PY0101EN-2-3-Sets

May 24, 2021

Sets in Python

Welcome! This notebook will teach you about the sets in the Python Programming Language. By the end of this lab, you'll know the basics set operations in Python, including what it is, operations and logic operations.

```
<a href="https://cocl.us/NotebooksPython101">
     <img src="https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass
</a>
```

Table of Contents

```
<ul>
    <li>
        <a href="#set">Sets</a>
        <ul>
            <li><a href="content">Set Content</a></li>
            <li><a href="op">Set Operations</a></li>
            <li><a href="logic">Sets Logic Operations</a></li>
        </ul>
    </li>
    <li>
        <a href="#quiz">Quiz on Sets</a>
    </li>
</ul>
<p>
    Estimated time needed: <strong>20 min</strong>
</p>
```

Sets

Set Content

A set is a unique collection of objects in Python. You can denote a set with a curly bracket {}. Python will automatically remove duplicate items:

[ ]:
```
# Create a set

set1 = {"pop", "rock", "soul", "hard rock", "rock", "R&B", "rock", "disco"}
set1
```

The process of mapping is illustrated in the figure:

You can also create a set from a list as follows:

[1]:
```
# Convert list to set

album_list = [ "Michael Jackson", "Thriller", 1982, "00:42:19", \
               "Pop, Rock, R&B", 46.0, 65, "30-Nov-82", None, 10.0]
album_set = set(album_list)
album_set
```

[1]:
```
{'00:42:19',
 10.0,
 1982,
 '30-Nov-82',
 46.0,
 65,
 'Michael Jackson',
 None,
 'Pop, Rock, R&B',
 'Thriller'}
```

Now let us create a set of genres:

[2]:
```
# Convert list to set

music_genres = set(["pop", "pop", "rock", "folk rock", "hard rock", "soul", \
                    "progressive rock", "soft rock", "R&B", "disco"])
music_genres
```

[2]:
```
{'R&B',
 'disco',
 'folk rock',
 'hard rock',
 'pop',
 'progressive rock',
 'rock',
 'soft rock',
 'soul'}
```

Set Operations

Let us go over set operations, as these can be used to change the set. Consider the set A:

[ ]:
```
# Sample set

A = set(["Thriller", "Back in Black", "AC/DC"])
A
```

We can add an element to a set using the add() method:

[ ]:
```
# Add element to set

A.add("NSYNC")
A
```

If we add the same element twice, nothing will happen as there can be no duplicates in a set:

[ ]:
```
# Try to add duplicate element to the set

A.add("NSYNC")
A
```

We can remove an item from a set using the remove method:

[ ]:
```
# Remove the element from set

A.remove("NSYNC")
A
```

We can verify if an element is in the set using the in command:

[ ]:
```
# Verify if the element is in the set

"AC/DC" in A
```

Sets Logic Operations

Remember that with sets you can check the difference between sets, as well as the symmetric difference, intersection, and union:

Consider the following two sets:

[ ]:
```
# Sample Sets

album_set1 = set(["Thriller", 'AC/DC', 'Back in Black'])
album_set2 = set([ "AC/DC", "Back in Black", "The Dark Side of the Moon"])
```

[ ]:
```
# Print two sets

album_set1, album_set2
```

As both sets contain AC/DC and Back in Black we represent these common elements with the intersection of two circles.

You can find the intersect of two sets as follow using &:

[ ]:
```
# Find the intersections

intersection = album_set1 & album_set2
intersection
```

You can find all the elements that are only contained in album_set1 using the difference method:

[ ]:
```
# Find the difference in set1 but not set2

album_set1.difference(album_set2)
```

You only need to consider elements in album_set1; all the elements in album_set2, including the intersection, are not included.

The elements in album_set2 but not in album_set1 is given by:

[ ]:
```
album_set2.difference(album_set1)
```

You can also find the intersection of album_list1 and album_list2, using the intersection method:

[ ]:
```
# Use intersection method to find the intersection of album_list1 and␣
↪album_list2

album_set1.intersection(album_set2)
```

This corresponds to the intersection of the two circles:

The union corresponds to all the elements in both sets, which is represented by coloring both circles:

The union is given by:

[ ]:
```
# Find the union of two sets

album_set1.union(album_set2)
```

And you can check if a set is a superset or subset of another set, respectively, like this:

[ ]:
```
# Check if superset

set(album_set1).issuperset(album_set2)
```

[ ]:
```
# Check if subset

set(album_set2).issubset(album_set1)
```

Here is an example where issubset() and issuperset() return true:

[ ]:
```
# Check if subset

set({"Back in Black", "AC/DC"}).issubset(album_set1)
```

[ ]:
```
# Check if superset

album_set1.issuperset({"Back in Black", "AC/DC"})
```

Quiz on Sets

Convert the list [‘rap’,‘house’,‘electronic music’, ‘rap’] to a set:

[4]:
```
# Write your code below and press Shift+Enter to execute
set(['rap','house','elctronic music','rap'])
```

[4]:
```
{'elctronic music', 'house', 'rap'}
```

Consider the list A = [1, 2, 2, 1] and set B = set([1, 2, 2, 1]), does sum(A) = sum(B)

[5]:
```
# Write your code below and press Shift+Enter to execute
A = [1, 2, 2, 1]
B = set([1, 2, 2, 1])
print("the sum of A is:", sum(A))
print("the sum of B is:", sum(B))
```

```
the sum of A is: 6
the sum of B is: 3
```

Create a new set album_set3 that is the union of album_set1 and album_set2:

[7]:
```
# Write your code below and press Shift+Enter to execute

album_set1 = set(["Thriller", 'AC/DC', 'Back in Black'])
album_set2 = set([ "AC/DC", "Back in Black", "The Dark Side of the Moon"])
album_set3 = album_set2.union(album_set2)
album_set3
```

[7]:
```
{'AC/DC', 'Back in Black', 'The Dark Side of the Moon'}
```

Double-click **here** for the solution.

Find out if album_set1 is a subset of album_set3:

[9]:
```
# Write your code below and press Shift+Enter to execute
album_set1.issubset(album_set3)
```

[9]:
```
False
```

The last exercise!

Congratulations, you have completed your first lesson and hands-on lab in Python. However, there is one more thing you need to do. The Data Science community encourages sharing work. The best way to share and showcase your work is to share it on GitHub. By sharing your notebook on GitHub you are not only building your reputation with fellow data scientists, but you can also show it off when applying for a job. Even though this was your first piece of work, it is never too early to start building good habits. So, please read and follow this article to learn how to share your work.

Get IBM Watson Studio free of charge!

```
<p><a href="https://cocl.us/NotebooksPython101bottom"><img src="https://s3-api.us-geo.objectst
```

About the Authors:

Joseph Santarcangelo is a Data Scientist at IBM, and holds a PhD in Electrical Engineering. His research focused on using Machine Learning, Signal Processing, and Computer Vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Other contributors: Mavis Zhou