1.Write a program that accepts the file name as an input from the user and count the number of times the character appears in the file and display only those sentences,which begin with an uppercase alphabet.

```
def letterFrequency(filename, letter):
  with open(filename, "r") as fobj:
    text = fobj.read()
    return text.count(letter)


def sentencesCapital(filename):
  fobj = open(filename, "r")
  text = fobj.read()
  listOfSentences = text.split(".")
  #print(listOfSentences)
  for sentence in listOfSentences:
    sentence = sentence.strip()
    if sentence and sentence[0].isupper():
      print(sentence)
  fobj.close()


fname = input("enter filename: ")
c = input("enter character: ")

print("number of times ", c , " occurs in file: ")
print(letterFrequency(fname, c))

print("sentences starting with uppercase letter in file: ")
sentencesCapital(fname)
```

```
enter filename: abc.txt
enter character: E
number of times  E  occurs in file:
3
sentences starting with uppercase letter in file:
HEY SUP THIS IS DEEKSHA!
```

1b.Write a program to enter the following records in a binary file: Item No integer Item_Name string Qty integer Price float Number of records to be entered should be accepted from the user. Read the file to display the records in the following format: Item No: Item Name : Quantity: Price per item: Amount: (to be calculated as Price * Qty)

```
import pickle
filename = input("binary filename: ")
n = int(input("number of records: "))
with open(filename, "wb+") as fobj2:
  for i in range(0, n):
```

```
    itemDict = { }              #creates new dictionary and dumps
into file on each iteration
    item_no = int(input("Item no.: "))
    item_name = input("Item name: ")
    qty = int(input("Quantity: "))
    price = int(input("Price: "))
    itemDict["item_no"] = item_no
    itemDict["item_name"] = item_name
    itemDict["qty"] = qty
    itemDict["price"] = price
    pickle.dump(itemDict, fobj2)
  fobj2.seek(0)
  for i in range(0, n):
    itemDict = pickle.load(fobj2)
    print("Item no: ", itemDict["item_no"])
    print("Item name: ", itemDict["item_name"])
    print("Qty: ", itemDict["qty"])
    print("price per item: ", itemDict["price"])
    print("amount: ", itemDict["price"]*itemDict["qty"])
    print()
```

```
binary filename: aa
number of records: 3
Item no.: 1
Item name: haha
Quantity: 100
Price: 200
Item no.: 123
Item name: hah
Quantity: 200
Price: 250
Item no.: 234
Item name: aaa
Quantity: 150
Price: 112
Item no:  1
Item name:  haha
Qty:  100
price per item:  200
amount:  20000

Item no:  123
Item name:  hah
Qty:  200
price per item:  250
amount:  50000

Item no:  234
Item name:  aaa
Qty:  150
```

```
price per item:  112
amount:  16800
```

c) Write a program that generates a quiz and uses two files- Questions.txt and Answers.txt. The program opens Questions txt, reads a question, and displays thequestion with options on the screen. The program then opens the Answers.txt file and displays the correct answers.

```
import pickle

qfname = "Questions.txt"
afname = "Answers.txt"

#make questions
questions = [{'qno' : 1 ,
      'qn' : 'what is the sum of 5 and 7?' ,
      'a' : 12 ,
      'b' : 18 ,
      'c' : 10},
            {'qno' : 2 ,
      'qn' : 'What is the capital city of Karnataka?' ,
      'a' : 'Mysore' ,
      'b' : 'Mangaluru' ,
      'c' : 'Bengaluru'}]

answers = [{'qno' : 1, 'ans' : 'a'},
           {'qno' : 2, 'ans' : 'c'}]

with open(qfname, 'wb') as qfile:
  for q in questions:
    pickle.dump(q, qfile)

with open(afname, 'wb') as afile:
  for a in answers:
    pickle.dump(a, afile)

qfile = open(qfname, 'rb')
afile = open(afname, 'rb')

for i in range(0, 2):
  q = pickle.load(qfile)
  print(q['qno'] , '. ' , q['qn'])
  print('a: ', q['a'])
  print('b: ', q['b'])
  print('c: ', q['c'])
  print()

  a = pickle.load(afile)
```

```
  print('answer: ', a['ans'])
  print()
  print()

1 .  what is the sum of 5 and 7?
a:  12
b:  18
c:  10

answer:  a


2 .  What is the capital city of Karnataka?
a:  Mysore
b:  Mangaluru
c:  Bengaluru

answer:  c
```

2a) Write the file mode that will be used for opening the following files. Also, write the Python statements to open the following files: a text file “example.txt” in both read and write mode. a binary file “bfile.dat” in write mode a text file “try.txt” in append and read mode a binary file “btry.dat” in read only mode

```
txtfile1 = open("example.txt", 'w+')
binfile1 = open("bfile.dat", 'wb')
txtfile2 = open("try.txt", 'a+')

binfile2 = open("btry.dat", 'wb')
binfile2 = open("btry.dat", 'rb')
```

2b)Demonstrate object serialization in python by creating a custom class called employee. This stores Employee name, age, salary, married and having kid. Save it and load it up into a separate object and display the new object.

```
import pickle
class employee:
  def __init__(self, name, age, married, kid):
    self.name = name
    self.age = age
    self.married = married
    self.kid = kid
  def showData(self):
    print("name: ", self.name)
    print("age: ", self.age)
    print("married: ", self.married)
    print("kid: ", self.kid)
```

```
e1 = employee("adt", 20, "n", "n")
e2 = employee("kirti", 20, "y", "y")

pickled_data = pickle.dumps(e1)
print("pickled representation: ", pickled_data)

unpickled_data_obj = pickle.loads(pickled_data)
print("unpickled data: ", unpickled_data_obj)

unpickled_data_obj.showData()
```

```
pickled representation:  b'\x80\x04\x95L\x00\x00\x00\x00\x00\x00\x00\
x8c\x08__main__\x94\x8c\x08employee\x94\x93\x94)\x81\x94}\x94(\x8c\
x04name\x94\x8c\x03adt\x94\x8c\x03age\x94K\x14\x8c\x07married\x94\x8c\
x01n\x94\x8c\x03kid\x94h\tub.'
unpickled data:  <__main__.employee object at 0x7f3af125ffd0>
name:  adt
age:  20
married:  n
kid:  n
```

c) Suppose we have created a file with 500 lines of data and the file object reference is “f”. Illustrate what each of these following operations does:

a. F.seek(0,)

b. F.seek(100,1)

c. F.seek(-10,2).

d. F.seek(0,2)

```
import random
fname = "500lines.txt"
f = open(fname , 'w')
slist = ['abc', 'cat', '29352', 'happy']
l = []
for i in range(0, 500):
  l.append(random.choice(slist) + '\n')
f.writelines(l)
f.close()

f = open(fname, 'rb')

f.seek(0,)
print(f.read())
print()

f.seek(100,1)
print(f.read())
```

```
print()

f.seek(-10, 2)
print(f.read())
print()

f.seek(0,2)
print(f.read())
print()

print("f.tell():", f.tell())
```

```
b'abc\n29352\nabc\nabc\nabc\n29352\ncat\ncat\ncat\ncat\nabc\nabc\nabc\
n29352\nhappy\n29352\nhappy\ncat\nhappy\ncat\nhappy\nabc\n29352\nabc\
nabc\nhappy\ncat\nabc\n29352\n29352\nabc\nabc\n29352\nabc\ncat\n29352\
nabc\ncat\nabc\nabc\ncat\nhappy\nhappy\nabc\nhappy\n29352\ncat\n29352\
n29352\n29352\nhappy\nabc\ncat\ncat\ncat\nhappy\nabc\nhappy\nabc\
nhappy\ncat\n29352\nhappy\n29352\nabc\nhappy\nabc\nabc\ncat\n29352\
nhappy\nhappy\ncat\nhappy\n29352\n29352\ncat\nhappy\nhappy\nhappy\
ncat\n29352\n29352\n29352\n29352\nabc\nabc\nhappy\ncat\ncat\nabc\nabc\
n29352\nabc\ncat\nabc\nabc\nhappy\n29352\ncat\nhappy\nhappy\nhappy\
ncat\nhappy\ncat\nhappy\ncat\ncat\nhappy\nabc\ncat\nhappy\ncat\n29352\
n29352\nabc\ncat\nabc\ncat\ncat\n29352\nhappy\n29352\ncat\n29352\
nhappy\nhappy\n29352\nabc\nhappy\n29352\n29352\nhappy\n29352\nabc\
n29352\n29352\nabc\ncat\nhappy\n29352\ncat\nabc\ncat\n29352\ncat\
n29352\nabc\ncat\nabc\nhappy\n29352\nhappy\ncat\ncat\nabc\nhappy\
n29352\ncat\nhappy\ncat\ncat\n29352\nhappy\nabc\nabc\ncat\n29352\nabc\
nhappy\ncat\ncat\nhappy\ncat\nabc\ncat\n29352\ncat\ncat\ncat\ncat\
nhappy\nhappy\nhappy\nhappy\n29352\ncat\ncat\n29352\n29352\nhappy\
nabc\ncat\nabc\n29352\nabc\nhappy\n29352\nhappy\nhappy\nabc\ncat\ncat\
n29352\n29352\nhappy\n29352\n29352\n29352\ncat\nabc\ncat\nabc\n29352\
n29352\ncat\nabc\ncat\nabc\nhappy\n29352\nhappy\nabc\nabc\nabc\n29352\
n29352\nabc\nabc\nhappy\nhappy\ncat\nhappy\nhappy\nabc\nabc\n29352\
n29352\ncat\n29352\nabc\ncat\nabc\n29352\n29352\nabc\nhappy\ncat\nabc\
ncat\nhappy\n29352\nabc\n29352\nhappy\ncat\ncat\nabc\ncat\n29352\nabc\
n29352\ncat\nhappy\ncat\ncat\ncat\ncat\nhappy\nabc\nhappy\nabc\n29352\
n29352\ncat\n29352\ncat\ncat\nhappy\nabc\n29352\nabc\nhappy\n29352\
n29352\n29352\nhappy\nhappy\nabc\nhappy\n29352\nabc\nhappy\ncat\ncat\
nabc\nhappy\nhappy\ncat\ncat\n29352\n29352\n29352\ncat\n29352\nhappy\
nabc\nhappy\ncat\nhappy\ncat\ncat\nabc\ncat\nhappy\nhappy\n29352\
nhappy\n29352\ncat\n29352\nabc\n29352\ncat\n29352\nhappy\nabc\nabc\
nabc\nhappy\ncat\nhappy\nabc\ncat\ncat\nhappy\nhappy\n29352\ncat\
nhappy\n29352\ncat\nabc\nhappy\n29352\ncat\nabc\nhappy\n29352\nhappy\
nabc\ncat\nabc\ncat\nabc\nhappy\nabc\nhappy\ncat\n29352\n29352\ncat\
nhappy\nabc\n29352\ncat\ncat\nabc\nhappy\ncat\nhappy\n29352\nabc\ncat\
nabc\nhappy\n29352\ncat\nhappy\nhappy\n29352\n29352\nabc\ncat\ncat\
nhappy\n29352\ncat\ncat\nhappy\nhappy\nhappy\n29352\nhappy\nabc\
nhappy\nabc\nhappy\n29352\n29352\n29352\ncat\nabc\nhappy\ncat\nhappy\
nhappy\nabc\ncat\ncat\n29352\ncat\nabc\nabc\n29352\ncat\nabc\nhappy\
n29352\nhappy\ncat\n29352\nhappy\n29352\ncat\nabc\nhappy\ncat\nhappy\
```

```
n29352\ncat\nabc\ncat\nhappy\n29352\nabc\n29352\ncat\n29352\n29352\
nabc\n29352\nhappy\ncat\n29352\nabc\ncat\nabc\nabc\nabc\ncat\nhappy\
n29352\nhappy\nabc\n29352\n29352\ncat\nhappy\nhappy\nabc\ncat\nabc\
nabc\nhappy\nabc\ncat\nhappy\nhappy\nhappy\ncat\n29352\nabc\ncat\nabc\
nhappy\nabc\ncat\ncat\nhappy\nhappy\nabc\n29352\ncat\n29352\nabc\
n29352\nabc\ncat\ncat\n29352\n29352\n29352\ncat\nabc\nabc\n29352\nabc\
nhappy\n'

b''

b'abc\nhappy\n'

b''

f.tell(): 2492
```

3 Create an Employee table with attributes such as emp_ssn, emp_name,

---

emp_category, gross_sal, basic_sal. Insert atleast three values in to the database. Demonstrate the database concepts for the following scenario: A company management wants to compute the net salary of each group of employee based on the category of the employee such as Category A, Category B, Category C. Compute the net salary based on the following table.

Category Tax Deducted Dearness Allowance(DA) A 30% of gross salary 80% of basic salary

B 20% of gross salary 50% of basic salary

C 10% of gross salary 30% of basic salary

```
import sqlite3
connection=sqlite3.connect("mydatabase.db")
c=connection.cursor()

c.execute("""create table employee
(
    emp_ssn int,
    emp_name text,
    emp_category text,
    gross_sal float default 'Null',
    basic_sal float
)
""")
c.execute("insert into employee values(1,'aaa','A','Null',10000)")
c.execute("insert into employee values(2,'bbb','B','Null',15000)")
c.execute("insert into employee values(3,'ccc','C','Null',20000)")
```

```
<sqlite3.Cursor at 0x7fa7ec400ab0>
```

```
c.execute("update employee set gross_sal=basic_sal+(0.8*basic_sal)
where emp_category='A' ")
c.execute("update employee set gross_sal=basic_sal+(0.5*basic_sal)
where emp_category='B' ")
c.execute("update employee set gross_sal=basic_sal+(0.3*basic_sal)
where emp_category='C' ")
```

```
<sqlite3.Cursor at 0x7fa7ec400ab0>
```

```
c.execute("alter table employee add taxamt float ")
c.execute("alter table employee add net_sal float ")
c.execute("update employee set taxamt=0.3*gross_sal where
emp_category='A' ")
c.execute("update employee set taxamt=0.2*gross_sal where
emp_category='B' ")
c.execute("update employee set taxamt=0.1*gross_sal where
emp_category='C' ")
```

```
<sqlite3.Cursor at 0x7fa7ec400ab0>
```

```
c.execute("update employee set net_sal=gross_sal-taxamt where
emp_category='A' ")
c.execute("update employee set net_sal=gross_sal-taxamt where
emp_category='B' ")
c.execute("update employee set net_sal=gross_sal-taxamt where
emp_category='C' ")
```

```
<sqlite3.Cursor at 0x7fa7ec400ab0>
```

```
c.execute("select * from employee")
print(c.fetchall())
```

```
[(1, 'aaa', 'A', 18000.0, 10000.0, 5400.0, 12600.0), (2, 'bbb', 'B',
22500.0, 15000.0, 4500.0, 18000.0), (3, 'ccc', 'C', 26000.0, 20000.0,
2600.0, 23400.0)]
```

1. Implement Library management where students can borrow as well as donate books. Books table: id INTEGER PRIMARY KEY name TEXT total_count INTEGER Insert values to the table 34,king,5 123,Harry Potter,3 Update the table based on user inputs: based on book id BORROW RETURN

```
import sqlite3
connection=sqlite3.connect("library2.db")
c=connection.cursor()
c.execute("""create table lib
(
   lib_id int primary key,
   name text,
   t_count int
)
""")
```

```
<sqlite3.Cursor at 0x7f8ca4c2a5e0>
```

```
c.execute("insert into lib values(34,'king',5)")
c.execute("insert into lib values(123,'Harry Potter',3)")
c.execute("select * from lib " )
print(c.fetchall())
```

```
[(34, 'king', 5), (123, 'Harry Potter', 3)]
```

```
id=int(input("enter the id of the book : "))
ch=input("enter b for burrow and r for return ")
if(ch=='b'):
  c.execute("update lib set t_count=t_count-1 where lib_id=? ",(id,))
  if(ch=='r'):
    c.execute("update lib set t_count=t_count+1 where lib_id=?",(id,))
c.execute("select * from lib ")
print(c.fetchall())
```

```
enter the id of the book : 34
enter b for burrow and r for return b
[(34, 'king', 4), (123, 'Harry Potter', 3)]
```

5) NUMPY: a)Create a Numpy array filled with all zeros[1d and 2d] b)Create a Numpy array filled with all ones[1d and 2d] c)Create a 5*4 numpy array which store: [[3 ,6, 9, 12], [15 ,18, 21, 24], [27 ,30, 33, 36], [39 ,42, 45, 48], [51 ,54, 57, 60]] Return array of odd rows and even columns from below numpy array. d)Create a 8x8 matrix and fill it with a checkerboard pattern (alternate 0 & 1) e)Aggregations: Min, Max, and Everything In Between – Write the Python code to print the maximum of 4,12,43.3,19,100 Check whether your able to find the minimum from the given set of values :: 4,12,43.3,19, "HelloProgramming" Write the python code to print the word occurring 1st among these in dict:: "GoodMorning", "Evening", "algorithm", "programming" f)SORTING: Create a list [[4,3,2],[2,1,4]], convert it to a numpy array and sort it along axis 1. Implement a program to take fruits names from array of fruits. To sort the array in alphabetical manner and display their index position.

```
a=np.zeros(5)
b=np.zeros([2,3])
print(a)
print(b)
```

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

```
c=np.ones(5)
d=np.ones([2,3])
print(c)
print(d)
```

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

```
import numpy as np
arr=np.array([[3 ,6, 9, 12], [15 ,18, 21, 24], [27 ,30, 33, 36],
[39,42, 45, 48], [51 ,54, 57, 60]])


print("\n Printing array of odd rows and even columns")
newArray = arr[::2, 1::2]
print(newArray)
```

```
 Printing array of odd rows and even columns
[[ 6 12]
 [30 36]
 [54 60]]
```

```
import numpy as np
arr=np.random.randint(1,size=(8,8))
arr[0::2,0::2]=1
arr[1::2,1::2]=1
print(arr)
```

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

```
array=np.array([4,12,43.3,19,100])
print(array.max())
```

```
100.0
```

```
list=[4,12,43.3,19, 'HelloProgramming']
print(min(list))
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-82-b2d922ef18aa> in <module>
      1 list=[4,12,43.3,19, 'HelloProgramming']
----> 2 print(min(list))

TypeError: '<' not supported between instances of 'str' and 'int'
```

```
a=np.array([ "GoodMorning", "Evening", "Algorithm", "programming"])
print(np.sort(a))
```

```
['Evening' 'GoodMorning' 'algorithm' 'programming']
```

```
list= [[4,3,2],[2,1,4]]
a=np.array(list)
print(np.sort(a,axis=1))
```

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

```
a=np.array(['Mango','Pinneaple','Apple'])
print(np.argsort(a))
```

```
[2 0 1]
```

6)Implement the program to create the data frame from the below data

i. Display only first two rows

ii. Display only last two rows

iii. Extract the py-score of Toronto column.

iv. Display of loc of last row

v. Calculate mean, min, max, standard deviation.

vi. Print the basic stats using describe () method.

```
import pandas as pd
df=pd.DataFrame({
                  'Name':['Xavier','Ann','Jana','Yi','Robin'],
                 'city':['Mexico
City','Toronto','Prague','Shandghai','Manchester'],
                 'age':[41,28,33,34,38],
                 'py-score':[88.0,79.0,81.0,80.0,68.0]})
df.index = pd.RangeIndex(start=101, stop=101+len(df), step=1)
df
```

```
       Name         city  age  py-score
101  Xavier  Mexico City   41      88.0
102     Ann      Toronto   28      79.0
103    Jana       Prague   33      81.0
104      Yi    Shandghai   34      80.0
105   Robin   Manchester   38      68.0
```

```
df.head(2)
```

```
       Name         city  age  py-score
101  Xavier  Mexico City   41      88.0
102     Ann      Toronto   28      79.0
```

```
df.tail(2)
```

```
      Name        city  age  py-score
104     Yi   Shandghai   34      80.0
105  Robin  Manchester   38      68.0
```

```
df['py-score'].iloc[1]
```

```
79.0
```

```
df.loc[df.index[-1]]
```

```
Name             Robin
city        Manchester
age                 38
py-score          68.0
Name: 105, dtype: object
```

```
age=df['age']
print("MEAN OF AGES",age.mean())
print("Maximum age",age.max())
print("Minimum age",age.min())
print("Standard Deviation",age.std())
```

```
MEAN OF AGES 34.8
Maximum age 41
Minimum age 28
Standard Deviation 4.969909455915671
```

```
df.describe()
```

```
             age   py-score
count   5.000000   5.000000
mean   34.800000  79.200000
std     4.969909   7.190271
min    28.000000  68.000000
25%    33.000000  79.000000
50%    34.000000  80.000000
75%    38.000000  81.000000
max    41.000000  88.000000
```

7)Implement a program to create a Data Frame which contains data given below: and obtain output following.

```
import pandas as pd
pd.DataFrame()
```

```
Empty DataFrame
Columns: []
Index: []
```

```
import pandas as pd

df1=pd.DataFrame({ 'Date':['2009-02-11','2009-02-12','2009-02-
13','2009-02-17','2009-02-18'],
```

```
  'High':[30.28,30.28,30.45,29.35,9.35],
   'Low':[29.41,29.32,29.96,28.74,28.56],
  'Close':[29.87,30.24,30.10,28.90,28.92]})
df2=df1.to_string(index=False)
print(df2)
```

```
      Date  High   Low  Close
2009-02-11 30.28 29.41  29.87
2009-02-12 30.28 29.32  30.24
2009-02-13 30.45 29.96  30.10
2009-02-17 29.35 28.74  28.90
2009-02-18  9.35 28.56  28.92
```

```
import numpy as np
data=np.array(['True','False','False','True','False'])
df2=pd.Series(data)
df2.index = pd.RangeIndex(start=1, stop=1+len(df), step=1)
df2
```

```
1     True
2    False
3    False
4     True
5    False
dtype: object
```

```
df1.iloc[[0,3]]
```

```
         Date   High    Low  Close
0  2009-02-11  30.28  29.41  29.87
3  2009-02-17  29.35  28.74  28.90
```