Sets and Booleans

March 17, 2017

1 Set and Booleans

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

1.1 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 [3]: # We add to sets with the add() method
        x.add(1)

In [4]: #Show
        x

Out[4]: {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 [5]: # Add a different element
        x.add(2)

In [6]: #Show
        x

Out[6]: {1, 2}

In [7]: # Try to add the same element
        x.add(1)

In [9]: #Show
        x

Out[9]: {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 [10]: # Create a list with repeats
         l = [1,1,2,2,3,4,5,6,1,1]

In [12]: # Cast as set to get unique values
         set(l)

Out[12]: {1, 2, 3, 4, 5, 6}
```

1.2 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 [13]: # Set object to be a boolean
         a = True

In [16]: #Show
         a

Out[16]: True
```

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

```
In [17]: # Output is boolean
         1 > 2

Out[17]: False
```

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

```
In [18]: # None placeholder
         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!

```
In [ ]:
```