Optional Coding Practice Questions with their Solutions

Question 1:-

Gambling 101
You are participating in a lottery game. A deck of cards numbered from 1-50 is shuffled and 5 cards are drawn out and laid out. You are given a coin. For each card, you toss the coin and pick it up if it says heads, otherwise you don't pick it up. The sum of the cards is what you win.
The lottery ticket costs c rupees. If the expected value of the sum of cards you pick up is less than the lottery ticket, then you buy another ticket otherwise you don't.

Input Format:
The first 5 lines of the input will contain 5 numbers between 1 to 50.
The next line will contain c, the cost of lottery ticket.
Output Format:
Print "Don't buy another" if the expected value is less than the ticket price and print "Buy another one" if the expected value is more than the ticket price.

Sample Input:
1
4
6
17
3
23
Sample Output:
Don't buy another

Sol:-

```
#write your code here
s=0
for i in range(0,5):
    s=s+int(input())/2
if s <= int(input()):
    print("Don't buy another")
else:
    print("Buy another one")
```

Question 2:-

Generating normal distribution
Generate an array of real numbers representing a normal distribution. You will be given the mean and standard deviation as input. You have to generate 10

such numbers.
Hint: You can use numpy's np.random here. You can read more about it here.
To keep the output consistent, you have to set the seed as a specific number which will be given to you as input. Setting a seed means that every time you generate random numbers, they will be the same for the same seed. You can read more about it here.

Input Format:
The input will contain 3 lines which have the seed, mean and standard deviation of the distribution in the same order.
The output will be a numpy array of the generated normal distribution.

Sample Input:
1
0
0.1

Sample Output:
[0.16243454 -0.06117564 -0.05281718 -0.10729686 0.08654076 -0.23015387
  0.17448118 -0.07612069 0.03190391 -0.02493704]
refresh

Sol:-

```
import numpy as np
seed=int(input())
mean=float(input())#mu
std_dev=float(input())#sigma
#write your code here
np.random.seed(seed)
s = np.random.normal(mean, std_dev, 10)#store your result here.
print(s)
```

Question 3:-

Confidence Intervals
For a given column in a dataframe, you have to calculate the 90 percent confidence interval for its mean value. (You can find Z* value for 90 percent confidence from previous segments)
The input will have the column name.
The output should have the confidence interval printed as a tuple.
Note: Do not use the inbuilt function via statmodels.api or any other libraries.

You should write the code on your own to get accurate answers.
The confidence interval values have to be approximated up to two decimal places.
Sample Input:
GRE Score

Sample Output:
(315.87, 317.75)

Sol:-

```
import pandas as pd
import numpy as np
df=pd.read_csv("https://media-doselect.s3.amazonaws.com/generic/
N9LKLvBAx1y14PLoBdL0yRn3/Admission_Predict.csv")
col=input()
#print(df.head())
#write your code here
mean = df[col].mean()
sd = df[col].std()
n = len(df)
Zstar=1.65

se = sd/np.sqrt(n)
lcb = mean - Zstar * se
ucb = mean + Zstar * se
print((round(lcb,2),round(ucb,2)))
```

Question 4:-

College admissions
The probability that a college will accept a student's application is x.
Consider that m students have applied to college. You have to find the probability that at most n students are accepted by the college.
The input will contain three lines with x, m and n respectively.
The output should be rounded off to four decimal places.
Sample Input:
0.3
5
2
Sample Output:
0.8369

Sol:-

```
import scipy.stats as ss
```

```
x=float(input())#number of applicants
m=int(input())#probability of accepting an application
n=int(input())#find the probability that at most n applications are accepted
#write your code here
dist=ss.binom(m,x)
sum=0.0
for i in range(0,n+1):
    sum=sum+dist.pmf(i)
print(round(sum,4))
```

Question 5:-

Tossing a coin
Given that you are tossing a coin n times, you have to find the probability of getting heads at most m times.
The input will have two lines containing n and m respectively.
Sample Input:
10
2
Sample Output:
0.0547

Sol:-

```
import scipy.stats as ss
n=int(input())#number of trials
m=int(input())# find the probability of getting at most m heads
dist=ss.binom(n,0.5)
sum=0.0
for i in range(0,m+1):
    sum=sum+dist.pmf(i)
print(round(sum,4))
```

Question 6:-

Combination Theory
You are given a list of n natural numbers. You select m numbers from the list at random.
Find the probability that at least one of the selected alphabets is "x" where x is a number given to you as input.
The first line of input will contain a list of numbers. The second line will contain m and the third line will contain x.
The output should be printed out as float.
Sample Input:
[1,2,3,4,5,6,6,6,6,7,7,7]

3
6
Sample Output:
0.7454545454545455

Sol:-

```
import ast,sys
from itertools import combinations
input_str = sys.stdin.read()
input_list = ast.literal_eval(input_str)
nums=input_list[0]#the list of numbers
m=int(input_list[1])#m numbers are chosen
x=int(input_list[2])#find probability of getting at least one x

num = 0
den = 0
for c in combinations(nums,m):
    den=den+1
    if x in c:
        num=num+1
print (float(num)/den)
```

Question 7:-

Rolling the dice
A die is rolled n times. You have to find the probability that a number i is rolled at least j times(up to four decimal places)
The input will contain the integers n, i and j in three lines respectively. You can assume that j<n and 0<i<7.
The output should be rounded off to four decimal places.
Sample Input:
4
1
2
Sample Output:
0.1319

Sol:-

```
import scipy.stats as ss
n=int(input())
i=int(input())
j=int(input())

distri = ss.binom(n,1/6)
print(round(1-distri.cdf(j-1),4))
```

Question 8:-

Lego Stack
You are given a row of Lego Blocks consisting of n blocks. All the blocks given have a square base whose side length is known. You need to stack the blocks over each other and create a vertical tower. Block-1 can go over Block-2 only if sideLength(Block-2)>sideLength(Block-1).
From the row of Lego blocks, you on only pick up either the leftmost or rightmost block.
Print "Possible" if it is possible to stack all n cubes this way or else print "Impossible".

Input Format:
The input will contain a list of n integers representing the side length of each block's base in the row starting from the leftmost.

Sample Input:
[5 ,4, 2, 1, 4 ,5]
Sample Output:
Possible

Sol:-

```
import ast,sys
input_str = sys.stdin.read()
sides = ast.literal_eval(input_str)#list of side lengths

n = len(sides)

my_stack = []
i = 0
while(i < n):
    if sides[i] >= sides[n-1]:
        my_stack.append(sides[i])
        i = i+1
    else:
        my_stack.append(sides[n-1])
        n = n - 1
flag = 0
i = 1
while i < len(my_stack):
    if(my_stack[i] > my_stack[i - 1]):
        flag = 1
    i += 1

if (not flag) :
```

```
    print ("Possible")
else :
    print ("Impossible")
```