07-Sets and Booleans

October 28, 2020

 Coursework delivered by: Alison Mukoma

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

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

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

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

[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?

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

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

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

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

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

[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:

[8]:
```
# Create a list with repeats
list1 = [1,1,2,2,3,4,5,6,1,1]
```

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

[9]:
```
{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).

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

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

[11]:
```
True
```

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

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

[12]:
```
False
```

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

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

[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!