Try-It Activity 2.1: Histograms in Python

IMPORTANT INSTRUCTIONS: This activity is designed for you to experiment with `Python` code that generates normally distributed data, histograms, and continuous distribution functions. Feel free to change any numerical value throughout the code in the activity to visualize different outcomes and results.

What is a Histogram?

A histogram is a display of statistical information that uses rectangles to show the frequency of data items in successive numerical intervals of equal size.

The `Python` plotting library `Matplotlib` provides the *`plt.hist()`* *function* for creating histogram *plots*.

Run the code cell below to import the necessary libraries for this activity.

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

Next, we need to define some data so we can generate our first histogram.

For this example, let's suppose we have available to us the weight measurements of 250 people (in lbs). Suppose further that the mean is around 170 and the data has a standard deviation equal to 10.

In the code cell below, fill in the ellipsis to generate the data described above.

In [2]:
```python
x = np.random.normal(170, 10, 250)
```

Note that above, we have used the *function* *`normal()`* from the `NumPy random()` *module* to generate our sample.

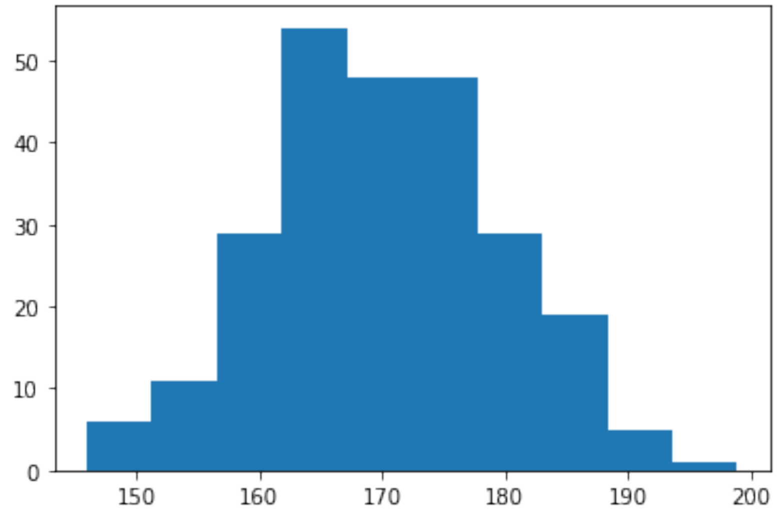
Next, let's try to produce our first *plot*.

As mentioned above, to accomplish this, we can use the *function* *`plt.hist()`*.

In the simplest case, this *function* only takes, as a parameter, the data we want to *plot*.

In the code cell below, plot the data generated above.

In [3]:
```
plt.hist(x)
plt.show()
```



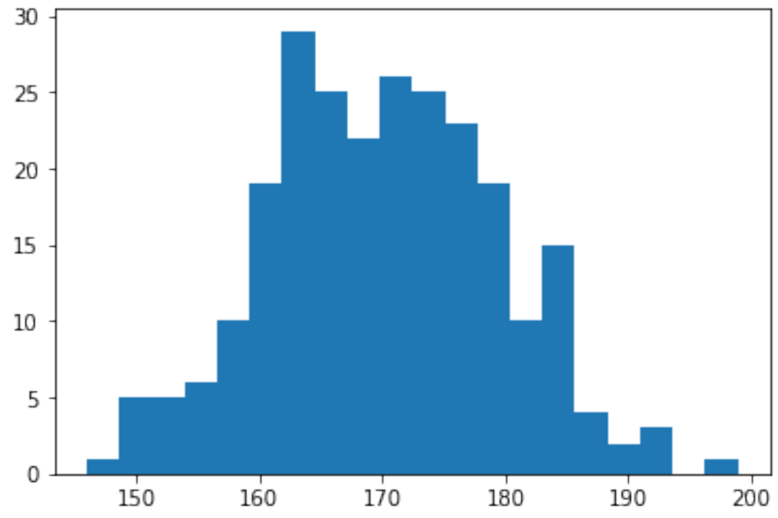
As you already know, a histogram can be interpreted as a *plot* that displays the frequency of each outcome.

Mathematically, a histogram is a mapping of bins (intervals) to frequencies. Each bin is plotted as a bar whose height corresponds to how many data points are in that bin.

Changing the number of bins changes the appearance of your *plot*. This parameter can be regulated by setting the argument `bins = n`, where n is the desired of number of bins, inside the *function* `hist()` .

In the code cell below, plot the data x by setting the number of bins equal to 20.

In [4]:
```
plt.hist(x, bins = 20)
plt.show()
```



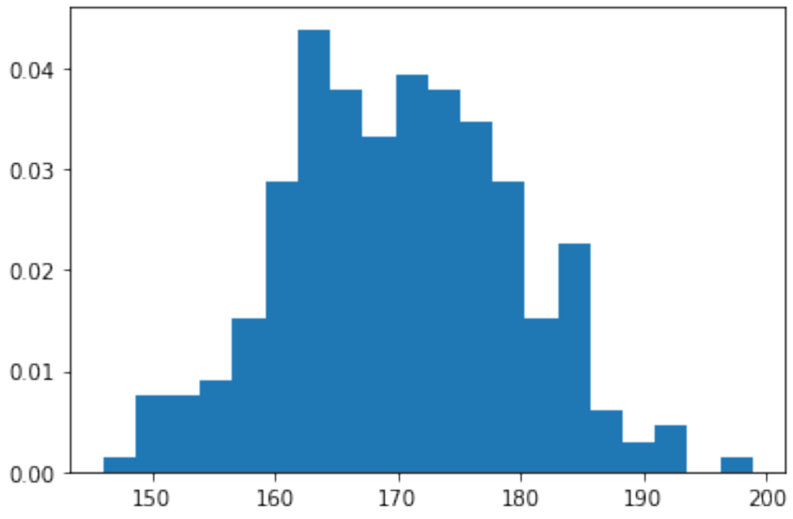
Feel free to change the number of bins above to any value you like to see how the distribution changes.

Visualizing the frequency of the available values using a histogram can be very useful. However, sometimes we might be interested in visualizing the probability of each outcome.

This can be accomplished by setting the argument `density` to true inside the *function* `hist()` .

In the code cell below, try to visualize the frequency of each outcome as a probability value.

In [5]:
```
plt.hist(x, bins = 20, density = True)
plt.show()
```



Continuous Probability Distribution and Probability Density Functions

A continuous distribution describes the probabilities of the possible values of a continuous random variable. A continuous random variable is a random variable with an infinite set of possible values.

A probability density describes the relationship between the continuous random variable observations and their probability.

The overall shape of the probability density is referred to as a probability distribution, and the calculation of probabilities for specific outcomes of a random variable is performed by a probability density function, or PDF for short.

Measuring the Height of the Rocky Mountains

Let's try to understand how to compute the probability density function of a continuous distribution with an example.

Suppose you are measuring the height of the Rocky Mountains. You find that the shortest mountain is 1,734 m tall and that the tallest mountain is 4,356 m tall. Since the mountain range is extensive, you want to find the probability that any random mountain will be shorter than 3,000 m.

Interestingly enough, you also find that the distribution of the mountain heights follows a normal distribution with a mean of 3,062 and a standard deviation of 500, as shown below:

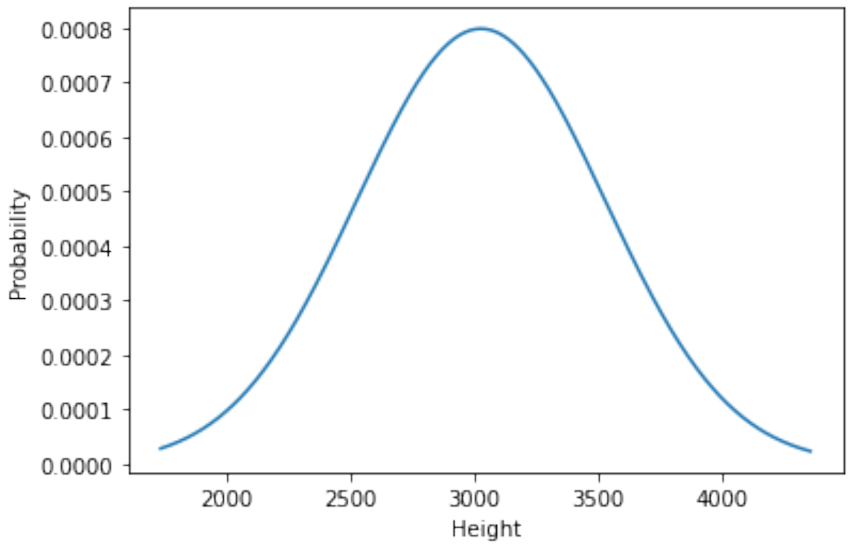
In the code cell below, fill in the ellipsis to generate a `NumPy` *array* with values between 1,734 and 4,356 and step 1.

In [6]:
```
# Plot between 1734 and 4356m with step 1.

x_axis = np.arange(1734, 4356, 1)
```

In the code cell below, we have given you the code to plot the values above.

The *function* `norm.pdf()` from `SciPy` helps us to visualize the PDF of our data.

```
In [7]: from scipy.stats import norm

        #plotting our distribution
        plt.plot(x_axis, norm.pdf(x_axis,3026,500))
        plt.xlabel("Height")
        plt.ylabel("Probability")
        plt.show()
```



Note that above, we have assumed that our measurements have a mean equal to 3,026 and a standard deviation equal to 500.

In the code below, fill in the ellipsis so that the variable H is a normal random variable with mean and standard deviation equal to the values above.

```
In [8]: H = norm(3026,500)
```

Cumulative Distribution Function

The cumulative distribution function (CDF) of a random variable is the probability that the random variable will take a value less than or equal to x.

In `Python` , if we want to compute the probability that a mountain is shorter than 3,000 m, we can use the function `cdf()` on our random variable.

Observe the code in the cell below.

```
In [9]: #computing the the probability density for P(<3000)
        print(H.cdf(3000))

        0.47926434670771967
```

It appears that the probability that a mountain is less than 3,000 m is around 0.47. In other words, 47% of mountains would be shorter that 3,000 m tall.

Change the code cell above to compute the probability that a mountain is less than 3,298 m.

To challenge yourself further, consider another data calculation. How would you change the code above to compute the probability that a mountain is taller than 3,000 m?

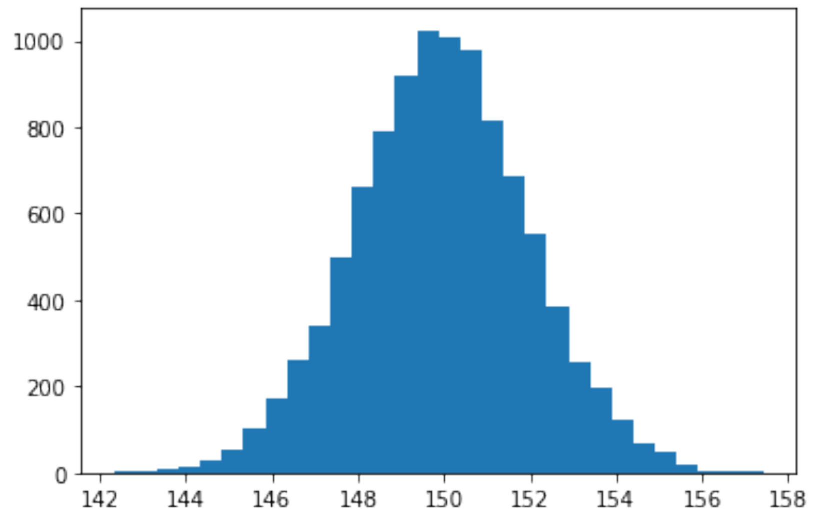
Distribution of Women's Weight

In our final example, suppose we have the following data available that represents the weights of women.

```
In [10]: w = np.random.normal(150, 2, 10000)
```

In the code cell below, fill in the ellipsis to plot the data `w` as a histogram.

```
In [11]: plt.hist(w, bins = 30)
         plt.show()
```



The shape of the distribution above is really similar to the shape of a very common continuous distribution, the normal distribution.

In fact, we can fit a normal distribution on top of the one we have just obtained.

In the code cell below, fill in the ellipsis to generate a normal random variable `W` with a mean of 150 and a standard deviation of 2.

```
In [12]: #generating the normal PDF
         W = norm(150, 2)
```

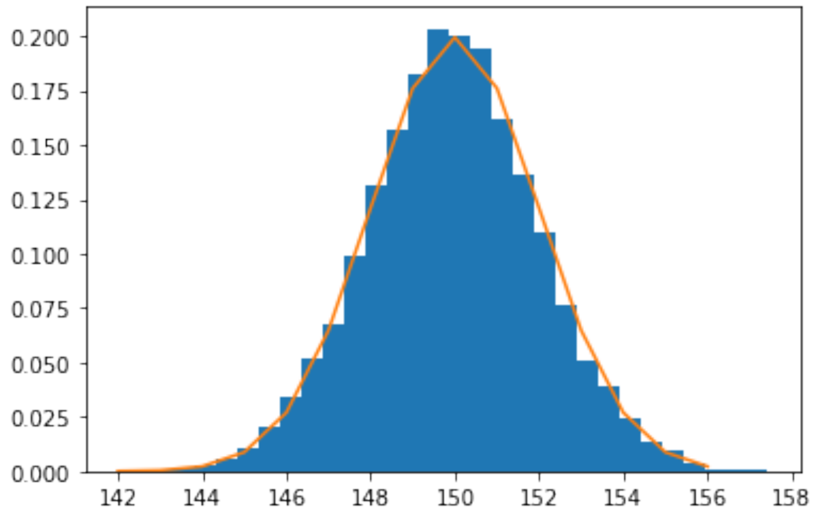
Next, we choose the range on which we'll plot the PDF and the probabilities for that range.

In [13]:
```
#lower bound
min_weight = np.min(w)
#upper bound
max_weight = np.max(w)
#generate a sequence of equally spaces values
values = list(range(int(min_weight), int(max_weight)))
#generate probabilities for each value (these will follow the shape of
probabilities = [W.pdf(v) for v in values]
```

Finally, we plot the sample distribution and the PDF we have generated.

In [14]:
```
#plotting weight distribution
plt.hist(w, bins=30, density = True)
#plotting normal distribution (orange line)
plt.plot(values, probabilities)
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x12a9082b0>]
```



In [ ]:
```
```