Day 18: Sets

- There are 4 built-in data types in Python used to store collections of data:
    1. List
    2. Tuple
    3. Set
    4. Dictionary

Unpacking a tuple/ list

```
a = (1, 2)
c, d = a

print(c)
print(d)
```

Sets

```
thisset = {"apple", "banana", "cherry"}
# Sets are written in curly brackets {}
```

- Sets are used to store multiple items in a single variable.
- A set is a collection which is:
    - unordered
    - unchangeable (Note: Set items are unchangeable, but you can add and remove items)
    - unindexed
    - do not allow duplicates

Unordered

- Sets are written with curly brackets.
- Sets are unordered, so you cannot be sure in which order the items will appear.

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

Unindexed

- Set items cannot be referred to by index or key.

```
thisset = {"apple", "banana", "cherry"}
print(thisset[1]) # error
```

Unchangeable

- Set items are unchangeable, meaning that we cannot change the items after the set has been created.
- Once a set is created, you cannot change its items, but you can remove items and add new items.

Duplicates Not Allowed

- Sets cannot have two items with the same value.

```
thisset = {"apple", "banana", "cherry", "apple"}

print(thisset) # output: {'banana', 'cherry', 'apple'}
```

- `True` and `1` is considered the same value.

```
thisset = {"apple", "banana", "cherry", True, 1, 2}

print(thisset)
```

- `False` and `0` is considered the same value.

```
thisset = {"apple", "banana", "cherry", False, True, 0}

print(thisset)
```

Set Methods

Length

```
thisset = {"apple", "banana", "cherry"}

print(len(thisset)) # output: 3

thisset = {"apple", "banana", "cherry", "apple"}

print(len(thisset)) # output: 3
```

The set() constructor

- It is also possible to use the set() constructor to make a set.

```
thisset = set(("apple", "banana", "cherry"))
print(thisset)
print(type(thisset))
```

Access Items

- You cannot access items in a set by referring to an index or a key.
- But you can loop through the set items using a `for` loop, or ask if a specified value is present in a set, by using the `in` keyword.

```
thisset = {"apple", "banana", "cherry"}
```

```
for x in thisset:
  print(x)

print("banana" in thisset) # output: True
print("banana" not in thisset) # output: False
```

Add Set Items

- add() method

```
thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)
```

- update() method

```
thisset = {"apple", "banana", "cherry"}
thatset = {"orange", "kiwi"}

thisset.update(thatset)

print(thisset)
```

Remove Set Items

- remove() method - removes the element, gives error if element is not present.

```
thisset = {"apple", "banana", "cherry"}
thisset.remove("apple")
print(thisset)
```

- discard() method - discards the element, does not give error if element not present.

```
thisset = {"apple", "banana", "cherry"}
thisset.discard("banana")
print(thisset)
```

- pop() method - removes a random element

```
thisset = {"apple", "banana", "cherry"}
thisset.pop()
print(thisset)
```

- clear() method - deletes all elements in the set but keeps the structure

```
thisset = {"apple", "banana", "cherry"}
thisset.pop()
print(thisset)
```

- del method - deletes the set

```
thisset = {"apple", "banana", "cherry"}
thisset.clear()
print(thisset)
```

📝 Test