Darshan UNIVERSITY

योग: कर्मसु कौशलम्

PDS - 2305CS303

Lab - 1

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

01) WAP to print “Hello World”

In [1]:
```
print('Hello World!')
```

```
Hello World!
```

02) WAP to print your address i) using single print ii) using multiple print

In [2]:
```
print("Bedipara Sitaram Road,Rajkot")
print("Bedipara")
print("Sitaram Road")
print("Rajkot")
```

```
Bedipara Sitaram Road,Rajkot
Bedipara
Sitaram Road
Rajkot
```

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

In [3]:
```
a = 10
b = 20
print(a+b)
```

```
30
```

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

In [4]:
```
num1=10
num2=20
```

```
print((num1+num2)/2)
```

```
15.0
```

05) WAP to add two number entered by user.

In [5]:
```
a = int(input("Enter 1 Number"))
b = int(input("Enter 2 Number"))
print(a+b)
```

```
Enter 1 Number12
Enter 2 Number22
34
```

06) Purposefully raise Indentation Error and Correct it.

In [7]:
```
if 1==1:
 print("Okey")
```

```
Okey
```

07) WAP to calculate simple interest.

In [8]:
```
P = float(input("Enter price"))
R = float(input("Enter Rate of Interest"))
N = int(input("Enter Number of days"))
print("The Simple Interest is:",(P*R*N)/100)
```

```
Enter price11
Enter Rate of Interest32
Enter Number of days5
The Simple Interest is: 17.6
```

08) WAP Calculate Area and Circumfrence of Circle

In [10]:
```
PI = 3.14
R = float(input("Enter Rate of Interest"))
print("The Area is",PI*R*R)
print("The Circumfrence of circle is:",2*PI*R)
```

```
Enter Rate of Interest12
The Area is 452.15999999999997
The Circumfrence of circle is: 75.36
```

09) WAP to print Multiplication table of given number without using loops.

In [12]:
```
num = int(input("Enter Number"))
print(num,"*",1,"=",(num*1))
print(num,"*",2,"=",(num*2))
print(num,"*",3,"=",(num*3))
print(num,"*",4,"=",(num*4))
print(num,"*",5,"=",(num*5))
```

```
print(num,"*",6,"=",(num*6))
print(num,"*",7,"=",(num*7))
print(num,"*",8,"=",(num*8))
print(num,"*",9,"=",(num*9))
print(num,"*",10,"=",(num*10))
```

```
Enter Number5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

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

In [13]:
```
h = int(input("Enter height:"))
b = int(input("Enter breath:"))
area = h*b*0.5
print("The area of Tringle is:",area)
```

```
Enter height2
Enter breath1
The area of Tringle is: 1.0
```

11) WAP to convert degree to Fahrenheit and vice versa.

In [16]:
```
degree = float(input("Enter degree to convert in fahrenhit:"))
fahrenheit = float(input("Enter fahrenhit to convert in degree:"))
print(degree,"Degree:",((degree*9/5)+32),"Fahrenheit")
print(fahrenheit,"fahrenheit:",((fahrenheit-32)*5/9),"Degree")
```

```
Enter degree to convert in fahrenhit:0
Enter fahrenhit to convert in degree:32
0.0 Degree: 32.0 Fahrenheit
32.0 fahrenheit: 0.0 Degree
```

12) WAP to calculate total marks and Percentage.

In [18]:
```
sub1 = int(input("Enter Marks of Subject1:"))
sub2 = int(input("Enter Marks of Subject2:"))
sub3 = int(input("Enter Marks of Subject3:"))
sub4 = int(input("Enter Marks of Subject4:"))
sub5 = int(input("Enter Marks of Subject5:"))

totalmarks = sub1+sub2+sub3+sub4+sub5
percentage = totalmarks/5
print("The Total Marks is:",totalmarks," and total percentage is:",percentag
```

```
Enter Marks of Subject1:11
Enter Marks of Subject2:22
Enter Marks of Subject3:33
Enter Marks of Subject4:44
Enter Marks of Subject5:55
The Total Marks is: 165  and total percentage is: 33.0
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

PDS - 2305CS303

Lab - 2

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

if..else..

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

In [1]:
```
num = int(input("Enter a Number:"))
if num < 0:
    print("The Number is Nagative:")
elif num > 0:
    print("The Number is Positive:")
else:
    print("The Number is Zero:")
```

```
Enter a Number: 5
The Number is Positive:
```

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

In [2]:
```
num = int(input("Enter a Number:"))
if num %2== 0:
    print("The Number is Even:")
else:
    print("The Number is odd:")
```

```
Enter a Number: 5
The Number is odd:
```

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

In [3]:
```
num1 = int(input("Enter a Number 1:"))
num2 = int(input("Enter a Number 2:"))
if num1>num2:
    print("The Num1 is Greater:")
else:
    print("The Num2 is Greater:")
# print(num1 if num1>num2 else num2)
```

```
Enter a Number 1: 22
Enter a Number 2: 34
The Num2 is Greater:
```

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

In [4]:
```
num1 = int(input("Enter a Number 1:"))
num2 = int(input("Enter a Number 2:"))
num3 = int(input("Enter a Number 2:"))
print(num1 if num1>num2 and num1>num3 else num2 if num2>num3 else num3 )
```

```
Enter a Number 1: 34
Enter a Number 2: 35
Enter a Number 2: 36
36
```

05) WAP to check whether the given year is leap year or not.

[If a year can be divisible by 4 but not divisible by 100 then it is leap year but if it is divisible by 400 then it is leap year]

In [5]:
```
num1 = int(input("Enter a Year:"))
if (num1%4==0 and num1%100!=0) or num1%400==0:
    print("The Given Year is Leap Year")
else:
     print("The Given Year is Not Leap Year")
```

```
Enter a Year: 2024
The Given Year is Leap Year
```

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

In [6]:
```
choice = int(input("Enter a Choice:"))
match choice :
    case 1:
        print("The Day is Monday")
    case 2:
        print("The Day is Tuesday")
    case 3:
        print("The Day is Wednesday")
    case 4:
        print("The Day is Thursday")
```

```
    case 5:
        print("The Day is Friday")
    case 6:
        print("The Day is Saturday")
    case 7:
        print("The Day is Sunday")
```

```
Enter a Choice: 4
The Day is Thursday
```

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

In [7]:
```
num1 = int(input("Enter a Number 1:"))
num2 = int(input("Enter a Number 2:"))
choice = input("Enter choice:")
if(choice=='+'):
    print(num1+num2)
elif(choice=='-'):
   print(num1-num2)
elif(choice=='*'):
    print(num1*num2)
elif(choice=='/'):
    print(int(num1/num2))
else:
    print("Invalid Choice!")
```

```
Enter a Number 1: 12
Enter a Number 2: 45
Enter choice: +
57
```

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

a. First 1 to 50 units – Rs. 2.60/unit b. Next 50 to 100 units – Rs. 3.25/unit c. Next 100 to 200 units – Rs. 5.26/unit d. above 200 units – Rs. 8.45/unit

In [8]:
```
unit = int(input("Enter unit:"))
rs = 0
if(unit<=50):
    rs=unit*2.60
elif(unit<=100):
    rs=50*2.60+(unit-50)*3.25
elif(unit<=200):
    rs = 50*2.60+50*3.25+(unit-100)*5.26
else:
    rs = 50*2.60+50*3.25+100*5.26+(unit-200)*8.45

print("The Bill is:",rs)
```

```
Enter unit: 12
The Bill is: 31.200000000000003
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2101CS405

Lab - 3

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

for and while loop

01) 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
```

02) WAP to print 1 to n

In [3]:
```
num = int(input("Enter a Number:"))
for i in range(1,num+1):
    print(i)
```

```
1
2
3
4
5
```

03) WAP to print odd numbers between 1 to n

In [4]:
```
num = int(input("Enter a Number:"))
for i in range(1,num+1):
    if i%2!=0:
        print(i)
```

```
1
3
5
7
9
```

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

In [1]:
```
num1 = int(input("Enter a 1 Number:"))
num2 = int(input("Enter a 2 Number:"))
for i in range(num1,num2+1):
    if i%2==0 and i%3!=0:
        print(i)
```

```
2
4
8
10
```

05) WAP to print sum of 1 to n numbers

In [7]:
```
num = int(input("Enter a Number:"))
sum = 0
for i in range(1,num+1):
    sum += i
print(sum)
```

```
15
```

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

In [15]:
```
for i in range(1,10+1):
    print(i**2)
```

```
1
4
9
16
25
36
49
64
81
100
```

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

In [11]:
```
sum=0
for i in range(1,10+1):
    if i%2==0:
        sum -= i
    else:
        sum += i
print(sum)
```

```
-5
```

08) WAP to print multiplication table of given number.

In [17]:
```
num = int(input("Enter a Number:"))
for i in range(1,11):
    print(f"{num} * {i} = {num*i}")
```

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

09) WAP to find factorial of the given number

In [19]:
```
num = int(input("Enter a Number:"))
fact=1
for i in range(1,num+1):
    fact*=i

print(fact)
```

```
120
```

10) WAP to find factors of the given number

In [3]:
```
num = int(input("Enter a Number:"))
for i in range(2,num):
   if num%i==0:
       print(i)
```

```
2
4
5
10
```

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

In [2]:
```
num = int(input("Enter a Number:"))
flag = True
for i in range(2,num):
    if num%i==0:
        flag=False
        break
if flag:
        print("Yes Prime")
else:
        print("No Prime")
```

```
Yes Prime
```

12) WAP to print sum of digits of given number

In [39]:
```
num = int(input("Enter a Number:"))
sum=0
while num!=0:
    module = num%10
    sum += module
    num = num//10

print("Sum of Digits is:",sum)
```

```
Sum of Digits is: 6
```

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

In [4]:
```
num = int(input("Enter a Number:"))
temp=num
rev=0
while temp>0:
    module = temp%10
    rev = rev * 10 + module
    temp = temp//10

if rev==num:
    print("The Num" ,num, "is Palindrome")
else:
    print("Not Palindrome")
```

```
The Num 3 is Palindrome
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

PDS - 2305CS303

Lab - 4

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

String

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

In [7]:
```
s = input("Enter String: ")
print(s)
rev = s[::-1]
if s==rev:
    print("Yes")
else:
    print("No")
```

```
Paresh Mori
No
```

02) WAP to reverse the words in given string.

In [8]:
```
str1 = input("Enter String: ")
l = s.split(" ")
for i in l:
    str1 = str1+" "+i[::-1]
print(str1)
```

```
Paresh hseraP iroM
```

3. WAP to remove ith character from given string

In [11]:
```
s = input("Enter String:")
i = int(input("Enter I Character to remove :"))
s1 = s[0:i]
s2 = s[i+1:]
print(s1+s2)
```

```
Enter String:Paresh
Enter I Character to remove :2
Paesh
```

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

In [15]:
```
s = input("Enter String:")
count=0
for i in s:
    count+=1
print(count)
```

```
Enter String:bnbnbnb
7
```

05) WAP to print even length word in string.

In [21]:
```
s = input("Enter String:")
s1 = s.split(" ")
for i in s1:
    if len(i)%2==0:
        print(i)
```

```
Enter String:Paresh Mor
Paresh
```

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

In [27]:
```
s = input("Enter String:")
count=0
s=s.lower()
for i in s:
    if i=='a' or i=='e' or i=='i' or i=='o' or i=='u':
        count+=1
print(count)
```

```
Enter String:Darshan University
6
```

07) WAP to convert given array to string.

In [38]:
```
l1 = []
while True:
    if l1.__contains__("Exit"):
        break;
    else:
        l1.append(input())
" ".join(l1[0:len(l1)-1])
```

```
Paresh
Rohit
Exit
```

Out[38]: 'Paresh Rohit'

01) WAP to find out duplicate characters in given string.

In [43]:
```
s = input("Enter String:")
l1 = list(s)
l = set(l1)
for i in l:
    c=l1.count(i)
    if c>1:
        print(i,c)
```

```
Enter String:adad
d 2
a 2
```

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

In [1]:
```
userStr = input("Enter the string here : ")
strList = userStr.split(" ")
for i in range(0, len(strList)):
    strList[i] = strList[i][0].upper() + strList[i][1:-1].lower() + strList[
print(" ".join(strList))
```

```
ParesH MorI
```

03) WAP to find Maximum frequency character in String.

In [2]:
```
userStr = input("Enter the string here : ")
strList = userStr.split(" ")
strDict = {}
for i in strList:
    for j in i:
        if j in strDict:
            strDict[j] += 1
        else:
            strDict[j] = 1
maxCount = 0
for key, value in strDict.items():
    if value > maxCount:
        maxCount = value
        maxChar = key
print(f"The Maximum Frequency Character is : {maxChar} with frequency : {max
```

```
The Maximum Frequency Character is : P with frequency : 1
```

04) WAP to find Minimum frequency character in String.

In [3]:
```
userStr = input("Enter the string here : ")
strList = userStr.split(" ")
strDict = {}
for i in strList:
```

```
    for j in i:
        if j in strDict:
            strDict[j] += 1
        else:
            strDict[j] = 1
minCount = 0
for key, value in strDict.items():
    if value < minCount or minCount == 0:
        minCount = value
        maxChar = key
print(f"The Minimum Frequency Character is : {maxChar} with frequency : {max
```

```
The Minimum Frequency Character is : e with frequency : 1
```

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

In [4]:
```
userStr = input("Enter the string here : ")
strList = list(userStr)

for i in strList:
    if i != "0" and i != "1":
        print("False")
        break
    else:
        print("True")
        break
```

```
False
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

PDS - 2305CS303

Lab - 5

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

list

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

In [3]:
```
size = int(input("Enter Size of a List : "))
l1 = []
count=0
for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))
sum = 0
for i in l1:
    sum = sum + i
print("Sum is : ",sum)
```

```
Sum is :  15
```

02) WAP to find largest element in a List.

In [4]:
```
size = int(input("Enter Size of a List : "))

l1 = []

for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))

largest = l1[0]

for i in l1:
    if i>largest :
        largest = i

print("Largest is : ",largest)
```

```
Largest is :  23
```

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

In [2]:
```
size = int(input("Enter Size of a List : "))
l1 = []
for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))
l2 = l1[0:int(size/2)]
print(l2)
l3 = l1[int(size/2) : ]
print(l3)
l4 = l3
l4.extend(l2)
print(l4)
```

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

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

In [3]:
```
size = int(input("Enter Size of a List : "))
l1 = []
for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))
temp = l1[size-1]
l1[size-1] = l1[0]
l1[0] = temp
print(l1)
```

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

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

In [4]:
```
size = int(input("Enter Size of a List : "))
l1 = []
for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))
x = int(input("Enter index to interchange : "))
y = int(input("Enter index to interchange : "))
temp = l1[y]
l1[y] = l1[x]
l1[x] = temp
print(l1)
```

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

06) WAP to reverses the list entered by user.

In [7]:
```
size = int(input("Enter Size of a List : "))
l1 = []
for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))
print(l1)
l1.reverse()
print(l1)
```

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

07) Python program to remove multiple elements from a list using list comprehension

In [9]:
```
size = int(input("Enter Size of a List : "))
l1 = []
for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))
print(l2)
l2 = [i for i in l1 if i%2==0]
print(l2)
```

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

08) Create a list from the specified start to end index of another list.

In [11]:
```
size = int(input("Enter Size of a List : "))
l1 = []
for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))
x = int(input("Enter index to Start : "))
y = int(input("Enter index to End : "))
print(l1)
l2=[]
for i in range(x,y+1):
    l2.append(l1[i])
print(l2)
```

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

09) Input comma separated elements, convert into list and print.

In [14]:
```
s1 = input("Enter Comma Separated Elements : ")
print(s1)
l1 = s1.split(",")
print(l1)
```

```
Paresh,Raj,Rohit
['Paresh', 'Raj', 'Rohit']
```

01) WAP to count Even and Odd numbers in a List.

In [15]:
```
size = int(input("Enter Size of a List : "))
l1 = []
for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))
countOdd = 0
countEven = 0
for i in l1:
    if i%2==0:
        countEven +=1
    else :
        countOdd +=1
print("Odd : ",countOdd)
print("Even : ",countEven)
```

```
Odd :  2
Even :  2
```

02) Python program to find N largest and smallest elements from the list

In [16]:
```
size = int(input("Enter Size of a List : "))
l1 = []
for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))

for i in range(0,size) :
    for j in range(i,size) :
        if l1[i]>l1[j] :
            temp = l1[j]
            l1[j] = l1[i]
            l1[i] = temp
print(l1)
num = int(input("Enter Number of Elements you want : "))
l2 = []
l2.extend(l1[size-num: ])
l3 = []
l3.extend(l1[0:num])
print(l2)
print(l3)
```

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

03) WAP to print duplicates from a list of integers

In [19]:
```
size = int(input("Enter Size of a List : "))
l1 = []
for i in range(0,size):
    l1.append(int(input(f"Enter Element {i} :")))
s = set(l1)
```

```
for i in s:
    if ll.count(i) > 1:
        print(i)
```

```
2
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

2305CS303 - Python for Data Science

Lab - 6

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

Tuples, dictionary, set

A

01) WAP to sort python dictionary by key or value.

In [1]:
```
# Section-A
# 1 Program
dict1 = {'n1':55,'n5':1,'n9':12,'n3':9,'n2':52}
sortedDict = {k: dict1[k] for k in sorted(dict1)}
print(sortedDict)
```

```
{'n1': 55, 'n2': 52, 'n3': 9, 'n5': 1, 'n9': 12}
```

02) WAP to merge two dictionaries given by user.

In [2]:
```
size1 = int(input("Enter size of Dictionary 1:"))
d1 = {}
for i in range(0,size1):
    key = input("Enter Key:")
    value = input("Enter Value:")
    d1[key] = value

size2 = int(input("Enter size of Dictionary 2:"))
d2 = {}
for i in range(0,size2):
    key2 = input("Enter Key:")
    value2 = input("Enter Value:")
    d2[key2] = value2
#using for loop
```

```
# for i in d2.keys():
#     d1[i] = d2[i]
#another method with update
d1.update(d2)
print(d1)
```

```
{'1': 'Paresh', '2': 'Raj', '3': 'Mohit', '4': 'Rohit'}
```

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

In [4]:
```
t1 = [(5,75,25,50),(40,100,23,76,5)]
k = 5
divbyk = [temp for temp in t1 if all(temp1%k == 0 for temp1 in temp)]
print(divbyk)
```

```
[(5, 75, 25, 50)]
```

04) WAP to find Tuples with positive elements in List of tuples.

In [7]:
```
t1 = [(5,75,5,50),(40,100,-23,76,5)]
pos = [temp for temp in t1 if all(temp1>0 for temp1 in temp)]
print(pos)
```

```
[(5, 75, 5, 50)]
```

05) WAP which perform union of two sets.

In [5]:
```
s1 = {"Paresh",3,76,"Devendra"}
s2 = {"Devendra",77,35,"Paresh"}
print(s1)
print(s2)
unionoftwo = s1.union(s2)
print(unionoftwo)
```

```
{3, 'Devendra', 'Paresh', 76}
{'Paresh', 'Devendra', 35, 77}
{3, 35, 'Devendra', 'Paresh', 76, 77}
```

B

01) WAP to convert binary tuple into integer.

In [6]:
```
size = int(input("Enter size of tuple"))
l1 = []
for i in range(size):
    element = int(input("Enter Elements:"))
    l1.append(element)
t1 = tuple(l1)
print(t1)
```

```
#logic 1
# bs = "".join(map(str,t1))
# ans = int(bs,2)
# print(ans)

#logic 2
dec = 0
length = len(t1)-1
for i in t1:
    dec += i*(2**length)
    length -= 1
print(f"The Binary Tuple Convert into Number is :{dec}")
```

```
(1, 2)
The Binary Tuple Convert into Number is :4
```

02) WAP to count frequency in list by dictionary.

In [7]:
```
# Program-2
size = int(input("Enter the Size of the List : "))
l1 = []
for i in range(size):
    elem = int(input(f"[{i}] : "))
    l1.append(elem)
freq = {}
for item in l1:
    if item in freq:
        freq[item] += 1
    else:
        freq[item] = 1
for key, value in freq.items():
    print(f"{key} : {value}")
```

```
1 : 1
2 : 1
3 : 1
4 : 1
```

03) WAP to remove all the duplicate words from the list using dictionary.

In [8]:
```
# Program-3
from collections import Counter
string = input("Enter the string here : ")
l1 = string.split(' ')
uniqueWord = Counter(l1)
s = ' '.join(uniqueWord.keys())
print(s)
```

```
Paresh Mori
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

2305CS303 - Python for Data Science

Lab - 7

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

Functions

In [ ]:
```
Subject Code
```

01) WAP to count simple interest using function.

In [27]:
```
def simpleinterest(p,r,n):
    return p*r*n/100
simpleinterest(100,10,1)
```

Out[27]: 10.0

02) WAP that defines a function to add first n numbers.

In [28]:
```
def addnumbers(n):
    sum = 0
    for i in range(1,n+1):
        sum += i
    return sum
addnumbers(5)
```

Out[28]: 15

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

In [16]:
```
def maxfromtwo(n1,n2):
    if n1>n2:
        print("The First number is Greater")
```

```
    else:
        print("The Second Number is Greatest")
maxfromtwo(10,20)
```

```
The Second Number is Greatest
```

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

In [26]:
```
def checkprime(num):
    for i in range(2,num):
        if num%i == 0:
            return 0
    return 1
checkprime(4)
```

Out[26]:
```
0
```

05) Write a function called primes that takes an integer value as an argument and returns a list of all prime numbers up to that number.

In [3]:
```
l1 = []
def rangeprime(n):

    for j in range(1,n+1):
        flag = True
        for i in range(2,j):
            if j%i == 0:
                flag = False
                break
        if flag:
            l1.append(j)
rangeprime(15)
print(l1)
```

```
[1, 2, 3, 5, 7, 11, 13]
```

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

In [5]:
```
def fibbo(n):
    a = 0
    b = 1
    print(a,b)
    for j in range(1,n+1):
        c = a+b
        print(c)
        a = b
        b = c
fibbo(7)
```

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

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

In [53]:
```
def factorial(n):
    if n==1:
        return 1
    else:
        return n*factorial(n-1)
factorial(5)
```

Out[53]:
```
120
```

08) WAP to implement simple calculator using lamda function.

In [57]:
```
s = lambda a,b,ch : a+b if ch == "+" else a-b if ch=="-" else a*b if ch=="*"
s(5,5,"*")
```

Out[57]:
```
25
```

09)Write a Python program that accepts a hyphen-separated sequence of words as input and prints the words in a hyphen-separated sequence after sorting them alphabetically

Sample Items : green-red-yellow-black-white

Expected Result : black-green-red-white-yellow

In [58]:
```
str = "green-red-yellow-black-white"
l1 = str.split("-")
l1.sort()
"-".join(l1)
```

Out[58]:
```
'black-green-red-white-yellow'
```

10) Write a python program to implement all function arguments type

Positional arguments

Default argument

Keyword arguments (named arguments)

Arbitrary arguments (variable-length arguments args and kwargs)

In [6]:
```
def add(*x):
    sum=0
    for i in x:
        sum +=i
    return sum
add(2,3)
```

Out[6]: 5

In [7]:
```
def add(x,y=3):
    return x+y
add(5)
```

Out[7]: 8

In [8]:
```
def add(x,y):
    return x+y
add(y=20,x=30)
```

Out[8]: 50

01) WAP to calculate power of a number using recursion.

In [22]:
```
def pow(num,i,p):
    if i == p-1:
        return num*num
    else:
        return num*pow(num,i+1,p)
pow(2,1,3)
```

Out[22]: 8

02) WAP to count digits of a number using recursion.

In [62]:
```
def countdigit(num):
    if num==0:
        return 0
    else:
        return 1+countdigit(num//10)
countdigit(1000)
```

Out[62]: 4

03) WAP to reverse an integer number using recursion.

In [12]:
```
def reverseFind(n,reverse):
    if n==0:
        return reverse
    else:
        reminder = n%10
        reverse = reverse*10 + reminder
        return reverseFind(int(n/10),reverse)
n = int(input("Enter a Number to find It's Reverse : "))
reverse =0
reverseFind(n,reverse)
```

Out[12]: 321

04) WAP to convert decimal number into binary using recursion.

In [21]:
```
def decimal_to_binary(n):
    if n > 1:
        decimal_to_binary(n // 2)
    print(n % 2, end='')

# Test the function
decimal = int(input("Enter a decimal number: "))
print("The binary representation is: ", end='')
decimal_to_binary(decimal)
```

```
The binary representation is: 101101
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

2305CS303 - Python for Data Science

Lab - 8

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

import numpy as np

In [2]:
```
import numpy as np
```

Create an array of 10 zeros

In [3]:
```
arr = np.zeros(10)
print(arr)
```

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

Output should be = array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

Create an array of 10 ones

In [4]:
```
arr = np.ones(10)
print(arr)
```

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

Output should be = array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])

Create an array of 10 fives

In [5]:
```
arr = np.ones(10)*5
print(arr)
```

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

Output should be = array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])

Create an array of the integers from 10 to 50

In [7]:
```
arr  = np.arange(10,51,1)
print(arr)
```

```
[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]
```

Output should be = array([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])

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

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

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

Output should be = array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50])

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

In [16]:
```
arr = np.arange(0,9).reshape(3,3)
print(arr)
```

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

Output should be =

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Create a 3x3 identity matrix

In [5]:
```
arr = np.eye(3)
arr
```

Out[5]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Output should be =

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Use NumPy to generate a random number between 0 and 1

In [23]:
```
arr = np.random.rand()
print(arr)
```

```
0.2299344122864706
```

Use NumPy to generate an array of 25 random numbers sampled from a standard normal distribution

In [27]:
```
arr = np.random.randn(25)
print(arr)
```

```
[ 0.97869879 -0.5935049  -0.18863757  0.51234066  1.48924556  0.96116134
  0.84252816  0.36957233  0.29505949  0.73535812 -1.32975231  0.00692104
 -0.22940629  0.70270525  0.05130406  0.89560829 -0.60860924  0.00905544
 -0.13929494  0.0272649  -1.40280973  0.21526839 -1.01978957 -0.29937078
  0.05944503]
```

Create the following matrix:

In [33]:
```
arr = np.linspace(0,1,101)
print(arr)
```

```
[0.   0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1  0.11 0.12 0.13
 0.14 0.15 0.16 0.17 0.18 0.19 0.2  0.21 0.22 0.23 0.24 0.25 0.26 0.27
 0.28 0.29 0.3  0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.4  0.41
 0.42 0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.5  0.51 0.52 0.53 0.54 0.55
 0.56 0.57 0.58 0.59 0.6  0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69
 0.7  0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.8  0.81 0.82 0.83
 0.84 0.85 0.86 0.87 0.88 0.89 0.9  0.91 0.92 0.93 0.94 0.95 0.96 0.97
 0.98 0.99 1.  ]
```

Output should be =

[[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1],
[0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2],
[0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3],
[0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4],
[0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5],
[0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6],
[0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7],
[0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8],
[0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9],
[0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.]]

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

In [34]:
```
arr = np.linspace(0,1,20)
print(arr)
```

```
[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.        ]
```

array([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.])

Numpy Indexing and Slicing

In [55]:
```
arr = np.arange(1,26,1).reshape(5,5)
print(arr)
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]]
```

Output should be =

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [43]:
```
arr = np.arange(1,26,1).reshape(5,5)
arr1 = arr[2:,1:]
print(arr1)
```

```
[[12 13 14 15]
 [17 18 19 20]
 [22 23 24 25]]
```

Output should be =

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

Output should be = 20 (element at specific index)

In [54]:
```
arr = np.arange(1,26,1).reshape(5,5)
arr1 = arr[3,4]
print(arr1)
```

```
20
```

Output should be =

```
array([[ 2],
       [ 7],
       [12]])
```

In [56]:
```
print(arr[0:3,1:2])
```

```
[[ 2]
 [ 7]
 [12]]
```

Output should be = array([21, 22, 23, 24, 25])

In [58]:
```
print(arr[4])
```

```
[21 22 23 24 25]
```

Output should be =

```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [62]:
```
print(arr[3:])
```

```
[[16 17 18 19 20]
 [21 22 23 24 25]]
```

In [66]:
```
condition = (arr % 3 == 0) & (arr % 5 != 0)
newarr = arr[condition]
newarr
```

Out[66]:
```
array([ 3,  6,  9, 12, 18, 21, 24])
```

Print all the number which are divisible by 3 but not by 5

Output should be = [3 6 9 12 18 21 24]

Now do the following

Get the sum of all the values in mat

In [67]:
```
print(arr.sum())
```

```
325
```

Output should be = 325

Get the standard deviation of the values in mat

In [71]:
```
print(arr.std())
```

```
7.211102550927978
```

Output should be = 7.211102550927978

Get the sum of all the columns in mat

In [70]:
```
print(arr.sum(axis=0))
```

```
[55 60 65 70 75]
```

Output should be = array([55, 60, 65, 70, 75])

Nice work, Keep the spark alive

========== End of Assignemnt ==========

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

PDS - 2305CS303

Lab - 9

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

01) Scrap the data of Faculty and news data from darshan university website.

In [33]:
```
import requests
import bs4
req = requests.get("https://darshan.ac.in/faculty-list/btech-computer")
soup = bs4.BeautifulSoup(req.text,"lxml")
data = soup.select("body > main > div:nth-child(5) > div > div > div > div "
for d in data :
    name = d.select("h2")
    for j in range(len(name)):
        print(name[j].text.strip())
```

In [ ]:
```
import requests
import bs4
req = requests.get("https://darshan.ac.in/faculty-list/btech-mechanical")
soup = bs4.BeautifulSoup(req.text,"lxml")
data = soup.select("body > main > div:nth-child(5) > div > div > div > div")
for d in data :
    name = d.select("h2")
    for j in range(len(name)):
        print(name[j].text.strip())
```

In [ ]:
```
import requests
import bs4
req = requests.get("https://darshan.ac.in/news-post/list/2024")
soup = bs4.BeautifulSoup(req.text,"lxml")
data = soup.select("body > main > div:nth-child(5) > div > div > div.col-lg-
for d in data :
    name = d.select("h2")
    for j in range(len(name)):
        print(name[j].text.strip())
```

01) Write a Python program to perform web scrapping from rajkot information technology association Website

In [ ]:
```
import bs4
import requests
req = requests.get("https://ritaindia.org/Member")
soup = bs4.BeautifulSoup(req.text,"lxml")
data = soup.select("body > section.team.pb-0 > div > div:nth-child(2) > div
for d in data :
    number = d.select("td:nth-child(1)")
    name = d.select("td:nth-child(2)")
    website = d.select("td:nth-child(3)")
    for j in range(len(name)):
        print(number[j].text +" -- "+ name[j].text +" -- "+ website[j].text)
```

In [ ]:
```
import bs4
import requests
req = requests.get("https://rajkotchamber.com/commitee-member/")
soup = bs4.BeautifulSoup(req.text,"lxml")
data = soup.select("#content > article > div > div > section.elementor-secti
for d in data :
    name = d.select("h3")
    for j in range(len(name)):
        print(name[j].text.strip())
```

Darshan UNIVERSITY

योग: कर्मसु कौशलम्

PDS - 2305CS303

Lab - 10

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

In [1]:
```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/
```

Step 2. Import the dataset from this address.

Step 3. Assign it to a variable called users and use the 'user_id' as index

In [3]:
```
users = pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/mast
```

Step 4. See the first 25 entries

In [4]:
```
users.head(25)
```

Out[4]:

user_id	age	gender	occupation	zip_code
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213
6	42	M	executive	98101
7	57	M	administrator	91344
8	36	M	administrator	05201
9	29	M	student	01002
10	53	M	lawyer	90703
11	39	F	other	30329
12	28	F	other	06405
13	47	M	educator	29206
14	45	M	scientist	55106
15	49	F	educator	97301
16	21	M	entertainment	10309
17	30	M	programmer	06355
18	35	F	other	37212
19	40	M	librarian	02138
20	42	F	homemaker	95660
21	26	M	writer	30068
22	25	M	writer	40206
23	30	F	artist	48197
24	21	F	artist	94533
25	39	M	engineer	55107

Step 5. See the last 10 entries

In [5]:
```
users.tail(10)
```

Out[5]:

user_id	age	gender	occupation	zip_code
934	61	M	engineer	22902
935	42	M	doctor	66221
936	24	M	other	32789
937	48	M	educator	98072
938	38	F	technician	55038
939	26	F	student	33319
940	32	M	administrator	02215
941	20	M	student	97229
942	48	F	librarian	78209
943	22	M	student	77841

Step 6. What is the number of observations in the dataset?

In [6]:
```
print(users.shape)
print(f'No of Observations : {users.shape[0]}')
```

```
(943, 4)
No of Observations : 943
```

Step 7. What is the number of columns in the dataset?

In [7]:
```
print(f'No of columns : {users.shape[1]}')
users.shape
```

```
No of columns : 4
```

Out[7]:
```
(943, 4)
```

Step 8. Print the name of all the columns.

In [8]:
```
users.columns
```

Out[8]:
```
Index(['age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

Step 9. How is the dataset indexed?

In [9]:
```
# "the index" (aka "the labels")
users.index
```

Out[9]:
```
Index([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
       ...
       934, 935, 936, 937, 938, 939, 940, 941, 942, 943],
      dtype='int64', name='user_id', length=943)
```

Step 10. What is the data type of each column?

In [10]:
```
users.dtypes
```

Out[10]:
```
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

Step 11. Print only the occupation column

In [11]:
```
users['occupation']
print(users['occupation'])
```

```
user_id
1         technician
2              other
3             writer
4         technician
5              other
           ...
939          student
940    administrator
941          student
942        librarian
943          student
Name: occupation, Length: 943, dtype: object
```

Step 12. How many different occupations are in this dataset?

In [12]:
```
users['occupation'].nunique()
```

Out[12]:
```
21
```

Step 13. What is the most frequent occupation?

In [13]:
```
users['occupation'].mode()[0]
```

Out[13]:
```
'student'
```

Step 14. Summarize the DataFrame.

In [14]:
```
print(users.describe())
```

```
              age
count  943.000000
mean    34.051962
std     12.192740
min      7.000000
25%     25.000000
50%     31.000000
75%     43.000000
max     73.000000
```

Step 15. Summarize all the columns

In [15]:
```
users['age'].describe()
```

Out[15]:
```
count    943.000000
mean      34.051962
std       12.192740
min        7.000000
25%       25.000000
50%       31.000000
75%       43.000000
max       73.000000
Name: age, dtype: float64
```

Step 16. Summarize only the occupation column

In [16]:
```
users['occupation'].describe()
```

Out[16]:
```
count         943
unique         21
top       student
freq          196
Name: occupation, dtype: object
```

Step 17. What is the mean age of users?

In [17]:
```
users['age'].mean()
```

Out[17]:
```
34.05196182396607
```

Step 18. What is the age with least occurrence?

In [18]:
```
all_count = df['age'].value_counts()
min_age=all_count.min()
all_data = all_count[all_count==min_age]
all_data
```

Out[18]:
```
age
7     1
66    1
11    1
10    1
73    1
Name: count, dtype: int64
```

Darshan UNIVERSITY
योग: कर्मसु कौशलम्

PDS - 2305CS303

Lab - 11

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

Ecommerce Purchases Exercise

In this Exercise you will be given some Fake Data about some purchases done through Amazon! Just go ahead and follow the directions and try your best to answer the questions and complete the tasks. Feel free to reference the solutions. Most of the tasks can be solved in different ways. For the most part, the questions get progressively harder.

Please excuse anything that doesn't make "Real-World" sense in the dataframe, all the data is fake and made-up.

Also note that all of these questions can be answered with one line of code.

** Import pandas and read in the Ecommerce Purchases csv file and set it to a DataFrame called ecom. **

In [2]:
```
import pandas as pd
df = pd.read_csv("Ecommerce Purchases.csv")
print(df)
```

```
                                                 Address    Lot AM or PM  \
0     16629 Pace Camp Apt. 448\nAlexisborough, NE 77...  46 in       PM
1     9374 Jasmine Spurs Suite 508\nSouth John, TN 8...  28 rn       PM
2                       Unit 0065 Box 5052\nDPO AP 27450  94 vE       PM
3                  7780 Julia Fords\nNew Stacy, WA 45798  36 vm       PM
4     23012 Munoz Drive Suite 337\nNew Cynthia, TX 5...  20 IE       AM
...                                                 ...    ...      ...
9995      966 Castaneda Locks\nWest Juliafurt, CO 96415  92 XI       PM
9996  832 Curtis Dam Suite 785\nNorth Edwardburgh, T...  41 JY       AM
9997              Unit 4434 Box 6343\nDPO AE 28026-0283  74 Zh       AM
9998               0096 English Rest\nRoystad, IA 12457  74 cL       PM
9999     40674 Barrett Stravenue\nGrimesville, WI 79682  64 Hr       AM

                                        Browser Info  \
0     Opera/9.56.(X11; Linux x86_64; sl-SI) Presto/2...
1     Opera/8.93.(Windows 98; Win 9x 4.90; en-US) Pr...
2     Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...
3     Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_0 ...
4     Opera/9.58.(X11; Linux x86_64; it-IT) Presto/2...
...                                                 ...
9995  Mozilla/5.0 (Windows NT 5.1) AppleWebKit/5352 ...
9996  Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...
9997  Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_7...
9998  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_8;...
9999  Mozilla/5.0 (X11; Linux i686; rv:1.9.5.20) Gec...

                              Company       Credit Card CC Exp Date  \
0                       Martinez-Herman  6011929061123406       02/20
1     Fletcher, Richards and Whitaker  3337758169645356       11/18
2          Simpson, Williams and Pham      675957666125       08/19
3     Williams, Marshall and Buchanan  6011578504430710       02/24
4           Brown, Watson and Andrews  6011456623207998       10/25
...                               ...               ...         ...
9995                    Randall-Sloan   342945015358701       03/22
9996          Hale, Collins and Wilson   210033169205009       07/25
9997                     Anderson Ltd  6011539787356311       05/21
9998                         Cook Inc   180003348082930       11/17
9999                      Greene Inc  4139972901927273       02/19

      CC Security Code                  CC Provider  \
0                  900                 JCB 16 digit
1                  561                   Mastercard
2                  699                 JCB 16 digit
3                  384                    Discover
4                  678  Diners Club / Carte Blanche
...                ...                          ...
9995               838                 JCB 15 digit
9996               207                 JCB 16 digit
9997                 1                VISA 16 digit
9998               987             American Express
9999               302                 JCB 15 digit

                                Email                                     Job
\
0                  pdunlap@yahoo.com  Scientist, product/process development
1                anthony41@reed.com                       Drilling engineer
```

```
2     amymiller@morales-harrison.com               Customer service manager
3        brent16@olson-robinson.info                      Drilling engineer
4        christopherwright@gmail.com                            Fine artist
...                               ...                                    ...
9995          iscott@wade-garner.com                         Printmaker
9996              mary85@hotmail.com                       Energy engineer
9997               tyler16@gmail.com                     Veterinary surgeon
9998         elizabethmoore@reid.net                Local government officer
9999           rachelford@vaughn.com                  Embryologist, clinical

           IP Address Language  Purchase Price
0     149.146.147.205       el           98.14
1        15.160.41.51       fr           70.73
2      132.207.160.22       de            0.95
3        30.250.74.19       es           78.04
4        24.140.33.94       es           77.82
...               ...      ...             ...
9995    29.73.197.114       it           82.21
9996   121.133.168.51       pt           25.63
9997    156.210.0.254       el           83.98
9998     55.78.26.143       es           38.84
9999  176.119.198.199       el           67.59

[10000 rows x 14 columns]
```

Check the head of the DataFrame.

In [ ]:

Out[ ]:

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	C Ex Dat
0	16629 Pace Camp Apt. 448\nAlexisborough, NE 77...	46 in	PM	Opera/9.56. (X11; Linux x86_64; sl-SI) Presto/2...	Martinez-Herman	6011929061123406	02/2
1	9374 Jasmine Spurs Suite 508\nSouth John, TN 8...	28 rn	PM	Opera/8.93. (Windows 98; Win 9x 4.90; en-US) Pr...	Fletcher, Richards and Whitaker	3337758169645356	11/1
2	Unit 0065 Box 5052\nDPO AP 27450	94 vE	PM	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Simpson, Williams and Pham	675957666125	08/1
3	7780 Julia Fords\nNew Stacy, WA 45798	36 vm	PM	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_0 ...	Williams, Marshall and Buchanan	6011578504430710	02/2
4	23012 Munoz Drive Suite 337\nNew Cynthia, TX 5...	20 IE	AM	Opera/9.58. (X11; Linux x86_64; it-IT) Presto/2...	Brown, Watson and Andrews	6011456623207998	10/2

** How many rows and columns are there? **

In [11]:
```
row_Count = len(df)
print(row_Count)

row_Count = df.shape[0]
print(row_Count)
```

```
10000
10000
```

** What is the average Purchase Price? **

In [13]:
```
df["Purchase Price"].mean()
```

Out[13]: 50.34730200000025

** What were the highest and lowest purchase prices? **

In [17]:
```
minimum = df["Purchase Price"].min()
print(minimum)
```

```
99.99
0.0
```

In [18]:
```
maximun = df["Purchase Price"].max()
print(maximun)
```

```
99.99
```

** How many people have English 'en' as their Language of choice on the website? **

In [21]:
```
len(df[df["Language"] == "en"])
```

Out[21]:
```
1098
```

** How many people have the job title of "Lawyer" ? **

In [22]:
```
len(df[df["Job"] == "Lawyer"])
```

Out[22]:
```
30
```

** How many people made the purchase during the AM and how many people made the purchase during PM ? **

**(Hint: Check out value_counts()) **

In [24]:
```
pm = len(df[df["AM or PM"] == "PM"])
am = len(df[df["AM or PM"] == "AM"])
print("This is Count of Am",am)
print("This is Count of Pm",pm)
```

```
This is Count of Am 4932
This is Count of Pm 5068
```

** What are the 5 most common Job Titles? **

In [28]:
```
df["Job"].value_counts().head()
```

Out[28]:
```
Interior and spatial designer    31
Lawyer                           30
Social researcher                28
Purchasing manager               27
Designer, jewellery              27
Name: Job, dtype: int64
```

** Someone made a purchase that came from Lot: "90 WT" , what was the Purchase Price for this transaction? **

In [44]:
```
price = df[df["Lot"] == "90 WT"]["Purchase Price"]
print(price)
```

```
513    75.1
Name: Purchase Price, dtype: float64
```

** What is the email of the person with the following Credit Card Number: 4926535242672853 **

In [48]:
```
df[df["Credit Card"] == 4926535242672853]["Email"]
```

Out[48]:
```
1234    bondellen@williams-garza.com
Name: Email, dtype: object
```

** How many people have American Express as their Credit Card Provider *and* made a purchase above $95 ?**

In [3]:
```
df[(df["CC Provider"] == "American Express") & (df["Purchase Price"] > 95)]
```

Out[3]:

	Address	Lot	AM or PM	Browser Info	Company	Credi
9	3795 Dawson Extensions\nLake Tinafort, ID 88739	15 Ug	AM	Mozilla/5.0 (X11; Linux i686; rv:1.9.7.20) Gec...	Rivera, Buchanan and Ramirez	4396283
280	81060 Dustin Causeway Apt. 503\nPort Danielche...	80 zh	PM	Mozilla/5.0 (Windows NT 5.01) AppleWebKit/5362...	Clay PLC	377737470
372	359 Stanley Coves\nSalasfort, SD 59457	75 Ub	PM	Opera/8.42.(X11; Linux x86_64; en-US) Presto/2...	Davis-Lawrence	371995567
677	4855 Peter Bridge\nJohnsonberg, PA 90599-0009	62 Nx	AM	Opera/9.49. (Windows 98; Win 9x 4.90; en-US) Pr...	Jones and Sons	4960556611
766	386 Alisha Unions\nSteelebury, ND 19782	28 pJ	PM	Opera/8.47.(X11; Linux x86_64; sl-SI) Presto/2...	Proctor PLC	30325623
1225	916 Amanda Heights\nNew Johnland, CA 52112-8572	09 vg	PM	Mozilla/5.0 (X11; Linux i686; rv:1.9.7.20) Gec...	Clark, Ross and Travis	371229555
1381	79284 Lisa Mews Suite 069\nKellyborough, CT 71...	45 MR	PM	Mozilla/5.0 (Macintosh; PPC Mac OS X 10_5_5; r...	Randall-Cohen	4855262
1385	67796 James Keys Suite 656\nSouth Katieshire, ...	06 ia	PM	Mozilla/5.0 (Windows NT 5.1) AppleWebKit/5330 ...	Gonzalez, Gross and Allen	304073329
1568	76108 Barker Manors\nEast Amy, PW 49544-5921	82 Qw	AM	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/53...	Nguyen, Branch and Wiley	514245185
1727	478 Anita Hill Apt. 766\nAverymouth, FM 50629-...	94 qM	AM	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/53...	Owen and Sons	302427579
2146	557 Barton Harbor\nFuentesport, VI 29961	96 MV	AM	Mozilla/5.0 (compatible; MSIE 6.0; Windows 98;...	Flores-Cook	333725599
2264	040 Ingram Way Suite 602\nEast Matthew, AR 67341	52 WA	AM	Mozilla/5.0 (Windows NT 6.0) AppleWebKit/5341 ...	Baker, Simmons and Pitts	309694645
2267	216 Olivia Court Apt. 439\nTonyshire, IN 19159...	11 za	PM	Mozilla/5.0 (Macintosh; PPC Mac OS X 10_6_4; r...	Olson Inc	538875733

	Address	Lot	AM or PM	Browser Info	Company	Credi
2642	899 Williams Prairie Suite 234\nEast Hunterbur...	46 hb	PM	Mozilla/5.0 (compatible; MSIE 7.0; Windows NT ...	Webb Inc	3096797127
2828	20135 Miller Green Apt. 763\nGarciaville, KY 6...	79 El	PM	Mozilla/5.0 (Windows; U; Windows CE) AppleWebK...	Zavala, Cox and Parker	5173968976
3291	58939 Reese Cove\nSouth Ross, MA 54531	50 cD	AM	Mozilla/5.0 (compatible; MSIE 9.0; Windows 95;...	Chang, Davis and Snyder	349464844
3349	1984 Christopher Turnpike Suite 904\nRichardbu...	30 xs	AM	Mozilla/5.0 (X11; Linux x86_64; rv:1.9.7.20) G...	Martin-Mitchell	3096398727
3619	03233 Angela Road Apt. 808\nEast Jilliantown, ...	41 nH	PM	Opera/9.53.(Windows 95; en-US) Presto/2.9.175 ...	Silva, Kim and Martinez	480725148
4071	8161 Craig Passage Apt. 939\nPort Emilystad, R...	45 ab	AM	Mozilla/5.0 (Windows; U; Windows NT 6.1) Apple...	Wilson-Robinson	550676769
4569	88284 Austin Summit Apt. 805\nKellerside, WY 6...	03 WL	AM	Mozilla/5.0 (Macintosh; U; PPC Mac OS X 10_5_5...	Fisher, Thompson and Lynch	308872688
5385	USNS Smith\nFPO AA 37829	80 BX	AM	Opera/9.56.(X11; Linux x86_64; en-US) Presto/2...	Patterson-Miller	441178605
5655	38674 Mckay Vista\nWilsonbury, MO 55611-8663	95 aB	PM	Mozilla/5.0 (X11; Linux i686) AppleWebKit/5331...	Watson-Bailey	309682098
5692	PSC 8264, Box 8124\nAPO AE 19700	97 sG	PM	Mozilla/5.0 (iPod; U; CPU iPhone OS 3_3 like M...	Moore Inc	601149970
5814	Unit 5232 Box 4650\nDPO AE 28335-7543	71 So	PM	Mozilla/5.0 (X11; Linux i686) AppleWebKit/5342...	Garcia LLC	30346960
5875	8256 Anderson Forest\nFordbury, PA 81027-8980	17 Oe	AM	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Kelly, Peterson and Oliver	517333572
5895	87003 Jason Tunnel Apt. 216\nDariusville, MS 8...	72 gL	AM	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Reyes, Fox and Wilson	180010957
6026	Unit 4385 Box 1428\nDPO AE 04110	58 Ex	AM	Mozilla/5.0 (Windows; U; Windows NT 5.01) Appl...	White Inc	311297557

	Address	Lot	AM or PM	Browser Info	Company	Credi
6524	344 Rivera Shore Apt. 224\nJoshuaborough, ND 3...	44 SM	AM	Opera/8.62.(X11; Linux i686; en-US) Presto/2.9...	Lawson, Moyer and Valencia	379348662
7239	81561 Stein Rue\nEast Melissaport, MA 64959	45 sZ	PM	Mozilla/5.0 (X11; Linux i686) AppleWebKit/5310...	Smith Ltd	503825
7381	40804 Belinda Roads Apt. 186\nWest Nicoleberg,...	30 GH	AM	Opera/9.32. (Windows NT 4.0; it-IT) Presto/2.9....	Blair-Keith	180089858
7416	1082 Jason Alley Suite 707\nWhitestad, SC 97805	08 bm	PM	Mozilla/5.0 (Windows NT 5.1; sl-SI; rv:1.9.2.2...	Carroll, Harrison and Escobar	3088903707
7514	678 David Manor Suite 323\nBobbymouth, PW 59942	91 NA	AM	Mozilla/5.0 (compatible; MSIE 8.0; Windows 98;...	Glenn, Lowe and Shah	676298
8233	327 Hamilton Hollow\nEast Nicolechester, RI 65868	53 dE	PM	Mozilla/5.0 (Macintosh; U; PPC Mac OS X 10_5_3...	Keller, Carter and Garcia	30299608
8381	274 Wallace Pine Apt. 189\nMartinberg, WY 7548...	32 rI	PM	Mozilla/5.0 (Windows; U; Windows 98; Win 9x 4....	Kelly-Newman	552710442
8408	Unit 4967 Box 4574\nDPO AP 11412	05 zs	PM	Mozilla/5.0 (compatible; MSIE 5.0; Windows 98;...	Acevedo-Smith	333772130
8434	474 Gonzalez Meadow Suite 581\nEast Vincent, M...	14 St	PM	Mozilla/5.0 (compatible; MSIE 6.0; Windows NT ...	Miller and Sons	180059754
8817	3658 Campos Lodge\nChambersfurt, MH 86260	24 ej	PM	Mozilla/5.0 (compatible; MSIE 5.0; Windows NT ...	Fisher-Jones	210046575
9168	18782 Hall Row Suite 810\nSouth Wesleyville, I...	15 QU	PM	Opera/9.57.(X11; Linux i686; it-IT) Presto/2.9...	Butler PLC	541261106
9856	058 Miranda Locks Suite 792\nPort Davidstad, P...	73 IW	PM	Mozilla/5.0 (Windows NT 5.0) AppleWebKit/5360 ...	Olson-Navarro	601103722

** Hard: How many people have a credit card that expires in 2025? **

In [62]:
```
len(df[(df["CC Exp Date"].str.contains('25'))])
```

Out[62]: 1033

** Hard: What are the top 5 most popular email providers/hosts (e.g. gmail.com, yahoo.com, etc...) **

In [71]:
```
df["Email"].str.split("@").str[1].value_counts().head()
```

Out[71]:
```
hotmail.com     1638
yahoo.com       1616
gmail.com       1605
smith.com         42
williams.com      37
Name: Email, dtype: int64
```

Great Job!

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

PDS - 2305CS303

Lab - 12

Enrollment No: 23030501037
Name : Mori Paresh Varsinghbhai

Welcome to a quick exercise for you to practice your pandas skills! We will be using the [SF Salaries Dataset] (https://www.kaggle.com/kaggle/sf-salaries) from Kaggle! Just follow along and complete the tasks outlined in bold below. The tasks will get harder and harder as you go along.

** Import pandas as pd.**

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

** Read Salaries.csv as a dataframe called sal.**

In [4]:
```
readCSV = pd.read_csv("Salaries.csv",index_col=0,header=0)
print(readCSV)
```

```
             EmployeeName                                       JobTitle  \
Id
1           NATHANIEL FORD  GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY
2             GARY JIMENEZ                 CAPTAIN III (POLICE DEPARTMENT)
3            ALBERT PARDINI                 CAPTAIN III (POLICE DEPARTMENT)
4         CHRISTOPHER CHONG            WIRE ROPE CABLE MAINTENANCE MECHANIC
5           PATRICK GARDNER    DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)
...                     ...                                             ...
148650      Roy I Tillery                                       Custodian
148651       Not provided                                    Not provided
148652       Not provided                                    Not provided
148653       Not provided                                    Not provided
148654          Joe Lopez                     Counselor, Log Cabin Ranch

          BasePay  OvertimePay   OtherPay  Benefits   TotalPay  \
Id
1       167411.18         0.00  400184.25       NaN  567595.43
2       155966.02    245131.88  137811.38       NaN  538909.28
3       212739.13    106088.18   16452.60       NaN  335279.91
4        77916.00     56120.71  198306.90       NaN  332343.61
5       134401.60      9737.00  182234.59       NaN  326373.19
...           ...          ...        ...       ...        ...
148650       0.00         0.00       0.00       0.0       0.00
148651        NaN          NaN        NaN       NaN       0.00
148652        NaN          NaN        NaN       NaN       0.00
148653        NaN          NaN        NaN       NaN       0.00
148654       0.00         0.00    -618.13       0.0    -618.13

        TotalPayBenefits  Year  Notes         Agency  Status
Id
1              567595.43  2011    NaN  San Francisco     NaN
2              538909.28  2011    NaN  San Francisco     NaN
3              335279.91  2011    NaN  San Francisco     NaN
4              332343.61  2011    NaN  San Francisco     NaN
5              326373.19  2011    NaN  San Francisco     NaN
...                  ...   ...    ...            ...     ...
148650              0.00  2014    NaN  San Francisco     NaN
148651              0.00  2014    NaN  San Francisco     NaN
148652              0.00  2014    NaN  San Francisco     NaN
148653              0.00  2014    NaN  San Francisco     NaN
148654           -618.13  2014    NaN  San Francisco     NaN

[148654 rows x 12 columns]
```

** Check the head of the DataFrame. **

In [28]:
```
readCSV.head()
```

Out[28]:

Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefit
1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	Na
2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	Na
3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	Na
4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	Na
5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT, (FIRE DEPARTMENT)	134401.60	9737.00	182234.59	Na

** Use the .info() method to find out how many entries there are.**

In [5]:
```
print(readCSV.info())
print(len(readCSV))
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 148654 entries, 1 to 148654
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   EmployeeName      148654 non-null  object 
 1   JobTitle          148654 non-null  object 
 2   BasePay           148045 non-null  float64
 3   OvertimePay       148650 non-null  float64
 4   OtherPay          148650 non-null  float64
 5   Benefits          112491 non-null  float64
 6   TotalPay          148654 non-null  float64
 7   TotalPayBenefits  148654 non-null  float64
 8   Year              148654 non-null  int64  
 9   Notes             0 non-null       float64
 10  Agency            148654 non-null  object 
 11  Status            0 non-null       float64
dtypes: float64(8), int64(1), object(3)
memory usage: 14.7+ MB
None
148654
```

What is the average BasePay ?

In [6]:
```
print(readCSV['BasePay'].mean())
```

```
66325.4488404877
```

** What is the highest amount of OvertimePay in the dataset ? **

In [7]:
```
print(readCSV['OvertimePay'].max())
```

```
245131.88
```

** What is the job title of JOSEPH DRISCOLL ? Note: Use all caps, otherwise you may get an answer that doesn't match up (there is also a lowercase Joseph Driscoll). **

In [11]:
```
readCSV[readCSV['EmployeeName'] == 'JOSEPH DRISCOLL']['JobTitle']
```

Out[11]:
```
Id
25    CAPTAIN, FIRE SUPPRESSION
Name: JobTitle, dtype: object
```

** How much does JOSEPH DRISCOLL make (including benefits)? **

In [13]:
```
readCSV[readCSV['EmployeeName'] == 'JOSEPH DRISCOLL']['TotalPayBenefits']
```

Out[13]:
```
Id
25    270324.91
Name: TotalPayBenefits, dtype: float64
```

** What is the name of highest paid person (including benefits)?**

In [7]:
```
readCSV[readCSV['TotalPayBenefits'] == readCSV['TotalPayBenefits'].max()]
```

Out[7]:

Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefit
1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.0	400184.25	Na

** What is the name of lowest paid person (including benefits)? Do you notice something strange about how much he or she is paid?**

In [8]:
```
readCSV[readCSV['TotalPayBenefits'] == readCSV['TotalPayBenefits'].min()]
```

Out[8]:

Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefit
148654	Joe Lopez	Counselor, Log Cabin Ranch	0.0	0.0	-618.13	0.(

** What was the average (mean) BasePay of all employees per year? (2011-2014) ? **

In [18]:
```
readCSV.groupby('Year')['BasePay'].mean()
```

Out[18]:
```
Year
2011    63595.956517
2012    65436.406857
2013    69630.030216
2014    66564.421924
Name: BasePay, dtype: float64
```

** How many unique job titles are there? **

In [20]:
```
readCSV['JobTitle'].nunique()
```

Out[20]:
```
2159
```

** What are the top 5 most common jobs? **

In [21]:
```
readCSV['JobTitle'].value_counts().head()
```

Out[21]:
```
JobTitle
Transit Operator                7036
Special Nurse                   4389
Registered Nurse                3736
Public Svc Aide-Public Works    2518
Police Officer 3                2421
Name: count, dtype: int64
```

** How many Job Titles were represented by only one person in 2013? (e.g. Job Titles with only one occurence in 2013?) **

In [22]:
```
sum(readCSV[readCSV['Year'] == 2013]['JobTitle'].value_counts() == 1)
```

Out[22]:
```
202
```

** How many people have the word Chief in their job title? (This is pretty tricky) **

In [23]:
```
sum(readCSV['JobTitle'].str.lower().str.contains('chief'))
```

Out[23]:
```
627
```

** Bonus: Is there a correlation between length of the Job Title string and Salary? **

In [25]:
```
readCSV['title_len'] = readCSV['JobTitle'].str.len()
```

In [26]:
```
readCSV[['title_len', 'TotalPayBenefits']].corr()
```

Out[26]:

	title_len	TotalPayBenefits
title_len	1.000000	-0.036878
TotalPayBenefits	-0.036878	1.000000

Great Job!