```
import numpy as np

# outcome of one coin flip
np.random.randint(0,2)

# outcomes of ten thousand coin flips
np.random.randint(0,2, size=10000)

# mean outcome of ten thousand coin flips
np.random.randint(0,2, size=10000).mean()

# outcome of ten thousand coin flips
np.random.choice([0, 1], size=10000)

# mean outcome of ten thousand coin flips
np.random.choice([0, 1], size=10000).mean()

# mean outcome of ten thousand biased coin flips
np.random.choice([0, 1], size=10000, p=[0.8, 0.2]).mean()
```

Quiz

```
# simulate 1 million tests of three fair coin flips
tests = np.random.randint(2, size=(int(1e6), 3))

# sums of all tests
test_sums =  tests.sum(axis=1)

# proportion of tests that produced exactly one head
(test_sums == 2).mean()

# simulate 1 million tests of three biased coin flips
# hint: use np.random.choice()

tests = np.random.choice([0, 1], size=(int(1e6), 3), p=[0.6, 0.4])


# sums of all tests

test_sums = tests.sum(axis=1)


# proportion of tests that produced exactly one head

(test_sums == 2).mean()


# simulate the first million die rolls

first = np.random.choice(np.arange(6), size=int(1e6))


# simulate the second million die rolls

second = np.random.choice(np.arange(6), size=int(1e6))


# proportion of tests where the 1st and 2nd die rolled the same number

(first == second).mean()
```

```
# number of heads from 10 fair coin flips

np.random.binomial(10, 0.5)


# results from 20 tests with 10 coin flips

np.random.binomial(10, 0.5, 20)


# mean number of heads from the 20 tests

np.random.binomial(10, 0.5, 20).mean()


import matplotlib.pyplot as plt

% matplotlib inline

# hist
```

```
plt.hist(np.random.binomial(10, 0.5, 1000000));
```

Quiz

```
# simulate 1 million tests of ten biased coin flips
tests = np.random.binomial(10, 0.15, int(1e6))

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

```
import pandas as pd
df = pd.read_csv('cancer_test_data.csv')

# proportion of patients with cancer
df.has_cancer.mean()

# proportion of patients with cancer who test negative
(df.query('has_cancer')['test_result'] == 'Negative').mean()

# proportion of patients without cancer who test positive
(df.query('has_cancer == False')['test_result'] == 'Positive').mean()

# What proportion of patients who tested positive has cancer?
df.query('test_result == \"Positive\"')['has_cancer'].mean()

# What proportion of patients who tested positive doesn't have cancer?
1 - df.query('test_result == \"Positive\"')['has_cancer'].mean()
```