**Python Collections (Arrays)**

There are four collection data types in the Python programming language:

- List is a collection which is ordered and changeable. Allows duplicate members.
    - thislist = list(("apple", "banana", "cherry")) # note the double round-brackets
    - mylist = ["apple", "banana", "cherry"]
- Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
    - thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets
    - mytuple = ("apple", "banana", "cherry")
- Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
    - thisset = set(("apple", "banana", "cherry")) # note the double round-brackets
    - myset = {"apple", "banana", "cherry"}
- Dictionary is a collection which is ordered** and changeable. No duplicate members. Dictionaries are used to store data values in key:value pairs.
    - thisdict = dict(name = "John", age = 36, country = "Norway")
    - thisdict = {

```
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
```

## List Methods

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

If you add new items to a list, the new items will be placed at the end of the list.

The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

Since lists are indexed, lists can have items with the same value

To determine how many items a list has, use the len() function

List items can be of any data type

A list can contain different data types

From Python's perspective, lists are defined as objects with the data type 'list'

| Method | Description |
|---|---|
| append() | Adds an element at the end of the list |
| clear() | Removes all the elements from the list |
| copy() | Returns a copy of the list |
| count() | Returns the number of elements with the specified value |
| extend() | Add the elements of a list (or any iterable), to the end of the current list |
| index() | Returns the index of the first element with the specified value |
| insert() | Adds an element at the specified position |
| pop() | Removes the element at the specified position |
| remove() | Removes the item with the specified value |
| reverse() | Reverses the order of the list |
| sort() | Sorts the list |

## Tuple

Since tuples are immutable, they do not have a built-in append() method, but there are other ways to add items to a tuple.

1. Convert into a list: Just like the workaround for changing a tuple, you can convert it into a list, add your item(s), and convert it back into a tuple.

```
thistuple = ("apple", "banana", "cherry")

y = list(thistuple)

y.append("orange")

thistuple = tuple(y)
```

2. Add tuple to a tuple. You are allowed to add tuples to tuples, so if you want to add one item, (or many), create a new tuple with the item(s), and add it to the existing tuple:

```
thistuple = ("apple", "banana", "cherry")

y = ("orange",)

thistuple += y


print(thistuple)
```

### Unpacking a Tuple

```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits
```

| Method | Description |
|---|---|
| count() | Returns the number of times a specified value occurs in a tuple |
| index() | Searches the tuple for a specified value and returns the position of where it was found |

## Sets Methods

| Method | Description |
|---|---|
| add() | Adds an element to the set |
| clear() | Removes all the elements from the set |
| copy() | Returns a copy of the set |
| difference() | Returns a set containing the difference between two or more sets |
| difference_update() | Removes the items in this set that are also included in another, specified set |
| discard() | Remove the specified item |
| intersection() | Returns a set, that is the intersection of two other sets |
| intersection_update() | Removes the items in this set that are not present in other, specified set(s) |
| isdisjoint() | Returns whether two sets have a intersection or not |
| issubset() | Returns whether another set contains this set or not |
| issuperset() | Returns whether this set contains another set or not |
| pop() | Removes an element from the set |
| remove() | Removes the specified element |
| symmetric_difference() | Returns a set with the symmetric differences of two sets |
| symmetric_difference_update() | inserts the symmetric differences from this set and another |
| union() | Return a set containing the union of sets |
| update() | Update the set with the union of this set and others |

## Dictionary Methods

| Method | Description |
|---|---|
| clear() | Removes all the elements from the dictionary |
| copy() | Returns a copy of the dictionary |
| fromkeys() | Returns a dictionary with the specified keys and value |
| get() | Returns the value of the specified key |
| items() | Returns a list containing a tuple for each key value pair |
| keys() | Returns a list containing the dictionary's keys |
| pop() | Removes the element with the specified key |
| popitem() | Removes the last inserted key-value pair |
| setdefault() | Returns the value of the specified key. If the key does not exist: insert the key, with the specified value |
| update() | Updates the dictionary with the specified key-value pairs |
| values() | Returns a list of all the values in the dictionary |