1. using a data set with at least 100 observations (like Kerala.csv but not Kerala.csv itself) obtain some probabilities following the below instructions:
   a) Define at least two events,
   b) Create a contingency table,
   c) Obtain probabilities like P(A), P(B), P(A'), P(B'), P(A∩B), P(A∪B), P(A|B), P(B|A).

```
import numpy as np
import pandas as pd

dataset = pd.read_csv("iris.data")
dataset.describe()
dataset.head(5)

dataset['class'].groupby(dataset['class']).count()
'''
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: class, dtype: int64
'''
# the output showing that we have fair distribution in target
def Mean(column):
    return np.mean(dataset[column])

dataset["sepal_lenght_GT_mean"] = (dataset["sepal_length"] > Mean('sepal_length')).astype("int")
dataset["sepal_width_GT_mean"] = (dataset["sepal_width"] > Mean('sepal_width')).astype("int")

pd.crosstab(dataset["class"],dataset["sepal_lenght_GT_mean"])
'''
sepal_lenght_GT_mean   0   1
class
Iris-setosa           50   0
Iris-versicolor       24  26
Iris-virginica         6  44
'''
pd.crosstab(dataset["class"],dataset["sepal_width_GT_mean"])
'''
sepal_width_GT_mean   0   1
class
Iris-setosa           8  42
Iris-versicolor      42   8
Iris-virginica       33  17
'''
```

```
40.# first event class is 'Iris-virginica' second event sepal_lenght_GT_mean
   is '1'
41.nA = 50
42.nB = 70
43.nS = 150
44.nAandnB = 44
45.
46.pA = nA / nS
47.pB = nB / nS
48.pAcomp = 1 - pA
49.pBcomp = 1 - pB
50.pAcappB = nAandnB / nS
51.pAcumppB = pA + pB - pAcappB
52.pA_given_pB = pAcappB / pB
53.pB_given_pA = pAcappB / pA
54.
55.print('''probability of class is Iris-virginica: {0}\nprobability of
   sepal_lenght_GT_mean: {1}
56.probability of class is Iris-virginica and sepal_lenght_GT_mean: {2}
57.probability of class is Iris-virginica or sepal_lenght_GT_mean: {3}
58.probability of class is not Iris-virginica: {4}\nprobability of
   sepal_lenght_LT_mean: {5}
59.probability of class is Iris-virginica when sepal_lenght_GT_mean is given:
   {6}
60.probability of sepal_lenght_GT_mean when class is Iris-virginica is given:
   {7}'''.format(pA, pB, pAcappB,
61.pAcumppB, pAcomp, pBcomp, pA_given_pB, pB_given_pA))
```

probability of class is Iris-virginica: 0.3333333333333333
probability of sepal_lenght_GT_mean: 0.4666666666666667
probability of class is Iris-virginica and sepal_lenght_GT_mean: 0.29333333333333333
probability of class is Iris-virginica or sepal_lenght_GT_mean: 0.5066666666666667
probability of class is not Iris-virginica: 0.6666666666666667
probability of sepal_lenght_LT_mean: 0.5333333333333333
probability of class is Iris-virginica when sepal_lenght_GT_mean is given:
0.6285714285714286
probability of sepal_lenght_GT_mean when class is Iris-virginica is given: 0.88

2. Construct a **Python document** on the example in which a school purchases 2 computers from a shipment of 20 computers containing 3 defectives and 17 nondefective. Try to obtain the values of the probability mass function, mean, and variance of the number of defective components purchased by the school.
a) In the case you couldn't find out that the random variable has a hypergeometric distribution (with
i) pure Python codes and
ii) using SymPy FiniteRV),
b) In the case you found out that the random variable has a hypergeometric distribution but you don't know that the hypergeometric distribution is already defined in Python
(with pure Python),
c) In the case you found out that the random variable has a hypergeometric distribution and you know that the hypergeometric distribution is already defined in Python (using
i) SymPy Hypergeometric and
ii) SciPy Hypergeometric).

A) i)

```
from math import factorial
#all computers 20, #defective 3, #purchase 2
nS , nD, nR = 20, 3, 2

def nCx(n,x):
    return factorial(n) / (factorial(n-x)*factorial(x))

def pmf(S, D, R):
    prob_l = []
    ND = S - D
    for i in range(R+1):
        up = nCx(D,i)*nCx(ND,R-i)
        prob = up / nCx(S,R)
        prob_l.append(prob)
    return prob_l

def Prob(x, S, D, R):
    return pmf(S, D, R)[x]

def E(x, S, D, R):
    moment = 0
    f = pmf(S, D, R)
    for i in range(len(f)):
        moment += i**x*f[i]
    return moment

def var(S, D, R):
    first_moment = E(1, S, D, R)
```

```
    second_moment = E(2, S, D, R)
    return second_moment - first_moment**2

print('''Probability of value none defectives: {0},
Probability of value one defective: {1},
Probability of value two defectives: {2},
expected value: {3},
variance: {4}'''.format(Prob(0, nS, nD, nR), Prob(1, nS, nD, nR), Prob(2, nS, nD,
nR),
E(1, nS, nD, nR), var(nS, nD, nR)))
```

Probability of value none defectives: 0.7157894736842111,
Probability of value one defective: 0.26842105263157895,
Probability of value two defectives: 0.01578947368421053,
expected value: 0.3,
variance: 0.24157894736842106

A) ii)

```
from sympy.stats import P, E, variance, FiniteRV

PMF = {}
for i in range(len(pmf(nS, nD, nR))):
    PMF[i] = Prob(i, nS, nD, nR)

X = FiniteRV('X',PMF)
print('''Probability of value none defectives: {0},
Probability of value one defective: {1},
Probability of value two defectives: {2},
expected value: {3},
variance: {4}'''.format(P(X<1), P(X<2)-P(X<1), P(X<=2)-P(X<2), E(X),
variance(X)))
```

Probability of value none defectives: 0.715789473684211,
Probability of value one defective: 0.268421052631579,
Probability of value two defectives: 0.0157894736842106,
expected value: 0.300000000000000,
variance: 0.241578947368421

B)

```
from math import factorial
#all computers 20, #defective 3, #purchase 2
nS = 20
nD = 3
```

```
nR = 2
nND = nS - nD

def nCx(n,x):
    return factorial(n) / (factorial(n-x)*factorial(x))

def pmf_Hyper(S, D, R):
    prob_l = []
    ND = S - D
    for i in range(R+1):
        up = nCx(D,i)*nCx(ND,R-i)
        prob = up / nCx(S,R)
        prob_l.append(prob)
    return prob_l

def Prob_Hyper(x, S, D, R):
    return pmf(S, D, R)[x]

def E_Hyper(S, D, R):
    return R*D/S

def var_Hyper(S, D, R):
    st1 = (S-R)/(S-1)
    st2 = R*D/S
    st3 = 1 - D/S
    return st1*st2*st3

print('''Probability of value none defectives: {0},
Probability of value one defective: {1},
Probability of value two defectives: {2},
expected value: {3},
variance: {4}'''.format(Prob_Hyper(0, nS, nD, nR), Prob_Hyper(1, nS, nD, nR),
Prob_Hyper(2, nS, nD, nR),
E_Hyper(nS, nD, nR), var_Hyper(nS, nD, nR)))
```

Probability of value none defectives: 0.7157894736842105,
Probability of value one defective: 0.26842105263157895,
Probability of value two defectives: 0.015789473684210527,
expected value: 0.3,
variance: 0.24157894736842103

C) i)

```
from sympy.stats import Hypergeometric, P, E, variance
```

```
Y = Hypergeometric('Y', nS, nD, nR)
print('''Probability of value none defectives: {0},
Probability of value one defective: {1},
Probability of value two defectives: {2},
expected value: {3},
variance: {4}'''.format(P(Y<1), P(Y<2)-P(Y<1), P(Y<=2)-P(Y<2), E(Y),
variance(Y)))
```

Probability of value none defectives: 68/95,
Probability of value one defective: 51/190,
Probability of value two defectives: 3/190,
expected value: 3/10,
variance: 459/1900

C) ii)

```
from scipy.stats import hypergeom
import numpy as np
Z = hypergeom(nS, nD, nR)
x = np.arange(0, nD)
pmf_scipy = Z.pmf(x)
mean_scipy = hypergeom.mean(nS, nD, nR)
var_scipy = hypergeom.var(nS, nD, nR)
print('''Probability of value none defectives: {0},
Probability of value one defective: {1},
Probability of value two defectives: {2},
expected value: {3},
variance: {4}'''.format(pmf_scipy[0], pmf_scipy[1], pmf_scipy[2], mean_scipy,
var_scipy))
```

Probability of value none defectives: 0.7157894736842111,
Probability of value one defective: 0.26842105263157895,
Probability of value two defectives: 0.01578947368421053,
expected value: 0.3,
variance: 0.24157894736842106

3. Construct a **Python document** for the following example: When a motorist stops at a red light at a certain intersection, the waiting time for the light to turn green (in seconds), is uniformly distributed on the interval (0, 30). Try to obtain some probabilities based on the waiting time of the motorist, mean, and variance of the waiting time.
a) Using SymPy ContinuousRV,

```
from sympy.stats import ContinuousRV, E, variance
from sympy import Symbol, Interval
a, b = 0, 30
X = Symbol('X')
Z = ContinuousRV(X, 1/b-a, set = Interval(a,b))

print('''expected value: {0},
variance: {1}'''.format(E(Z), variance(Z)))
```

expected value: 15.0000000000000,
variance: 75.0000000000000

b) Using SymPy Uniform,

```
from sympy.stats import Uniform
from sympy import Symbol
a, b = 0, 30
Y = Symbol('Y')
Z = Uniform(Y,a,b)
print('''expected value: {0},
variance: {1}'''.format(E(Z), variance(Z)))
```

expected value: 15,
variance: 75

c) Using SciPy Uniform.

```
from scipy.stats import uniform
Z = uniform(0,30)
print('''expected value: {0},
variance: {1}'''.format(Z.mean(), Z.var()))
```

expected value: 15.0,
variance: 75.0