Set and Booleans

There are two other object types in Python that we should quickly cover: Sets and Booleans.

Sets

Sets are an unordered collection of *unique* elements. We can construct them by using the set() function. Let's go ahead and make a set to see how it works

```
In [1]:  x = set()
```

```
In [2]:  # We add to sets with the add() method
         x.add(1)
```

```
In [3]:  #Show
         x
```

```
Out[3]:  {1}
```

Note the curly brackets. This does not indicate a dictionary! Although you can draw analogies as a set being a dictionary with only keys.

We know that a set has only unique entries. So what happens when we try to add something that is already in a set?

```
In [4]:  # Add a different element
         x.add(2)
```

```
In [5]:  #Show
         x
```

```
Out[5]:  {1, 2}
```

```
In [6]:  # Try to add the same element
         x.add(1)
```

```
In [7]:  #Show
         x
```

```
Out[7]:  {1, 2}
```

Notice how it won't place another 1 there. That's because a set is only concerned with unique elements! We can cast a list with multiple repeat elements to a set to get the unique elements. For example:

```
In [8]:  # Create a list with repeats
```

```
list1 = [1,1,2,2,3,4,5,6,1,1]
```

In [9]:
```
# Cast as set to get unique values
set(list1)
```

Out[9]: {1, 2, 3, 4, 5, 6}

Booleans

Python comes with Booleans (with predefined True and False displays that are basically just the integers 1 and 0). It also has a placeholder object called None. Let's walk through a few quick examples of Booleans (we will dive deeper into them later in this course).

In [10]:
```
# Set object to be a boolean
a = True
```

In [11]:
```
#Show
a
```

Out[11]: True

We can also use comparison operators to create booleans. We will go over all the comparison operators later on in the course.

In [12]:
```
# Output is boolean
1 > 2
```

Out[12]: False

We can use None as a placeholder for an object that we don't want to reassign yet:

In [13]:
```
# None placeholder
b = None
```

In [14]:
```
# Show
print(b)
```

None

Thats it! You should now have a basic understanding of Python objects and data structure types. Next, go ahead and do the assessment test!