GRADED Questions: (Inferential Statistics)

1. Basics of Probability

A picture containing object, curve

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A picture containing object, curve

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. Discrete Probability Distributions

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. Continuous Probability Distribution

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. Central Limit Theorem

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Coding Graded:

**Generating Binomial Distribution**

**Description**

Generate a binomial distribution, tested 10 times, given the number of trials(n) and probability(p) of each trial.  
The input will contain seed, n and p in the same order.  
The output should contain a numpy array with 10 numbers representing the required binomial distribution.  
Hint: You can use numpy's [random number generator](https://docs.scipy.org/doc/numpy-1.9.2/reference/generated/numpy.random.binomial.html) here too. Remember to set the seed before generating numbers to ensure correct output.  
Sample Input:  
0  
10  
0.5  
Sample Output:

[5 6 5 5 5 6 5 7 8 5]

Solution:

import numpy as np

seed=int(input())

n=int(input())

p=float(input())

#write your code here

np.random.seed(seed)

s = np.random.binomial(n,p,(10,1))

print(s.T[0])

**Description**

A card stack contains white and black cards. Two cards are drawn randomly without replacement. The probability of selecting a white and then a black card is x. The probability of selecting a white card in the first draw is y. You have to find the probability of drawing a black card, given that the first card drawn was white.  
The input will contain two lines with x and y respectively.    
The output should be displayed as a float(no need to round it off).  
Sample Input:  
0.2  
0.5  
Sample Output:  
0.4

Solution:

x=float(input())

y=float(input())

#write your code here

p\_black=x/y

print(p\_black)

**Entrance Test**

**Description**

Consider that the scores of an entrance test are normally distributed.   
If the test has a mean of m and a standard deviation of s.  
You have set the cut off as 90 marks for the test. What is the probability of a student passing the test?  
The input will contain m and s in two lines in the same order. The output has to be rounded off to two decimal places.  
  
Please note that if the answer is 3.6, it is to be reported as 3.6 and not 3.60  
  
Sample Input:  
100  
10  
Sample Output:  
0.84

Solution:

import scipy.stats

m=float(input())#mean

sd=float(input())#standard deviation

p=scipy.stats.norm(m,sd).cdf(90)

students\_passing\_test=1-p

print((round(students\_passing\_test,2)))

**Symmetric triangle**

**Description**

A symmetric triangle of level 5 would look as follows:

1

121

12321

1234321

123454321

As you can see, each row here is a palindrome. Your task is to print an n level symmetric triangle.  
Note: There should be no spaces between the numbers in the triangle.  
  
Sample Input:  
4  
Sample Output:

1

121

12321

1234321

**Solution:**

**n=int(input())**

**count=1**

**for i in range(1,n+1):**

**print(count\*\*2)**

**count=count\*10+1;**

**Seating Arrangement**

**Description**

N people sit around a circular table. You have to find the probability that two particular people won't be sitting together.  
  
The input will have the number N and the output should have the probability printed as a float type number rounded off to four decimal places.  
  
**Hint:** This problem follows the logic of circular permutation which you can read about [here.](https://www.dyclassroom.com/aptitude/circular-permutation)  
  
Please note that if the answer is 3.6, it is to be reported as 3.6 and not 3.6000  
  
Sample Input:  
15  
  
Sample Output:

0.8571

**Solution:**

**import math as m**

**n=int(input())**

**p\_final=round(1-m.factorial(n-2)\*m.factorial(2)/m.factorial(n-1),4)**

**print(p\_final)**

1. **Application Of Sampling Methods**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**