# Stochastic conditions inside loops

## Random numbers and seeds

Python and other programming languages generate random numbers using a certain algorithm that produces a sequence with characteristics of actual random numbers. However, the sequence that the algorithm produces is not really random, since it is completely determined by its initial value, the seed. When you use a random function in your code, usually python just takes some seed, for example based on time, so that the random numbers are different every time you run the program. However, you can also define the seed yourself so that the random numbers will be the same every time you run the program. Using a seed can be useful, for example, to compare results with others.

## Exercise: Rolling dice

import numpy.random as rd

rd.seed(141)

for n in range(30):

r = rd.randint(1,7)

print(r)

This simulates the rolling of a dice 30 times. As you can see, a seed has been set, so that the program should generate the same random numbers every time. In this case, the last few random numbers should be 4,5,4,1,3,6,3,6,3. If you changed the seed from 141 to some other number, the program would generate another sequence of random numbers (and the answer would change).

Extend the program to print the number of times a 6 was obtained.

What number does the program print?

Answer: 5

## Exercise: Rolling dice repeated

Suppose you're playing a dice game with a friend of yours and, after rolling a dice 30 times, he has 10 sixes. Is he just lucky or is he cheating? In order to find out, you decide to change the program you just made to simulate the game 10 times. The program should now print 10 numbers: the number of sixes for each of the simulated games.

How many times does the program print a number that is 10 or higher?

Answer: 1

What is, estimated based on this simulation, the probability that your friend is honest?

Answer: 0.1, or 10%

## Exercise: Coins

Consider the following program.

import numpy.random as rd

rd.seed(15)

for n in range(1000):

r = rd.randint(1,3)

print(r)

Let's say 1 stands for head and 2 stands for tail. Then this simulates the tossing of a coin 1000 times.The last few random numbers should be 2212122121

Extend this program to output the length of the longest consecutive sequence of the same number (e.g. if 111111 then output 6, or if 2222222 then output 7). How long is the longest found sequence?

Answer: 13

You may wish to try the warm-up exercises below before attempting this one.

### Warmup: Consecutive sequences

In order to do the exercise with the coins, two things need to be done: counting the length of consecutive sequences and finding a maximum number. In these warm-up questions, these two steps are considered separately.

Let's start with the consecutive sequence.

If we have the following code:

import numpy.random as rd

rd.seed(5)

for n in range(6):

r = rd.randint(1,3)

print(r)

The printed output will be:

2

1

2

2

1

1

Change the code above, so that it prints the number of consecutive ones instead. The output should thus be:

0

1

0

0

1

2

Now change the code so that it also prints the sum of the printed number of consecutive ones. For the example above, it should thus print 4. Once this works, change range(6) to range(500).

What is the sum of the numbers of consecutive ones in this case?

Answer: 481

You cannot easily check all 1000 output numbers. Therefore, in order to check your program yourself, it may be practical to first run the loop 100 times instead of 1000 times. Testing a simpler situation first is a commonly used programming strategy.

### Warmup: Maximum

import numpy.random as rd

rd.seed(5)

for n in range(6):

r = rd.randint(1,5000)

print(r)

This code prints:

2916

2255

4080

1726

3047

2122

Change the code, so that it prints the maximum number that was generated. In this case that would be 4080. If you don't know how to approach this, you can read the hint below.

Change range(6) to range(500). What is the maximum number in this case?

Answer: 4997