Introduction to Functions, Counting in Numpy, and Accelerating Simulations

1. Introduction to Functions

Let's consider the coin flip experiment with the sample space:

In [1]:
```
import random
faces=['H','T']
```

Consider the simulation from last time for determining the relative frequency of getting 6 or fewer heads on 20 flips of a fair coin:

In [2]:
```
num_sims=1000000
flips=20

event_count=0
for sim in range(num_sims):
    coins=random.choices(faces, k=flips)
    num_heads=coins.count('H')
    if num_heads <= 6:
        event_count+=1

print("Relative frequency of 6 or fewer heads is ", event_count/num_sims)
```

```
Relative frequency of 6 or fewer heads is  0.057838
```

Let's consider how to further improve this code.

We begin by turning this simulation into a **function**.

- New functions in Python are defined using the `def` keyword, followed by the function name, the arguments in parentheses, and then a colon. The commands to be run in the function follow in an indented block.

Note that it is helpful to know how to indent a whole block of code in Jupyter. Choose the Help->Keyboard Shortcuts menu and then look under the Edit Mode section for the Indent command. For instance, on the Mac, it is Command-]. When you want to turn a code block into a function, copy and paste it into a new cell and then indent it using the keyboard command. Then add the `def` statement above it.

- It is easiest to understand through and example:

In [3]:
```
def coinsim(num_sims=1000000, flips=20, threshold=6):

    event_count=0
    for sim in range(num_sims):
        coins=random.choices(faces, k=flips)
        num_heads=coins.count('H')
        if num_heads <= threshold:
            event_count+=1

    print("Relative frequency of ",threshold," or fewer heads is ", event_count/num_sim
```

Now we can call the function by its name followed by parentheses. Since we have provided **default values** for all of the function's arguments, we do not have to even provide any arguments:

```
In [4]: coinsim()
```

```
Relative frequency of  6  or fewer heads is  0.05788
```

We can pass arguments to the function according to their **position**, **keyword**, or both. For instance, to only run 100k simulations, we can do either of the following:

```
In [5]: coinsim(100000)
```

```
Relative frequency of  6  or fewer heads is  0.05819
```

```
In [6]: coinsim(num_sims=100000)
```

```
Relative frequency of  6  or fewer heads is  0.05765
```

Keyword arguments can appear in any order and can appear after positional arguments:

```
In [7]: coinsim(100000, threshold=4, flips=16)
```

```
Relative frequency of  4  or fewer heads is  0.03908
```

However, positional arguments cannot follow keyword arguments:

```
In [8]: coinsim(100000, flips=16, 4)
```

```
  File "<ipython-input-8-5b5963aef832>", line 1
    coinsim(100000, flips=16, 4)
                             ^
SyntaxError: positional argument follows keyword argument
```

Now let's see how long it takes to run this function. We will use Jupyter's built-in `%timeit` magic:

```
In [9]: %timeit coinsim()
```

```
Relative frequency of  6  or fewer heads is  0.0579
Relative frequency of  6  or fewer heads is  0.057468
Relative frequency of  6  or fewer heads is  0.057976
Relative frequency of  6  or fewer heads is  0.057804
Relative frequency of  6  or fewer heads is  0.057365
Relative frequency of  6  or fewer heads is  0.057753
Relative frequency of  6  or fewer heads is  0.057691
Relative frequency of  6  or fewer heads is  0.057894
4.39 s ± 24.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

If you have programmed in Matlab, you may have heard to avoid `for` loops because they slow everything down. The same is true in Python. Instead, we replace the lists with 2-D arrays, where one dimension is for the different dice, and the other dimension is for the different experiments.

Since we are creating an *array* of values, we will be generating 1s and 0s instead of 'H's and 'T's. We will use the `numpy.random` module. It will be convenient to import it as `npr`. We will also use other parts of `numpy`, so we will import it as `np`, as usual:

```
In [10]:
import numpy.random as npr
import numpy as np
```

Let's start by simulating flipping a fair coin 20 times again. Here we just randomly choose 20 random values that are equally likely to be 0 (representing tails) or 1 (representing heads). We use the `randint()` method:

```
In [11]:
npr.randint(2, size=20)
```

```
Out[11]:
array([1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1])
```

Now, we can generate multiple rows like this by changing the size to a tuple. The tuple is interpreted as (rows, columns), so to conduct 5 simulations, we can do:

```
In [16]:
results = npr.randint(2, size=(5,20))
results
```

```
Out[16]:
array([[1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0],
       [1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1],
       [0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0],
       [1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1],
       [0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1]])
```

Next, we need to learn how to translate the simulated coin flips into the counts of the number of heads. We can do that by summing across the columns. The rows are dimension 0, and the columns are dimension 1. We can use numpy's sum method to carry out the sum over the columns as follows:

```
In [17]:
np.sum(results)
```

```
Out[17]:
49
```

```
In [19]:
num_heads = np.sum(results, axis=1)
num_heads
```

```
Out[19]:
array([10,  8,  7, 11, 13])
```

We can perform comparisons on numpy arrays, and it will compare every element:

```
In [20]:
num_heads<10
```

```
Out[20]:
array([False,  True,  True, False, False])
```

If we sum over an array of True/False values, it will treat True as 1 and False as 0. Thus, we can count how many items satisfy some condition easily:

In [22]:
```
np.sum(num_heads<10)
```
Out[22]:
```
2
```

Now we are ready to put all of that into practice. Let's make a new function using these principles:

In [23]:
```
def coinsim2(num_sims=1000000, flips=20, threshold=6):

    results = npr.randint(2,size=(num_sims,flips))
    num_heads=np.sum(results, axis=1)
    event_count = np.sum(num_heads<=threshold)

    print("Relative frequency of ",threshold," or fewer heads is ", event_count/num_sim
```

In [24]:
```
coinsim2()
```
```
Relative frequency of  6  or fewer heads is  0.05766
```

In [25]:
```
%timeit coinsim2()
```
```
Relative frequency of  6  or fewer heads is  0.057475
Relative frequency of  6  or fewer heads is  0.057602
Relative frequency of  6  or fewer heads is  0.057811
Relative frequency of  6  or fewer heads is  0.057482
Relative frequency of  6  or fewer heads is  0.057446
Relative frequency of  6  or fewer heads is  0.058163
Relative frequency of  6  or fewer heads is  0.05793
Relative frequency of  6  or fewer heads is  0.057349
Relative frequency of  6  or fewer heads is  0.057563
Relative frequency of  6  or fewer heads is  0.057738
Relative frequency of  6  or fewer heads is  0.057716
Relative frequency of  6  or fewer heads is  0.057751
Relative frequency of  6  or fewer heads is  0.057729
Relative frequency of  6  or fewer heads is  0.057635
Relative frequency of  6  or fewer heads is  0.057435
Relative frequency of  6  or fewer heads is  0.057227
Relative frequency of  6  or fewer heads is  0.057901
Relative frequency of  6  or fewer heads is  0.057381
Relative frequency of  6  or fewer heads is  0.057759
Relative frequency of  6  or fewer heads is  0.057698
Relative frequency of  6  or fewer heads is  0.058468
Relative frequency of  6  or fewer heads is  0.057544
Relative frequency of  6  or fewer heads is  0.057478
Relative frequency of  6  or fewer heads is  0.057326
Relative frequency of  6  or fewer heads is  0.057518
Relative frequency of  6  or fewer heads is  0.058026
Relative frequency of  6  or fewer heads is  0.057506
Relative frequency of  6  or fewer heads is  0.057239
Relative frequency of  6  or fewer heads is  0.057471
Relative frequency of  6  or fewer heads is  0.057574
Relative frequency of  6  or fewer heads is  0.057987
Relative frequency of  6  or fewer heads is  0.058012
Relative frequency of  6  or fewer heads is  0.057419
Relative frequency of  6  or fewer heads is  0.058007
Relative frequency of  6  or fewer heads is  0.057637
Relative frequency of  6  or fewer heads is  0.057575
Relative frequency of  6  or fewer heads is  0.057763
Relative frequency of  6  or fewer heads is  0.057739
Relative frequency of  6  or fewer heads is  0.057712
Relative frequency of  6  or fewer heads is  0.057867
Relative frequency of  6  or fewer heads is  0.05785
Relative frequency of  6  or fewer heads is  0.057562
Relative frequency of  6  or fewer heads is  0.057431
Relative frequency of  6  or fewer heads is  0.057202
Relative frequency of  6  or fewer heads is  0.057839
Relative frequency of  6  or fewer heads is  0.057915
Relative frequency of  6  or fewer heads is  0.057521
Relative frequency of  6  or fewer heads is  0.057758
Relative frequency of  6  or fewer heads is  0.057308
Relative frequency of  6  or fewer heads is  0.057489
Relative frequency of  6  or fewer heads is  0.057447
Relative frequency of  6  or fewer heads is  0.057917
Relative frequency of  6  or fewer heads is  0.057526
Relative frequency of  6  or fewer heads is  0.057905
Relative frequency of  6  or fewer heads is  0.057375
Relative frequency of  6  or fewer heads is  0.057544
Relative frequency of  6  or fewer heads is  0.057685
Relative frequency of  6  or fewer heads is  0.05753
Relative frequency of  6  or fewer heads is  0.057185
Relative frequency of  6  or fewer heads is  0.057356
Relative frequency of  6  or fewer heads is  0.057834
Relative frequency of  6  or fewer heads is  0.057367
Relative frequency of  6  or fewer heads is  0.057994
Relative frequency of  6  or fewer heads is  0.057425
Relative frequency of  6  or fewer heads is  0.057788
Relative frequency of  6  or fewer heads is  0.057311
Relative frequency of  6  or fewer heads is  0.057199
Relative frequency of  6  or fewer heads is  0.057516
Relative frequency of  6  or fewer heads is  0.057302
Relative frequency of  6  or fewer heads is  0.057493
Relative frequency of  6  or fewer heads is  0.0575
Relative frequency of  6  or fewer heads is  0.057488
Relative frequency of  6  or fewer heads is  0.058016
Relative frequency of  6  or fewer heads is  0.057826
Relative frequency of  6  or fewer heads is  0.057398
Relative frequency of  6  or fewer heads is  0.057757
Relative frequency of  6  or fewer heads is  0.057615
Relative frequency of  6  or fewer heads is  0.057442
Relative frequency of  6  or fewer heads is  0.057747
Relative frequency of  6  or fewer heads is  0.057404
Relative frequency of  6  or fewer heads is  0.057543
98.9 ms ± 647 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

That is about a 45 times speed up!

```
In [ ]:
```