Introduction to Sets

Python Sets

Unordered collection of values

Denoted using curly braces

Set items are unordered, meaning they cannot be indexed by bracket notation

We can access contents using other ways, such as a for ... in loop

Syntax:
{*item1*, *item2*, ...}

Example:
my_set = {“ALPHA”, 123, (True, False)}

~~print(my_set[0])~~

Set contents

A set is mutable, but its items **cannot** have mutable data types

Mutable data types: list, dictionary, set

✓
```
my_set = {“ALPHA”, 123, (True,
                          False)}
```

Set contents

A set is mutable, but its items **cannot** have mutable data types

Mutable data types: list, dictionary, set

❌ my_set = {"ALPHA", 123, [True, False]}

Set contents

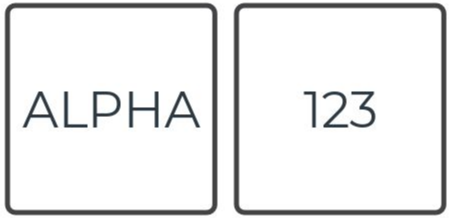
A set is mutable, but its items **cannot** have mutable data types

Mutable data types: list, dictionary, set

Items cannot have duplicate values

Useful for when you don't want any duplicates in your data

❌ my_set = {"ALPHA", 123, 123}



Empty Set

{} creates an empty dictionary, not an empty set

To create an empty set, use the set() function

~~empty_set = {}~~

empty_set = set()

Sets recap

Unordered collection of values, denoted with curly braces

Items are comma-separated

Items are not indexed, cannot use bracket notation

Set is mutable, but items must be of immutable data type
(no list, dictionary, set)

Duplicates will be removed

Create empty set using set() function with no arguments