Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

Lab - 1

Roll No. : 110

Name : Udit Khalpada

1. WAP to print “Hello World”

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

```
Hello World
```

2. WAP to print your address i) using single print ii) using multiple print

In [2]:
```
# 2. WAP to print your address i) using single print ii) using multiple print
print("Udit Khalpada", "45 Sunrise Avenue", "Rajkot, Gujarat - 360001", sep="\
# multiple print
print("====================")
print("Udit Khalpada")
print("45 Sunrise Avenue")
print("Rajkot, Gujarat - 360001")
```

```
Udit Khalpada
45 Sunrise Avenue
Rajkot, Gujarat - 360001
====================
Udit Khalpada
45 Sunrise Avenue
Rajkot, Gujarat - 360001
```

3. WAP to print addition of 2 numbers (without input function)

In [3]:
```
# 3. WAP to print addition of 2 numbers (without input function)
num_a = 15
num_b = 35
print("Sum of", num_a, "and", num_b, "is", num_a + num_b)
```

```
Sum of 15 and 35 is 50
```

4. WAP to calculate and print average of 2 numbers (without input function)

In [18]:
```
# 4. WAP to calculate and print average of 2 numbers (without input function)
score1 = 42
score2 = 58
average = (score1 + score2) / 2
print("Average of", score1, "and", score2, "is", average)
```

```
Average of 42 and 58 is 50.0
```

5. WAP to add two number entered by user.

In [16]:
```
# 5. WAP to add two number entered by user.
x = float(input("Enter first value: "))
y = float(input("Enter second value: "))
print("The total is", x + y)
```

```
The total is 35.0
```

6. WAP to calculate area of circle.

In [15]:
```
# 6. WAP to calculate area of circle.
radius = float(input("Enter the radius of the circle: "))
circle_area = 3.14159 * radius * radius
print("Circle area =", circle_area)
```

```
Circle area = 2827.431
```

7. Purposefully raise Indentation Error and Correct it.

In [14]:
```
# 7. Purposefully raise Indentation Error and Correct it.
num1, num2 = 25, 10
#error
if num1 < num2:
print(num1)
#corrected
if num1 > num2:
    print("num1 is greater than num2")
```

```
  Cell In[14], line 5
    print(num1)
    ^
IndentationError: expected an indented block after 'if' statement on line 4
```

8. WAP to calculate simple interest

In [13]:
```
# 8. WAP to calculate simple interest
principal = float(input("Principal amount: "))
rate = float(input("Interest rate: "))
time = float(input("Time period: "))
simple_interest = (principal * rate * time) / 100
print("Simple Interest calculated:", simple_interest)
```

```
Simple Interest calculated: 720.0
```

9. WAP Calculate Area and Circumference of Circle.

In [12]:
```
# 9. WAP Calculate Area and Circumference of Circle.
r = float(input("Radius for calculations: "))
area = 3.14159 * r * r
circumference = 2 * 3.14159 * r
print("Calculated Area:", area)
print("Calculated Circumference:", circumference)
```

```
Calculated Area: 1520.52956
Calculated Circumference: 138.22996
```

10. WAP to print Multiplication table of given number.

In [11]:
```
n = int(input("Enter a number: "))
print(n, "* 1 =", n * 1)
print(n, "* 2 =", n * 2)
print(n, "* 3 =", n * 3)
print(n, "* 4 =", n * 4)
print(n, "* 5 =", n * 5)
print(n, "* 6 =", n * 6)
print(n, "* 7 =", n * 7)
print(n, "* 8 =", n * 8)
print(n, "* 9 =", n * 9)
print(n, "* 10 =", n * 10)
```

```
11 * 1 = 11
11 * 2 = 22
11 * 3 = 33
11 * 4 = 44
11 * 5 = 55
11 * 6 = 66
11 * 7 = 77
11 * 8 = 88
11 * 9 = 99
11 * 10 = 110
```

11. WAP to calculate Area of Triangle. (hint: a = h*b*0.5)

In [10]:
```
# 11. WAP to calculate Area of Triangle. (hint: a = h*b*0.5)
base = float(input("Base of triangle: "))
height = float(input("Height of triangle: "))
triangle_area = 0.5 * base * height
print("Triangle area is", triangle_area)
```

```
Triangle area is 517.5
```

12. WAP to convert Degree to Fahrenheit and vice versa.

In [9]:
```
# 12. WAP to convert Degree to Fahrenheit and vice versa.
deg_c = float(input("Celsius temperature: "))
deg_f = (deg_c * 9/5) + 32
print("Converted to Fahrenheit:", deg_f)

deg_f = float(input("Fahrenheit temperature: "))
deg_c = (deg_f - 32) * 5/9
print("Converted to Celsius:", deg_c)
```

```
Converted to Fahrenheit: 86.0
Converted to Celsius: -5.0
Converted to Celsius: -5.0
```

13.WAP to calculate total marks and Percentage.

In [8]:
```
# 13. WAP to calculate total marks and Percentage.
sub1 = float(input("Subject 1 marks: "))
sub2 = float(input("Subject 2 marks: "))
sub3 = float(input("Subject 3 marks: "))
sub4 = float(input("Subject 4 marks: "))
sub5 = float(input("Subject 5 marks: "))
total_marks = sub1 + sub2 + sub3 + sub4 + sub5
percent = (total_marks / 500) * 100
print("Total marks obtained:", total_marks)
print("Percentage scored:", percent)
```

```
Total marks obtained: 150.0
Percentage scored: 30.0
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

Lab - 2

Roll No. : 110

Name : Udit Khalpada

1. WAP to check whether the given number is Positive or Negative.

In [1]:
```
n = int(input("Enter any number : "))
if n < 0:
    print("Number is negative")
else:
    print("Positive")
```

```
Positive
```

2. WAP to check whether the given number is Odd or Even.

In [2]:
```
n = int(input("Enter any number : "))
if n % 2== 0:
    print("Number is even")
else:
    print("odd")
```

```
odd
```

3. WAP to find out Largest number from given two numbers using simple if and ternary operator.

In [3]:
```
a = int(input("Enter number 1 : "))
b = int(input("Enter number 2 : "))
#using simple if
```

```
if a > b:
    print(a ," is greater")
else:
    print(b ," is greater")

#using ternary operator
ans = a if a > b else b
print(ans, " is greater")
```

```
45  is greater
45  is greater
```

4. WAP to find out Largest number from given three numbers.

In [4]:
```
a = int(input("Enter number 1 : "))
b = int(input("Enter number 2 : "))
c = int(input("Enter number 3 : "))
if a > b and a > c:
    print(a, " is greater")
elif b > c:
    print(b, " is greater")
else:
    print(c, " is greater")
```

```
6  is greater
```

5. 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 [5]:
```
y = int(input("Enter any year : "))
if y % 4 == 0 and y % 100 != 0 or y % 400 == 0:
    print(y," is a leap year")
else:
    print(y," is a Not leap year")
```

```
2004  is a leap year
```

6. WAP to display the name of the Day according to the number given by the user.

In [6]:
```
d = int(input("Enter any day number : "))
match d:
    case 1:
        print("Monday")
    case 2:
        print("Tuesday")
```

```
    case 3:
        print("Wednesday")
    case 4:
        print("Thursday")
    case 5:
        print("Friday")
    case 6:
        print("Saturday")
    case 7:
        print("Sunday")
    case _:
        print("Enter valid day")
```

```
Thursday
```

7. WAP to implement simple Calculator which performs (add,sub,mul,div) of two numbers based on user input.

In [7]:
```
a = int(input("Enter number 1 :"))
b = int(input("Enter number 2 :"))
op = input("Enter Operator : ")

match op:
    case '+':
        print(a+b)
    case '-':
        print(a-b)
    case '*':
        print(a*b)
    case '/':
        print(a/b)
    case '%':
        print(a%b)
    case _:
        print("Enter valid operator")
```

```
6
```

8. WAP to calculate electricity bill based on following criteria. Which takes the unit from the user.

-First 1 to 50 units – Rs. 2.60/unit

-Next 50 to 100 units – Rs. 3.25/unit

-Next 100 to 200 units – Rs. 5.26/unit

-200 units – Rs. 8.45/unit

In [8]:
```
u = int(input("Enter units :"))
if u < 50:
```

```
    print(u*2.60)
elif u < 100:
    rem = u-50
    res = (50*2.60) + (rem*3.25)
    print(res)
elif u < 200:
    rem = u-100
    res = (50*2.60) + (50*3.25) + (rem*5.26)
    print(res)
else:
    rem = u-200
    res = (50*2.60) + (50*3.25) + (100*5.26) + (rem*8.45)
    print(res)
```

```
59.800000000000004
```

9. WAP to find second largest number from the given three numbers.

In [9]:
```
a = int(input("Enter a : "))
b = int(input("Enter b : "))
c = int(input("Enter c : "))

if (a > b and a < c) or (a > c and a < b):
    second_largest = a
elif (b > a and b < c) or (b > c and b < a):
    second_largest = b
else:
    second_largest = c

print("Second largest:", second_largest)
```

```
Second largest: 6
```

10. Student marks class

In [10]:
```
m1 = int(input("Enter m1 : "))
m2 = int(input("Enter m2 : "))
m3 = int(input("Enter m3 : "))
m4 = int(input("Enter m4 : "))
m5 = int(input("Enter m5 : "))
total = m1+m2+m3+m4+m5;
pr = (total / 500) * 100

print("Total : ",total)
print("Percentage : ",pr)

if pr < 33:
    print("failed")
elif pr >33 and pr <= 50:
    print("Pass class")
elif pr >50 and pr <= 70:
```

```
    print("Second class")
elif pr >70 and pr <= 90:
    print("First class")
elif pr>90:
    print("Dsitinction")
else:
    print("Not valid")
```

```
Total :  265
Percentage :  53.0
Second class
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

Lab - 3

Roll No. : 110

Name : Udit Khalpada

1. WAP to print 1 to 10.

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

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

2. WAP to print 1 to n.

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

```
Enter n : 12
1
2
3
4
5
6
7
8
9
10
11
12
```

3.WAP to print odd numbers between 1 to n.

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

```
Enter n : 12
1
3
5
7
9
11
```

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

In [4]:
```
n =int(input("Enter n : "))
n2 = int(input("Enter n2 :"))
for i in range(n,n2):
    if i % 2==0 and i%3 != 0:
        print(i)
```

```
Enter n : 23
Enter n2 :45
26
28
32
34
38
40
44
```

5. WAP to print sum of 1 to n numbers.

In [5]:
```
n =int(input("Enter n : "))
```

```
sum=0
for i in range(1,n+1):
    sum +=i
print(sum)
```

```
Enter n : 23
276
```

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

In [6]:
```
n =int(input("Enter n : "))
sum =0
for i in range(1,n+1):
    sum += (i*i)
print(sum)
```

```
Enter n : 20
2870
```

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

In [8]:
```
n =int(input("Enter n : "))
sum =0
for i in range(1,n+1):
    if i % 2!=0:
        sum +=i
    else:
        sum -= i
print(sum)
```

```
Enter n : 10
-5
```

8. WAP to print multiplication table of given number.

In [9]:
```
n =int(input("Enter n : "))
for i in range(1,11):
    print(n," X ",i," = ",n*i)
```

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

9. WAP to find factorial of the given number.

In [11]:
```
n = int(input("Enter n:"))
fact=1
for i in range(1,n+1):
    fact *= i
print(fact)
```

```
Enter n:5
120
```

10. WAP to find factors of the given number.

In [12]:
```
n = int(input("Enter n:"))
for i in range(1,n+1):
    if n%i==0:
        print(i)
```

```
Enter n:5
1
5
```

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

In [14]:
```
n = int(input("Enter n:"))
prime=True
for i in range(2,n):
    if n%i==0:
        Prime=False
        break
if prime==True:
    print("Prime")
else:
    print("Not a prime")
```

```
Enter n:7
Prime
```

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

In [21]:
```
n = int(input("Enter n:"))
sum=0
while n >0:
    d= n%10
    sum += d
    n=n//10
print(sum)
```

```
Enter n:123
6
```

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

In [20]:
```
n = int(input("Enter n:"))
temp=n
rev=0
while n>0:
    d = n%10
    rev = rev * 10 +d
    n = n//10
if rev==temp:
    print("Palindrome")
else:
    print("Not palindrome")
```

```
Enter n:121
Palindrome
```

Patterns

14. Right angle triangle

In [25]:
```
n = int(input("Enter n:"))
for i in range(1, n + 1):
    for j in range(i):
        print("*", end=" ")
    print()
```

```
Enter n:4
*
* *
* * *
* * * *
```

15. Left Angle triangle

In [29]:
```
n = int(input("Enter n:"))
for i in range(1, n + 1):
    for j in range(n - i):
        print(" ", end="")

    for k in range(i):
        print("*", end="")
    print()
```

```
Enter n:4
   *
  **
 ***
****
```

16. Pyramid

In [30]:
```python
n = int(input("Enter n:"))
for i in range(1, n + 1):
    for j in range(n - i):
        print(" ", end="")

    for k in range(i):
        print("* ", end="")
    print()
```

```
Enter n:4
   *
  * *
 * * *
* * * *
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

Lab - 4

Roll No. : 110

Name : Udit Khalpada

1. WAP to check given string is palindrome or not.

In [11]:
```
s = input("Enter any string : ")
revstr = s[::-1]
if revstr == s:
    print("String is palindrome")
else:
    print("String is not palindrome")
```

```
String is palindrome
```

2.WAP to reverse the words in given string.

In [12]:
```
s1 = input("Enter any string : ")
li = s1.split()
reverseli = li[::-1]
str=''.join(reverseli)
print(str)
```

3.WAP to remove ith character from given string.

In [13]:
```
s2 = input("Enter any string : ")
i = int(input("Enter index of character you wanna remove : "))
news3 = s2[:i] + s2[i+1:]
print(news3)
```

```
Tiu
```

4. WAP to find length of String without using len function..

In [14]:
```
s4 = input("Enter string : ")
i=0
for char in s4:
    i += 1
print(i)
```

```
4
```

5. WAP to print even length word in string.

In [15]:
```
s5 = input('Enter string : ')
l5 = s5.split(" ")
flag=0
for i in l5:
    if len(i) % 2 == 0:
        print(i)
        flag=1
if(flag==0):
    print("not even length word in given string")
```

6.WAP to count numbers of vowels in given string.

In [16]:
```
s6 = input("Enter any string : ")
lows6 = s6.lower()
count=0
for i in s6:
    if i=='a' or i=='e' or i=='o' or i=='u' or i=='i':
        count = count + 1
print(count)
```

```
1
```

7. WAP to convert given array to string.

In [17]:
```
l7 = ["Hello", "I", "Am", "Udit", "K."]
s7 = ",".join(l7)
news7 = s7.replace(',',' ')
print(news7)
```

```
Hello I Am Udit K.
```

8. Check if the password and confirm password is same or not.

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

In [18]:
```
password = input("Enter password : ")
```

```
cpwd = input("Confirm password : ")
if cpwd == password:
    print("Account Created!")
else:
    print("Passwords do not match")
```

```
Account Created!
```

Darshan UNIVERSITY

Python for Data Science - 2305CS303

Lab - 5

Roll No. : 110

Name : Udit Khalpada

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

In [1]:
```
li = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
total = 0
for i in li:
    total += i
print(total)
```

```
55
```

2. WAP to find largest element in a List.

In [2]:
```
li = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
maxval = li[0]
for i in li:
    if i > maxval:
        maxval = i
print(maxval)
```

```
10
```

3. WAP to interchange first and last elements in a list.

In [3]:
```
li = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
li[0], li[-1] = li[-1], li[0]
print(li)
```

```
[10, 2, 3, 4, 5, 6, 7, 8, 9, 1]
```

4. WAP to reverse the list entered by user.

In [4]:
```
li = input("Enter list elements separated by space: ").split()
rev = []
for i in range(len(li)-1, -1, -1):
    rev.append(li[i])
print(rev)
```

```
Enter list elements separated by space:  1 2 3 4 5
['5', '4', '3', '2', '1']
```

5. WAP to print even numbers in a list.

In [5]:
```
li = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for i in li:
    if i % 2 == 0:
        print(i)
```

```
2
4
6
8
10
```

6. WAP to count occurrences of an element in a list.

In [6]:
```
li = [1, 2, 3, 2, 4, 2, 5, 6, 2]
element = 2
count = 0
for i in li:
    if i == element:
        count += 1
print(count)
```

```
4
```

7. WAP to extract elements with frequency greater than K.

In [7]:
```
li = [1, 2, 3, 2, 4, 2, 5, 6, 2, 3, 3, 3]
k = 2
for i in li:
    if li.count(i) > k:
        print(i)
```

2
3
2
2
2
3
3
3

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

Lab - 5 Part-2

Roll No. : 110

Name : Udit Khalpada

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

In [3]:
```
# With Comprehension
l1 = [i*i for i in range(10)]
print(l1)

#Wihtout comprehension
li1 = []
for i in range(10):
    li1.append(i*i)
print(li1)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

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

In [4]:
```
t = (5, 1, 9, 4, 7, 3)
k = 3
sorted_t = sorted(t)
mink = sorted_t[:k]
maxk = sorted_t[-k:]
print("Min K:", mink)
print("Max K:", maxk)
```

```
Min K: [1, 3, 4]
Max K: [5, 7, 9]
```

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

In [5]:
```
data = [(10, 20), (15, 25), (30, 60), (12, 18)]
k = 5
result = []
for tup in data:
    if all(x % k == 0 for x in tup):
        result.append(tup)
print(result)
```

```
[(10, 20), (15, 25), (30, 60)]
```

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

In [7]:
```
l4 = [1, 2, 3, 4, 5]
l5 = []
for i in l4:
    l5.append((i, i**3))
print(l5)
```

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

5. WAP to remove tuples of length K.

In [8]:
```
l1 = [(1, 2), (3, 4, 5), (6,), (7, 8)]
k = 2
l2 = []
for i in l1:
    if len(i) != k:
        l2.append(i)
print(l2)
```

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

Darshan UNIVERSITY

Python for Data Science - 2305CS303

Lab - 6

Roll No. : 110

Name : Udit Khalpada

1. WAP to iterate over a set.

In [1]:
```
s1 = {10, 20, 30, 40, 50}
for i in s1:
    print(i)
```

```
50
20
40
10
30
```

2. WAP to convert set into list, string and tuple.

In [2]:
```
s2 = {1, 2, 3, 4}

l1 = list(s2)
str1 = ''.join(str(i) for i in s2)
t1 = tuple(s2)

print(l1)
print(str1)
print(t1)
```

```
[1, 2, 3, 4]
1234
(1, 2, 3, 4)
```

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

In [3]:
```
l1 = [1, 2, 3, 4]
l2 = [5, 6, 3, 7]
l3 = False

for i in l1:
    if i in l2:
        l3 = True
        break

print(l3)
```

```
True
```

4. WAP to remove duplicates from list.

In [4]:
```
l1 = [1, 2, 2, 3, 4, 4, 5]
l2 = []
for i in l1:
    if i not in l2:
        l2.append(i)
print(l2)
```

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

5. WAP to find unique words in the given string.

In [5]:
```
s1 = "this is a test this is simple"
l1 = s1.split()
l2 = []
for word in l1:
    if word not in l2:
        l2.append(word)
print(l2)
```

```
['this', 'is', 'a', 'test', 'simple']
```

6. WAP to iterate over a dictionary.

In [6]:
```
d1 = {'a': 1, 'b': 2, 'c': 3}
for key in d1:
    print(key, d1[key])
```

```
a 1
b 2
c 3
```

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

In [8]:
```
d1 = {}
n = int(input("Enter number of items: "))
for i in range(n):
    key = input("Enter key: ")
    value = int(input("Enter value: "))
    d1[key] = value

total = 0
for v in d1.values():
    total += v

print("Sum:", total)
```

```
Enter number of items:  3
Enter key:  a
Enter value:  10
Enter key:  b
Enter value:  20
Enter key:  c
Enter value:  30
Sum: 60
```

8. WAP to sort dictionary by key or value.

In [9]:
```
d1 = {'b': 3, 'a': 1, 'c': 2}

sorted_by_key = dict(sorted(d1.items()))
sorted_by_value = dict(sorted(d1.items(), key=lambda x: x[1]))

print("Sorted by key:", sorted_by_key)
print("Sorted by value:", sorted_by_value)
```

```
Sorted by key: {'a': 1, 'b': 3, 'c': 2}
Sorted by value: {'a': 1, 'c': 2, 'b': 3}
```

9. 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 [10]:
```
dict1 = {'a': 5, 'c': 8, 'e': 2}
key = 'd'

if key in dict1:
    print(dict1[key])
```

```
else:
    print("Key Not Found")
```

```
Key Not Found
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

Lab - 7

Roll No. : 110

Name : Udit Khalpada

1. WAP to count simple interest using function.

In [1]:
```
def si(p, r, t):
    return (p * r * t) / 100

p = float(input("P: "))
r = float(input("R: "))
t = float(input("T: "))
print("SI:", si(p, r, t))
```

```
P:  20000
R:  2
T:  2
SI: 800.0
```

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

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

m = float(input("Mass (kg): "))
h = float(input("Height (m): "))
print("BMI:", bmi(m, h))
```

```
Mass (kg):  60
Height (m):  234
BMI: 0.0010957703265395574
```

3. WAP that defines a function to add first n numbers.

In [3]:
```
def add_n(n):
    return n * (n + 1) // 2

n = int(input("Enter n: "))
print("Sum:", add_n(n))
```

```
Enter n:  10
Sum: 55
```

4. WAP to find maximum number from given two numbers using function.

In [4]:
```
def max_num(a, b):
    return a if a > b else b

x = int(input("First: "))
y = int(input("Second: "))
print("Max:", max_num(x, y))
```

```
First:  44
Second:  55
Max: 55
```

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

In [5]:
```
def is_palindrome(s):
    return s == s[::-1]

txt = input("Enter string: ")
print(is_palindrome(txt))
```

```
Enter string:  aabbaa
True
```

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

In [6]:
```
def sum_list(lst):
    return sum(lst)

nums = list(map(int, input("Enter numbers: ").split()))
print("Sum:", sum_list(nums))
```

```
Enter numbers:  23 34 56
Sum: 113
```

7. WAP that defines a function which returns 1 if the number is prime otherwise return 0.

In [7]:
```
def is_prime(n):
    if n < 2:
        return 0
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return 0
    return 1

num = int(input("Enter number: "))
print(is_prime(num))
```

```
Enter number:  13
1
```

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

In [8]:
```
def prime_range(a, b):
    return [x for x in range(a, b+1) if is_prime(x)]

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

start = int(input("Start: "))
end = int(input("End: "))
print("Primes:", prime_range(start, end))
```

```
Start:  10
End:  20
Primes: [11, 13, 17, 19]
```

9. WAP to generate Fibonacci series of N given number using function name fibbo. (e.g. 0 1 1 2 3 5 8...).

In [9]:
```
def fibbo(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=' ')
        a, b = b, a + b

num = int(input("Enter N: "))
fibbo(num)
```

```
Enter N:  10
```

```
0 1 1 2 3 5 8 13 21 34
```

10. WAP to find the factorial of a given number using recursion.

In [10]:
```python
def fact(n):
    if n == 0 or n == 1:
        return 1
    return n * fact(n - 1)

num = int(input("Enter number: "))
print("Factorial:", fact(num))
```

```
Enter number:  5
Factorial: 120
```

11. WAP to implement simple calculator using lamda function.

In [11]:
```python
add = lambda a, b: a + b
sub = lambda a, b: a - b
mul = lambda a, b: a * b
div = lambda a, b: a / b if b != 0 else 'Error'

a = float(input("First: "))
b = float(input("Second: "))
op = input("Op (+ - * /): ")

calc = {'+': add, '-': sub, '*': mul, '/': div}
print("Result:", calc[op](a, b) if op in calc else "Invalid op")
```

```
First:  23
Second:  34
Op (+ - * /):  +
Result: 57.0
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 7 (Part-2)

User Defined Function

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

In [16]:
```
def sumFirstElements(tuplesList):
    total = 0
    for t in tuplesList:
        total += t[0]
    return total

sampleTuplesList = [(4, 2), (7, 8), (1, 3), (9, 5)]
print(sumFirstElements(sampleTuplesList))
```

```
21
```

13. 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 [17]:
```
def getStudentName(rollno, dict1):
    for key in dict1:
        if key == rollno:
            return dict1[key]

dict1 = {101: 'Udit', 102: 'Tipu', 103: 'ABC', 104: 'XYZ'}
print(getStudentName(103, dict1))
```

```
ABC
```

14. 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 [18]:
```
def sumScoresEndingWithZero(scores):
    total = 0
    for score in scores:
        if score % 10 == 0:
            total += score
    return total

scores = [200, 456, 300, 100, 234, 678]
print(sumScoresEndingWithZero(scores))
```

```
600
```

15. 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 [19]:
```
def invertDictionary(originalDict):
    invertedDict = {}
    for key in originalDict:
        value = originalDict[key]
        invertedDict[value] = key
    return invertedDict

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

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

16. 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 [20]:
```
def countCase(s1):
    noUpper = 0
    noLower = 0
    for ch in s1:
        if ch >= 'A' and ch <= 'Z':
```

```
            noUpper += 1
        elif ch >= 'a' and ch <= 'z':
            noLower += 1
    return noUpper, noLower

s1 = "AbcDEfgh"
upperCount, lowerCount = countCase(s1)
print("no_upper =", upperCount)
print("no_lower =", lowerCount)
```

```
no_upper = 3
no_lower = 5
```

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

In [21]:
```
getSmallest = lambda a, b: a if a < b else b
print(getSmallest(10, 25))
```

```
10
```

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

In [22]:
```
names = ['Siddharth', 'Aman', 'Priyanka', 'Rohit', 'Chandresh', 'Neha']
longNames = list(filter(lambda name: len(name) > 7, names))
print(longNames)
```

```
['Siddharth', 'Priyanka', 'Chandresh']
```

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

In [23]:
```
names = ['siddharth', 'aman', 'priyanka', 'rohit', 'chandresh', 'neha']
capitalizedNames = list(map(lambda name: name[0].upper() + name[1:], names))
print(capitalizedNames)
```

```
['Siddharth', 'Aman', 'Priyanka', 'Rohit', 'Chandresh', 'Neha']
```

20. 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 [24]:

```python
#Positional Arguments
def showDetailspos(name, age):
    print("Positional Arguments Name:", name)
    print("Positional Arguments Age:", age)

showDetailspos("Udit", 20)

#Keyword Arguments
def showDetailskey(name, age):
    print("Keyword Arguments Name:", name)
    print("Keyword Arguments Age:", age)

showDetailskey(age=20, name="Udit")

#Default Arguments
def showDetailsdef(name, age=18):
    print("Name:", name)
    print("Age:", age)

showDetailsdef("Udit")

#Variable Length Positional
def totalMarksvlp(*marks):
    total = 0
    for m in marks:
        total += m
    print("Variable Length Positional Total Marks:", total)

totalMarksvlp(70, 80, 90, 85)

#Variable length keywords
def studentInfovlk(**kwargs):
    for key in kwargs:
        print("Variable length keywords" + key + ":", kwargs[key])

studentInfovlk(name="Udit", age=20, grade="A")

#Keyword only
def showResultko(name, *, grade, marks):
    print("Keyword only Name:", name)
    print("Keyword only Grade:", grade)
    print("Keyword only Marks:", marks)

showResultko("Udit", grade="A", marks=92)

#Positional Only
def showStudentpo(name, age, /):
    print("Positional Only Name:", name)
    print("Positional Only Age:", age)

showStudentpo("Udit", 20)
```

```
Positional Arguments Name: Udit
Positional Arguments Age: 20
Keyword Arguments Name: Udit
Keyword Arguments Age: 20
Name: Udit
Age: 18
Variable Length Positional Total Marks: 325
Variable length keywordsname: Udit
Variable length keywordsage: 20
Variable length keywordsgrade: A
Keyword only Name: Udit
Keyword only Grade: A
Keyword only Marks: 92
Positional Only Name: Udit
Positional Only Age: 20
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

Lab - 8

Roll No. : 110

Name : Udit Khalpada

1. import numpy library.

In [1]:
```
import numpy as np
```

2.Create an array of 10 zeros

In [2]:
```
print(np.zeros(10))
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

3. Create an array of 10 ones.

In [3]:
```
print(np.ones(10))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

4. Create an array of 10 fives

In [4]:
```
print(np.full(10, 5))
```

```
[5 5 5 5 5 5 5 5 5 5]
```

5. Create an array of integers from 10 to 50.

In [5]:
```
print(np.arange(10, 51))
```

```
[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50]
```

6. Create an array of all the even integers from 10 to 50.

In [6]:
```
print(np.arange(10, 51, 2))
```

```
[10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50]
```

7. Create a 3x3 matrix with values ranging from 0 to 8.

In [7]:
```
print(np.arange(9).reshape(3, 3))
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

8. Create a 3x3 identity matrix.

In [8]:
```
print(np.eye(3))
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

9. Use Numpy to generate a random number between 0 and 1

In [9]:
```
print(np.random.rand())
```

```
0.7805854170542816
```

10. Use Numpy to generate an array of 25 random numbers sampled from a standard normal distribution.

In [10]:
```
print(np.random.randn(25))
```

```
[-0.93844195  1.3223431  -0.12042212 -0.13327516 -1.0452298   0.22507651
 -2.00390552  0.59983835  0.29420393 -0.65297752  2.25313795 -0.91518061
  1.09865065  0.92361313 -1.73410482  0.57499933  0.04665246  0.37916231
  0.10967248 -2.08761967 -0.3193682  -1.07495095 -1.04178092  2.08070052
  0.70429462]
```

11. Create linspace array

In [11]:
```
print(np.linspace(0, 10, 5))
```

```
[ 0.   2.5  5.   7.5 10. ]
```

12. Create an array of 20 linearly spaced points between 0 and 1.

In [12]:
```
print(np.linspace(0, 1, 20))
```

```
[0.         0.05263158 0.10526316 0.15789474 0.21052632 0.26315789
 0.31578947 0.36842105 0.42105263 0.47368421 0.52631579 0.57894737
 0.63157895 0.68421053 0.73684211 0.78947368 0.84210526 0.89473684
 0.94736842 1.        ]
```

13. Create Random Integer Array

In [13]:
```
print(np.random.randint(1, 100, 10))
```

```
[ 1 75 42 27 32 91 58 97 93 49]
```

14. Create Random Integer Array and Reshape that Array

In [14]:
```
arr = np.random.randint(1, 100, 12).reshape(3, 4)
print(arr)
```

```
[[56  9 98 60]
 [15 57 10 94]
 [39 81 73  8]]
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

Lab - 9

Roll No. : 110

Name : Udit Khalpada

1. Create a Pandas Series containing names of 5 students.

In [1]:
```
import pandas as pd

students = pd.Series(["Udit", "Tipu", "ABC", "XYZ", "PQR"])
print(students)
```

```
0    Udit
1    Tipu
2     ABC
3     XYZ
4     PQR
dtype: object
```

2. Create a Series with student roll numbers as index and their IAT scores as values..

In [2]:
```
import pandas as pd

scores = pd.Series([85, 78, 92, 74, 88], index=[101, 102, 103, 104, 105])
print(scores)
```

```
101    85
102    78
103    92
104    74
105    88
dtype: int64
```

3. Create a time series (daily) from 2025-08-01 to 2025-08-10 representing attendance tracking for a student.

In [3]:
```
import pandas as pd

dates = pd.date_range(start="2025-08-01", end="2025-08-10", freq="D")
attendance = pd.Series([1, 1, 0, 1, 1, 1, 0, 1, 1, 1], index=dates)
print(attendance)
```

```
2025-08-01    1
2025-08-02    1
2025-08-03    0
2025-08-04    1
2025-08-05    1
2025-08-06    1
2025-08-07    0
2025-08-08    1
2025-08-09    1
2025-08-10    1
Freq: D, dtype: int64
```

4. Create a DataFrame for 10 students with the following columns: Roll No, Name, PDS, CA, CN, IAT.

(Use NumPy random module to generate scores)

In [3]:
```
import pandas as pd
import numpy as np

np.random.seed(1)
roll_no = range(101, 111)
names = ["Udit", "Riya", "Karan", "Sneha", "Vikram", "Meena", "Arjun", "Pooja"
pds = np.random.randint(60, 100, 10)
ca = np.random.randint(60, 100, 10)
cn = np.random.randint(60, 100, 10)
iat = np.random.randint(60, 100, 10)

df = pd.DataFrame({
    "Roll No": roll_no,
    "Name": names,
    "PDS": pds,
    "CA": ca,
    "CN": cn,
    "IAT": iat
})
print(df)
```

```
   Roll No    Name  PDS  CA  CN  IAT
0      101    Udit   97  72  89   67
1      102    Riya   72  67  74   82
2      103   Karan   68  66  64   61
3      104   Sneha   69  85  83   60
4      105  Vikram   71  80  83   77
5      106   Meena   65  97  90   68
6      107   Arjun   75  78  92   84
7      108   Pooja   60  80  82   73
8      109   Rahul   76  71  73   68
9      110    Neha   61  88  69   90
```

5. Display the first 3 rows of the DataFrame.

In [5]:
```
print(df.head(3))
```

```
   Roll No   Name  PDS  CA  CN  IAT
0      101   Amit   97  72  89   67
1      102   Riya   72  67  74   82
2      103  Karan   68  66  64   61
```

6. Display the last 2 rows of the DataFrame.

In [6]:
```
print(df.tail(2))
```

```
   Roll No   Name  PDS  CA  CN  IAT
8      109  Rahul   76  71  73   68
9      110   Neha   61  88  69   90
```

7. Use .describe() to summarize the numeric data.

In [7]:
```
print(df.describe())
```

```
         Roll No        PDS         CA         CN        IAT
count   10.00000  10.000000  10.000000  10.000000  10.000000
mean   105.50000  71.400000  78.400000  79.900000  73.000000
std      3.02765  10.469002   9.811558   9.480389  10.033278
min    101.00000  60.000000  66.000000  64.000000  60.000000
25%    103.25000  65.750000  71.250000  73.250000  67.250000
50%    105.50000  70.000000  79.000000  82.500000  70.500000
75%    107.75000  74.250000  83.750000  87.500000  80.750000
max    110.00000  97.000000  97.000000  92.000000  90.000000
```

8. Select only the Name column.

In [8]:
```
print(df["Name"])
```

```
0      Amit
1      Riya
2     Karan
3     Sneha
4    Vikram
5     Meena
6     Arjun
7     Pooja
8     Rahul
9      Neha
Name: Name, dtype: object
```

9. Select the columns PDS, CN, and IAT.

In [9]:
```
print(df[["PDS", "CN", "IAT"]])
```

```
   PDS  CN  IAT
0   97  89   67
1   72  74   82
2   68  64   61
3   69  83   60
4   71  83   77
5   65  90   68
6   75  92   84
7   60  82   73
8   76  73   68
9   61  69   90
```

10. Select the row with Roll No = 105 using loc.

In [10]:
```
print(df.loc[df["Roll No"] == 105])
```

```
   Roll No    Name  PDS  CA  CN  IAT
4      105  Vikram   71  80  83   77
```

11. Select the 4th row using iloc.

In [11]:
```
print(df.iloc[3])
```

```
Roll No      104
Name       Sneha
PDS           69
CA            85
CN            83
IAT           60
Name: 3, dtype: object
```

12. Select students with marks in PDS > 80.

In [12]:
```
print(df[df["PDS"] > 80])
```

```
   Roll No  Name  PDS  CA  CN  IAT
0      101  Amit   97  72  89   67
```

13. Select students with marks in CA < 70.

In [13]:
```
print(df[df["CA"] < 70])
```

```
   Roll No   Name  PDS  CA  CN  IAT
1      102   Riya   72  67  74   82
2      103  Karan   68  66  64   61
```

14. Select students with marks in CN > 85 and PDS > 80

In [14]:
```
print(df[(df["CN"] > 85) & (df["PDS"] > 80)])
```

```
   Roll No  Name  PDS  CA  CN  IAT
0      101  Amit   97  72  89   67
```

15. Add a new column Total Marks = PDS + CA + CN + IAT.

In [15]:
```
df["Total Marks"] = df["PDS"] + df["CA"] + df["CN"] + df["IAT"]
print(df)
```

```
   Roll No    Name  PDS  CA  CN  IAT  Total Marks
0      101    Amit   97  72  89   67          325
1      102    Riya   72  67  74   82          295
2      103   Karan   68  66  64   61          259
3      104   Sneha   69  85  83   60          297
4      105  Vikram   71  80  83   77          311
5      106   Meena   65  97  90   68          320
6      107   Arjun   75  78  92   84          329
7      108   Pooja   60  80  82   73          295
8      109   Rahul   76  71  73   68          288
9      110    Neha   61  88  69   90          308
```

16. Create a new DataFrame of students with Total Marks > 320.

In [16]:
```
df_high = df[df["Total Marks"] > 320]
print(df_high)
```

```
   Roll No   Name  PDS  CA  CN  IAT  Total Marks
0      101   Amit   97  72  89   67          325
6      107  Arjun   75  78  92   84          329
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

Lab - 10

Roll No. : 110

Name : Udit Khalpada

Student Score (.csv file)

1. Load the file student_scores.csv.

In [1]:
```
import pandas as pd
df = pd.read_csv("students_score.csv")
print("1. File loaded successfully with", len(df), "rows and", len(df.columns)
```

```
1. File loaded successfully with 10 rows and 5 columns.
```

2. Show the first 5 rows.

In [2]:
```
print(df.head())
```

```
   RollNo   Name  Math  Science  English
0     101   Aman    78       85       90
1     102   Riya    65       82       75
2     103  Kiran    90       88       92
3     104   Ravi    70       79       85
4     105  Meera    88       92       91
```

3. Display the index and column names.

In [3]:
```
print("Index ->", df.index)
print("Columns ->", df.columns.tolist())
```

```
Index -> RangeIndex(start=0, stop=10, step=1)
Columns -> ['RollNo', 'Name', 'Math', 'Science', 'English']
```

4. Get descriptive statistics using .describe().

In [4]:
```
print(df.describe())
```

```
          RollNo      Math    Science   English
count   10.00000  10.00000  10.000000  10.00000
mean   105.50000  77.40000  84.200000  85.60000
std      3.02765   8.40899   6.033241   6.60303
min    101.00000  65.00000  73.000000  75.00000
25%    103.25000  70.50000  79.750000  81.25000
50%    105.50000  77.50000  86.000000  87.00000
75%    107.75000  83.25000  88.000000  90.75000
max    110.00000  90.00000  92.000000  93.00000
```

5. Select the Name and Math columns.

In [5]:
```
print(df[["Name", "Math"]])
```

```
    Name  Math
0   Aman    78
1   Riya    65
2  Kiran    90
3   Ravi    70
4  Meera    88
5   John    81
6   Sara    77
7    Tom    69
8  Alice    84
9   Neha    72
```

6. Find all students who scored more than 80 in Science.

In [6]:
```
print(df[df["Science"] > 80])
```

```
   RollNo   Name  Math  Science  English
0     101   Aman    78       85       90
1     102   Riya    65       82       75
2     103  Kiran    90       88       92
4     105  Meera    88       92       91
5     106   John    81       87       93
6     107   Sara    77       90       89
8     109  Alice    84       88       85
```

7. Find all students with English < 75.

In [7]:
```
print(df[df["English"] < 75])
```

```
Empty DataFrame
Columns: [RollNo, Name, Math, Science, English]
Index: []
```

8. Extract the last 3 rows.

In [8]:
```
print(df.tail(3))
```

```
   RollNo   Name  Math  Science  English
7     108    Tom    69       73       80
8     109  Alice    84       88       85
9     110   Neha    72       78       76
```

9. Sort the DataFrame by Math in descending order.

(Hint : use df.sort_values(by = "column_name", ascending = True/ False))

In [9]:
```
print(df.sort_values(by="Math", ascending=False))
```

```
   RollNo   Name  Math  Science  English
2     103  Kiran    90       88       92
4     105  Meera    88       92       91
8     109  Alice    84       88       85
5     106   John    81       87       93
0     101   Aman    78       85       90
6     107   Sara    77       90       89
9     110   Neha    72       78       76
3     104   Ravi    70       79       85
7     108    Tom    69       73       80
1     102   Riya    65       82       75
```

10. Set RollNo as the index and rename it "Student ID".

In [10]:
```
df_idx = df.set_index("RollNo")
df_idx.index.name = "Student ID"
print(df_idx.head())
```

```
             Name  Math  Science  English
Student ID
101          Aman    78       85       90
102          Riya    65       82       75
103         Kiran    90       88       92
104          Ravi    70       79       85
105         Meera    88       92       91
```

11. Reset the index back.

In [11]:
```
print(df_idx.reset_index().head())
```

```
   Student ID   Name  Math  Science  English
0         101   Aman    78       85       90
1         102   Riya    65       82       75
2         103  Kiran    90       88       92
3         104   Ravi    70       79       85
4         105  Meera    88       92       91
```

12. Add a new column Total = Math + Science + English.

In [12]:
```
df["Total"] = df[["Math", "Science", "English"]].sum(axis=1)
print(df.head())
```

```
   RollNo   Name  Math  Science  English  Total
0     101   Aman    78       85       90    253
1     102   Riya    65       82       75    222
2     103  Kiran    90       88       92    270
3     104   Ravi    70       79       85    234
4     105  Meera    88       92       91    271
```

13. Find the student with the highest Total score.

In [13]:
```
print(df[df["Total"] == df["Total"].max()])
```

```
   RollNo   Name  Math  Science  English  Total
4     105  Meera    88       92       91    271
```

14. Get the Top 3 students with the highest total score.

In [14]:
```
print(df.sort_values(by="Total", ascending=False).head(3))
```

```
   RollNo   Name  Math  Science  English  Total
4     105  Meera    88       92       91    271
2     103  Kiran    90       88       92    270
5     106   John    81       87       93    261
```

15. Get the average marks in each subject.

In [15]:
```
print(df[["Math", "Science", "English"]].mean())
```

```
Math       77.4
Science    84.2
English    85.6
dtype: float64
```

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

Lab - 11

Roll No. : 110

Name : Udit Khalpada

GroupBy

In [5]:
```
import pandas as pd
```

In [6]:
```
students = {
    'RollNo': [101, 102, 103, 104, 105, 106],
    'Name': ['Aarav', 'Diya', 'Ishaan', 'Meera', 'Kabir', 'Anaya'],
    'Dept': ['CSE', 'CSE', 'ECE', 'ECE', 'ME', 'CSE'],
    'Math': [88, 92, None, 74, 69, 85],
    'Science': [91, None, 78, 84, 76, 89],
    'English': [85, 87, 80, None, 74, 90]
}
```

1. Group students by Dept and find the average marks in each subject.

In [9]:
```
df = pd.DataFrame(students)
```

In [10]:
```
df_grp = df.groupby('Dept')

df_grp[['Math','Science','English']].mean()
```

Out[10]:

Dept	Math	Science	English
CSE	88.333333	90.0	87.333333
ECE	74.000000	81.0	80.000000
ME	69.000000	76.0	74.000000

2. Find the highest Math score in each department.

In [11]:
```
df_grp['Math'].max()
```

Out[11]:
```
Dept
CSE    92.0
ECE    74.0
ME     69.0
Name: Math, dtype: float64
```

3. Count how many students belong to each department.

In [12]:
```
df_grp['RollNo'].count()
```

Out[12]:
```
Dept
CSE    3
ECE    2
ME     1
Name: RollNo, dtype: int64
```

4. Compute the minimum, maximum, and mean of Science marks.

In [17]:
```
df['Science'].min()
```

Out[17]:
```
76.0
```

In [16]:
```
df['Science'].max()
```

Out[16]:
```
91.0
```

In [18]:
```
df['Science'].mean()
```

Out[18]:
```
83.6
```

5. For each department, apply multiple aggregations:

Math: mean, max

Science: min, count

In [19]:
```
df_grp['Math'].mean()
```

Out[19]:
```
Dept
CSE    88.333333
ECE    74.000000
ME     69.000000
Name: Math, dtype: float64
```

In [20]:
```
df_grp['Math'].max()
```

Out[20]:
```
Dept
CSE    92.0
ECE    74.0
ME     69.0
Name: Math, dtype: float64
```

In [21]:
```
df_grp['Science'].min()
```

Out[21]:
```
Dept
CSE    89.0
ECE    78.0
ME     76.0
Name: Science, dtype: float64
```

In [22]:
```
df_grp['Science'].count()
```

Out[22]:
```
Dept
CSE    2
ECE    2
ME     1
Name: Science, dtype: int64
```

Merge

In [23]:
```
attendance = {
    'RollNo': [101, 102, 103, 104, 107],
    'Attendance(%)': [92, 85, 88, 76, 90]
}
```

In [24]:
```
df1 = pd.DataFrame(students)
```

In [25]:
```
df2 = pd.DataFrame(attendance)
```

6. Merge students and attendance on RollNo (inner join).

In [26]:
```
pd.merge(df1, df2, on = 'RollNo', how = "inner")
```

Out[26]:

	RollNo	Name	Dept	Math	Science	English	Attendance(%)
0	101	Aarav	CSE	88.0	91.0	85.0	92
1	102	Diya	CSE	92.0	NaN	87.0	85
2	103	Ishaan	ECE	NaN	78.0	80.0	88
3	104	Meera	ECE	74.0	84.0	NaN	76

7. Merge students and sports (outer join) – identify students without sports info.

In [28]:
```
df3 = pd.DataFrame(sports)
```

In [40]:
```
merge_df = pd.merge(df1, df3, on = 'RollNo', how = "outer")
merge_df = merge_df[merge_df['Sport'].isna()]
merge_df
```

Out[40]:

	RollNo	Name	Dept	Math	Science	English	Sport
1	102	Diya	CSE	92.0	NaN	87.0	NaN
3	104	Meera	ECE	74.0	84.0	NaN	NaN
5	106	Anaya	CSE	85.0	89.0	90.0	NaN

join

8. Convert students and attendance into DataFrames with RollNo as index. Perform a left join on index.

In [59]:
```
df_students = pd.DataFrame(students)
df_students = df_students.set_index('RollNo')
df_students
```

Out[59]:

RollNo	Name	Dept	Math	Science	English
101	Aarav	CSE	88.0	91.0	85.0
102	Diya	CSE	92.0	NaN	87.0
103	Ishaan	ECE	NaN	78.0	80.0
104	Meera	ECE	74.0	84.0	NaN
105	Kabir	ME	69.0	76.0	74.0
106	Anaya	CSE	85.0	89.0	90.0

In [58]:
```
df_attendance = pd.DataFrame(attendance)
df_attendance = df_attendance.set_index('RollNo')
df_attendance
```

Out[58]:

RollNo	Attendance(%)
101	92
102	85
103	88
104	76
107	90

In [60]:
```
df_students.join(df_attendance, how='left')
```

Out[60]:

RollNo	Name	Dept	Math	Science	English	Attendance(%)
101	Aarav	CSE	88.0	91.0	85.0	92.0
102	Diya	CSE	92.0	NaN	87.0	85.0
103	Ishaan	ECE	NaN	78.0	80.0	88.0
104	Meera	ECE	74.0	84.0	NaN	76.0
105	Kabir	ME	69.0	76.0	74.0	NaN
106	Anaya	CSE	85.0	89.0	90.0	NaN

concat

9. Create a new small DataFrame of newly admitted students:

In [62]:
```
new_students = {
    'RollNo': [109, 110],
    'Name': ['Rohan', 'Sara'],
    'Dept': ['ECE', 'CSE'],
    'Math': [81, 95],
    'Science': [79, 88],
    'English': [83, 91]
}
```

In [63]:
```
df_new_students = pd.DataFrame(new_students)
```

10. Concatenate this DataFrame with the original students.

In [64]:
```
pd.concat([df1,df_new_students])
```

Out[64]:

	RollNo	Name	Dept	Math	Science	English
0	101	Aarav	CSE	88.0	91.0	85.0
1	102	Diya	CSE	92.0	NaN	87.0
2	103	Ishaan	ECE	NaN	78.0	80.0
3	104	Meera	ECE	74.0	84.0	NaN
4	105	Kabir	ME	69.0	76.0	74.0
5	106	Anaya	CSE	85.0	89.0	90.0
0	109	Rohan	ECE	81.0	79.0	83.0
1	110	Sara	CSE	95.0	88.0	91.0

11. Concatenate students[['RollNo','Name']] with sports column-wise.

In [65]:
```
sports_df = pd.DataFrame(sports)
```

In [67]:
```
# student_df = pd.DataFrame()
pd.concat([df1[['RollNo','Name']], sports_df], axis= 1)
```

Out[67]:

	RollNo	Name	RollNo	Sport
0	101	Aarav	101.0	Cricket
1	102	Diya	103.0	Football
2	103	Ishaan	105.0	Badminton
3	104	Meera	107.0	Hockey
4	105	Kabir	NaN	NaN
5	106	Anaya	NaN	NaN

In [27]:
```
sports = {
    'RollNo': [101, 103, 105, 107],
    'Sport': ['Cricket', 'Football', 'Badminton', 'Hockey']
}
```

Handle missing value

In [71]:
```
import numpy as np
```

In [78]:
```
di = {'Score1': [100, 90, np.nan, 95],
        'Score2': [30, 45, 56, np.nan],
        'Score3': [np.nan, 40, 80, 98]}
```

In [79]:
```
df = pd.DataFrame(di)
```

In [80]:
```
df.to_csv('Scores.csv')
```

12. Read one csv file of your choice

Use different techniques to deal with missing values in the file

In [87]:
```
df = pd.read_csv('Scores.csv',index_col=0)
```

In [88]:
```
df
```

Out[88]:

	Score1	Score2	Score3
0	100.0	30.0	NaN
1	90.0	45.0	40.0
2	NaN	56.0	80.0
3	95.0	NaN	98.0

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Python for Data Science - 2305CS303

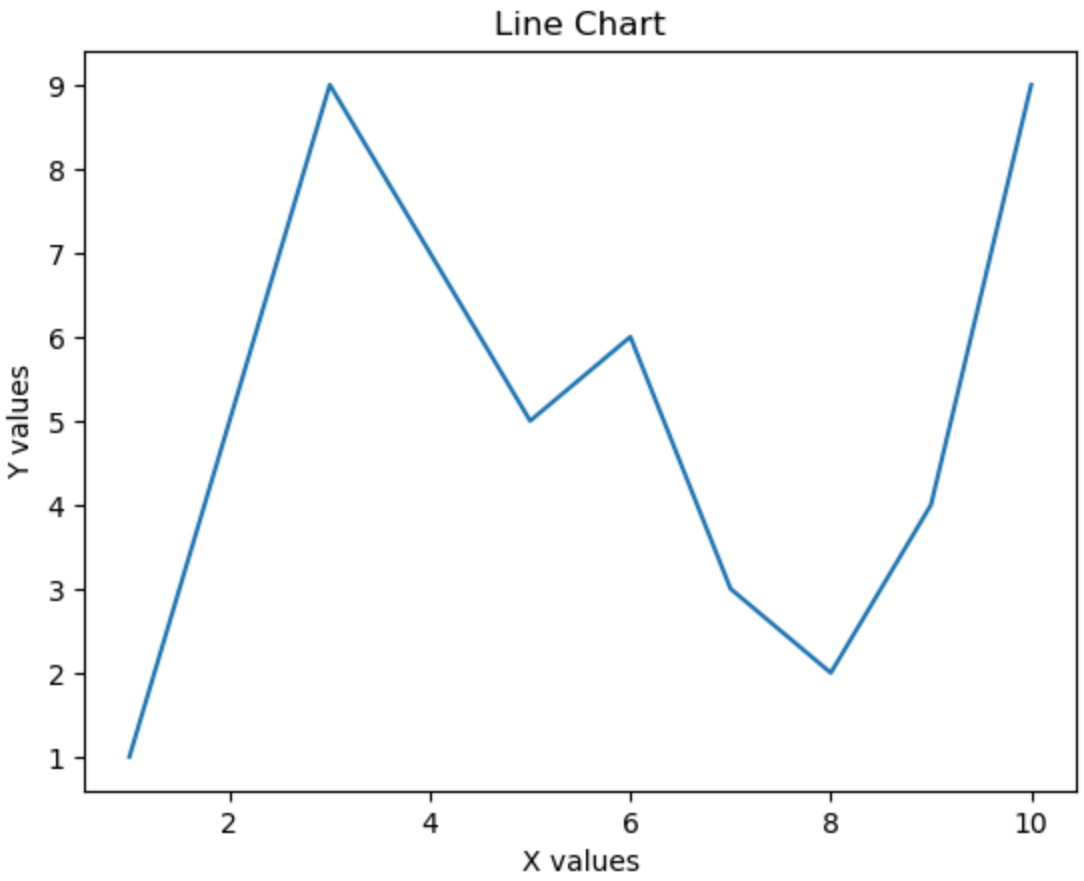
Lab - 12

Roll No. : 110

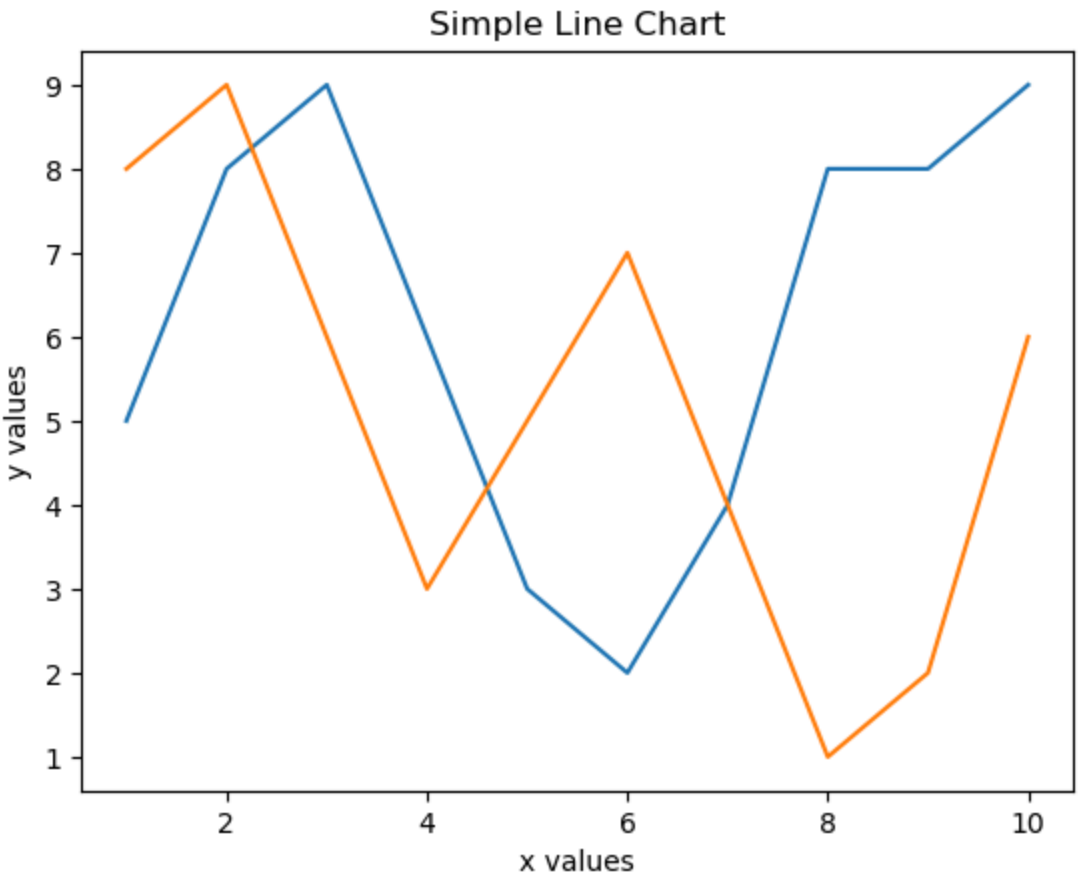
Name : Udit Khalpada

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

In [10]:
```
x = range(1,11)
y = [1,5,9,7,5,6,3,2,4,9]

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

plt.plot(x, y)          # plot line
plt.xlabel("X values")  # label for x-axis
plt.ylabel("Y values")  # label for y-axis
plt.title("Line Chart")
plt.show()
```



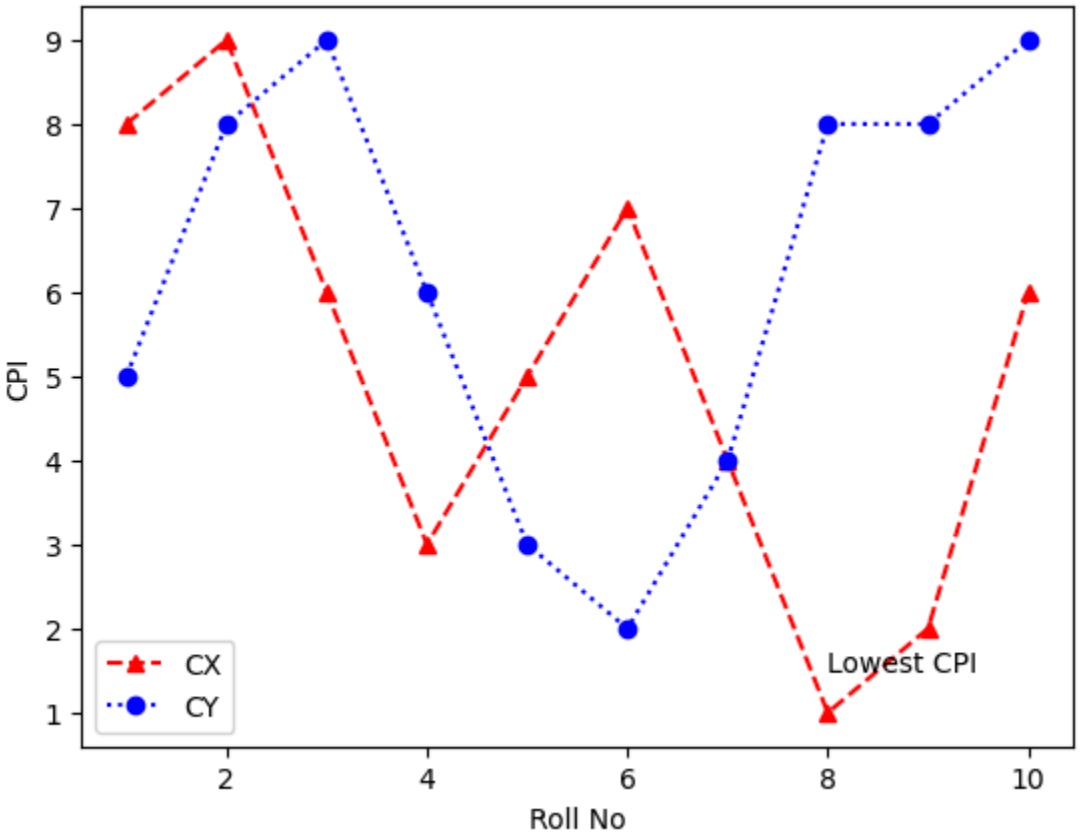
In [22]:
```
x = [1,2,3,4,5,6,7,8,9,10]
cxMarks = [5,8,9,6,3,2,4,8,8,9]
cyMarks = [8,9,6,3,5,7,4,1,2,6]

# write a code to display two lines in a line chart (data given above)
plt.plot(x, cxMarks)
plt.plot(x, cyMarks)
plt.xlabel("x values")
plt.ylabel("y values")
plt.title("Simple Line Chart")
plt.show()
```



In [21]:

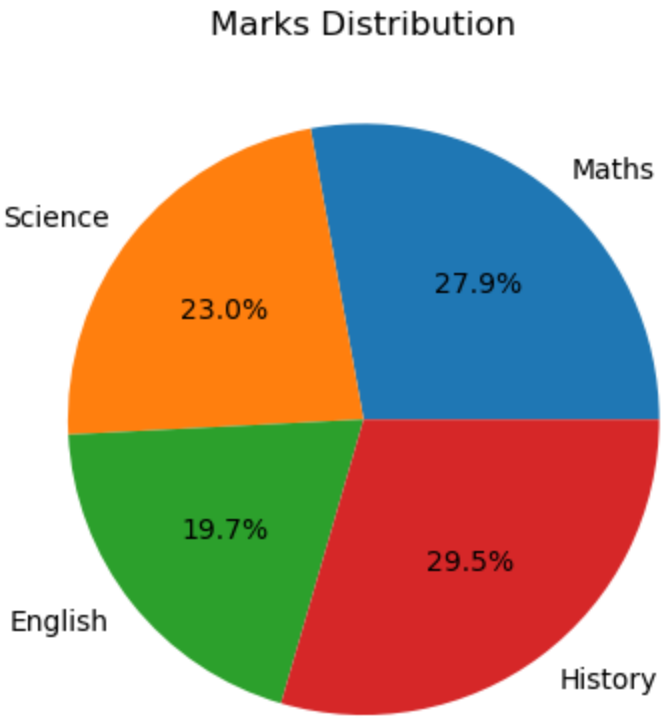
```
x = range(1,11,1)
cxMarks= [8,9,6,3,5,7,4,1,2,6]
cyMarks= [5,8,9,6,3,2,4,8,8,9]


# write a code to generate below graph

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

# Plot CX with red dashed line, triangle markers
plt.plot(x, cxMarks, 'r--^', label="CX")

# Plot CY with blue dotted line, circle markers
plt.plot(x, cyMarks, 'b:o', label="CY")

# Labels and title
plt.xlabel("Roll No")
plt.ylabel("CPI")

# Annotate the lowest CPI point
plt.annotate("Lowest CPI",
             xy=(8,1),        # point to annotate
             xytext=(8,1.5))  # text position
```

```
# Show legend
plt.legend()

# Display graph
plt.show()
```



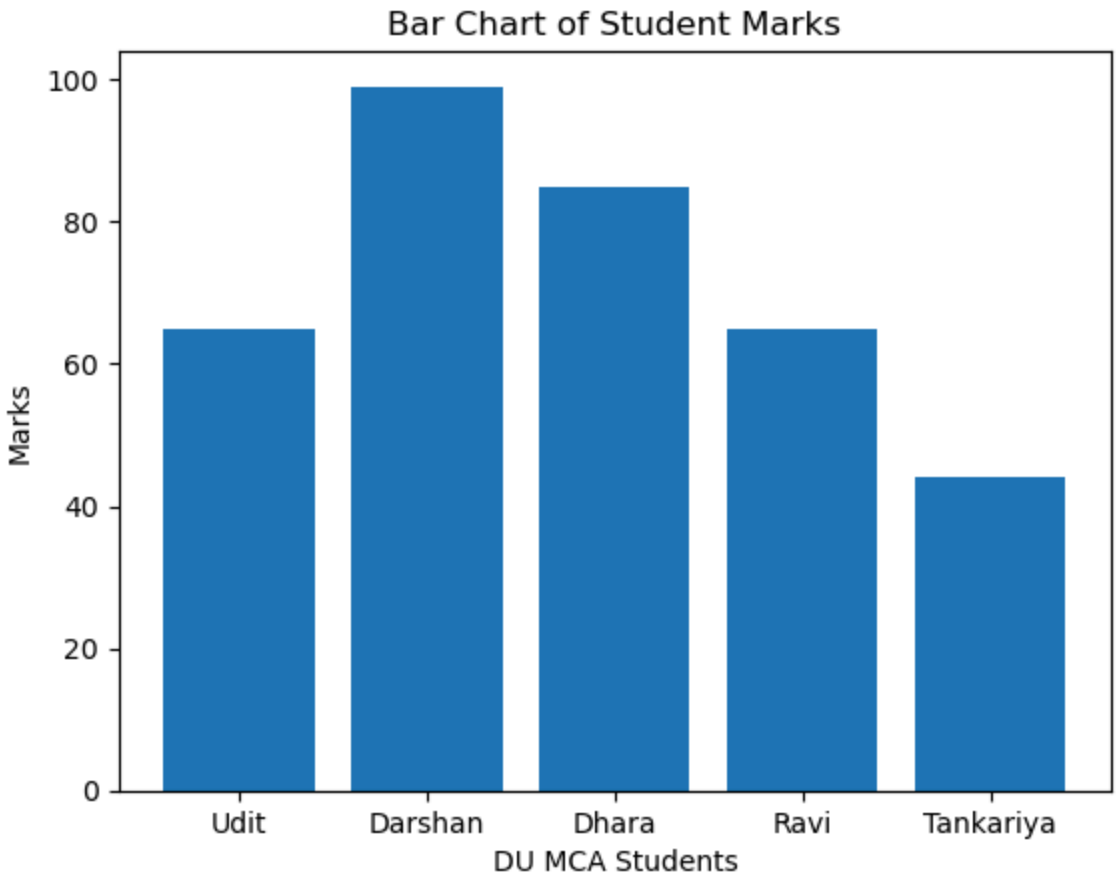
04) WAP to demonstrate the use of Pie chart.

In [7]:
```
subjects = ["Maths", "Science", "English", "History"]
marks = [85, 70, 60, 90]

plt.pie(marks, labels=subjects, autopct="%1.1f%%")
plt.title("Marks Distribution")
plt.show()
```



05) WAP to demonstrate the use of Bar chart.

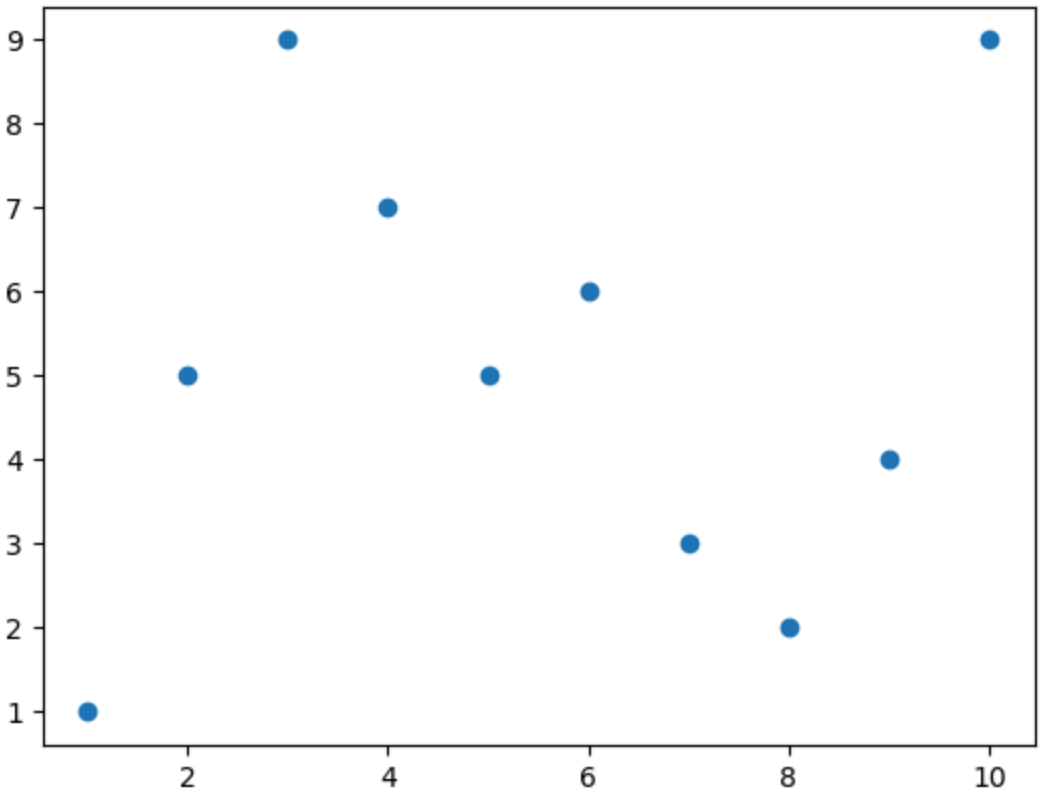
In [35]:
```
students = ["Udit", "Darshan", "Dhara", "Ravi", "Tankariya"]
marks = [65, 99, 85, 65, 44]

plt.bar(students, marks)
plt.xlabel("DU MCA Students")
plt.ylabel("Marks")
plt.title("Bar Chart of Student Marks")
plt.show()
```



06) WAP to demonstrate the use of Scatter Plot.

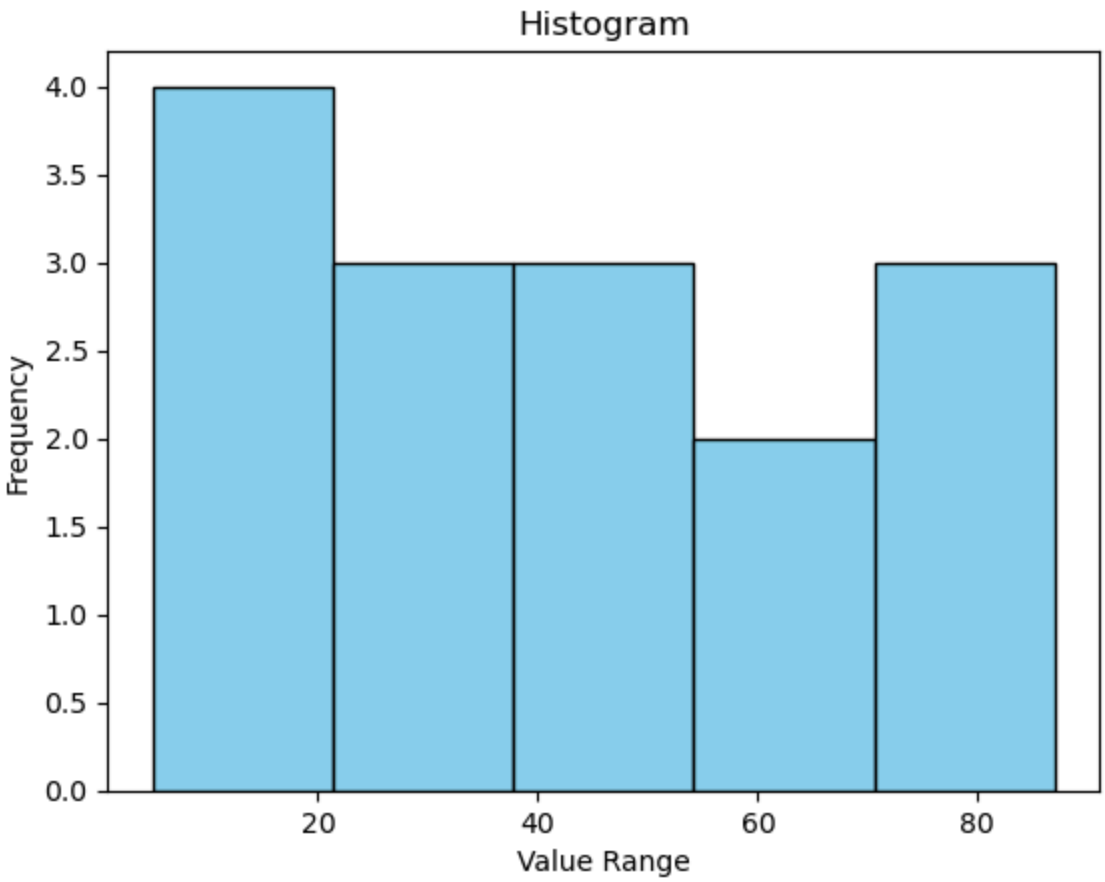
In [29]:
```
plt.scatter(x,y)
plt.show
```

Out[29]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



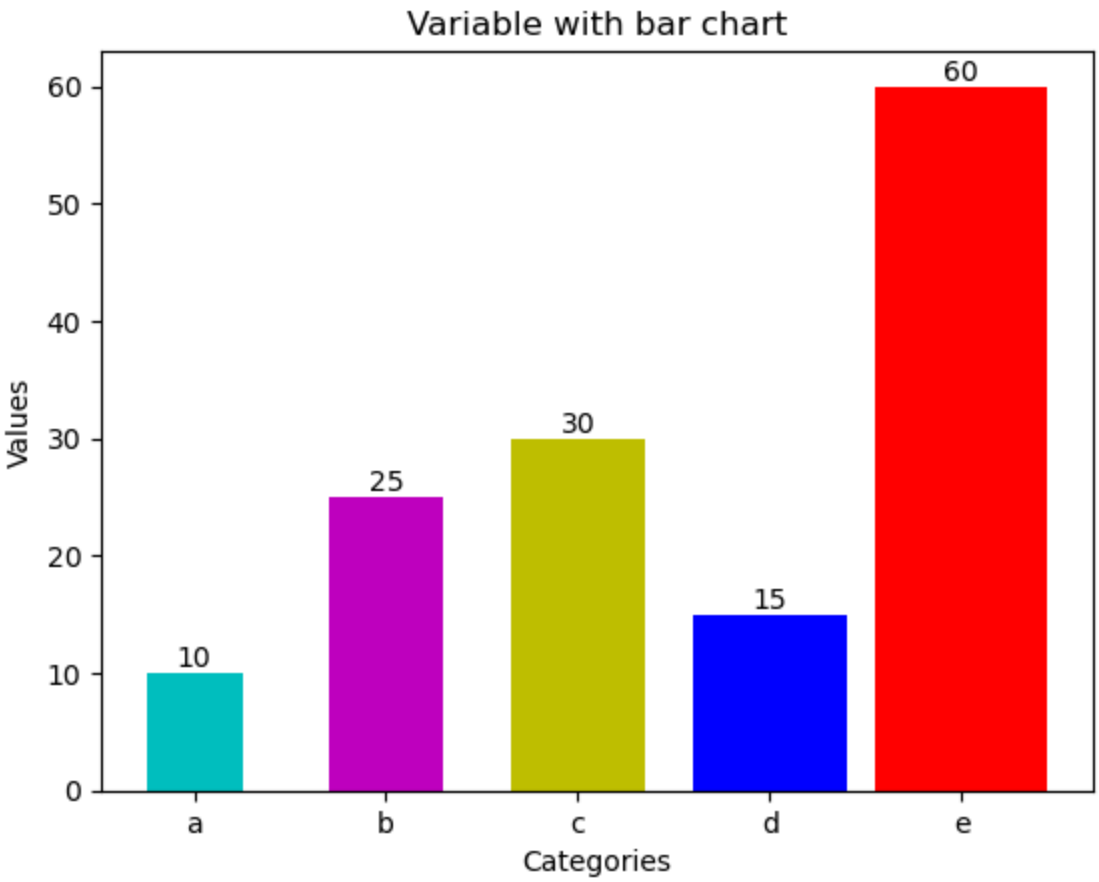
07) WAP to demonstrate the use of Histogram.

In [23]:
```
data = [22,87,5,43,56,73,55,54,11,20,51,5,79,31,27]

plt.hist(data, bins=5, color="skyblue", edgecolor="black")
plt.xlabel("Value Range")
plt.ylabel("Frequency")
plt.title("Histogram")
plt.show()
```



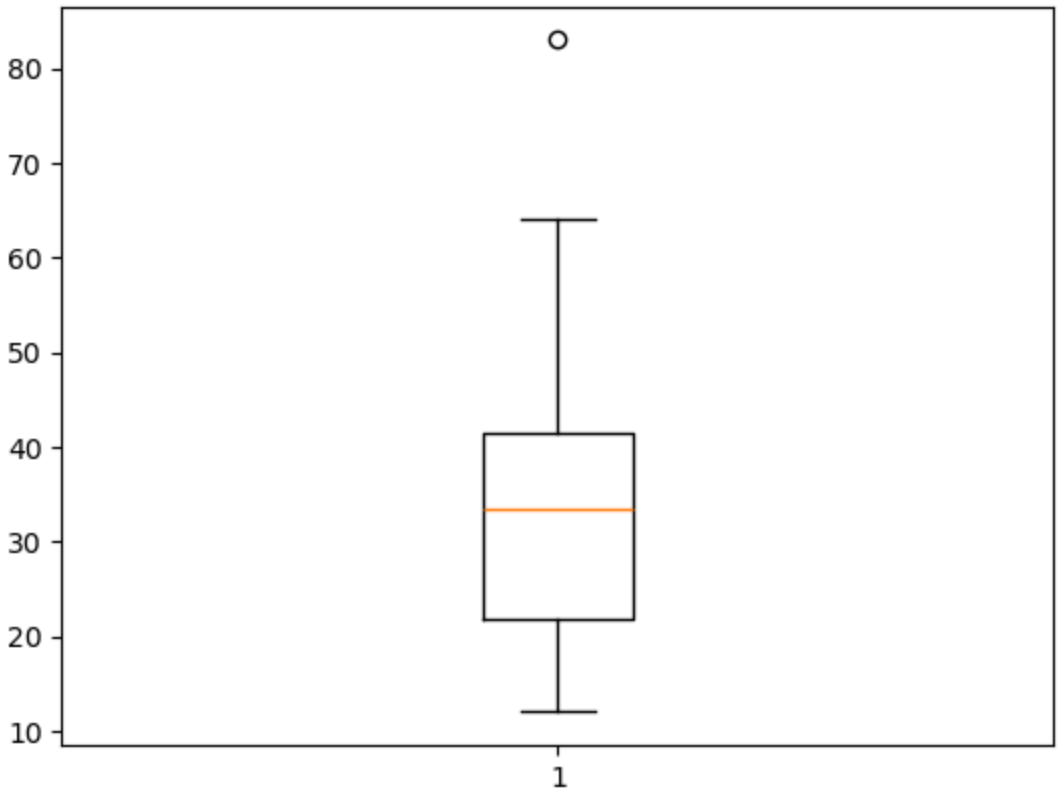
08) WAP to display the value of each bar in a bar chart using Matplotlib.

In [37]:
```
import matplotlib.pyplot as plt

x = ['a', 'b', 'c', 'd', 'e']
y = [10, 25, 30, 15, 60]
widthh = [0.5, 0.6, 0.7, 0.8, 0.9]
c = ['c', 'm', 'y', 'b', 'r']

bars = plt.bar(x, y, width=widthh, color=c)
plt.title("Variable with bar chart")
plt.xlabel("Categories")
plt.ylabel("Values")

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

plt.show()
```



09) WAP to create a Box Plot.

In [36]:
```
data = [83,22,33,12,34,64,21,34]
plt.boxplot(data)
plt.show()
```


80
70
60
50
40
30
20
10
1