Binomial Distributions

Use NumPy to create simulations and compute proportions for the following outcomes. The first one is done for you.

In [2]:

```
# import numpy
import numpy as np
```

1. A fair coin flip produces heads

In [2]:

```
# simulate 1 million tests of one fair coin flip
# remember, the output of these tests are the # successes, or # heads
tests = np.random.binomial(1, 0.5, int(1e6))

# proportion of tests that produced heads
(tests == 1).mean()
```

Out[2]:

```
0.50007999999999997
```

2. Five fair coin flips produce exactly one head

In [3]:

```
# simulate 1 million tests of five fair coin flips
tests = np.random.binomial(5, 0.5, int(1e6))

# proportion of tests that produced 1 head
(tests == 1).mean()
```

Out[3]:

```
0.15629599999999999
```

3. Ten fair coin flips produce exactly four heads

In [4]:

```
# simulate 1 million tests of ten fair coin flips
tests = np.random.binomial(10, 0.5, int(1e6))

# proportion of tests that produced 4 heads
(tests == 4).mean()
```

Out[4]:

```
0.204512
```

4. Five bias coin flips with P(H) = 0.8 produce exactly five heads

```
In [5]: # simulate 1 million tests of five bias coin flips
        tests = np.random.binomial(5, 0.8, int(1e6))

        # proportion of tests that produced 5 heads
        (tests == 5).mean()
```

```
Out[5]: 0.32767099999999999
```

5. Ten bias coin flips with P(H) = 0.15 produce at least 3 heads

```
In [3]: # simulate 1 million tests of ten bias coin flips
        tests = np.random.binomial(10, 0.15, int(1e6))

        # proportion of tests that produced at least 3 heads
        (tests >= 3).mean()
```

```
Out[3]: 0.18032400000000001
```

```
In [ ]:
```