Day 3: Sets

SETS

Definition: Unordered collection of unique items.

Syntax:

Curly Braces **{}**

Example: `my_set = {'apple', 'banana', 'cherry'}`

Key Operations:

REMEMBER: Sets do not allow duplicate values and are unordered (no specific order or indexing).

ADDING ELEMENTS:

Add an item to my_set:

```
my_set = {'apple', 'banana', 'cherry'}
my_set.add('orange')
```

Result: {'apple', 'banana', 'cherry', 'orange'}

REMOVING ELEMENTS:

Remove an item from my_set using `.remove()` (raises an error if the item is not found):

```
my_set = {'apple', 'banana', 'cherry'}
my_set.remove('banana')
```

Result: {'apple', 'cherry'}

Remove an item safely using `.discard()` (doesn't raise an error if the item isn't found):

```
my_set = {'apple', 'banana', 'cherry'}
my_set.discard('banana')
```

Result: {'apple', 'cherry'}

SET OPERATIONS:

Union: Combines two sets, keeping only unique items.

```
set1 = {'apple', 'banana'}
set2 = {'banana', 'orange'}
set1.union(set2)
```

Result: {'apple', 'banana', 'orange'}

Intersection: Finds common items between two sets.

```
set1 = {'apple', 'banana'}
set2 = {'banana', 'orange'}
set1.intersection(set2)
```

Result: {'banana'}

Difference: Shows items in one set but not in the other.

```
set1 = {'apple', 'banana', 'cherry'}
set2 = {'banana', 'orange'}
set1.difference(set2)
```

Result: {'apple', 'cherry'}

Use Case

Sets are useful for storing collections of unique items, like unique usernames or product codes, and for performing quick comparisons between collections.

PRACTICE EXERCISES

ADDING AND REMOVING ELEMENTS:

Create a set called **'fruits'** with the items `{'apple', 'banana', 'cherry'}`. Add **'orange'** to the set, then remove **'banana'**.

Expected Result: `{'apple', 'cherry', 'orange'}`

UNION AND INTERSECTION:

Given two sets, `set_a = {1, 2, 3, 4}` and `set_b = {3, 4, 5, 6}`, find the union of set_a and set_b and the intersection of set_a and set_b

Expected Results:
Union: `{1, 2, 3, 4, 5, 6}`
Intersection: `{3, 4}`

DIFFERENCE OPERATION:

Given two sets, `set_x = {'cat', 'dog', 'fish'}` and

`set_y = {'dog', 'bird'}`, find the items that are in set_x but not in set_y.

Expected Result: `{'cat', 'fish'}`