😊 Advanced Sets

1. In this lab, we will learn about the various methods for sets that you may not have seen yet. We'll go over the basic ones you already know and then dive a little deeper.

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

2. We are going to look at the **add method**, this method **adds elements** to a set. Remember, a set won't duplicate elements; it will only present them once (that's why it's called a set!)

```
[2]: s.add(1)
```

```
[3]: s.add(2)
```

```
[4]: s
```

```
[4]: {1, 2}
```

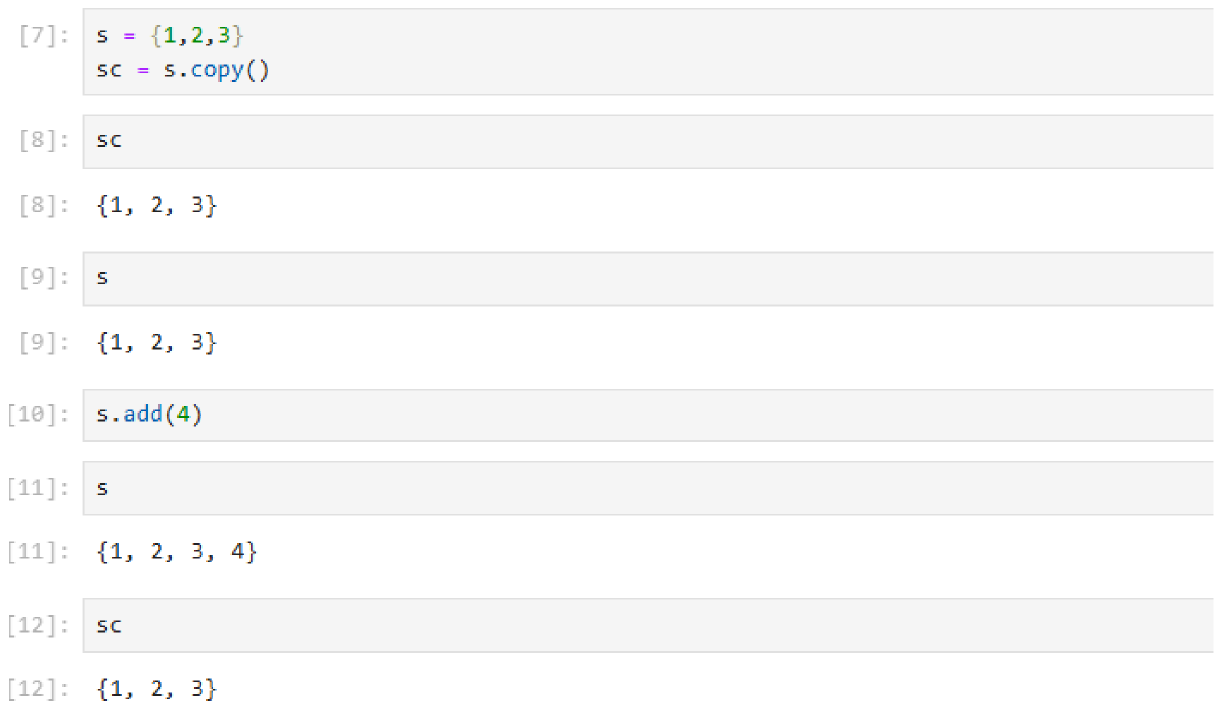
3. The **clear** method removes all the elements from the set.

```
[5]: s.clear()
```

```
[6]: s
```

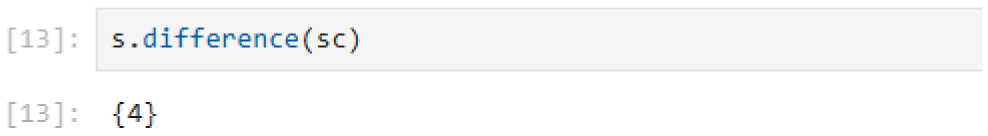
```
[6]: set()
```

4. The **copy** method returns a copy of the set. Note, it is a copy, so changes to the original don't affect the copy.



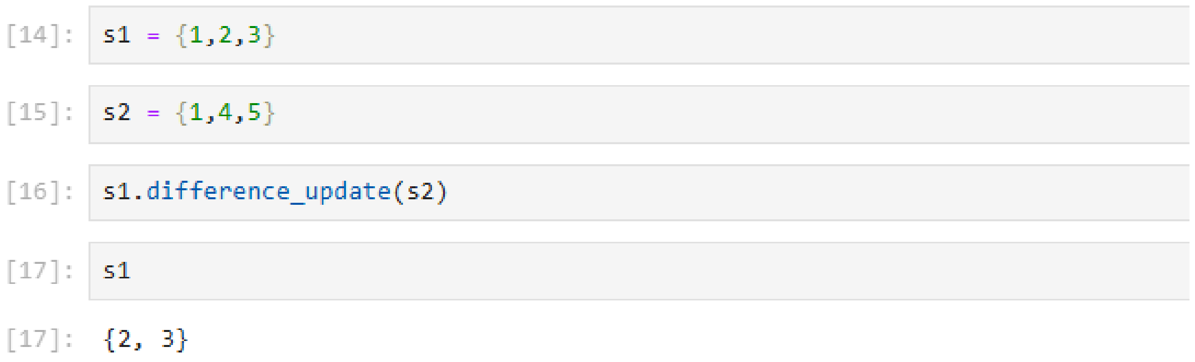
```
[7]:  s = {1,2,3}
      sc = s.copy()

[8]:  sc

[8]:  {1, 2, 3}

[9]:  s

[9]:  {1, 2, 3}

[10]: s.add(4)

[11]: s

[11]: {1, 2, 3, 4}

[12]: sc

[12]: {1, 2, 3}
```

5. The **difference** method returns the difference of two or more sets.



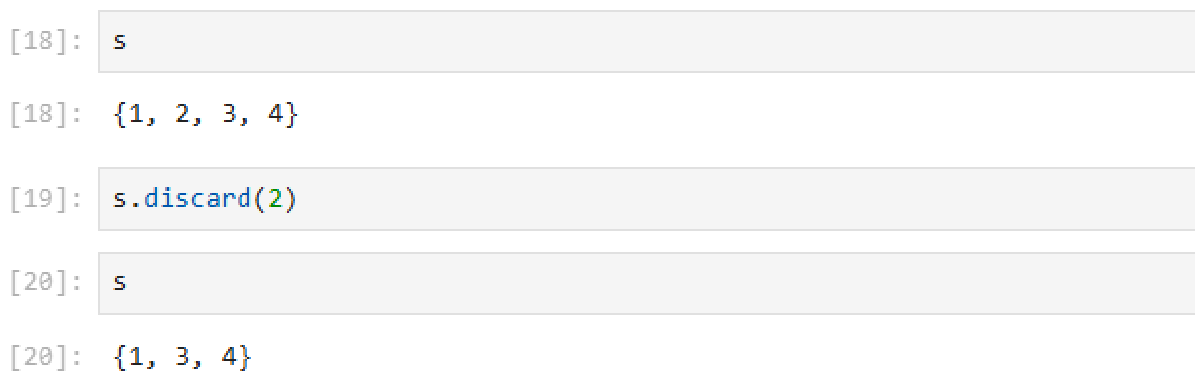
```
[13]: s.difference(sc)

[13]: {4}
```

6. The **difference_update** method returns set1 after removing elements found in set2.



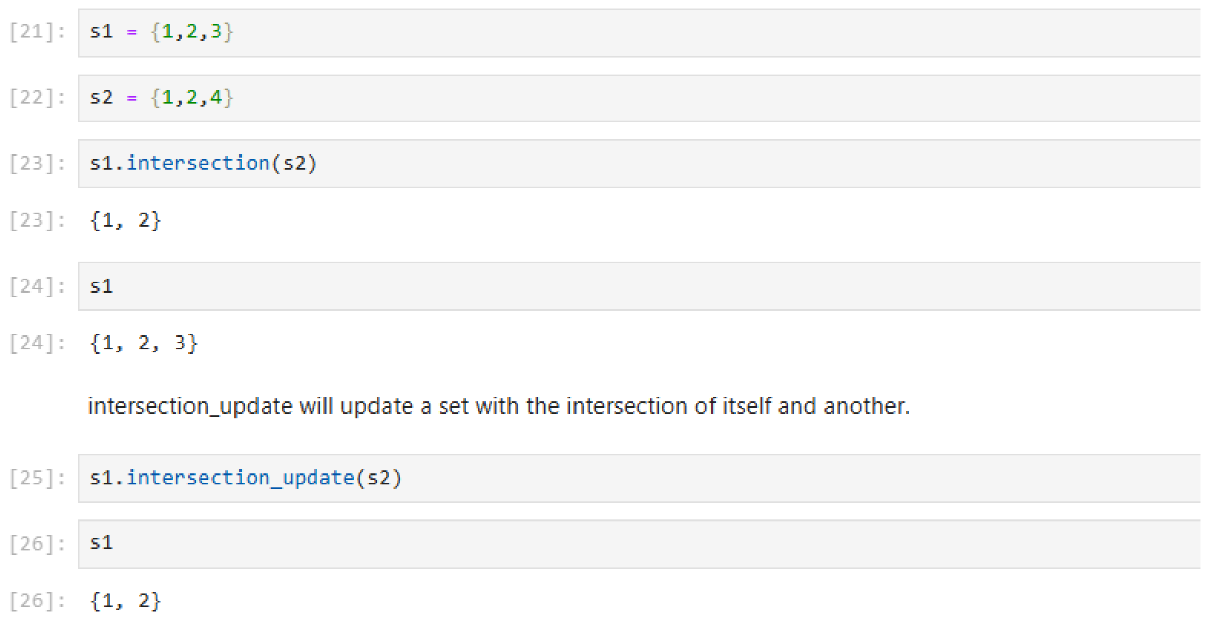
```
[14]: s1 = {1,2,3}

[15]: s2 = {1,4,5}

[16]: s1.difference_update(s2)

[17]: s1

[17]: {2, 3}
```

7. The **discard** method removes an element from a set if it is a member. If the element is not a member, do nothing.



```
[18]: s
[18]: {1, 2, 3, 4}

[19]: s.discard(2)

[20]: s
[20]: {1, 3, 4}
```

8. The **intersection and intersection_update** return the intersection of two or more sets as a new set. (i.e., elements that are common to all of the sets.)

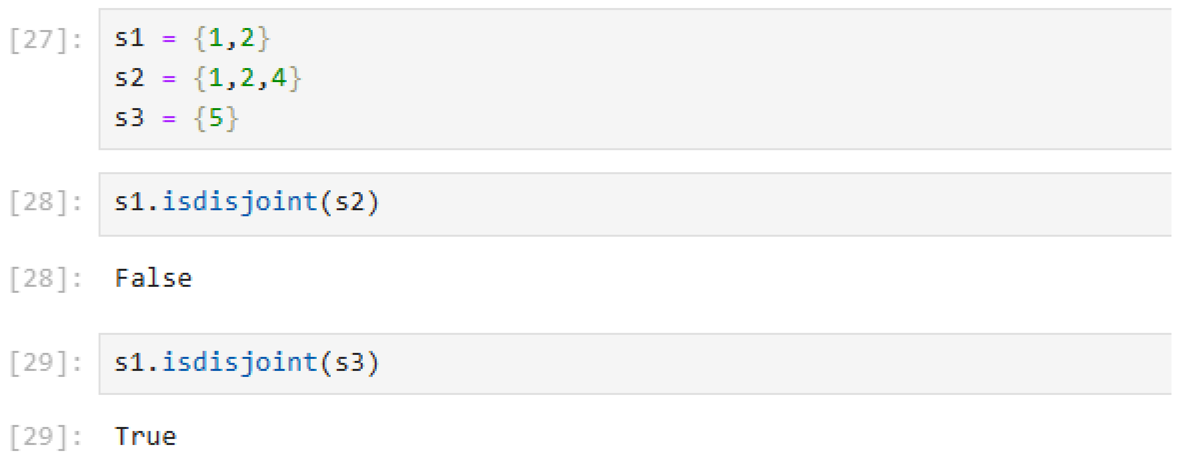


```
[21]: s1 = {1,2,3}

[22]: s2 = {1,2,4}

[23]: s1.intersection(s2)
[23]: {1, 2}

[24]: s1
[24]: {1, 2, 3}
```

intersection_update will update a set with the intersection of itself and another.

```
[25]: s1.intersection_update(s2)

[26]: s1
[26]: {1, 2}
```

9. This method (**isdisjoint**) will return True if two sets have a null intersection.



```
[27]: s1 = {1,2}
      s2 = {1,2,4}
      s3 = {5}

[28]: s1.isdisjoint(s2)
[28]: False

[29]: s1.isdisjoint(s3)
[29]: True
```

10. **The issubset** method reports whether another set is a subset of this set.

```
[30]: s1
[30]: {1, 2}
```

```
[31]: s2
[31]: {1, 2, 4}
```

```
[32]: s1.issubset(s2)
[32]: True
```

11. The **issuperset** method will report whether this set contains another set.

```
[33]: s2.issuperset(s1)
[33]: True
```

```
[34]: s1.issuperset(s2)
[34]: False
```

12. **symmetric_difference and symmetric_update** return the symmetric difference of two sets as a new set. (i.e., all elements that are in exactly one of the sets.)

```
[35]: s1
[35]: {1, 2}
```

```
[36]: s2
[36]: {1, 2, 4}
```

```
[37]: s1.symmetric_difference(s2)
[37]: {4}
```

13. The union method returns the union of two sets (i.e., all elements that are in either set).
14. The update method returns by updating a set with the union of itself and others.

```
[38]: s1.union(s2)
[38]: {1, 2, 4}
```

```
[39]: s1.update(s2)
```

```
[40]: s1
```

```
[40]: {1, 2, 4}
```