]
```

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

```
import random

lottery_tickets = random.sample(range(1000, 10000), 100)

winner, runner_up = random.sample(lottery_tickets, 2)

print("Winner Ticket:", winner)
print("Runner-up Ticket:", runner_up)

Winner Ticket: 8359
Runner-up Ticket: 4483
```

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

```
from datetime import datetime

current_time = datetime.now()

print("Current Date and Time:", current_time)
```

```
Current Date and Time: 2025-02-15 09:02:55.127242
```

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

```
from datetime import datetime, timedelta

given_date = input("Enter a date (YYYY-MM-DD): ")
date_obj = datetime.strptime(given_date, "%Y-%m-%d")

new_date = date_obj - timedelta(days=7)

print("Date after subtracting a week:", new_date.strftime("%Y-%m-%d"))
```

```
Enter a date (YYYY-MM-DD):  2025-02-15

Date after subtracting a week: 2025-02-08
```

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

```
from datetime import datetime

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

date_obj1 = datetime.strptime(date1, "%Y-%m-%d")
date_obj2 = datetime.strptime(date2, "%Y-%m-%d")

days_difference = abs((date_obj2 - date_obj1).days)

print("Number of days between the given dates:", days_difference)
```

```
Enter first date (YYYY-MM-DD):  2025-02-15
Enter second date (YYYY-MM-DD):  2024-02-15

Number of days between the given dates: 366
```

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

```
from datetime import datetime

date_input = input("Enter a date (YYYY-MM-DD): ")
date_obj = datetime.strptime(date_input, "%Y-%m-%d")
```

```
day_of_week = date_obj.strftime("%A")

print("Day of the week:", day_of_week)

Enter a date (YYYY-MM-DD):  2025-02-15

Day of the week: Saturday
```

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

```
from datetime import datetime, timedelta

current_datetime = datetime.now()
print("Current Date and Time:", current_datetime.strftime("%Y-%m-%d
%H:%M:%S"))

current_date = current_datetime.date()
print("Current Date:", current_date)

current_time = current_datetime.time()
print("Current Time:", current_time.strftime("%H:%M:%S"))

future_date = current_datetime + timedelta(days=7)
print("Date after 7 days:", future_date.strftime("%Y-%m-%d"))

past_date = current_datetime - timedelta(days=7)
print("Date 7 days ago:", past_date.strftime("%Y-%m-%d"))

date_input = "2025-02-15"
date_obj = datetime.strptime(date_input, "%Y-%m-%d")
print(f"The day of {date_input} is:", date_obj.strftime("%A"))

Current Date and Time: 2025-02-15 09:10:33
Current Date: 2025-02-15
Current Time: 09:10:33
Date after 7 days: 2025-02-22
Date 7 days ago: 2025-02-08
The day of 2025-02-15 is: Saturday
```

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

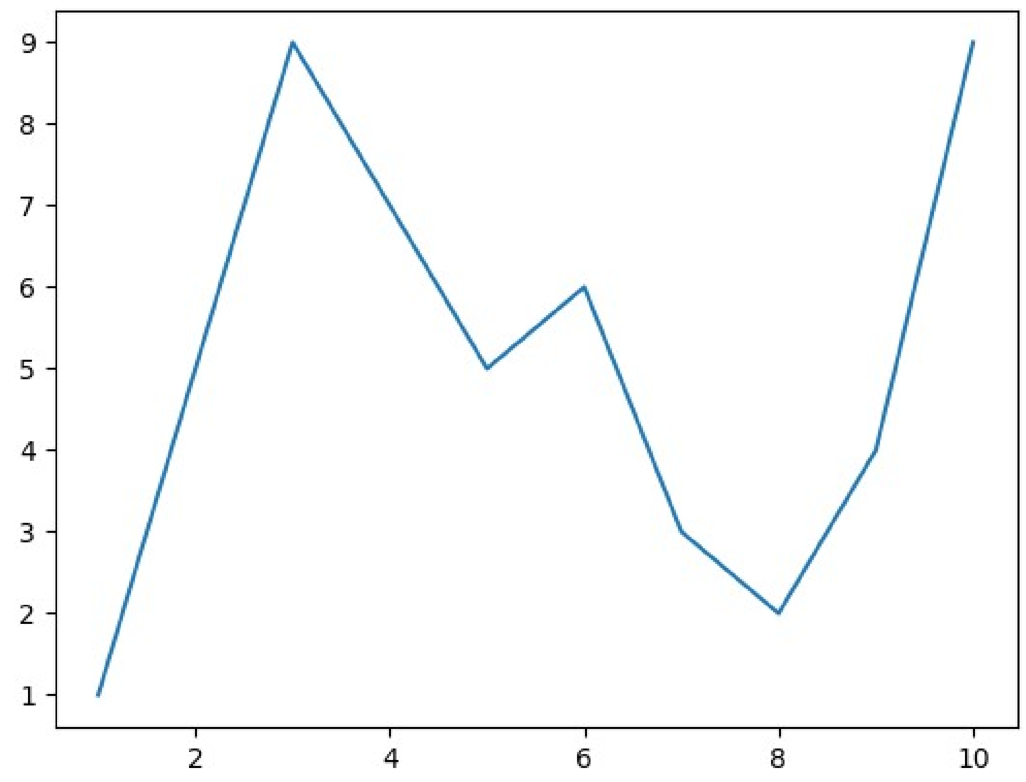
```
import math

num = 25
print("Square root of", num, "is:", math.sqrt(num))

base, exponent = 2, 5
print(f"{base} raised to the power {exponent} is:", math.pow(base,
exponent))

num = 5
```

```python
print(f"Factorial of {num} is:", math.factorial(num))

angle = math.radians(30)
print("Sin(30°):", math.sin(angle))
print("Cos(30°):", math.cos(angle))
print("Tan(30°):", math.tan(angle))

num = 10
print("Natural log of", num, "is:", math.log(num))
print("Log base 10 of", num, "is:", math.log10(num))

print("Value of Pi:", math.pi)
print("Value of Euler's number (e):", math.e)
```

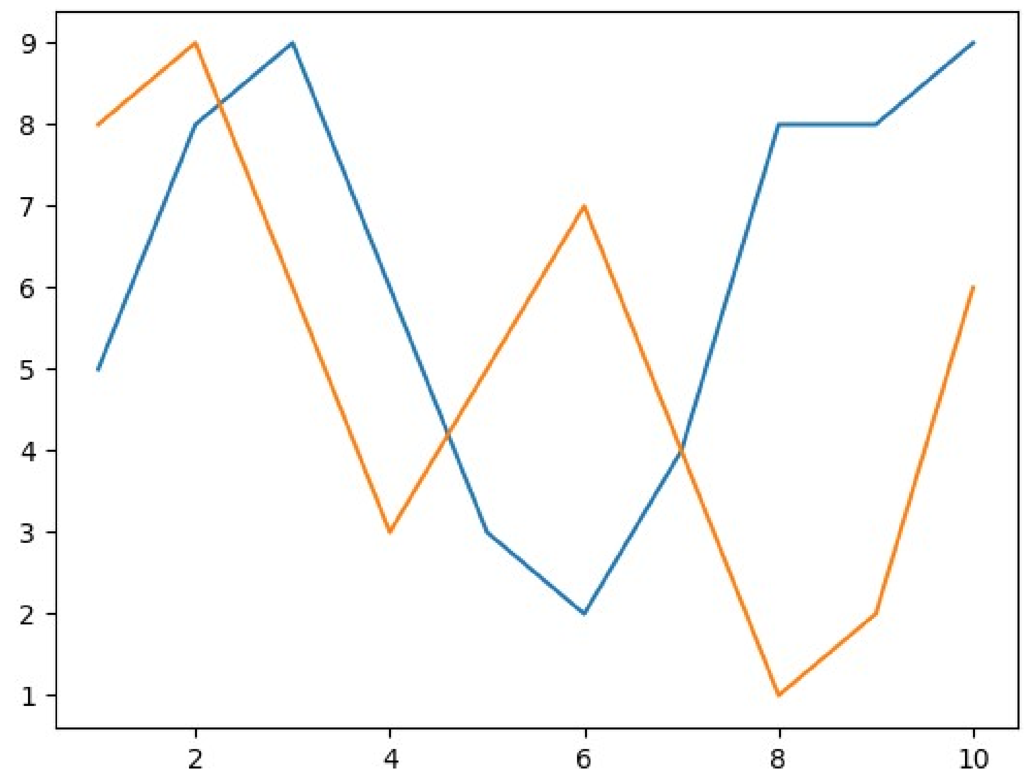
```
Square root of 25 is: 5.0
2 raised to the power 5 is: 32.0
Factorial of 5 is: 120
Sin(30°): 0.49999999999999994
Cos(30°): 0.8660254037844387
Tan(30°): 0.5773502691896257
Natural log of 10 is: 2.302585092994046
Log base 10 of 10 is: 1.0
Value of Pi: 3.141592653589793
Value of Euler's number (e): 2.718281828459045
```

```
#import matplotlib below

import matplotlib.pyplot


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

plt.plot(x,y)
plt.show

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

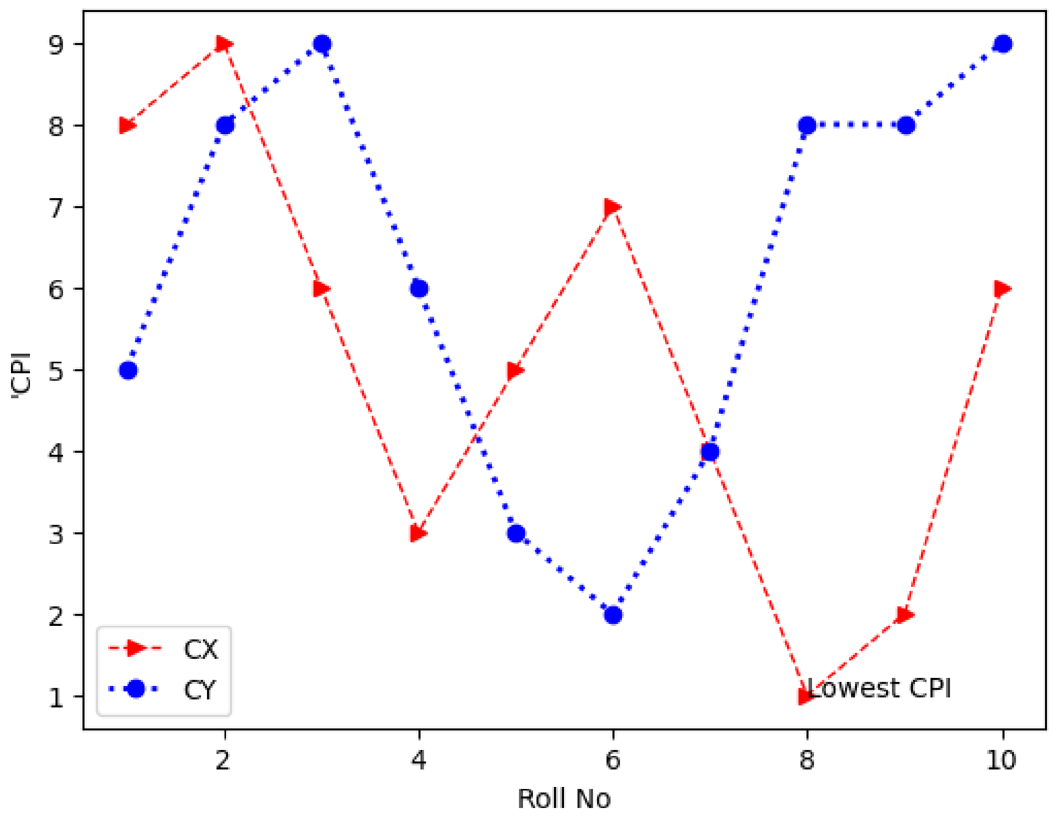
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



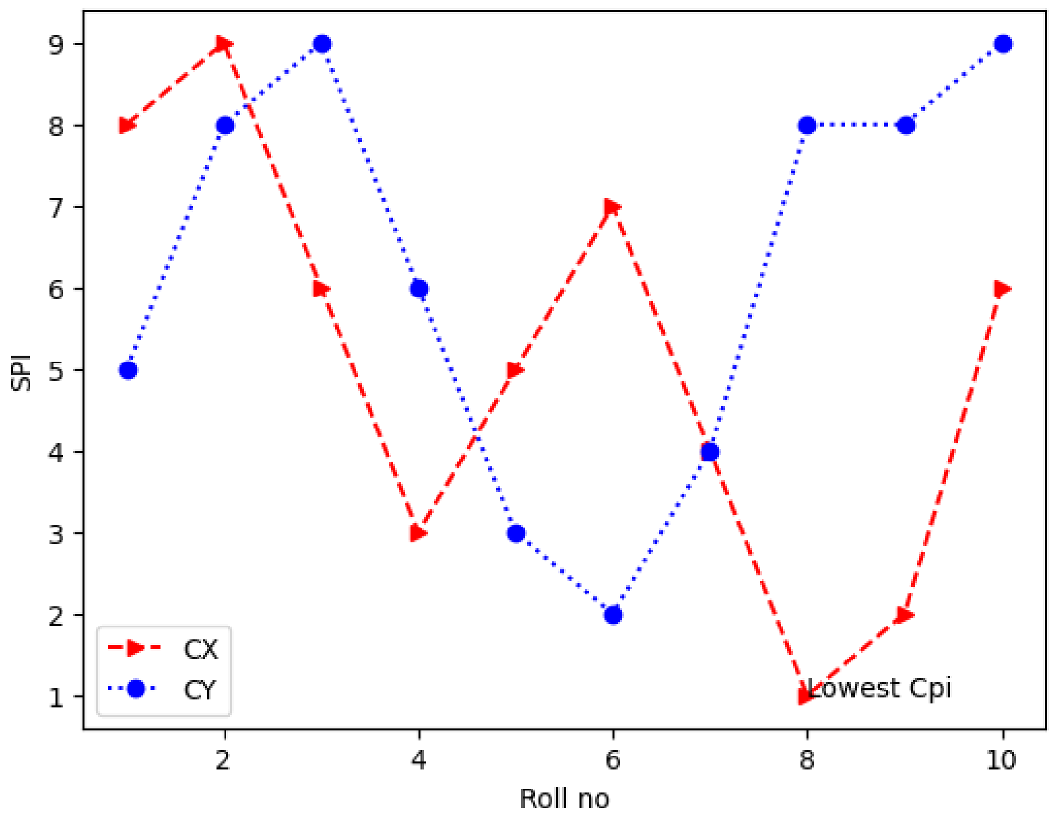
```
import matplotlib.pyplot as plt

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(x,cxMarks)
plt.plot(x,cyMarks)
plt.show()

# write a code to display two lines in a line chart (data given above)
```



```
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
```



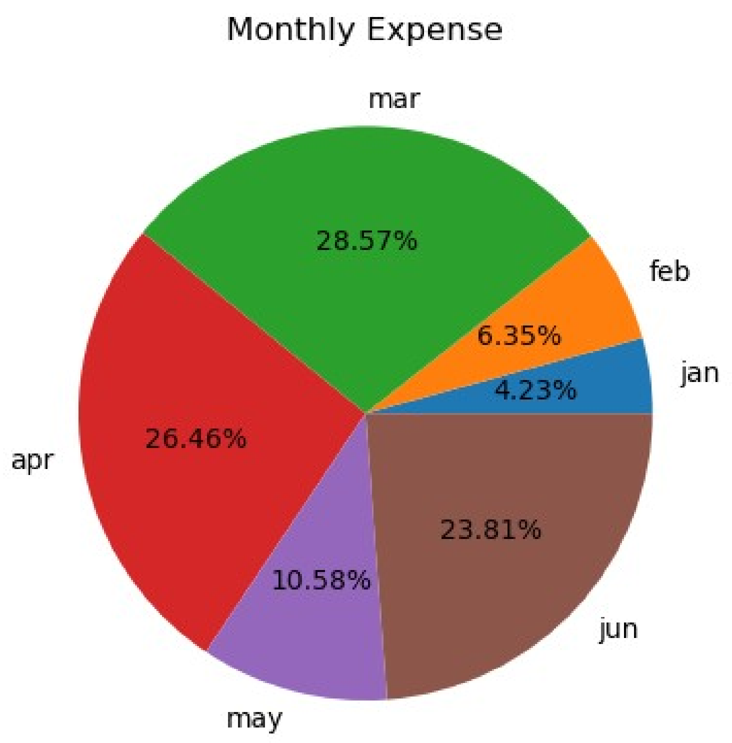
```
import matplotlib.pyplot as plt

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

plt.plot(x,cxMarks,c='r',marker='>',ls='--',label="CX")
plt.plot(x,cyMarks,c='b',marker='o',ls=':',label='CY')
plt.annotate("Lowest Cpi",(8,1))
plt.xlabel('Roll no')
plt.ylabel("SPI")
plt.legend()
plt.show()
```



04) WAP to demonstrate the use of Pie chart.

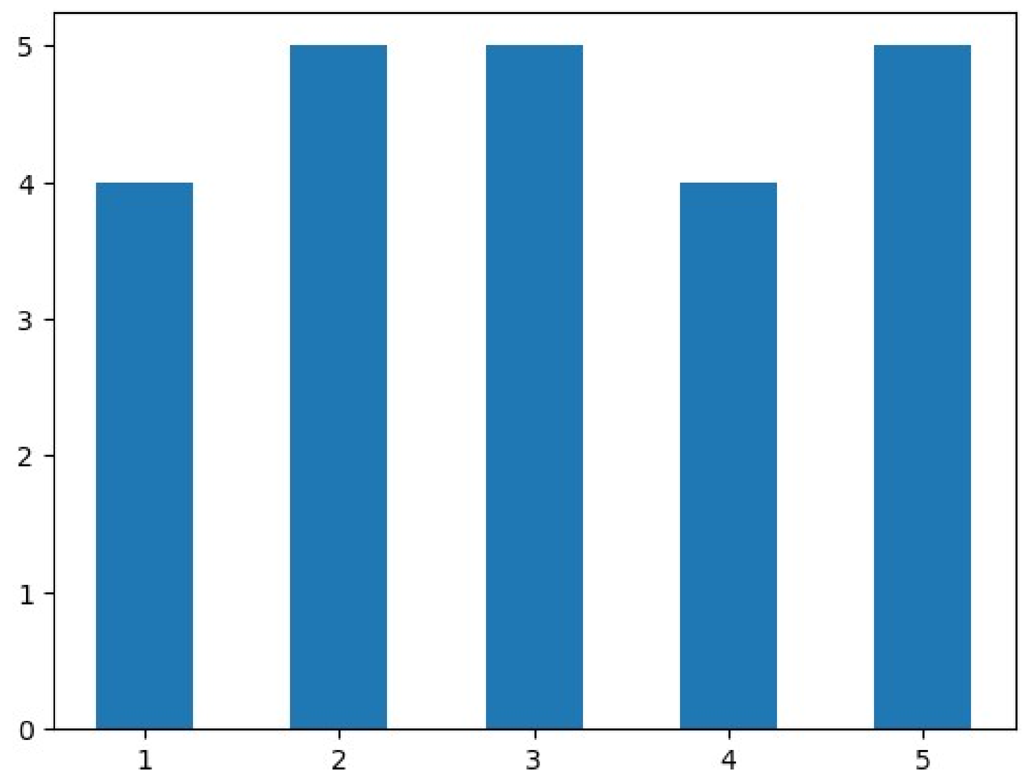
```
import matplotlib.pyplot as plt

value = [800,1200,5400,5000,2000,4500]
lable = ['jan','feb','mar','apr','may','jun']

plt.pie(value,labels=lable,autopct='%1.2f%%')
plt.title("Monthly Expense")
```

```
Text(0.5, 1.0, 'Monthly Expense')
```



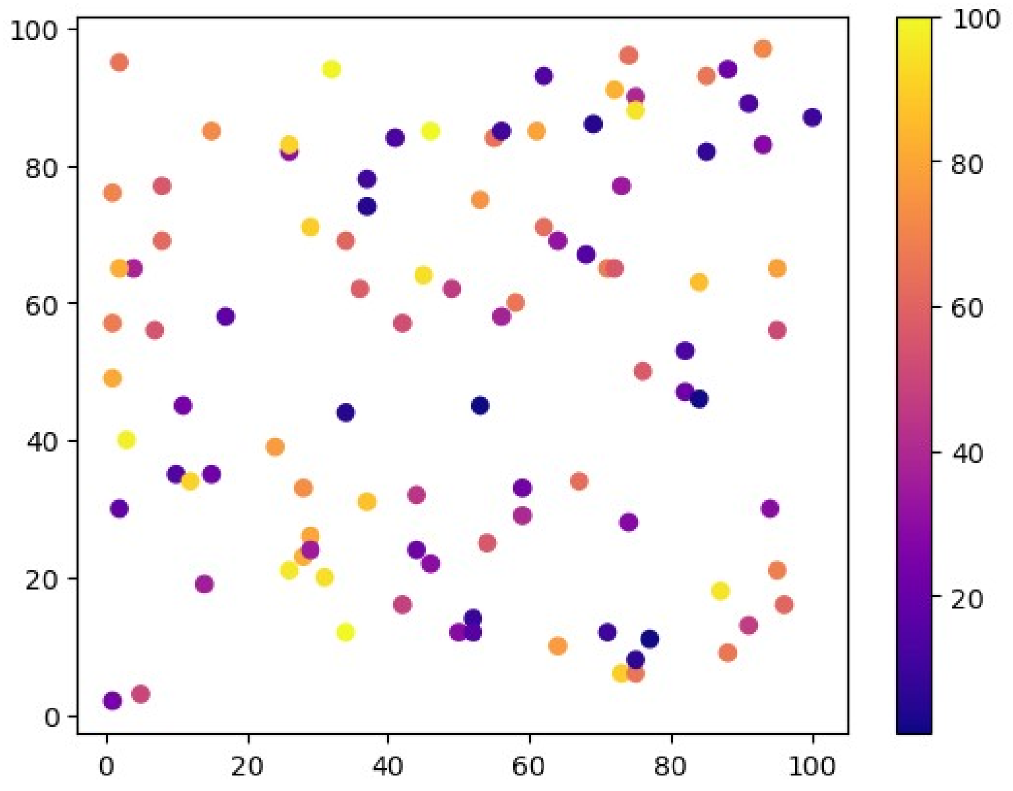
05) WAP to demonstrate the use of Bar chart.

```
import matplotlib.pyplot as plt
import random
x = [random.randint(1,5) for i in range(15)]
y = [random.randint(1,5) for i in range(15)]

bars = plt.bar(x, y, width=0.5)
plt.show()
```



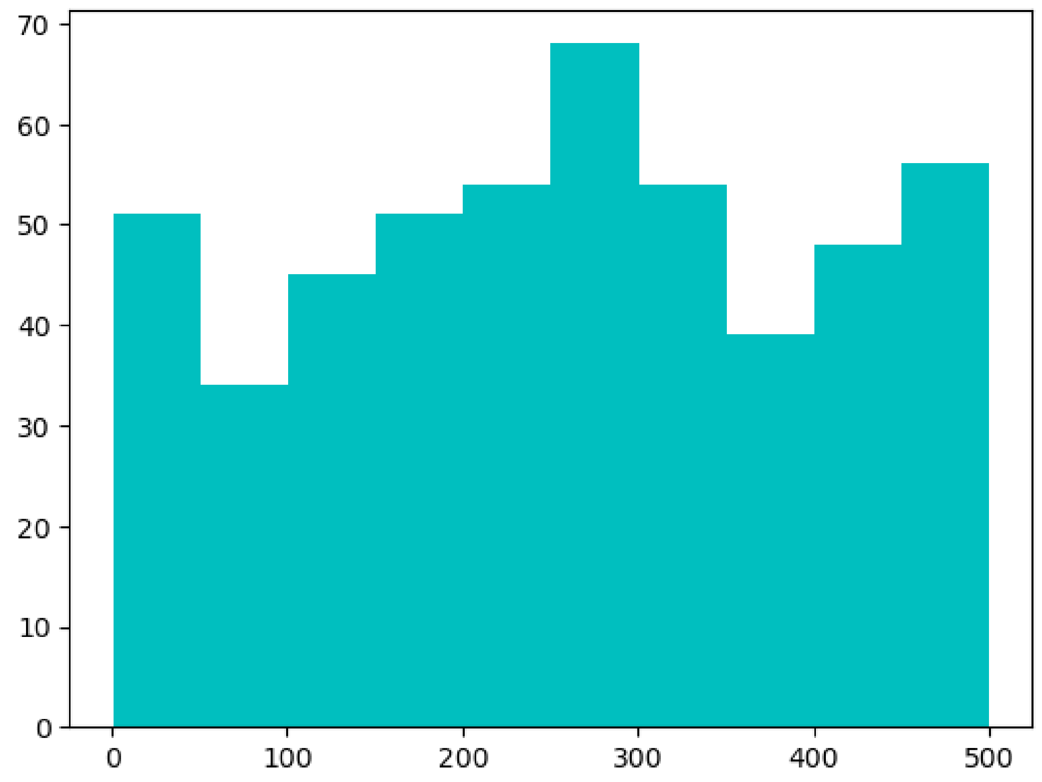
06) WAP to demonstrate the use of Scatter Plot.

```
import matplotlib.pyplot as plt
import random

x = [random.randint(1,100) for i in range(100)]
y = [random.randint(1,100) for i in range(100)]
a = [random.randint(1,100) for i in range(100)]

plt.scatter(x,y,c=a,cmap='plasma')
plt.colorbar()
plt.show()
```



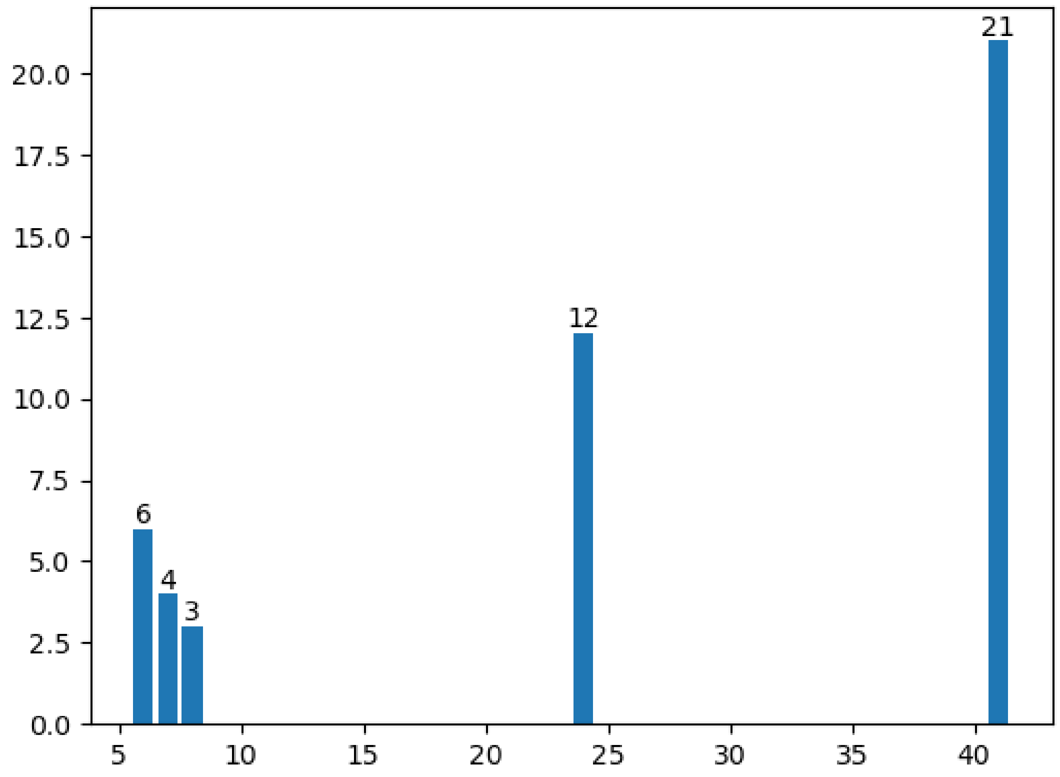
07) WAP to demonstrate the use of Histogram.

```
import matplotlib.pyplot as plt
import random

x = [random.randint(1,500) for i in range(500)]
# y = [random.randint(1,1000) for i in range(1000)]


plt.hist(x,color='c',histtype='bar')

plt.show()
```



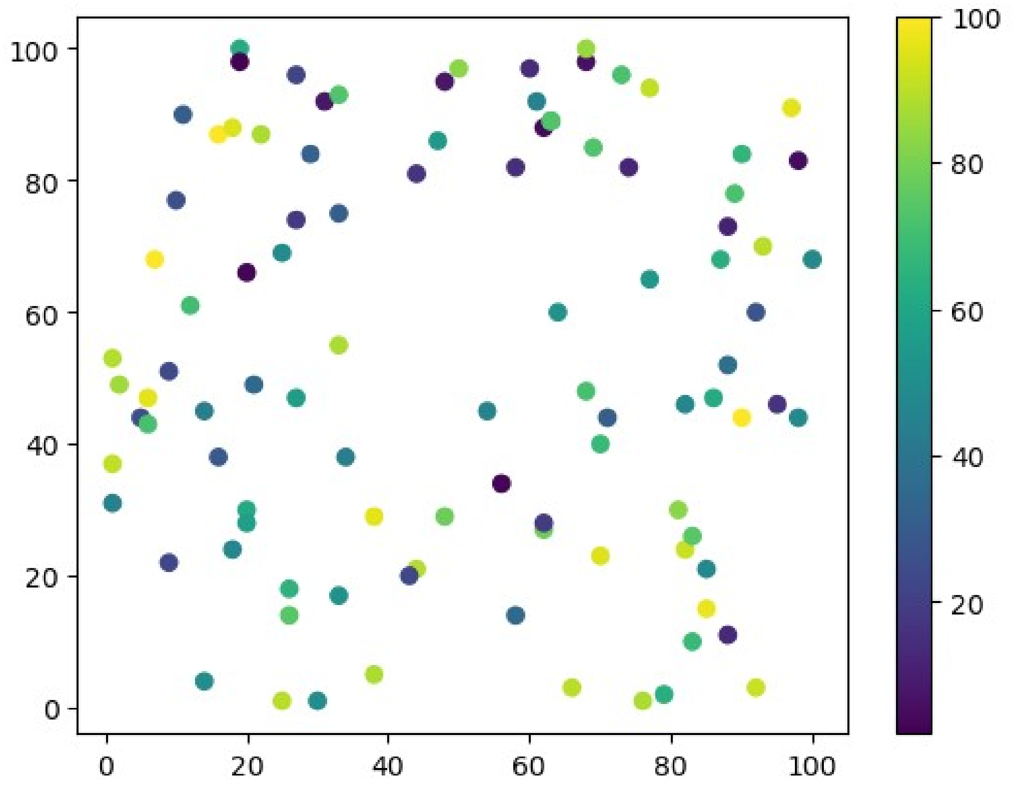
08) WAP to display the value of each bar in a bar chart using Matplotlib.

```python
import matplotlib.pyplot as plt
import random

x = [random.randint(1, 50) for _ in range(5)]
y = [random.randint(1, 50) for _ in range(5)]

bars = plt.bar(x, y)

for i in bars:
    cy = i.get_height()
    plt.text(i.get_x() + i.get_width()/2, cy, f'{cy}', ha='center',
va='bottom')

plt.show()
```



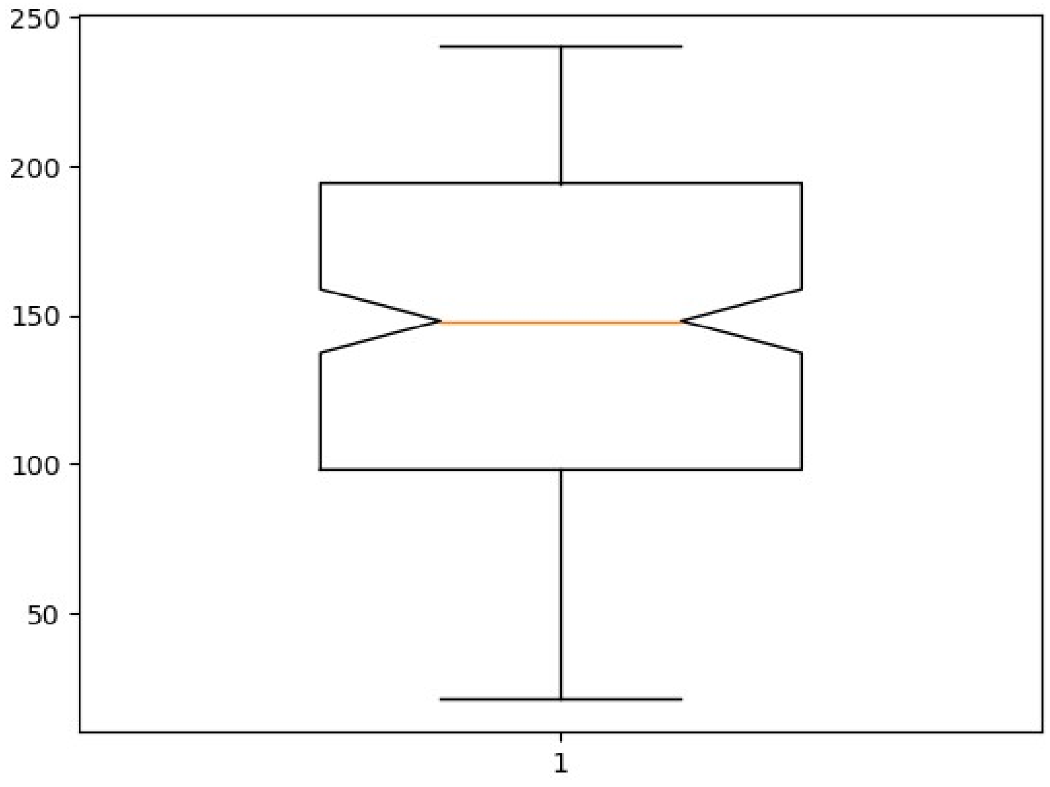
09) WAP create a Scatter Plot with several colors in Matplotlib?

```
import matplotlib.pyplot as plt
import random

x = [random.randint(1,100) for i in range(100)]
y = [random.randint(1,100) for i in range(100)]
a = [random.randint(1,100) for i in range(100)]

plt.scatter(x,y,c=a)
plt.colorbar()
plt.show()
```



10) WAP to create a Box Plot.

```
import matplotlib.pyplot as plt

random.seed(51)
test = [ random.randint(20,240) for i in range(200)]
plt.boxplot(test, notch=True,vert=True, widths=0.5)
plt.show()
```



250
200
150
100
50
1

OOP

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

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

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

student1 = Students("John Doe", 16, "10th Grade")

student1.display_info()
```

```
Name: John Doe, Age: 16, Grade: 10th 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.

```
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 Number: ")
        self.user_name = input("Enter User Name: ")
        self.email = input("Enter Email: ")
        self.account_type = input("Enter Account Type: ")
        self.account_balance = float(input("Enter Account Balance: "))

    def DisplayAccountDetails(self):
        print("\nAccount 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}")

def main():
    account = Bank_Account("", "", "", "", 0.0)
    account.GetAccountDetails()
    account.DisplayAccountDetails()

if __name__ == "__main__":
    main()

Enter Account Number:  161116111611
Enter User Name:  Parth
Enter Email:  parth@gmail.com
Enter Account Type:  current
Enter Account Balance:  25000


Account Details:
Account Number: 161116111611
User Name: Parth
Email: parth@gmail.com
Account Type: current
Account Balance: 25000.0
```

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

```
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

circle1 = Circle(5)
print(f"Area of Circle: {circle1.area()}")
print(f"Perimeter of Circle: {circle1.perimeter()}")

Area of Circle: 78.53981633974483
Perimeter of Circle: 31.41592653589793
```

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

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

    def update_info(self, name=None, age=None, salary=None):
        if name:
            self.name = name
        if age:
            self.age = age
        if salary:
            self.salary = salary

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

employee1 = Employee("Alice", 30, 50000)
employee1.display_info()

employee1.update_info(age=31, salary=55000)
print("\nUpdated Employee Information:")
employee1.display_info()
```

```
Employee Name: Alice
Age: 30
Salary: 50000

Updated Employee Information:
Employee Name: Alice
Age: 31
Salary: 55000
```

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

```python
class BankAccount:
    def __init__(self, account_number, account_holder, balance=0.0):
        self.account_number = account_number
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited {amount}. New balance: {self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdrawn {amount}. Remaining balance:
{self.balance}")
        else:
            print("Insufficient balance or invalid amount.")

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

account_number = input("Enter Account Number: ")
account_holder = input("Enter Account Holder Name: ")
balance = float(input("Enter Initial Balance: "))

account = BankAccount(account_number, account_holder, balance)

while True:
    print("\n1. Check Balance\n2. Deposit\n3. Withdraw\n4. Exit")
    choice = input("Enter your choice: ")

    if choice == "1":
        account.check_balance()
    elif choice == "2":
        amount = float(input("Enter deposit amount: "))
        account.deposit(amount)
    elif choice == "3":
        amount = float(input("Enter withdrawal amount: "))
        account.withdraw(amount)
    elif choice == "4":
        print("Exiting... Thank you!")
        break
    else:
        print("Invalid choice! Please try again.")
```

```
Enter Account Number:  161116111611
Enter Account Holder Name:  Parth
Enter Initial Balance:  10000


1. Check Balance
2. Deposit
3. Withdraw
4. Exit
```

```
Enter your choice:  2
Enter deposit amount:  5000

Deposited 5000.0. New balance: 15000.0

1. Check Balance
2. Deposit
3. Withdraw
4. Exit

Enter your choice:  1

Account Balance: 15000.0

1. Check Balance
2. Deposit
3. Withdraw
4. Exit

Enter your choice:  4

Exiting... Thank you!
```

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

```python
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, name, price, quantity):
        if name in self.items:
            self.items[name]['quantity'] += quantity
        else:
            self.items[name] = {'price': price, 'quantity': quantity}
        print(f"Added {quantity} of {name} at ${price} each.")

    def remove_item(self, name, quantity):
        if name in self.items:
            if self.items[name]['quantity'] >= quantity:
                self.items[name]['quantity'] -= quantity
                print(f"Removed {quantity} of {name}.")
                if self.items[name]['quantity'] == 0:
                    del self.items[name]
            else:
                print("Insufficient quantity.")
        else:
            print("Item not found.")
    def display_inventory(self):
        if not self.items:
            print("\nInventory is empty.")
        else:
            print("\nInventory:")
            for name, details in sorted(self.items.items()):
                print(f"{name}: ${details['price']}, Qty:
{details['quantity']}")

inventory = Inventory()

while True:
    print("\n1. Add Item\n2. Remove Item\n3. Display Inventory\n4.
Exit")
    choice = input("Enter your choice: ")

    if choice == "1":
        name = input("Item name: ")
        price = float(input("Price: "))
        quantity = int(input("Quantity: "))
        inventory.add_item(name, price, quantity)
    elif choice == "2":
        name = input("Item name to remove: ")
        quantity = int(input("Quantity to remove: "))
        inventory.remove_item(name, quantity)
    elif choice == "3":
        inventory.display_inventory()
    elif choice == "4":
        print("Exiting... Thank you!")
        break
    else:
        print("Invalid choice! Try again.")
```

```
1. Add Item
2. Remove Item
3. Display Inventory
4. Exit

Enter your choice:  1
Item name:  Laptop
Price:  3
Quantity:  3

Added 3 of Laptop at $3.0 each.

1. Add Item
2. Remove Item
3. Display Inventory
4. Exit
```

```
Enter your choice:  4

Exiting... Thank you!
```

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

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

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

student1 = Student("Alice", 20, "A")
student2 = Student("Bob", 22, "B")

student1.display_details()
student2.display_details()

Name: Alice, Age: 20, Grade: A
Name: Bob, Age: 22, Grade: B
```

08) Create one class student_kit

Within the student_kit class create one class attribute principal name (Mr ABC)

Create one attendance method and take input as number of days.

While creating student take input their name .

Create one certificate for each student by taking input of number of days present in class.

```
class StudentKit:
    principal_name = "Mr. ABC"

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

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

    def generate_certificate(self):
        print("\n-----------------------------")
        print(f"Certificate of Attendance")
        print(f"Presented to: {self.name}")
        print(f"Principal: {StudentKit.principal_name}")
        print(f"Days Present: {self.attendance_days}")
        print("----------------------------\n")

name = input("Enter student name: ")
student = StudentKit(name)
days_present = int(input("Enter number of days present: "))
student.attendance(days_present)
student.generate_certificate()

Enter student name:  Parth
Enter number of days present:  90


----------------------------
Certificate of Attendance
Presented to: Parth
Principal: Mr. ABC
Days Present: 90
----------------------------
```

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

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

    def add_time(self, other):
        total_minutes = self.minute + other.minute
        extra_hours = total_minutes // 60
        final_minutes = total_minutes % 60
        final_hours = self.hour + other.hour + extra_hours
        return Time(final_hours, final_minutes)

    def display_time(self):
        print(f"Time: {self.hour} hours and {self.minute} minutes")

time1 = Time(2, 50)
time2 = Time(1, 30)
sum_time = time1.add_time(time2)
sum_time.display_time()

Time: 4 hours and 20 minutes
```

Continued..

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

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

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

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

rect1 = Rectangle(10, 5)
print("Area:", calculate_area(rect1))
```

```
Area: 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().

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

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

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

square1 = Square(4)
square1.area()
```

```
Area of Square: 16
```

12) Calculate the area of a rectangle.

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

Also define a class method that compares the two sides of reactangle. An object is instantiated only if the two sides are different; otherwise a message should be displayed : THIS IS SQUARE.

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

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

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

    @classmethod
    def compare_sides(cls, length, width):
        if length == width:
            print("THIS IS SQUARE.")
        else:
            return cls(length, width)

rect1 = Rectangle.compare_sides(10, 5)
if rect1:
    rect1.area()

rect2 = Rectangle.compare_sides(6, 6)
```

```
Area of Rectangle: 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.

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

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

square1 = Square(4)
print("Side:", square1.get_side())
square1.set_side(6)
print("Updated Side:", square1.get_side())
```

```
Side: 4
Updated Side: 6
```

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

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

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

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

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

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

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

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

```
Enter Profit:  2000
Enter Loss:  1000
```

```
Balance: 1000.0
```

15) WAP to demonstrate all types of inheritance.

```python
# Single Inheritance
class Animal:
    def sound(self):
        print("Animal makes a sound")

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

dog = Dog()
dog.sound()
dog.bark()

# Multiple Inheritance
class Father:
    def func1(self):
        print("Function from Father")

class Mother:
    def func2(self):
        print("Function from Mother")

class Child(Father, Mother):
    def func3(self):
        print("Function from Child")

child = Child()
child.func1()
child.func2()
child.func3()

# Multilevel Inheritance
class Grandfather:
    def grandparent_func(self):
        print("Function from Grandfather")

class Father(Grandfather):
    def parent_func(self):
        print("Function from Father")

class Son(Father):
    def child_func(self):
        print("Function from Son")

son = Son()
son.grandparent_func()
son.parent_func()
son.child_func()

# Hierarchical Inheritance
class Parent:
    def parent_func(self):
        print("Function from Parent")

class Son(Parent):
    def son_func(self):
        print("Function from Son")

class Daughter(Parent):
    def daughter_func(self):
        print("Function from Daughter")

son = Son()
daughter = Daughter()
son.parent_func()
son.son_func()
daughter.parent_func()
daughter.daughter_func()

# Hybrid Inheritance
class Person:
    def person_func(self):
        print("Function from Person")

class Employee(Person):
    def employee_func(self):
        print("Function from Employee")

class Manager(Person):
    def manager_func(self):
        print("Function from Manager")

class CEO(Employee, Manager):
    def ceo_func(self):
        print("Function from CEO")

ceo = CEO()
ceo.person_func()
ceo.employee_func()
ceo.manager_func()
ceo.ceo_func()
```

```
Animal makes a sound
Dog barks
Function from Father
Function from Mother
Function from Child
Function from Grandfather
Function from Father
Function from Son
Function from Parent
Function from Son
Function from Parent
Function from Daughter
Function from Person
Function from Employee
Function from Manager
Function from CEO
```

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.

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

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

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

emp = Employee("John", 30, 50000)
emp.display()
```

```
Name: John, 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.

```python
from abc import ABC, abstractmethod

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

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

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

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

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

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

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