(https://www.darshan.ac.in/)

**Python for Data Science - 3150713**

**Lab - 1**

# 01) WAP to add two numbers entered by user

In [ ]:
```
a = int(input("Enter number 1 : "))
b = int(input("Enter number 2 : "))
print("{} + {} = {}".format(a,b,a+b))
```

```
Enter number 1 : 12864
Enter number 2 : 54973
12864 + 54973 = 67837
```

# 02) WAP to calculate simple interest.

In [ ]:
```
P = int(input("Enter Principal Amount : "))
R = float(input("Enter Rate per Annum : "))
N = int(input("Enter Time in Months  : "))
I = (P*R*(N/12))/100
print("Total Interest on Amount {amount} at rate {rate} for {months} months :
{interest}".format(amount=P,rate=R,months=N,interest=I))
print("Total Amount needed to pay : ",P+I)
```

```
Enter Principal Amount : 100000
Enter Rate per Annum : 7.25
Enter Time in Months  : 18
Total Interest on Amount 100000 at rate 7.25 for 18 months : 10875.0
Total Amount needed to pay :  110875.0
```

## 03) WAP to calculate area of circle.

In [ ]:
```
import math
r = float(input("Enter radius of circle : "))
print("Area of circle with radius {} is : {} ".format(r,(math.pi)*(r**2)))
```

```
Enter radius of circle : 3.5
Area of circle with radius 3.5 is : 38.48451000647496
```

## 04) WAP to print Multiplication table of given number without using loop.

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

```
Enter n : 5
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
```

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

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

```
Enter a number : -1
Number is negative
```

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

In [ ]:
```
n1 = int(input("Enter number 1 : "))
n2 = int(input("Enter number 2 : "))
n3 = int(input("Enter number 3 : "))
if n1 > n2:
  if n1 > n3:
    print("Largest :",n1)
  else:
    print("Largest :",n3)
else:
  if n2 > n3:
    print("Largest :",n2)
  else:
    print("Largest :",n3)
```

```
Enter number 1 : 3
Enter number 2 : 3
Enter number 3 : 3
Largest : 3
```

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

In [ ]:
```
n1 = int(input("Enter number 1 : "))
n2 = int(input("Enter number 2 : "))

print(n1,"+",n2,"=",n1+n2)
print(n1,"-",n2,"=",n1-n2)
print(n1,"*",n2,"=",n1*n2)
print(n1,"/",n2,"=",n1//n2)
```

```
Enter number 1 : 10
Enter number 2 : 2
10 + 2 = 12
10 - 2 = 8
10 * 2 = 20
10 / 2 = 5
```

## 08) WAP to print 1 to n

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

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

## 09) WAP to print odd numbers between 1 to n

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

# or
print()

for j in range(1,n+1,2):
  print(j)
```

```
Enter n : 10
1
3
5
7
9

1
3
5
7
9
```

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

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

```
Enter n : 9
5
```

## 11) WAP to print multiplication table of given number.

In [ ]:
```
n = int(input("Enter n : "))
for i in range(1,11):
  print("{num} * {times} = {value}".format(num=n,times=i,value=n*i))
```

```
Enter n : 49
49 * 1 = 49
49 * 2 = 98
49 * 3 = 147
49 * 4 = 196
49 * 5 = 245
49 * 6 = 294
49 * 7 = 343
49 * 8 = 392
49 * 9 = 441
49 * 10 = 490
```

## 12) WAP to find factorial of the given number.

In [ ]:
```
n = int(input("Enter a number : "))
factorial = 1
for i in range(1,n+1):
  factorial *= i
print("Factorial of {num} is : {fact}".format(num=n,fact=factorial))
```

```
Enter a number : 6
Factorial of 6 is : 720
```

## 13) WAP to find factors of the given number.

In [ ]:
```
n = int(input("Enter a number : "))
i=1
print("Factors of number",n,"are :")
while( i <= n ):
  if not n%i:
    print(i,end=" ")
  i+=1
```

```
Enter a number : 18
Factors of number 18 are :
1 2 3 6 9 18
```

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

In [ ]:
```
n = int(input("Enter a number : "))
prime=True
i=2
while(i <= math.sqrt(n)):
  if not n%i:
    prime=False
    break
  i += 1

if prime:
  print("Number {} is a prime number".format(n))
else:
  print("Number {} is not a prime number".format(n))
```

```
Enter a number : 23
Number 23 is a prime number
```

## 15) WAP to print sum of digits of given number.

In [ ]:
```
n = int(input("Enter a number : "))
sum = 0
while(n>0):
  sum += n%10
  n //= 10
print("Sum of digit is :",sum)
```

```
Enter a number : 54123
Sum of digit is : 15
```

# 16) WAP to find out prime numbers between given two numbers.

In [ ]:
```
import math
n1 = int(input("Enter number 1 : "))
n2 = int(input("Enter number 2 : "))
for num in range(min(max(n1,2),max(n2,2)),max(n1,n2)+1):
  prime=True
  i=2
  while(i<=math.sqrt(num)):
    if not num%i:
      prime=False
      break
    i+=1
  if prime:
    print(num)
```

```
Enter number 1 : 100
Enter number 2 : 1
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```

# 17) WAP to print Fibonacci series up to number given by user.

In [ ]:
```
n = int(input("Enter a number : "))
n1,n2,n3=0,1,0

print("Fibonacci series up to number",n)
while not n1>n:
  print(n1,end=" ")
  n3 = n1 + n2
  n1 = n2
  n2 = n3
```

```
Enter a number : 5
Fibonacci series up to number 5
0 1 1 2 3 5
```

# 18) WAP to Find largest prime factor of given number.

In [ ]:
```
n = int(input('Enter a number : '))
m = n
while(m>0):
  if not n%m:
    prime = True
    i = 2
    while(i<=math.sqrt(m)):
      if not m%i:
        prime = False
        break
      i += 1
    if prime:
      print(m)
      break
  m -= 1
```

```
Enter a number : 15
5
```

# 19) WAP to print sum for cube of first n natural numbers.

In [ ]:
```
n = int(input("Enter a number : "))
cubeTotal = 0
for i in range(1,n+1):
  cubeTotal += i**3

print("Sum of cube of first {num} natural numbers is : {ans}".format(num=n,ans
=cubeTotal))
```

```
Enter a number : 3
Sum of cube of first 3 natural numbers is : 36
```

# 20) WAP to converts the given decimal number to its binary equivalent.

In [ ]:
```
n = int(input("Enter a number : "))
binary = ""
while n > 0:
  binary = str(n%2) + binary
  n //= 2
print(binary)
```

```
Enter a number : 13
1101
```

# 21) WAP to print following pattern

```
*
* *
* * *
* * * *
```

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

```
Enter n : 5
*
* *
* * *
* * * *
* * * * *
```

## 22) WAP to print following pattern

```
$ $ $ $
$ $ $
$ $
$
```

In [ ]:
```
n = int(input("Enter n : "))
for outer in range(n,0,-1):
  for inner in range(outer, 0, -1):
    print("$",end=" ")
  print()
```

```
Enter n : 5
$ $ $ $ $
$ $ $ $
$ $ $
$ $
$
```

## 23) WAP to print following pattern

```
# # # # #
  # # #
    #
  # # #
# # # # #
```

In [ ]:
```
n = int(input("Enter n : "))
reverse,i = False,n

while not (i>n and reverse):
  if i <= 2:
    reverse = True
  for space in range((n-i)//2):
    print(end="  ")
  for symbol in range(i):
    print("#",end=" ")
  print()
  if reverse:
    i += 2
  else:
    i -= 2
```

```
Enter n : 5
# # # # #
  # # #
    #
  # # #
# # # # #
```

# 24) WAP to print following pattern

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

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

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


(https://www.darshan.ac.in/)

**Python for Data Science - 3150713**

**Lab - 2**

*Rewrite all the programs of Lab-1 with Function*

# 01) WAP to add two numbers entered by user

In [ ]:
```
def sum(a,b):
  return (a+b)
a = int(input("Enter number 1 : "))
b = int(input("Enter number 2 : "))
print("{} + {} = {}".format(a,b,sum(a,b)))
```

```
Enter number 1 : 5
Enter number 2 : 7
5 + 7 = 12
```

# 02) WAP to calculate simple interest.

In [ ]:
```
def sInterest(p,r,n):
  return (p*r*n)/100

P = int(input("Enter Principal Amount : "))
R = float(input("Enter Rate per Annum : "))
N = int(input("Enter Time in Months  : "))
I = sInterest(P,R,N/12)
print("Total Interest on Amount {amount} at rate {rate} for {months} months :
{interest}".format(amount=P,rate=R,months=N,interest=I))
print("Total Amount needed to pay : ",P+I)
```

```
Enter Principal Amount : 100
Enter Rate per Annum : 10
Enter Time in Months  : 9
Total Interest on Amount 100 at rate 10.0 for 9 months : 7.5
Total Amount needed to pay :  107.5
```

# 03) WAP to calculate area of circle.

In [ ]:
```
import math

def circleArea(r):
  return (math.pi)*(r**2)
r = float(input("Enter radius of circle : "))
print("Area of circle with radius {} is : {} ".format(r,circleArea(r)))
```

```
Enter radius of circle : 10
Area of circle with radius 10.0 is : 314.1592653589793
```

# 04) WAP to print Multiplication table of given number without using loop.

In [ ]:
```
def printTable(n):
  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)

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

```
Enter n : 7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```

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

In [ ]:
```
def isPositive(n):
  if n < 0:
    return False
  return True
n = int(input("Enter a number : "))
if isPositive(n):
  print("Positive")
else:
  print("Negative")
```

```
Enter a number : 10
Positive
```

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

In [ ]:
```
def isGreater(n1,n2):
  if n1>n2:
    return True
  return False

def largest(n1,n2,n3):
  if isGreater(n1,n2):
    if isGreater(n1,n3):
      return n1
    else:
      return n3
  else:
    if isGreater(n2,n3):
      return n2
    else:
      return n3
n1 = int(input("Enter number 1 : "))
n2 = int(input("Enter number 2 : "))
n3 = int(input("Enter number 3 : "))

print(largest(n1,n2,n3))
```

```
Enter number 1 : 10
Enter number 2 : 20
Enter number 3 : 30
30
```

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

In [ ]:
```
def calculate(n1,n2):
  print(n1,"+",n2,"=",n1+n2)
  print(n1,"-",n2,"=",n1-n2)
  print(n1,"*",n2,"=",n1*n2)
  print(n1,"/",n2,"=",n1//n2)

n1 = int(input("Enter number 1 : "))
n2 = int(input("Enter number 2 : "))
calculate(n1,n2)
```

```
Enter number 1 : 10
Enter number 2 : 2
10 + 2 = 12
10 - 2 = 8
10 * 2 = 20
10 / 2 = 5
```

# 08) WAP to print 1 to n

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

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

## 09) WAP to print odd numbers between 1 to n

In [23]:
```
def printOdd(n):
  for j in range(1,n+1,2):
    print(j)
n = int(input("Enter n : "))
printOdd(n)
```

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

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

In [27]:
```
def seriesSum(n):
  sum = 0
  for i in range(1,n+1):
    sum += i * -(-1)**i
  return sum
n = int(input("Enter n : "))
print(seriesSum(n))
```

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

## 11) WAP to print multiplication table of given number.

In [30]:
```
def printTable(n):
  for i in range(10):
    print("{num} * {times} = {value}".format(num=n,times=i+1,value=n*(i+1)))
n = int(input("Enter n : "))
printTable(n)
```

```
Enter n : 7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```

# 12) WAP to find factorial of the given number.

In [32]:
```
def factorial(n):
  if(n<=1):
    return 1
  return n*factorial(n-1)
n = int(input("Enter a number : "))
print("Factorial of {num} is : {fact}".format(num=n,fact=factorial(n)))
```

```
Enter a number : 6
Factorial of 6 is : 720
```

# 13) WAP to find factors of the given number.

In [35]:
```
def factors(n):
  for i in range(n):
    if not n%(i+1):
      print(i+1,end=" ")
    i+=1
n = int(input("Enter a number : "))
print("Factors of number",n,"are :")
factors(n)
```

```
Enter a number : 18
Factors of number 18 are :
1 2 3 6 9 18
```

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

In [48]:
```
def isPrime(n):
  for i in range(2,math.ceil(math.sqrt(n))):
    if not n%i:
      return False
    i += 1
  return True

n = int(input("Enter a number : "))
if isPrime(n):
  print("Number {} is a prime number".format(n))
else:
  print("Number {} is not a prime number".format(n))
```

```
Enter a number : 101
Number 101 is a prime number
```

# 15) WAP to print sum of digits of given number.

In [50]:
```
def digitSum(n):
  sum = 0
  while(n>0):
    sum += n%10
    n //= 10
  return sum

n = int(input("Enter a number : "))
print("Sum of digit is :",digitSum(n))
```

```
Enter a number : 12345
Sum of digit is : 15
```

# 16) WAP to find out prime numbers between given two numbers.

In [52]:
```
def prime(n1,n2):
  for num in range(max(n1,2),n2+1):
    prime=True
    i=2
    while(i<=math.sqrt(num)):
      if not num%i:
        prime=False
        break
      i+=1
    if prime:
      print(num)
n1 = int(input("Enter number 1 : "))
n2 = int(input("Enter number 2 : "))
prime(n1,n2)
```

```
Enter number 1 : 1
Enter number 2 : 100
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```

## 17) WAP to print Fibonacci series up to number given by user.

In [54]:
```
def fibonacci(n):
  n1,n2,n3=0,1,0
  while not n1>n:
    print(n1,end=" ")
    n3 = n1 + n2
    n1 = n2
    n2 = n3

n = int(input("Enter a number : "))
print("Fibonacci series up to number",n)
fibonacci(n)
```

```
Enter a number : 50
Fibonacci series up to number 50
0 1 1 2 3 5 8 13 21 34
```

# 18) WAP to Find largest prime factor of given number.

In [56]:
```
def largest_prime_factor(n):
  m = n
  while(m>0):
    if not n%m:
      prime = True
      i = 2
      while(i<=math.sqrt(m)):
        if not m%i:
          prime = False
          break
        i += 1
      if prime:
        return m
    m -= 1
n = int(input('Enter a number : '))
print(largest_prime_factor(n))
```

```
Enter a number : 15
5
```

# 19) WAP to print sum for cube of first n natural numbers.

In [57]:
```
def cubeTotal(n):
  cubeTotal = 0
  for i in range(1,n+1):
    cubeTotal += i**3
  return cubeTotal

n = int(input("Enter a number : "))
print("Sum of cube of first {num} natural numbers is : {ans}".format(num=n,ans
=cubeTotal(n)))
```

```
Enter a number : 5
Sum of cube of first 5 natural numbers is : 225
```

## 20) WAP to converts the given decimal number to its binary equivalent.

In [61]:
```
def binary(n):
  if n == 0:
    return
  binary(n//2)
  print(n%2,end="")

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

```
Enter a number : 20
10100
```

## 21) WAP to print following pattern

```
*
* *
* * *
* * * *
```

In [67]:
```
def triangle(n):
  for outer in range(n):
    for inner in range(outer+1):
      print("*",end=" ")
    print()

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

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

## 22) WAP to print following pattern

```
$ $ $ $
$ $ $
$ $
$
```

In [66]:
```
def reverse_triangle(n):
  for outer in range(n,0,-1):
    for inner in range(outer, 0, -1):
      print("$",end=" ")
    print()

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

```
Enter n : 4
$ $ $ $
$ $ $
$ $
$
```

## 23) WAP to print following pattern

```
# # # # #
  # # #
    #
  # # #
# # # # #
```

In [69]:
```
def hourglass(n):
  reverse,i = False,n
  while not (i==n+2 and reverse):
    if i <= 2:
      reverse = True
    for space in range((n-i)//2):
      print(end="  ")
    for symbol in range(i):
      print("#",end=" ")
    print()
    if reverse:
      i += 2
    else:
      i -= 2

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

```
Enter n : 5
# # # # #
  # # #
    #
  # # #
# # # # #
```

# 24) WAP to print following pattern

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

In [70]:
```
def numPattern(n):
  i=1
  for outer in range(n):
    for inner in range(outer+1):
      print(i,end=" ")
      i += 1
    print()
n = int(input("Enter n : "))
numPattern(n)
```

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


(https://www.darshan.ac.in/)

**Python for Data Science - 3150713**

**Lab - 3**

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

In [ ]:
```
str1 = input("Enter a string : ")
if str1 == str1[::-1]:
    print("Palindrome")
else:
    print("Not Palindrome")
```

```
Enter a string : aba
Palindrome
```

# 02) WAP to reverse the word in given string.

In [ ]:
```
string = input("Enter a string : ")
n = int(input("Enter word number : "))
words = str1.split(" ")
words[n-1] = (words[n-1])[::-1]
for i in words:
  print(i,end=" ")
```

```
Enter a string : abc def ghi jkl
Enter word number : 2
abc fed ghi jkl
```

## 03) WAP to remove ith character from given string.

In [ ]:
```
string = input("Enter a string : ")
i = int(input("Enter ith place to remove character : "))
string = string[:i]+string[i+1:]
print(string)
```

```
Enter a string : abcdef
Enter ith place to remove character : 2
abdef
```

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

In [ ]:
```
string = input("Enter a string : ")
len = 0
for char in string:
  len+=1
print("Length :",len)
```

```
Enter a string : Madhav
Length : 6
```

## 05) WAP to print even length word in string

In [ ]:
```
string = input("Enter a string : ")
words = string.split(" ")
for word in words:
  if not (len(word) % 2):
    print(word, end=" ")
```

```
Enter a string : abc defg hij klmnop
defg klmnop
```

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

In [ ]:
```
string = input("Enter a string : ")
vowel = ('a','e','i','o','u')
count = 0
for v in vowel:
  count += string.count(v)
  count += string.count(v.upper())
print(count)
```

```
Enter a string : AbCdefGhIjKlMnopQrStUvWxYz
5
```

## 07) WAP which accepts a string as input to print "Yes" if the string is "yes" or "YES" or "Yes", otherwise print "No".

In [1]:
```
string = input("Enter a string : ")
if string.istitle() or string.islower() or string.isupper():
  print("Yes")
else:
  print("No")
```

```
Enter a string : YES
Yes
```

## 08) WAP to check whether string contains any special character or not.

In [ ]:
```
string = input("Enter a string : ")
if not string.isalnum():
  print("Yes")
else:
  print("No")
```

```
Enter a string : dfhioxv
No
```

```
# This is formatted as code
```

## 09) WAP to generate random strings until a given string is generated.

In [ ]:
```
import random
```

In [ ]:

```
string = input("Enter a string : ")
string = string.upper()
randString = ''
for i in string:
  randomNum = random.randint(65,90)
  while randomNum != ord(i):
    print(randString + chr(randomNum))
    randomNum = random.randint(65,90)
  randString += chr(randomNum)
  print(randString)
```

```
Enter a string : Python
V
Y
O
W
J
Y
G
E
S
H
K
L
H
Z
A
M
K
S
W
O
M
C
H
M
K
J
G
N
W
B
M
L
H
P
PK
PN
PS
PQ
PX
PS
PE
PA
PQ
PO
PY
PYF
PYK
PYN
PYJ
PYD
PYY
PYU
PYG
PYL
PYT
PYTF
```

```
PYTU
PYTW
PYTT
PYTZ
PYTI
PYTX
PYTW
PYTU
PYTY
PYTO
PYTI
PYTH
PYTHO
PYTHOG
PYTHOK
PYTHON
```

## 10) WAP to check given string data in binary formats or not.

In [ ]:
```
string = input("Enter a string : ")
if string.isnumeric() and string.count("0")+string.count("1") == len(string):
  print("Binary")
else:
  print("Not Binary")
```

```
Enter a string : 100101001111
Binary
```

## 11) WAP to find all duplicate characters in string.

In [5]:
```
string = input("Enter a string : ")
for i in range(ord('a'),ord('z')+1):
  if string.count(chr(i))>1:
    print(chr(i))
  if string.count(chr(i-ord('a')+ord('A')))>1:
    print(chr(i-ord('a')+ord('A')))
```

```
Enter a string : abcdaABCAd
a
A
d
```

## 12) WAP to read an ASCII string and to convert it to a Unicode string encoded by utf-8.

In [ ]:
```
string = input("Enter a string : ")
string = string.encode('utf-8')
print(string)
```

```
Enter a string : Madhav
b'Madhav'
```

# 13) WAP to do String slicing

a) Left (Or anticlockwise) rotate the given string by d elements (where d <= n).

b) Right (Or clockwise) rotate the given string by d elements (where d <= n).

In [ ]:
```
string = input('Enter a string : ')
d = int(input("Enter d : "))
left = string[d:] + string[:d]
right = string[len(string)-d:] + string[:len(string)-d]
print("Left rotate by {} is {}".format(d,left))
print("Right rotate by {} is {}".format(d,right))
```

```
Enter a string : Python
Enter d : 2
Left rotate by 2 is thonPy
Right rotate by 2 is onPyth
```


(https://www.darshan.ac.in/)

**Python for Data Science - 3150713**

**Lab - 4**

In [ ]:
```
n = int(input("Enter number of elements of list : "))
elements = []
for i in range(n):
  elements.append(int(input()))
```

```
Enter number of elements of list : 5
30
50
10
40
20
```

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

In [ ]:
```
print("Sum :",sum(elements))
```

```
Sum : 150
```

# 02) WAP to find largest element in a List.

In [ ]:
```
print("Max :",max(elements))
```

```
Max : 50
```

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

In [ ]:
```
i = int(input("Enter ith place from which you want to split list : "))
list1 = elements[:i]
list2 = elements[i:]

print("List 1 :",list1)
print("List 2 :",list2)
list2.extend(list1)
print("After appending the first part to the end :",list2)
```

```
Enter ith place from which you want to split list : 3
List 1 : [30, 50, 10]
List 2 : [40, 20]
After appending the first part to the end : [40, 20, 30, 50, 10]
```

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

In [ ]:
```
print("Before interchanging first and last element :",elements)
temp = elements[0]
elements[0] = elements[-1]
elements[-1] = temp
print("After interchanging first and last element :",elements)
```

```
Before interchanging first and last element : [30, 50, 10, 40, 20]
After interchanging first and last element : [20, 50, 10, 40, 30]
```

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

In [ ]:
```
pos1 = int(input("Enter 1st position : "))
pos2 = int(input("Enter 2nd position : "))
print("Before interchanging first and last element :",elements)
temp = elements[pos1]
elements[pos1] = elements[pos2]
elements[pos2] = temp
print("After interchanging first and last element :",elements)
```

```
Enter 1st position : 1
Enter 2nd position : 3
Before interchanging first and last element : [20, 50, 10, 40, 30]
After interchanging first and last element : [20, 40, 10, 50, 30]
```

# 06) WAP to remove Nth occurrence of the given word in the list entered by user.

In [2]:
```
n = int(input("Enter number of elements of list : "))
list1 = []
for i in range(n):
  list1.append(input())
```

```
Enter number of elements of list : 7
Python
Java
C++
Python
Javascript
Kotlin
C
```

In [3]:
```
word = input("Enter word to remove : ")
occurance = int(input("Enter occurance to remove word : "))
print(list1)

for index in range(n):
  if list1[index]==word:
    occurance -= 1
    if occurance == 0:
      list1.pop(index)
      break

print(list1)
```

```
Enter word to remove : Python
Enter occurance to remove word : 2
['Python', 'Java', 'C++', 'Python', 'Javascript', 'Kotlin', 'C']
['Python', 'Java', 'C++', 'Javascript', 'Kotlin', 'C']
```

# 07) WAP to check whether given element exits in list or not.

In [4]:
```
element = input("Enter element to check : ")
if list1.count(element):
  print("Element",element,"exist")
else:
  print("Element",element,"doesn't exist")
```

```
Enter element to check : Python
Element Python exist
```

## 08) WAP to reverser the list entered by user.

In [ ]:
```
print("Original list :",list1)
list1.reverse()
print("Reversed list :",list1)
```

```
Original list : ['Python', 'Java', 'C++', 'Kotlin', 'Go', 'Python']
Reversed list : ['Python', 'Go', 'Kotlin', 'C++', 'Java', 'Python']
```

## 09) WAP to print all positive number in list entered by user.

In [ ]:
```
n = int(input("Enter number of elements of list : "))
list1 = []
for i in range(n):
  list1.append(int(input()))
```

```
Enter number of elements of list : 5
-4
3
-2
1
0
```

In [ ]:
```
print("\nPositive numbers are :")
for element in list1:
  if element >= 0:
    print(element)
```

```
Positive numbers are :
3
1
0
```

## 10) WAP to check if given array is monotonic or not.

In [ ]:
```
n = int(input("Enter number of elements of list : "))
list1 = []
for i in range(n):
  list1.append(int(input()))
```

```
Enter number of elements of list : 5
10
21
32
43
54
```

In [ ]:
```
ascending = True if list1[0] > list1[1] else False
monotonic = True
for i in range(2,n):
  if (ascending and list1[i-1] < list1[i]) or (not ascending and list1[i-1] >
list1[i]):
    monotonic = False
    break
print("Monotonic" if monotonic else "Not Monotonic")
```

```
Monotonic
```

## 11) WAP which takes 2 digits, X,Y as input and generates a 2-dimensional array of size X *Y. The element value in the i-th row and j-th column of the array should be i*j. Note: i=0,1.., X-1; j=0,1,¡Y-1.

In [ ]:
```
import random
X = int(input("Enter X : "))
Y = int(input("Enter Y : "))
array = []

for i in range(X):
  temp = []
  for j in range(Y):
    temp.append(random.randint(1,100))
  array.append(temp)
```

```
Enter X : 3
Enter Y : 4
```

In [ ]:
```
for outer in array:
  for inner in outer:
    print(inner,end="\t")
  print()
```

```
3       1       46      25
43      15      21      8
34      93      99      37
```

## 12) WAP to find median of array.

In [5]:
```
import numpy as np
```

In [7]:
```
values = [np.random.randint(1,100) for i in range(10)]
print(values)
```

```
[2, 28, 44, 95, 80, 66, 66, 47, 99, 52]
```

In [8]:
```
np.median(values)
```

Out[8]:
```
59.0
```

## 13) Write a program that interchanges the odd and even elements of an array.

In [9]:
```
for i in range(0,10,2):
  temp = values[i]
  values[i] = values[i+1]
  values[i+1] = temp
print(values)
```

```
[28, 2, 95, 44, 66, 80, 47, 66, 52, 99]
```

## 14) WAP to find given number in the list and print the position of number using sequential search.

In [ ]:
```
n = int(input("Enter number of elements of list : "))
list1 = []
for i in range(n):
  list1.append(int(input()))
```

```
Enter number of elements of list : 5
19
28
37
46
55
```

In [ ]:
```
find = int(input("Enter a number to find : "))
for index in range(len(list1)):
  if list1[index]==find:
    print("Position :",index+1)
    break
```

```
Enter a number to find : 46
Position : 4
```

# 15) WAP to multiply two 3*3 matrix.

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

In [ ]:
```
matrix1 = np.random.randint(1,100,9).reshape(3,3)
matrix2 = np.random.randint(1,100,9).reshape(3,3)
print("Matrix 1",matrix1,sep="\n")
print("Matrix 2",matrix2,sep="\n")
```

```
Matrix 1
[[17 61 50]
 [37 80 11]
 [ 1 93 11]]
Matrix 2
[[27 77 65]
 [76 68 34]
 [90 57 98]]
```

In [ ]:
```
way1 = np.matmul(matrix1,matrix2)
print(way1)
```

```
[[9595 8307 8079]
 [8069 8916 6203]
 [8085 7028 4305]]
```

In [ ]:
```
way2 = matrix1 @ matrix2
print(way2)
```

```
[[9595 8307 8079]
 [8069 8916 6203]
 [8085 7028 4305]]
```

In [ ]:
```
way3 = matrix1.dot(matrix2)
print(way3)
```

```
[[9595 8307 8079]
 [8069 8916 6203]
 [8085 7028 4305]]
```

# 16) Write one line of Python code that returns list of even elements of the given list.

In [ ]:
```
randList = np.random.randint(1,1000,10)
print(randList)
```

```
[ 26 178 171 374 410 414 966 222 682  25]
```

In [ ]:
```
print([element for element in randList if not element%2])
```

```
[26, 178, 374, 410, 414, 966, 222, 682]
```

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

In [ ]:
```
randTuples = [tuple(random.randint(1,1000) for _ in range(3)) for _ in range(1
00)]
print(randTuples)
```

```
[(59, 758, 172), (970, 82, 168), (490, 844, 590), (747, 743, 523), (275, 320,
420), (495, 267, 882), (551, 388, 779), (255, 687, 160), (121, 192, 411), (93
3, 984, 850), (829, 29, 447), (127, 628, 178), (881, 677, 479), (618, 71, 26
2), (961, 688, 111), (991, 616, 447), (440, 581, 673), (227, 732, 776), (154,
73, 516), (723, 459, 329), (437, 264, 649), (288, 884, 542), (668, 527, 772),
(859, 600, 492), (891, 889, 689), (143, 833, 403), (222, 303, 81), (721, 424,
279), (450, 423, 187), (994, 180, 826), (44, 639, 176), (176, 213, 772), (85
6, 628, 599), (445, 662, 31), (290, 393, 72), (975, 508, 739), (1, 613, 697),
(940, 377, 990), (548, 321, 138), (965, 897, 401), (941, 447, 236), (894, 27
6, 711), (838, 619, 130), (152, 9, 546), (519, 395, 407), (840, 119, 338), (8
23, 452, 979), (505, 168, 137), (461, 640, 298), (72, 458, 657), (800, 630, 9
92), (836, 298, 243), (855, 401, 802), (607, 499, 26), (673, 362, 320), (395,
756, 188), (421, 983, 489), (781, 525, 980), (573, 580, 502), (527, 15, 367),
(848, 783, 316), (993, 121, 868), (333, 191, 649), (322, 636, 576), (382, 13
3, 38), (310, 586, 561), (975, 142, 802), (266, 455, 664), (698, 169, 939),
(101, 267, 278), (889, 75, 782), (43, 773, 909), (657, 908, 62), (423, 976, 1
66), (690, 902, 975), (27, 771, 866), (473, 591, 447), (648, 452, 55), (252,
761, 331), (735, 791, 66), (367, 338, 826), (522, 868, 513), (312, 310, 542),
(657, 886, 721), (378, 219, 41), (247, 746, 684), (725, 609, 569), (785, 604,
413), (503, 355, 428), (520, 998, 753), (651, 158, 443), (89, 1000, 812), (76
7, 25, 162), (190, 668, 709), (246, 835, 866), (975, 400, 611), (600, 852, 65
7), (974, 102, 481), (347, 528, 233), (909, 471, 89)]
```

In [ ]:
```
K = int(input("Enter K : "))
divisibleTuples = []
for tup in randTuples:
  isDivisible = True
  for element in tup:
    if element%K != 0:
      isDivisible = False
      break
  if isDivisible:
    divisibleTuples.append(tup)
print(divisibleTuples)
```

```
Enter K : 3
[(495, 267, 882), (222, 303, 81), (894, 276, 711), (600, 852, 657)]
```

# 18) WAP to convert tuple into list by adding the given string after every element.

In [ ]:
```
randTuple = tuple(random.randint(1,100) for _ in range(5))
print(randTuple)
string = input("Enter a string : ")
print(string)
```

```
(18, 78, 90, 14, 95)
Enter a string : Python
Python
```

In [ ]:
```
newList = [element for tup in randTuple for element in [tup, string]]
print(newList)
```

```
[18, 'Python', 78, 'Python', 90, 'Python', 14, 'Python', 95, 'Python']
```

# 19) WAP to sort python dictionary by key or value.

In [ ]:
```
subject4 = {"OS":5, "OOP":5, "COA":4, "DM":5, "PEM":3}
subcode4 = {3140702,3140705,3140707,3140708,3140709,3140702,3140705,3140707,31
40708,3140709,3150713,310709,3150710}
subject5 = {"ADA":5, "PE":3, "CN":5, "SE":4, "PDS":3}
subcode5 = {3150713,3150711,3150703,3150713,3150709,3150710,3140705,3140707,31
40708,3140709}
```

In [ ]:
```
for i in sorted(subject4):
  print(i, subject4[i])
```

```
COA 4
DM 5
OOP 5
OS 5
PEM 3
```

## 20) WAP to merge two dictionaries given by user.

In [ ]:
```
subject4.update(subject5)
print(subject4)
```

```
{'OS': 5, 'OOP': 5, 'COA': 4, 'DM': 5, 'PEM': 3, 'ADA': 5, 'PE': 3, 'CN': 5,
'SE': 4, 'PDS': 3}
```

## 21) WAP to merge two sets given by user.

In [ ]:
```
subcodes = subcode4.union(subcode5)
print(subcodes)
```

```
{3140705, 3140707, 3140708, 3140709, 3150703, 3150711, 310709, 3150710, 31507
09, 3150713, 3140702}
```

## 22) WAP to find intersection of two sets given by user.

In [ ]:
```
commonSubjects = subcode4.intersection(subcode5)
print(commonSubjects)
```

```
{3140705, 3140707, 3140708, 3140709, 3150710, 3150713}
```

(https://www.darshan.ac.in/)

**Python for Data Science - 3150713**

**Lab - 5**

# 1) Scrape the data of Faculty and news data from darshan university website.

In [ ]:
```
import requests
import bs4
```

In [ ]:
```
branches = ("computer","civil","electrical","mechanical","humanities-and-scien
ce")
print("Name","Designation","Qualification","Experience","Since",sep="\t\t")
for branch in branches:
  data = requests.get("https://darshan.ac.in/engineering/"+branch+"/faculty")
  soup = bs4.BeautifulSoup(data.text,"lxml")
  allFaculty = soup.select("body > main > div")
  allFaculty = soup.select("div > div > div.col-lg-8.col-xl-9.g-font-size-14 >
div > div")
  for faculty in allFaculty:
    name = faculty.select("h2")[0].text.strip().title().replace("  "," ")
    details = faculty.select("span")
    print(name, end="\t")
    for i in range(5):
      if i==2:
        continue
      detail = details[i].text.strip().title().replace("  "," ")
      print(detail,end="\t")
    print()
```

# 2) Scrape the data of student placement from darshan university website.

In [ ]:
```
years = [str(year) for year in range(2023,2018,-1)]

for year in years:
  data = requests.get("https://darshan.ac.in/placement/list/engineering/computer/"+year)
  soup = bs4.BeautifulSoup(data.text,"lxml")
  allStudents = soup.select("body > main > div > div > div > div.col-lg-8.col-xl-9.g-font-size-14.g-mb-30 > div.g-pt-20 > div.row.g-px-4.g-px-5--sm.g-px-5--md > div")
  print("\n\n------ YEAR : ",year,"------\n")
  print("Name","Enrollment No.","CTC per Annum", "Placed in",sep="\t\t\t\t\t\t\t")
  for student in allStudents:
    name = str(student.select("h2 > span")[0]).split("<br/>")
    name[0] = name[0].split(">")[1].strip().title()
    name[1] = name[1].split("<")[0].strip().title()
    details = student.select("p")
    enrollment = details[0].text.strip()
    company = str(student.select("div > div > a > img")[0]).split("alt=\"")[1].split("\"")[0].strip()

    ctc = details[2].text.strip()
    print(name[0]+" "+name[1],enrollment,ctc,company,sep="\t\t\t\t\t\t\t")
```

# Write a python program to perform database operations on database using mysql and sqlite3.

1) Display Student table from database.
2) Insert new record into student table by user.

In [ ]:
```
import sqlite3
import pandas as pd
```

In [15]:
```
db_connection=sqlite3.connect("student.db")
cursor = db_connection.cursor()
```

In [ ]:
```
"""(1) Display Student table from database."""
cursor.execute("SELECT * FROM student")
rows = cursor.fetchall()
for id,name,roll in rows:
  print(id,name,roll)
print()

"""(2) Insert new record into student table by user."""
insert = "INSERT INTO student values(?, ?, ?);"
data = (4,"XXX",111)
cursor.execute(insert,data)
db_connection.commit()

"""Display Table"""
cursor.execute("SELECT * FROM student")
rows = cursor.fetchall()
for id,name,roll in rows:
  print(id,name,roll)

"""Delete Data"""
cursor.execute("DELETE FROM student where S_id = 4")
db_connection.commit()
```

In [17]:
```
cursor.close()
```


(https://www.darshan.ac.in/)

**Python for Data Science - 3150713**

**Lab - 6**

# 1. WAP to read entire file named abc.txt

In [2]:
```
with open('D://abc.txt', 'r') as f:
  print(f.read())
```

```
TXT test file
Purpose: Provide example of this file type
Line 1
Line 2
Line 3
Line 4
Line 5
```

# 2. WAP to read first 5 lines from the file named abc.txt

In [3]:
```
with open('D://abc.txt', 'r') as f:
 for i in range(5):
    print(f.readline().strip())
```

```
TXT test file
Purpose: Provide example of this file type
Line 1
Line 2
Line 3
```

## 3. WAP to read file line by line and store lines as a List.

In [4]:
```
n = int(input("Enter number of lines : "))
with open('D:\\abc.txt', 'r') as f:
  lst = []
  for i in range(n):
    lst.append(f.readline().rstrip())
  print(lst)
```

```
Enter number of lines : 5
['TXT test file', 'Purpose: Provide example of this file type', 'Line 1', 'Li
ne 2', 'Line 3']
```

## 4. WAP to find the longest word in a file named abc.txt

In [7]:
```
def longest_word(filename):
    with open(filename, 'r') as infile:
              words = infile.read().split()
    max_len = len(max(words, key=len))
    return [word for word in words if len(word) == max_len]

print(longest_word('D:\\abc.txt'))
```

```
['Purpose:']
```

## 5. WAP to count lines, word, and characters within a text file.

In [9]:
```
with open('D:\\abc.txt', 'r') as f:
  data = f.read()
  lines = len(data.split('\n'))
  words = len(data.split())
  characters = sum([len(i) for i in data.split('\n')])
  print("Lines :",lines)
  print("Words :",words)
  print("Characters :",characters)
```

```
Lines : 8
Words : 20
Characters : 85
```

## 6. Write an application that reads a file and counts the number of occurrences of word.

In [10]:
```
word = input("Enter word to find in file :")
with open('D://abc.txt', 'r') as f:
  data = f.read().split()
  print(data.count(word))
```

```
Enter word to find in file :Line
5
```

## 7. WAP to copy content of abc.txt to xyz.txt

In [12]:
```
with open('D://abc.txt', 'r') as source:
  with open('D://xyz.txt', 'w') as destination:
    destination.write(source.read())
    print("Successful !")
```

```
Successful !
```

## 8. WAP to append next 50 prime numbers to a file named primenumbers.txt)

In [13]:
```
def prime(number):
  flag = True
  for i in range(2,number):
    if(number%i == 0):
      flag = False
      break
  return flag

with open('primenos.txt', 'a') as primenos:
  i=0
  number = 2
  while i != 50:
    if(prime(number)):
        primenos.write(str(number)+"\n")
        i = i + 1
    number = number + 1
  print("Successful !")
```

```
Successful !
```

In [ ]:
```

```


(https://www.darshan.ac.in/)

**Python for Data Science - 3150713**

**Lab - 7**

# NumPy Exercises

Now that we've learned about NumPy let's test your knowledge. We'll start off with a few simple tasks, and then you'll be asked some more complicated questions.

#### import numpy as np

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

#### Create an array of 10 zeros

```
In [2]: np.zeros(10)
Out[2]: array([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 [3]: np.ones(10)
Out[3]: array([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 [4]: np.ones(10)*5
Out[4]: array([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 [5]: np.arange(10, 51)
Out[5]: 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])
```

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 [6]: np.arange(10, 51, 2)
Out[6]: array([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 [7]: np.matrix('0 1 2; 3 4 5; 6 7 8')
Out[7]: matrix([[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 [8]: np.identity(3)
```

```
Out[8]: 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 [9]: np.random.random()
```

```
Out[9]: 0.3469888820832657
```

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

```
In [10]: np.random.normal(10, 5, 25)
```

```
Out[10]: array([-0.5396173 ,  6.72732007, 12.73403742,  6.92558252,  4.02809709,
                9.64820535, 16.38796372, 13.63325724, 13.16164058, 14.81313907,
               10.5621281 , 13.01840862,  8.38834482,  6.55038334,  9.11087143,
                9.58782259, 16.98767405,  0.74276178, 11.95106998,  2.9776923 ,
                7.7641266 , 16.46566161, 12.0885632 , 16.50510623,  9.526635  ])
```

**Create the following matrix:**

```
In [11]: np.linspace(0.01, 1, 100).reshape(10, 10)

Out[11]: array([[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 [12]: np.linspace(0, 1, 20)

Out[12]: 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.        ])
```

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 [13]: #create array here
         mat =np.arange(1, 26).reshape(5,5)
         mat

Out[13]: 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]])
```

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 [14]: mat[2:,1:]

Out[14]: array([[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]])
```

```
In [15]: mat[3,4]

Out[15]: 20
```

Output should be = 20 (element at specific index)

```
In [16]: mat[:3,1:2]

Out[16]: array([[ 2],
                [ 7],
                [12]])
```

Output should be =

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

```
In [17]: mat[-1]
```

```
Out[17]: array([21, 22, 23, 24, 25])
```

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

```
In [18]: mat[-2:]
```

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

Output should be =

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

```
In [19]: mat[(mat%3==0) & (mat%5!=0)]
```

```
Out[19]: 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 [20]: mat.sum()
```

```
Out[20]: 325
```

Output should be = 325

#### Get the standard deviation of the values in mat

```
In [21]: mat.std()
Out[21]: 7.211102550927978
```

Output should be = 7.211102550927978

**Get the sum of all the columns in mat**

```
In [22]: mat.sum(axis=0)
Out[22]: array([55, 60, 65, 70, 75])
```

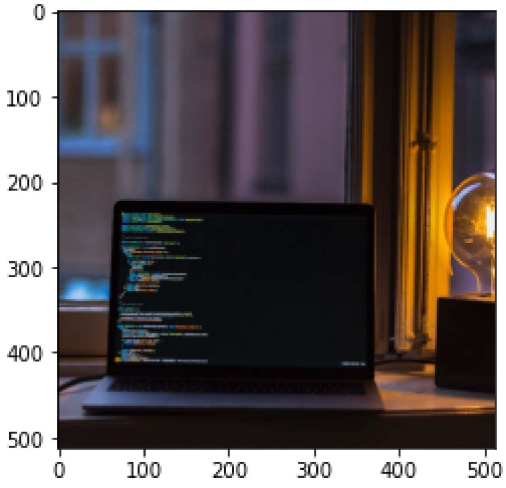
Output should be = array([55, 60, 65, 70, 75])

# Convert image into gray scale

```
In [29]: from skimage import io
         import matplotlib.pyplot as plt
```

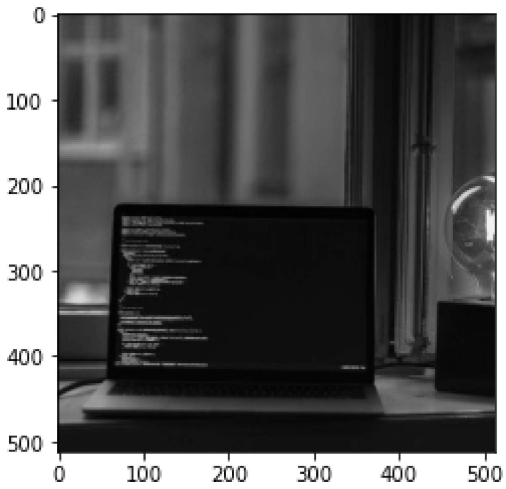
```
In [33]: image = io.imread("demo.jpg")
```

```
In [34]: plt.imshow(image)
Out[34]: <matplotlib.image.AxesImage at 0x7f8720322b50>
```



In [35]:
```
r,g,b = image[::,::,0],image[::,::,1],image[::,::,2]

imgGray = 0.2989 * r + 0.5870 * b + 0.1140 * g

image = imgGray

plt.imshow(image,cmap='gray')
```

Out[35]: <matplotlib.image.AxesImage at 0x7f8723f4d3a0>




# Nice work, Keep the spark alive

In [ ]:


(https://www.darshan.ac.in/)

Python for Data Science - 3150713

Lab - 8

# Pandas SF Salaries Exercise

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 [1]: import pandas as pd
```

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

```
In [2]: df = pd.read_csv("Salaries.csv")
```

**Check the head of the DataFrame.**

In [3]:
```
df
```

Out[3]:

| | Id | EmployeeName | JobTitle | BasePay | OvertimePay | OtherPay | Benefits |
|---|---|---|---|---|---|---|---|
| **0** | 1 | NATHANIEL FORD | GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY | 167411.18 | 0.00 | 400184.25 | NaN |
| **1** | 2 | GARY JIMENEZ | CAPTAIN III (POLICE DEPARTMENT) | 155966.02 | 245131.88 | 137811.38 | NaN |
| **2** | 3 | ALBERT PARDINI | CAPTAIN III (POLICE DEPARTMENT) | 212739.13 | 106088.18 | 16452.60 | NaN |
| **3** | 4 | CHRISTOPHER CHONG | WIRE ROPE CABLE MAINTENANCE MECHANIC | 77916.00 | 56120.71 | 198306.90 | NaN |
| **4** | 5 | PATRICK GARDNER | DEPUTY CHIEF OF DEPARTMENT, (FIRE DEPARTMENT) | 134401.60 | 9737.00 | 182234.59 | NaN |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **148649** | 148650 | Roy I Tillery | Custodian | 0.00 | 0.00 | 0.00 | 0.0 |
| **148650** | 148651 | Not provided | Not provided | NaN | NaN | NaN | NaN |
| **148651** | 148652 | Not provided | Not provided | NaN | NaN | NaN | NaN |
| **148652** | 148653 | Not provided | Not provided | NaN | NaN | NaN | NaN |
| **148653** | 148654 | Joe Lopez | Counselor, Log Cabin Ranch | 0.00 | 0.00 | -618.13 | 0.0 |

148654 rows × 13 columns

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

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148654 entries, 0 to 148653
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Id                148654 non-null  int64  
 1   EmployeeName      148654 non-null  object 
 2   JobTitle          148654 non-null  object 
 3   BasePay           148045 non-null  float64
 4   OvertimePay       148650 non-null  float64
 5   OtherPay          148650 non-null  float64
 6   Benefits          112491 non-null  float64
 7   TotalPay          148654 non-null  float64
 8   TotalPayBenefits  148654 non-null  float64
 9   Year              148654 non-null  int64  
 10  Notes             0 non-null       float64
 11  Agency            148654 non-null  object 
 12  Status            0 non-null       float64
dtypes: float64(8), int64(2), object(3)
memory usage: 14.7+ MB
```

**What is the average BasePay ?**

In [5]:
```
df["BasePay"].mean()
```

Out[5]:
```
66325.44884050643
```

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

In [6]:
```
df['OvertimePay'].max()
```

Out[6]:
```
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 [7]:
```
df[df["EmployeeName"] == "JOSEPH DRISCOLL"]['JobTitle']
```

Out[7]:
```
24    CAPTAIN, FIRE SUPPRESSION
Name: JobTitle, dtype: object
```

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

In [8]:
```
df[df['EmployeeName']=='JOSEPH DRISCOLL']['TotalPayBenefits']
```

Out[8]:
```
24    270324.91
Name: TotalPayBenefits, dtype: float64
```

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

In [9]:
```
df[df['TotalPayBenefits'] == df['TotalPayBenefits'].max()]
```

Out[9]:

| | Id | EmployeeName | JobTitle | BasePay | OvertimePay | OtherPay | Benefits | TotalPay |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | NATHANIEL FORD | GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY | 167411.18 | 0.0 | 400184.25 | NaN | 567595.43 |

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

In [10]:
```
df[df["TotalPayBenefits"] == df['TotalPayBenefits'].min()]
```

Out[10]:

| | Id | EmployeeName | JobTitle | BasePay | OvertimePay | OtherPay | Benefits | TotalPay |
|---|---|---|---|---|---|---|---|---|
| **148653** | 148654 | Joe Lopez | Counselor, Log Cabin Ranch | 0.0 | 0.0 | -618.13 | 0.0 | -618.13 |

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

In [11]:
```
df.groupby('Year').mean()["BasePay"]
```

Out[11]:
```
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 [12]:
```
df['JobTitle'].nunique()
```

Out[12]: 2159

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

In [13]:
```
df['JobTitle'].value_counts().head()
```

Out[13]:
```
Transit Operator                7036
Special Nurse                   4389
Registered Nurse                3736
Public Svc Aide-Public Works    2518
Police Officer 3                2421
Name: JobTitle, 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 [14]:
```
(df[df['Year']==2013]['JobTitle'].value_counts()==1).sum()
```

Out[14]:
```
202
```

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

In [17]:
```
def find_chief(job_title):
    if 'chief' in job_title.lower().split():
        return True
    else:
        return False
```

In [18]:
```
sum(df['JobTitle'].apply(lambda x: find_chief(x)))
```

Out[18]:
```
477
```

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

In [19]:
```
df['title_len'] = df['JobTitle'].apply(len)
```

In [20]:
```
df[['title_len', 'TotalPayBenefits']].corr()
```

Out[20]:

| | title_len | TotalPayBenefits |
|---|---|---|
| **title_len** | 1.000000 | -0.036878 |
| **TotalPayBenefits** | -0.036878 | 1.000000 |

# Great Job!


(https://www.darshan.ac.in/)

**Python for Data Science - 3150713**

**Lab - 9**

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

In [3]:
```
ecom=pd.read_csv('Ecommerce Purchases.csv')
ecom
```

Out[3]:

| | Address | Lot | AM or PM | Browser Info | Company | Credit Card | CC Exp Date | Se |
|---|---|---|---|---|---|---|---|---|
| **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 | 02/20 | |
| **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 | 11/18 | |
| **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 | 08/19 | |
| **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 | 02/24 | |
| **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 | 10/25 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **9995** | 966 Castaneda Locks\nWest Juliafurt, CO 96415 | 92 XI | PM | Mozilla/5.0 (Windows NT 5.1) AppleWebKit/5352 ... | Randall-Sloan | 342945015358701 | 03/22 | |
| **9996** | 832 Curtis Dam Suite 785\nNorth Edwardburgh, T... | 41 JY | AM | Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ... | Hale, Collins and Wilson | 210033169205009 | 07/25 | |
| **9997** | Unit 4434 Box 6343\nDPO AE 28026-0283 | 74 Zh | AM | Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_7... | Anderson Ltd | 6011539787356311 | 05/21 | |
| **9998** | 0096 English Rest\nRoystad, IA 12457 | 74 cL | PM | Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_8;... | Cook Inc | 180003348082930 | 11/17 | |
| **9999** | 40674 Barrett Stravenue\nGrimesville, WI 79682 | 64 Hr | AM | Mozilla/5.0 (X11; Linux i686; rv:1.9.5.20) Gec... | Greene Inc | 4139972901927273 | 02/19 | |

10000 rows × 14 columns

**Check the head of the DataFrame.**

In [5]:
```
ecom.head()
```

Out[5]:

| | Address | Lot | AM or PM | Browser Info | Company | Credit Card | CC Exp Date | CC Security Code | Pro |
|---|---|---|---|---|---|---|---|---|---|
| 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 | 02/20 | 900 | J |
| 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 | 11/18 | 561 | Mast |
| 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 | 08/19 | 699 | J |
| 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 | 02/24 | 384 | Dis |
| 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 | 10/25 | 678 | I<br>Bl |

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

In [6]:
```
ecom.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Address           10000 non-null  object 
 1   Lot               10000 non-null  object 
 2   AM or PM          10000 non-null  object 
 3   Browser Info      10000 non-null  object 
 4   Company           10000 non-null  object 
 5   Credit Card       10000 non-null  int64  
 6   CC Exp Date       10000 non-null  object 
 7   CC Security Code  10000 non-null  int64  
 8   CC Provider       10000 non-null  object 
 9   Email             10000 non-null  object 
 10  Job               10000 non-null  object 
 11  IP Address        10000 non-null  object 
 12  Language          10000 non-null  object 
 13  Purchase Price    10000 non-null  float64
dtypes: float64(1), int64(2), object(11)
memory usage: 1.1+ MB
```

**What is the average Purchase Price?**

In [7]:
```
ecom['Purchase Price'].mean()
```

Out[7]:
```
50.34730200000025
```

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

In [8]:
```
ecom['Purchase Price'].max()
```

Out[8]:
```
99.99
```

In [9]:
```
ecom['Purchase Price'].min()
```

Out[9]:
```
0.0
```

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

In [10]:
```
len(ecom[ecom['Language'].str.contains('en')])
```

Out[10]:
```
1098
```

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

```
In [11]: len(ecom[ecom['Job']=='Lawyer'])

Out[11]: 30
```

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

**(Hint: Check out value_counts() (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.value_counts.html) )**

```
In [12]: ecom['AM or PM'].value_counts()

Out[12]: PM    5068
         AM    4932
         Name: AM or PM, dtype: int64
```

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

```
In [13]: ecom['Job'].value_counts().head(5)

Out[13]: 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 [14]: ecom[ecom['Lot']=='90 WT']['Purchase Price']

Out[14]: 513    75.1
         Name: Purchase Price, dtype: float64
```

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

```
In [15]: ecom[ecom['Credit Card']==4926535242672853]['Email']

Out[15]: 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 [16]: sum(ecom[ecom['CC Provider']=='American Express']['Purchase Price']>95)
Out[16]: 39
```

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

```
In [17]: len(ecom[ecom['CC Exp Date'].str.contains('25')])
Out[17]: 1033
```

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

```
In [18]: ecom['Email'].str.split('@').str[1].value_counts().head()
Out[18]: hotmail.com     1638
         yahoo.com       1616
         gmail.com       1605
         smith.com         42
         williams.com      37
         Name: Email, dtype: int64
```

# Great Job!

(https://www.darshan.ac.in/)

Python for Data Science - 3150713

Lab - 10

# MatPlotLib

# import matplotlib below

In [1]:
```
import matplotlib.pyplot as plt
```

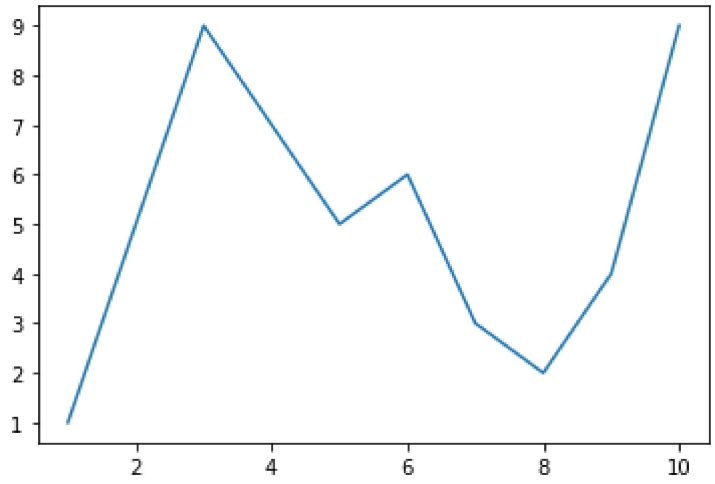
# set matplotlib inline below

In [2]:
```
%matplotlib inline
```

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

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

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

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

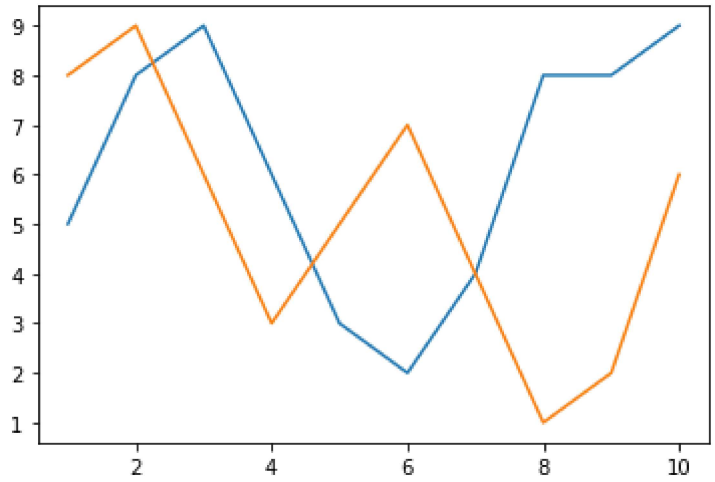



# write a code to display two lines in a line chart (data given below)

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]

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


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

Out[4]:
```
[<matplotlib.lines.Line2D at 0x7f7bb1225e80>]
```




# write a code to save a graph as image named 'classMarks.png'

In [5]:
```
plt.savefig('classMarks.png',format='png')
```

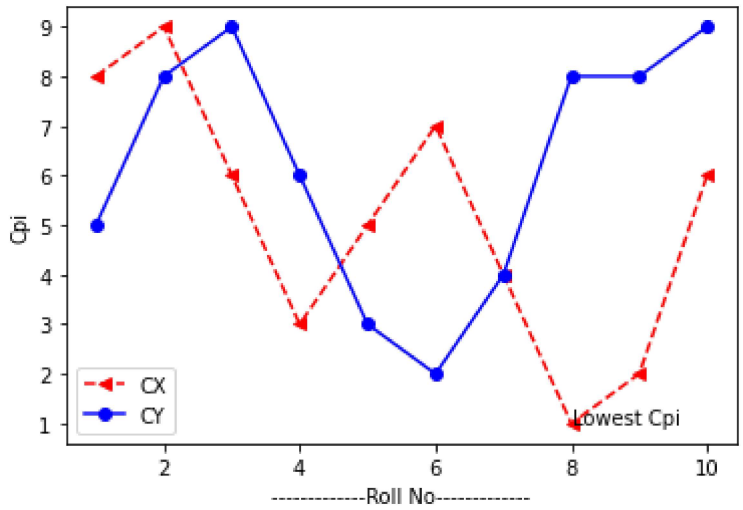
```
<Figure size 432x288 with 0 Axes>
```

# write a code to generate below graph

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]

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

plt.plot(x,cxMarks,ls='--',c='r',marker='<')
plt.plot(x,cyMarks,ls='-',c='b',marker='o')

plt.xlabel('-------------Roll No-------------')
plt.ylabel('Cpi')

plt.annotate(xy=[8,1],text='Lowest Cpi')

plt.legend(['CX','CY'])
```

Out[6]: <matplotlib.legend.Legend at 0x7f7bb105e490>

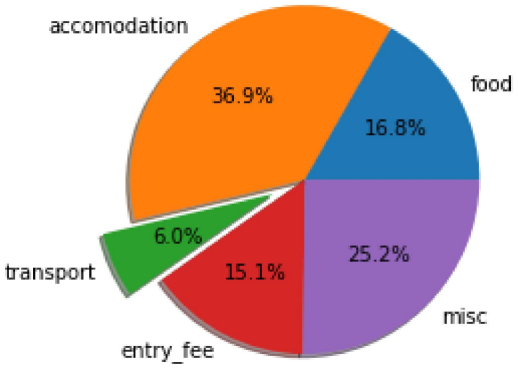



# write a code to generate below pie chart

v = [100,220,36,90,150]

l = ['food','accomodation','transport','entry_fee','misc']

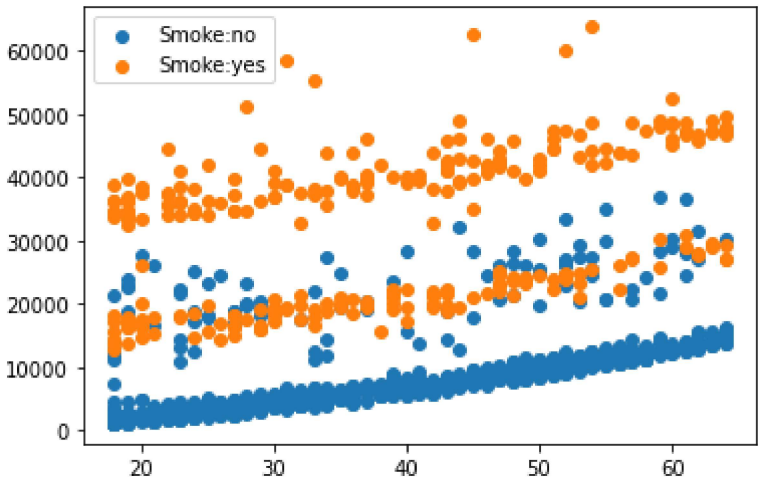
In [7]:
```
v = [100,220,36,90,150]
l = ['food','accomodation','transport','entry_fee','misc']
e = [0,0,.2,0,0]
plt.pie(v,labels=l,explode=e,autopct='%.1f%%',shadow=True)
plt.show()
```




## Generate scatter plot grouped by smoker field and age on x axis and charges on y axis

df = pd.read_csv('insurance.csv')

In [8]:
```
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline

df = pd.read_csv('insurance.csv')
grouped = df.groupby(['smoker'])
for key,group in grouped :
    plt.scatter(group['age'],group['charges'],label=key)

plt.legend(['Smoke:no','Smoke:yes'])
plt.show()
```



(https://www.darshan.ac.in/)

Python for Data Science - 3150713

Lab - 11

# Data Wrangling

# WAP to implement linear regression with iphone csv

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

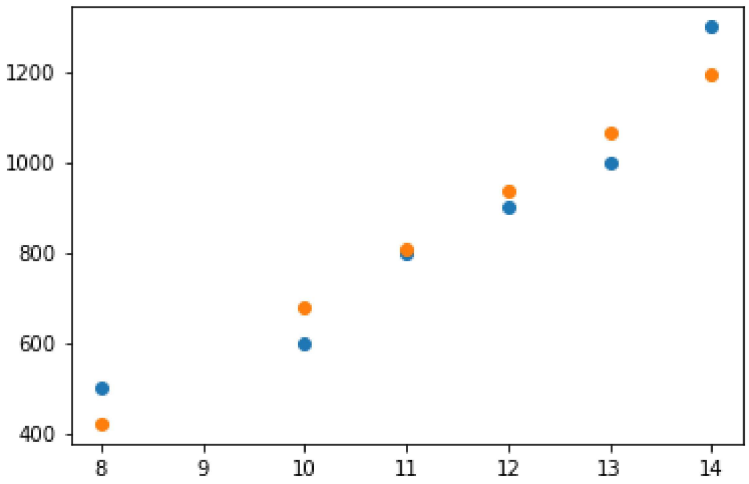
In [2]:
```
df = pd.read_csv("iphone.csv")
```

In [6]:
```
df
```

Out[6]:

| | number | Ipnone name | Price |
|---|---|---|---|
| **0** | 8 | Iphone 8 | 499 |
| **1** | 10 | Iphone X | 600 |
| **2** | 11 | Iphone 11 | 800 |
| **3** | 12 | Iphone 12 | 900 |
| **4** | 13 | Iphone 13 | 1000 |
| **5** | 14 | Iphone 14 | 1299 |

```
In [7]: from sklearn.linear_model import LinearRegression

In [8]: m = LinearRegression()

In [9]: x = df["number"]
        y = df["Price"]

In [10]: newx = x.values.reshape(-1,1)
         newx

Out[10]: array([[ 8],
                [10],
                [11],
                [12],
                [13],
                [14]])

In [11]: m.fit(newx,y)

Out[11]: LinearRegression()

In [12]: predictval = m.predict(newx)

In [13]: import matplotlib.pyplot as plt

In [15]: plt.scatter(x,y)
         plt.scatter(x,predictval)

Out[15]: <matplotlib.collections.PathCollection at 0x7f90a6d615b0>
```




# WAP to implement TF-IDF on following data

a = [ 'this is the first document', 'this document is the second document', 'and this is the third one', 'is this the first document', ]

In [16]:
```
a = [
      'this is the first document',
      'this document is the second document',
      'and this is the third one',
      'is this the first document',
 ]
```

In [18]:
```
from sklearn.feature_extraction.text import *
```

In [19]:
```
tv = TfidfVectorizer()
```

In [20]:
```
result = tv.fit_transform(a)
```

In [21]:
```
result.toarray()
```

Out[21]:
```
array([[0.        , 0.46979139, 0.58028582, 0.38408524, 0.        ,
        0.        , 0.38408524, 0.        , 0.38408524],
       [0.        , 0.6876236 , 0.        , 0.28108867, 0.        ,
        0.53864762, 0.28108867, 0.        , 0.28108867],
       [0.51184851, 0.        , 0.        , 0.26710379, 0.51184851,
        0.        , 0.26710379, 0.51184851, 0.26710379],
       [0.        , 0.46979139, 0.58028582, 0.38408524, 0.        ,
        0.        , 0.38408524, 0.        , 0.38408524]])
```

In [22]:
```
tv.vocabulary_
```

Out[22]:
```
{'this': 8,
 'is': 3,
 'the': 6,
 'first': 2,
 'document': 1,
 'second': 5,
 'and': 0,
 'third': 7,
 'one': 4}
```

# Perform EDA on titanic dataset from kaggle, download dataset from : https://www.kaggle.com/c/titanic (https://www.kaggle.com/c/titanic)

In [24]:
```
df = pd.read_csv("titanic.csv")
```

In [26]:
```
df
```

Out[26]:

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **886** | 887 | 0 | 2 | Montvila, Rev. Juozas | male | 27.0 | 0 | 0 | 211536 | 13.0000 |
| **887** | 888 | 1 | 1 | Graham, Miss. Margaret Edith | female | 19.0 | 0 | 0 | 112053 | 30.0000 |
| **888** | 889 | 0 | 3 | Johnston, Miss. Catherine Helen "Carrie" | female | NaN | 1 | 2 | W./C. 6607 | 23.4500 |
| **889** | 890 | 1 | 1 | Behr, Mr. Karl Howell | male | 26.0 | 0 | 0 | 111369 | 30.0000 |
| **890** | 891 | 0 | 3 | Dooley, Mr. Patrick | male | 32.0 | 0 | 0 | 370376 | 7.7500 |

891 rows × 12 columns

In [27]:
```
df.mean()
```

```
/var/folders/30/3wcvnzrj19z45z0zlz3pvb7h0000gn/T/ipykernel_10334/3698961737.p
y:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (wit
h 'numeric_only=None') is deprecated; in a future version this will raise Typ
eError.  Select only valid columns before calling the reduction.
  df.mean()
```

Out[27]:
```
PassengerId    446.000000
Survived         0.383838
Pclass           2.308642
Age             29.699118
SibSp            0.523008
Parch            0.381594
Fare            32.204208
dtype: float64
```

In [28]:
```
df.max()
```

```
/var/folders/30/3wcvnzrj19z45z0zlz3pvb7h0000gn/T/ipykernel_10334/1151452817.p
y:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (wit
h 'numeric_only=None') is deprecated; in a future version this will raise Typ
eError.  Select only valid columns before calling the reduction.
  df.max()
```

Out[28]:
```
PassengerId                             891
Survived                                  1
Pclass                                    3
Name           van Melkebeke, Mr. Philemon
Sex                                    male
Age                                    80.0
SibSp                                     8
Parch                                     6
Ticket                           WE/P 5735
Fare                              512.3292
dtype: object
```

In [29]:
```
df.min()
```

```
/var/folders/30/3wcvnzrj19z45z0zlz3pvb7h0000gn/T/ipykernel_10334/3962516015.p
y:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (wit
h 'numeric_only=None') is deprecated; in a future version this will raise Typ
eError.  Select only valid columns before calling the reduction.
  df.min()
```

Out[29]:
```
PassengerId                     1
Survived                        0
Pclass                          1
Name           Abbing, Mr. Anthony
Sex                         female
Age                           0.42
SibSp                            0
Parch                            0
Ticket                      110152
Fare                           0.0
dtype: object
```

In [30]:
```
df.describe()
```

Out[30]:

| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|---|
| **count** | 891.000000 | 891.000000 | 891.000000 | 714.000000 | 891.000000 | 891.000000 | 891.000000 |
| **mean** | 446.000000 | 0.383838 | 2.308642 | 29.699118 | 0.523008 | 0.381594 | 32.204208 |
| **std** | 257.353842 | 0.486592 | 0.836071 | 14.526497 | 1.102743 | 0.806057 | 49.693429 |
| **min** | 1.000000 | 0.000000 | 1.000000 | 0.420000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 223.500000 | 0.000000 | 2.000000 | 20.125000 | 0.000000 | 0.000000 | 7.910400 |
| **50%** | 446.000000 | 0.000000 | 3.000000 | 28.000000 | 0.000000 | 0.000000 | 14.454200 |
| **75%** | 668.500000 | 1.000000 | 3.000000 | 38.000000 | 1.000000 | 0.000000 | 31.000000 |
| **max** | 891.000000 | 1.000000 | 3.000000 | 80.000000 | 8.000000 | 6.000000 | 512.329200 |

In [31]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [33]:
```
df['Sex'].value_counts()
```

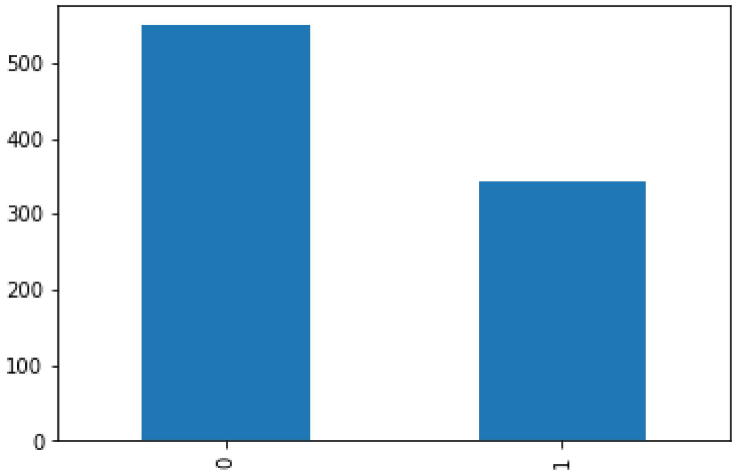
Out[33]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [34]:
```
df['Survived'].value_counts()
```

Out[34]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [37]:
```
df['Survived'].value_counts().plot(kind='bar')
```

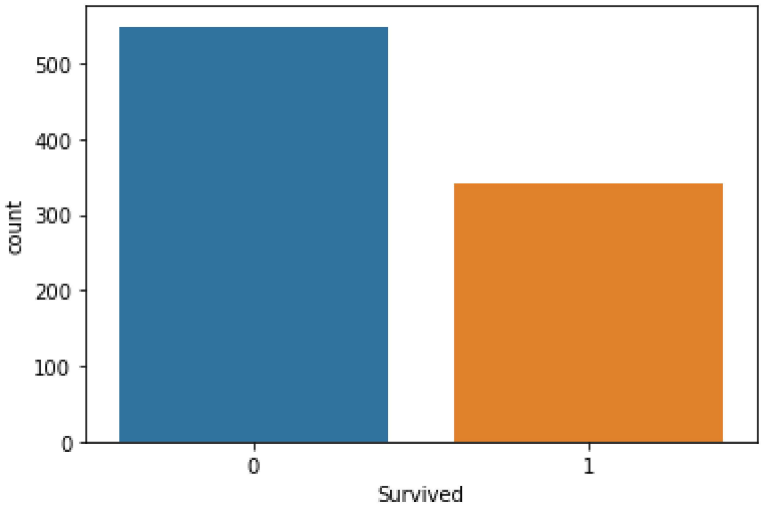
Out[37]: <AxesSubplot:>




In [38]:
```
import seaborn as sns
```

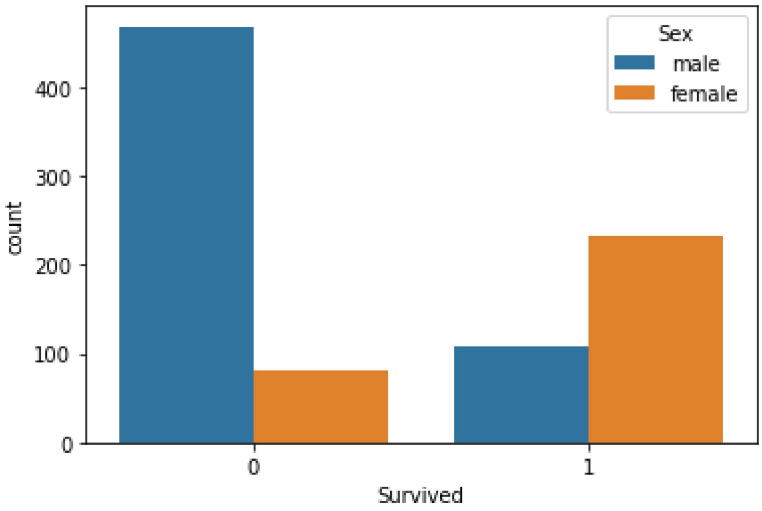
In [39]:
```
sns.countplot(x=df['Survived'],data=df)
```

Out[39]: <AxesSubplot:xlabel='Survived', ylabel='count'>



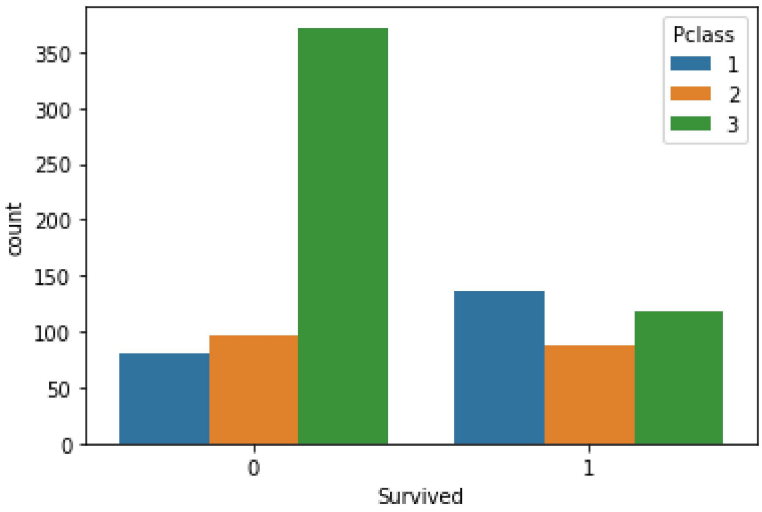
In [41]:
```
sns.countplot(x=df['Survived'],data=df,hue=df['Sex'])
```

Out[41]: <AxesSubplot:xlabel='Survived', ylabel='count'>



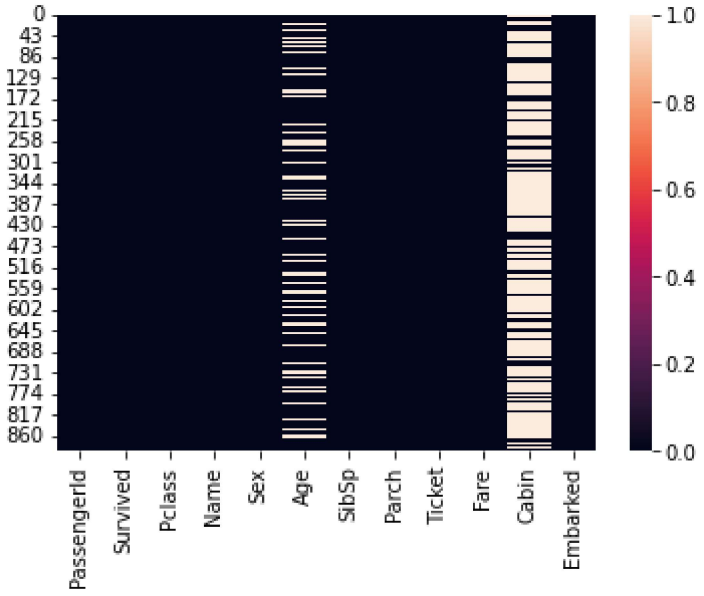

In [42]:
```
sns.countplot(x=df['Survived'],data=df,hue=df['Pclass'])
```

Out[42]: <AxesSubplot:xlabel='Survived', ylabel='count'>



In [47]:
```
sns.heatmap(data=df.isnull())
```

Out[47]: <AxesSubplot:>

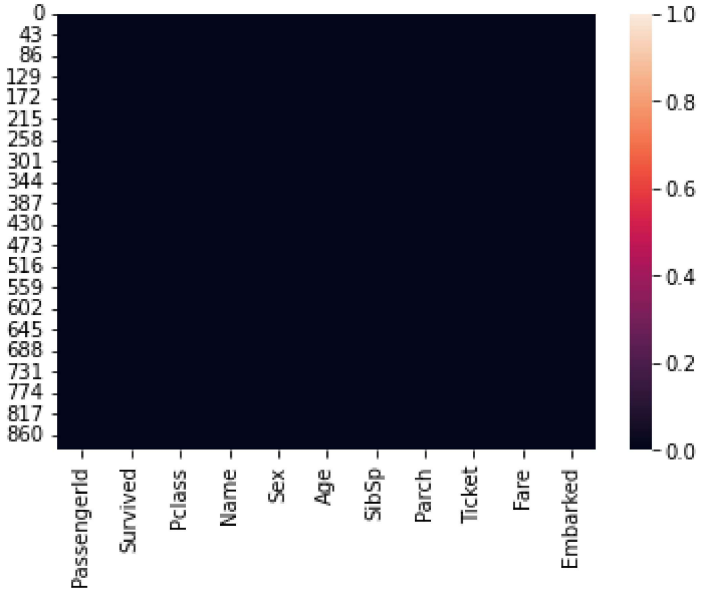



In [51]:
```
df.drop("Cabin",axis=1,inplace=True)
```

In [53]:
```
df['Age'].fillna("29",inplace=True)
```

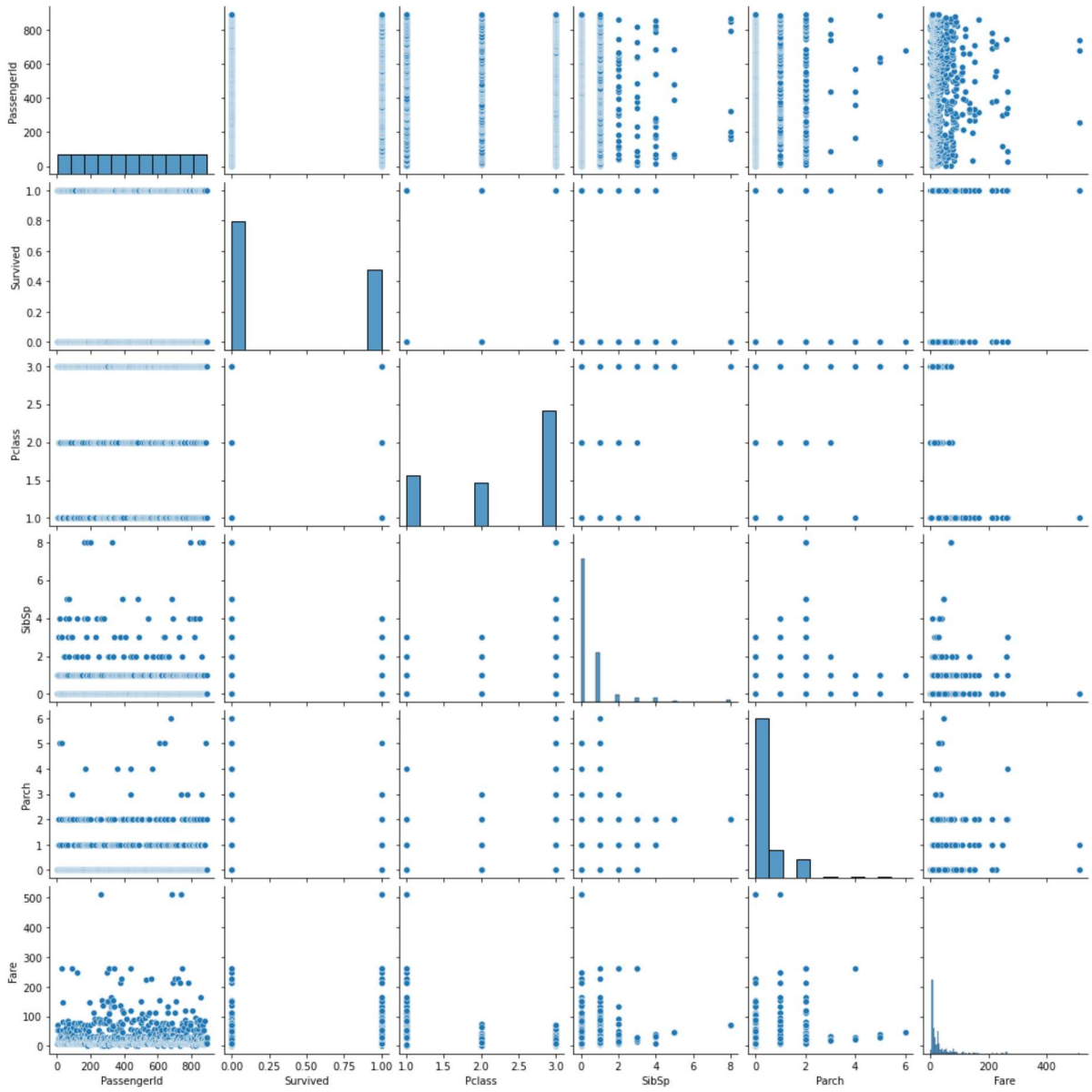
In [54]:
```
sns.heatmap(data=df.isnull())
```

Out[54]: <AxesSubplot:>



In [55]:
```
sns.pairplot(df)
```

Out[55]:
```
<seaborn.axisgrid.PairGrid at 0x7f90a84ff340>
```




In [ ]:
```

```