Set and Booleans

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

1. Sets

Sets are an unordered collection of *unique* elements. We can construct them by using the set() function.

Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc. Let's go ahead and make a set to see how it works

In [1]:

```
x = set()
```

In [2]:

```
# We add to sets with the add() method
x.add(1)
```

In [3]:

```
#Show
x
```

Out[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?

We can add single element using the add() method and multiple elements using the update() method. The update() method can take tuples, lists, strings or other sets as its argument. In all cases, duplicates are avoided.

In [4]:

```
# Add a different element
x.add(2)
```

In [5]:

```
#Show
x
```

Out[5]:

```
{1, 2}
```

In [6]:

```python
# Try to add the same element
x.add(1)
```

In [7]:

```python
#Show
x
```

Out[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:

In [8]:

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

In [9]:

```python
# Cast as set to get unique values
set(list1)
```

Out[9]:

```
{1, 2, 3, 4, 5, 6}
```

Python Set Operations

Sets can be used to carry out mathematical set operations like union, intersection, difference and symmetric difference. We can do this with operators or methods.

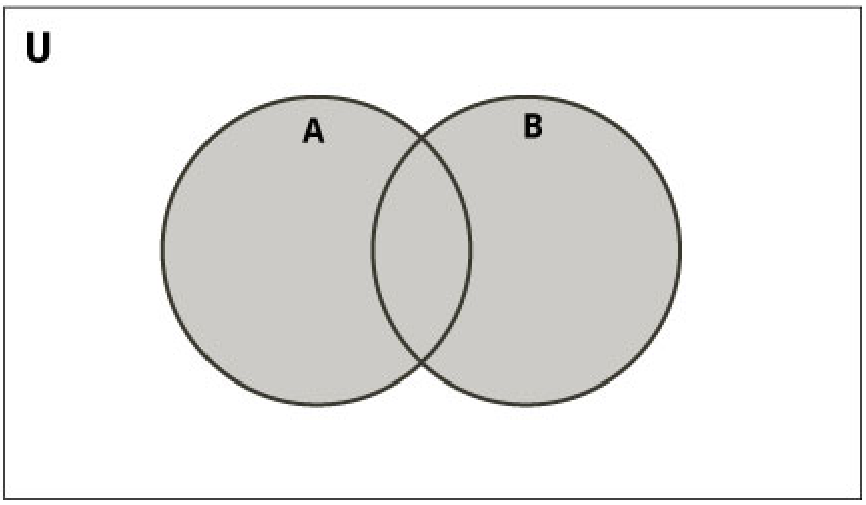
Let us consider the following two sets for the following operations.

- A = {1, 2, 3, 4, 5}
- B = {4, 5, 6, 7, 8}

Set Union

Union of A and B is a set of all elements from both sets.

Union is performed using | operator. Same can be accomplished using the method union().



In [10]:

```python
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use | operator
print(A | B)
# or
A.union(B)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

Out[10]:

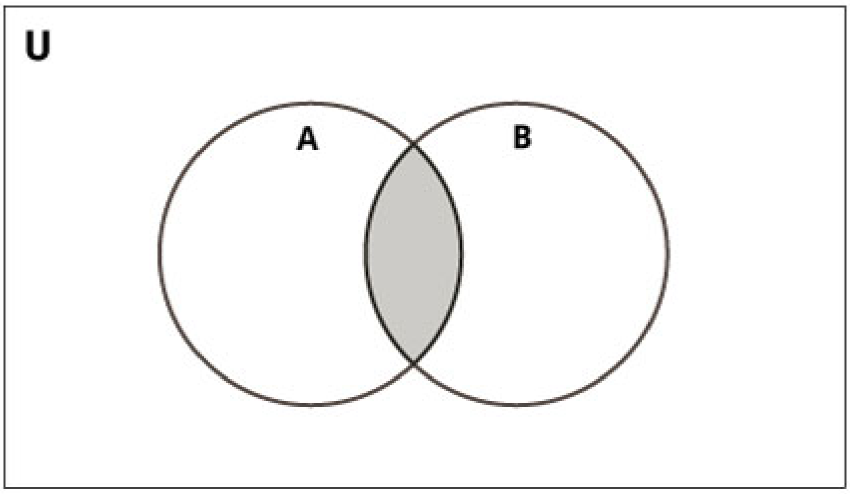
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

Set Intersection

Set Intersection in Python

Intersection of A and B is a set of elements that are common in both sets.

Intersection is performed using & operator. Same can be accomplished using the method intersection().



In [11]:

```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use & operator
print(A & B)
# or
A.intersection(B)
```

```
{4, 5}
```

Out[11]:

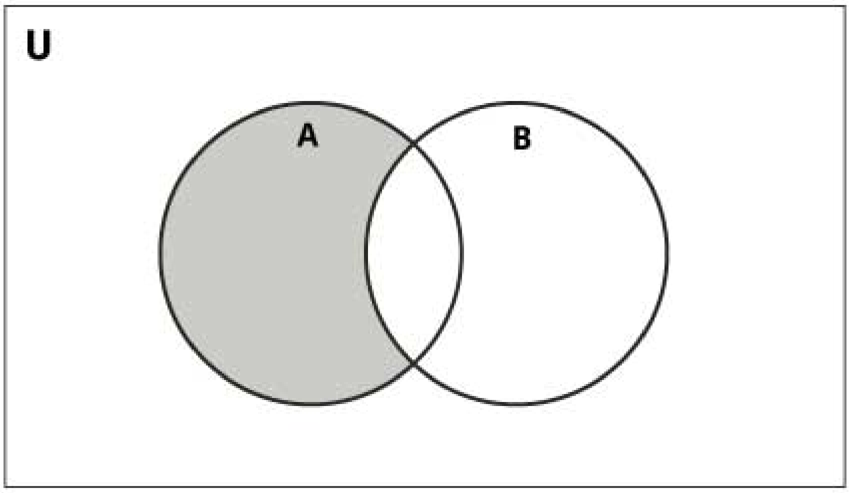
```
{4, 5}
```

Set Difference

Set Difference in Python

Difference of A and B (A - B) is a set of elements that are only in A but not in B. Similarly, B - A is a set of element in B but not in A.

Difference is performed using - operator. Same can be accomplished using the method difference().



In [12]:

```python
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use - operator on A
print(A - B)
# or
A.difference(B)
{1, 2, 3}
```

```
{1, 2, 3}
```

Out[12]:

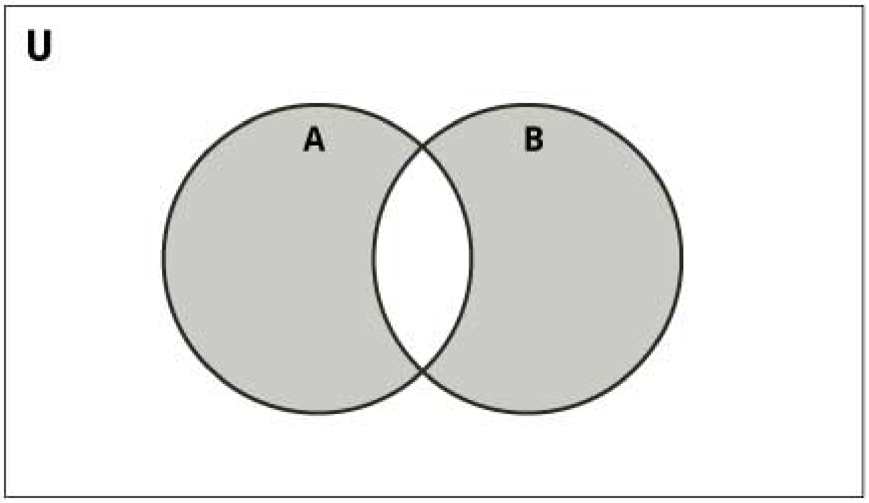
```
{1, 2, 3}
```

Set Symmetric Difference

Set Symmetric Difference in Python

Symmetric Difference of A and B is a set of elements in both A and B except those that are common in both.

Symmetric difference is performed using ^ operator. Same can be accomplished using the method symmetric_difference().



In [13]:

```python
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use ^ operator
print(A ^ B)
# or
A.symmetric_difference(B)
{1, 2, 3, 6, 7, 8}
```

```
{1, 2, 3, 6, 7, 8}
```

Out[13]:

```
{1, 2, 3, 6, 7, 8}
```

Built-in Functions with Set

- **all()**: Return True if all elements of the set are true (or if the set is empty).
- **any()**: Return True if any element of the set is true. If the set is empty, return False.
- **enumerate()**: Return an enumerate object. It contains the index and value of all the items of set as a pair.
- **len()**: Return the length (the number of items) in the set.
- **max()**: Return the largest item in the set.
- **min()**: Return the smallest item in the set.
- **sorted()**: Return a new sorted list from elements in the set(does not sort the set itself).
- **sum()**: Retrun the sum of all elements in the set.

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 [14]:

```python
# Set object to be a boolean
a = True
```

In [15]:

```python
#Show
a
```

Out[15]:

```
True
```

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

In [16]:

```
# Output is boolean
1 > 2
```

Out[16]:

```
False
```

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

In [17]:

```
# None placeholder
b = None
```

In [18]:

```
# 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!