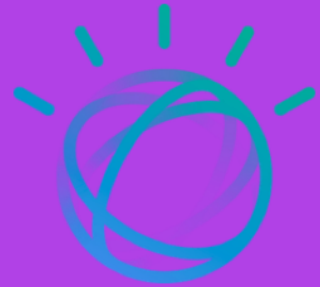


Simplifying AI and Machine-Learning with Watson Studio

- Get your free account and use the Lite plan forever
- No credit card and no autorenewals

Click Here



(https://cocl.us/PYTHON101EN_portal_add_top)



(<https://cognitiveclass.ai/>)

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.

Table of Contents

- [Sets](#)
 - [Set Content \(content\)](#)
 - [Set Operations \(op\)](#)
 - [Sets Logic Operations \(logic\)](#)
- [Quiz on Sets](#)

Estimated time needed: **20 min**

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:

In [1]:

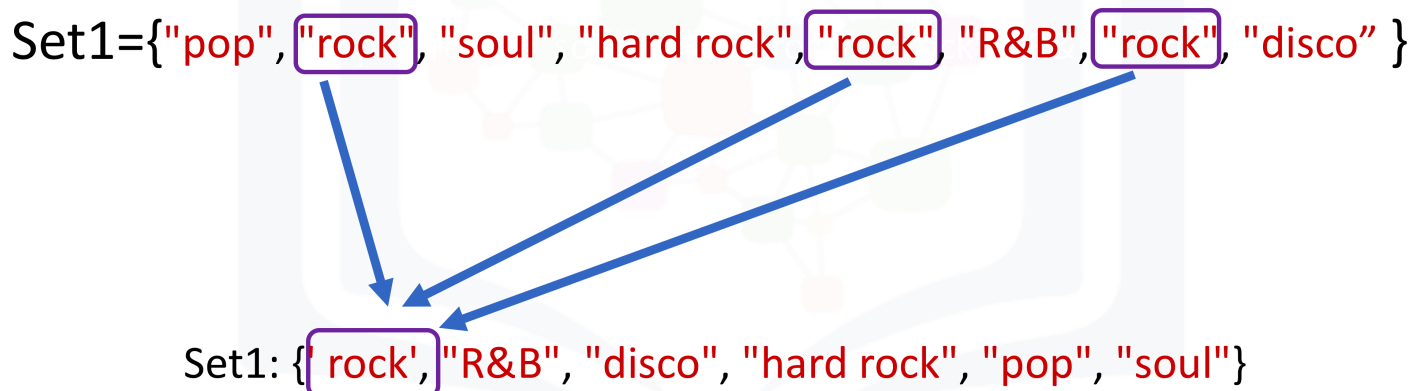
```
# Create a set
```

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

Out[1]:

```
{'R&B', 'disco', 'hard rock', 'pop', 'rock', 'soul'}
```

The process of mapping is illustrated in the figure:



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

In [2]:

```
# 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
```

Out[2]:

```
{'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:

In [3]:

```
# Convert list to set

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

Out[3]:

```
{'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**:

In [4]:

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

Out[4]:

```
{'AC/DC', 'Back in Black', 'Thriller'}
```

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

In [5]:

```
# Add element to set  
  
A.add("NSYNC")  
A
```

Out[5]:

```
{'AC/DC', 'Back in Black', 'NSYNC', 'Thriller'}
```

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

In [6]:

```
# Try to add duplicate element to the set  
  
A.add("NSYNC")  
A
```

Out[6]:

```
{'AC/DC', 'Back in Black', 'NSYNC', 'Thriller'}
```

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

In [7]:

```
# Remove the element from set  
  
A.remove("NSYNC")  
A
```

Out[7]:

```
{'AC/DC', 'Back in Black', 'Thriller'}
```

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

In [8]:

```
# Verify if the element is in the set
```

```
"AC/DC" in A
```

Out[8]:

True

Sets Logic Operations

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

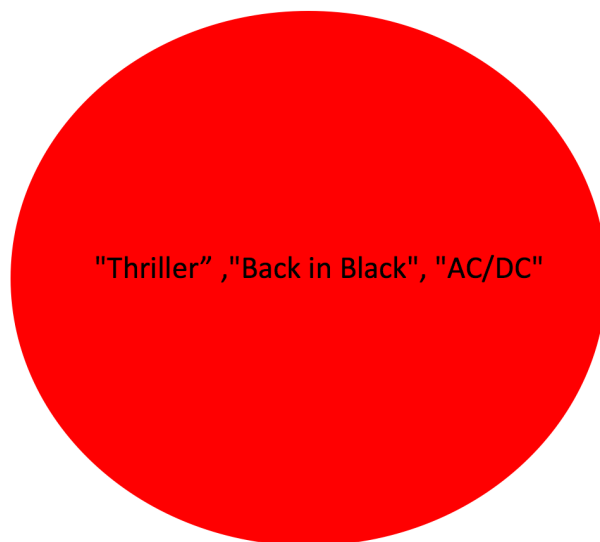
Consider the following two sets:

In [9]:

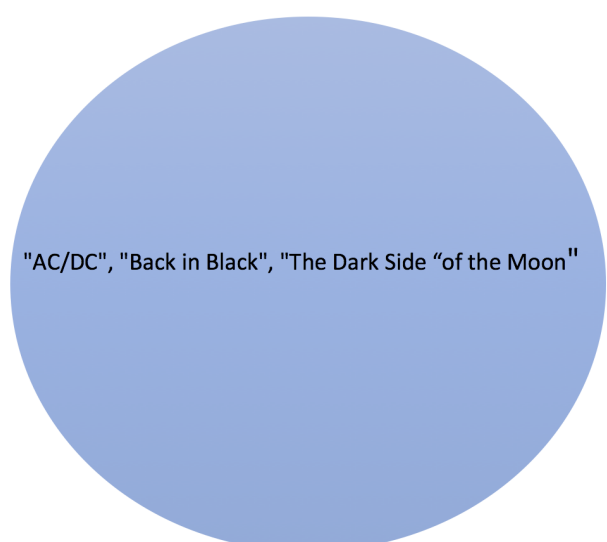
```
# Sample Sets
```

```
album_set1 = set(["Thriller", 'AC/DC', 'Back in Black'])
```

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



album_list1



album_list2

In [10]:

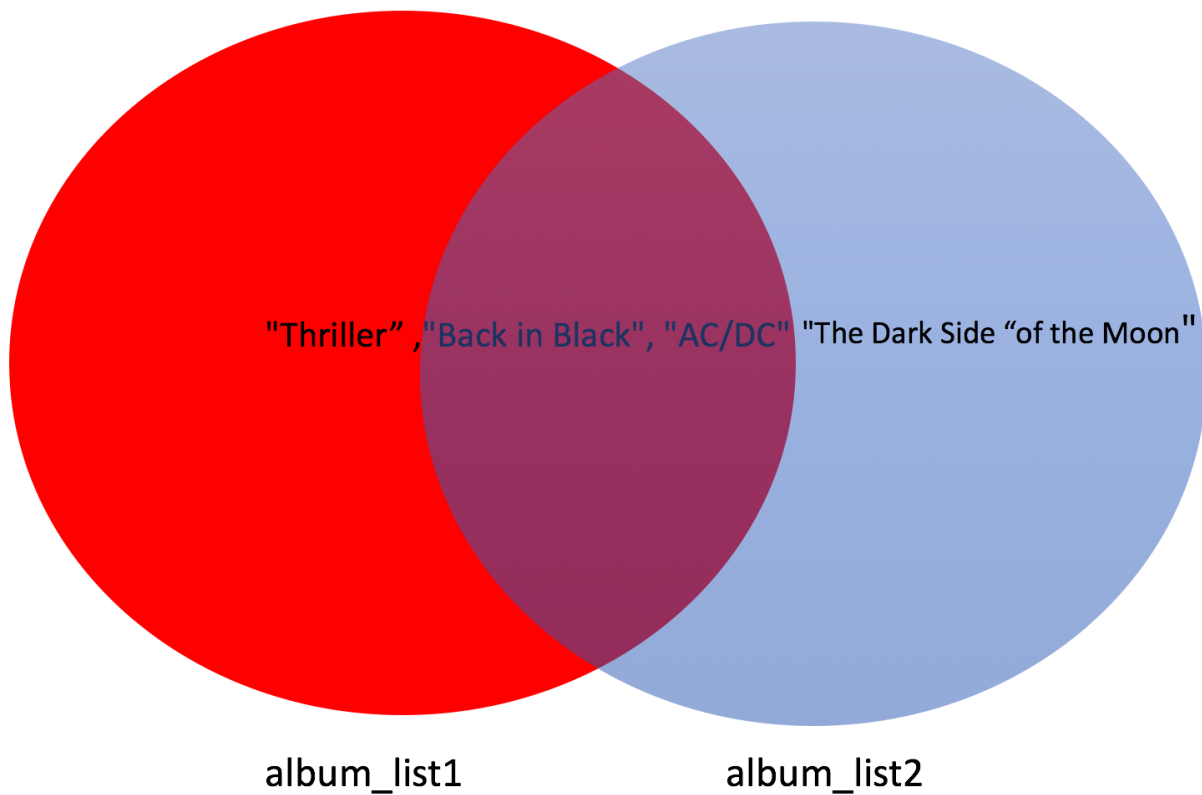
```
# Print two sets
```

```
album_set1, album_set2
```

Out[10]:

```
{'AC/DC', 'Back in Black', 'Thriller'},  
{'AC/DC', 'Back in Black', 'The Dark Side of the Moon'}
```

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 `&` :

In [11]:

```
# Find the intersections
```

```
intersection = album_set1 & album_set2  
intersection
```

Out[11]:

```
{'AC/DC', 'Back in Black'}
```

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

In [12]:

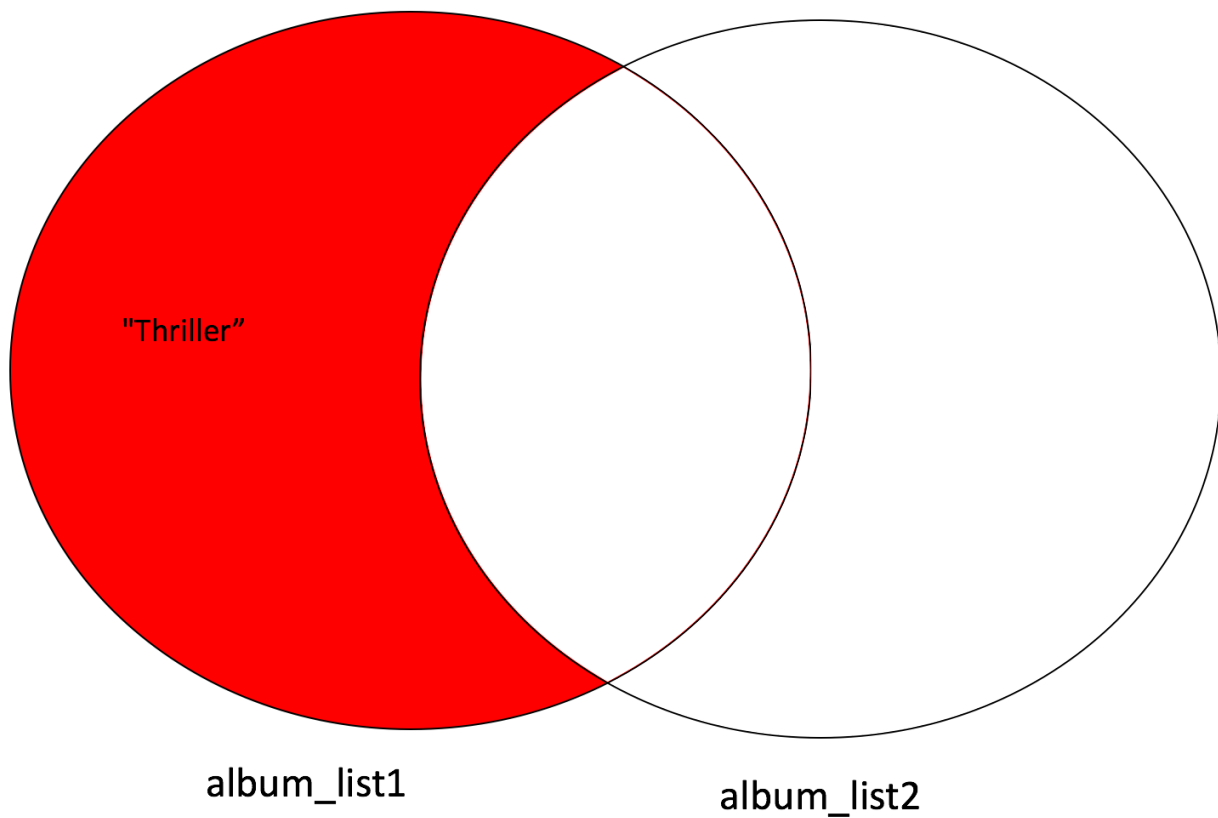
```
# Find the difference in set1 but not set2
```

```
album_set1.difference(album_set2)
```

Out[12]:

```
{'Thriller'}
```

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



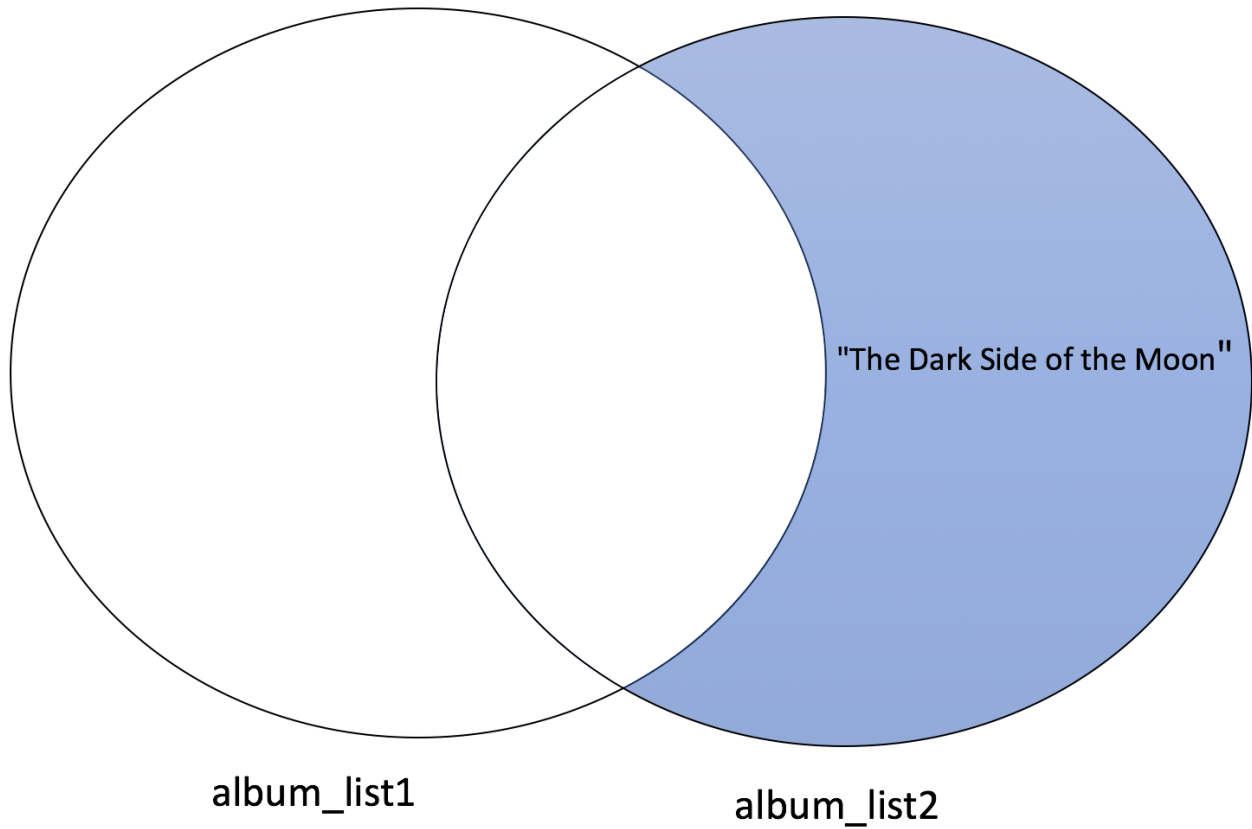
The elements in `album_set2` but not in `album_set1` is given by:

In [13]:

```
album_set2.difference(album_set1)
```

Out[13]:

```
{'The Dark Side of the Moon'}
```



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

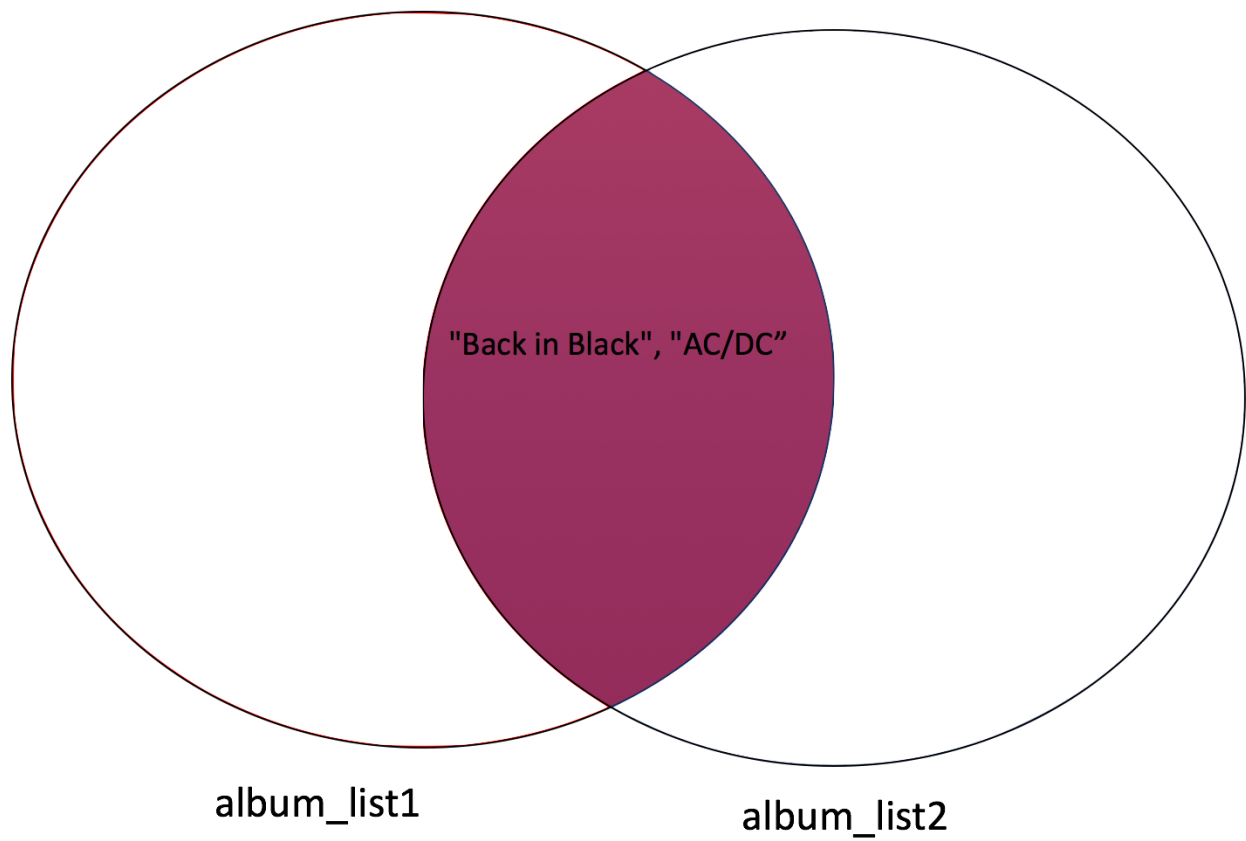
In [14]:

```
# Use intersection method to find the intersection of album_list1 and album_list2  
album_set1.intersection(album_set2)
```

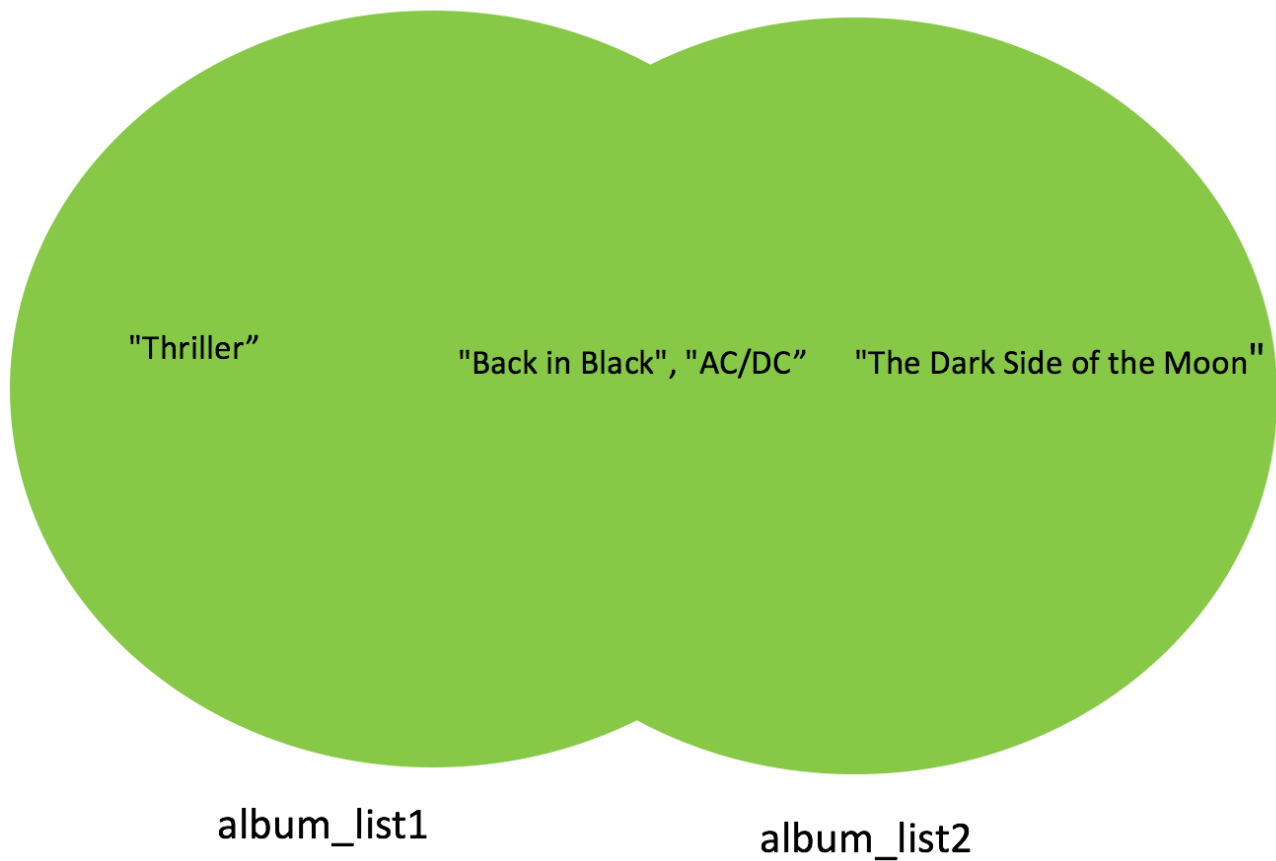
Out[14]:

```
{'AC/DC', 'Back in Black'}
```

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:

In [15]:

```
# Find the union of two sets
```

```
album_set1.union(album_set2)
```

Out[15]:

```
{'AC/DC', 'Back in Black', 'The Dark Side of the Moon', 'Thriller'}
```

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

In [21]:

```
# Check if superset  
  
set(album_set1).issuperset(album_set2)
```

Out[21]:

False

In [22]:

```
# Check if subset  
  
set(album_set2).issubset(album_set1)
```

Out[22]:

False

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

In [23]:

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

Out[23]:

True

In [24]:

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

Out[24]:

True

Quiz on Sets

Convert the list `['rap','house','electronic music', 'rap']` to a set:

In [25]:

```
# Write your code below and press Shift+Enter to execute
list = ['rap', 'house', 'electronic music', 'rap']
set = set(list)
type(set)
```

Out[25]:

set

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

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

In [26]:

```
# 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))
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-26-e2454573ad0b> in <module>
      1 # Write your code below and press Shift+Enter to execute
      2 A = [1, 2, 2, 1]
----> 3 B = set([1, 2, 2, 1])
      4 print("the sum of A is:", sum(A))
      5 print("the sum of B is:", sum(B))
```

TypeError: 'set' object is not callable

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

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

In [30]:

```
# 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_set1.union(album_set2)
album_set3
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-30-8f3a4cc3fe6b> in <module>
      1 # Write your code below and press Shift+Enter to execute
      2
----> 3 album_set1 = set(["Thriller", 'AC/DC', 'Back in Black'])
      4 album_set2 = set([ "AC/DC", "Back in Black", "The Dark Side of the Moon"])
      5 album_set3 = album_set1.union(album_set2)
```

TypeError: 'set' object is not callable

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

Find out if `album_set1` is a subset of `album_set3` :

In [32]:

```
# Write your code below and press Shift+Enter to execute
album_set1.issubset(album_set2)
```

Out[32]:

False

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

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](https://cognitiveclass.ai/blog/data-scientists-stand-out-by-sharing-your-notebooks/) (<https://cognitiveclass.ai/blog/data-scientists-stand-out-by-sharing-your-notebooks/>) to learn how to share your work.

Get IBM Watson Studio free of charge!

Get IBM Watson Studio free of charge!

Build and train AI & machine learning models, prepare and analyze data – all in a flexible, hybrid cloud environment. Get IBM Watson Studio Lite Plan free of charge.



Learn

Get started or get better with built-in learning.



Create

Use the best of open source tooling with IBM innovation.



Collaborate

Work smarter using community, work faster with your team.

[Sign Up For a Free Trial](#)

(https://cocl.us/PYTHON101EN_portal_add_bottom).

About the Authors:

[Joseph Santarcangelo](https://www.linkedin.com/in/joseph-s-50398b136/) (<https://www.linkedin.com/in/joseph-s-50398b136/>) 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](https://www.linkedin.com/in/jiahui-mavis-zhou-a4537814a) (www.linkedin.com/in/jiahui-mavis-zhou-a4537814a)

Copyright © 2018 IBM Developer Skills Network. This notebook and its source code are released under the terms of the [MIT License \(https://cognitiveclass.ai/mit-license/\)](https://cognitiveclass.ai/mit-license/).