For quick reference, you can refer this curated list of ready-to-use data structures and methods.

List

A list is a collection which is ordered and changeable. In Python lists are written with square brackets.

```
list = ["Apple", "Banana", "Cherry"]
```

1. You access the list items by referring to the index number:

   ```
   print(list[1])       //  Start with 0th index so Output is Banana
   ```

2. To change the value of a specific item, refer to the index number:

   ```
   list[1] = "Orange"
   ```

3. You can loop through the list items by using a `for` loop:

   ```
   for x in list:
         print(x)          // Output is Apple , Orange , Cherry
   ```

4. To determine if a specified item is present in a list use the `in` keyword:

   ```
   if "Apple" in list:

         print (“Yes”)               // Yes if Apple is present in list

   else:

         print(“No”)                 // No  if it is not Present
   ```

5. To determine how many items a list have, use the `len()` method:

   ```
   print(len(list))    // Output is 3 as contains 3 elements
   ```

6. To add an item to the end of the list, use the `append()` method:

```
list.append("Mango")    // Append at the end of list
```

7. To add an item at the specified index, use the `insert()` method:

```
list.insert(1, "Mango")  // insert at index 1 of list
```

8. The `remove()` method removes the specified item:

```
list.remove("Banana")        // Remove the element Banana if present
```

9. The `pop()` method removes the specified index, (or the last item if index is not specified)

```
list.pop()
```

10.The `del` keyword removes the specified index:

```
    del list[0]  // removes the specified index
```

Tuple

A tuple is a collection which is ordered and **unchangeable**. In Python tuples are written with round brackets.

```
tuple = ("Apple", "Banana", "Cherry")
```

1. You can access tuple items by referring to the index number, inside square brackets:

```
print(tuple[1])  // Output is Banana the specified index
```

2. Once a tuple is created, you cannot change its values. Tuples are **unchangeable**.

```
tuple[1] = "Orange" // Gives error the value remain unchanged
```

3. You can loop through the tuple items by using a `for` loop.

```
for x in tuple:
        print(x)          // Generate all element present in tuple
```

4. To determine if a specified item is present in a tuple use the `in` keyword:

```
if "Apple" in tuple:
        print("Yes")          // Output is Yes if Apple is present in tuple
```

5. To determine how many items a list have, use the `len()` method:
   `print(len(tuple)) // Output is 3 as 3 element are in tuple`
6. Tuples are **unchangeable**, so you cannot add or remove items from it, but you can delete the tuple completely:
7. Python has two built-in methods that you can use on tuples.

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

Set

A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets.

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

1. You cannot access items in a set by referring to an index, since sets are unordered the items has no index. 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.

   ```
   for x in set:
     print(x)       // Output contains all element present in set
   ```

2. Once a set is created, you cannot change its items, but you can add new items.

   To add one item to a set use the `add()` method.

   ```
        set.add("Orange")                 // Add one element at end
   ```

   To add more than one item to a set use the `update()` method.

   ```
        set.update(["Orange", "Mango", "Grapes"])   // Add all
   ```

   // element in the end

3. To determine how many items a set have, use the `len()` method.

   ```
   print(len(set))       // output is length of set
   ```

4. To remove an item in a set, use the `remove()`, or the `discard()` method.

   ```
   set.remove("Banana") //Remove element if present else raise error

   set.discard("Banana") // Remove element if present else don't

                         // raise error
   ```

5. Remove last element by using `pop()` method:

   ```
   x = set.pop()    //Remove and Return last element from the set

   print(x)         // print the last element of set
   ```

Dictionary

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

```
dict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

1. You can access the items of a dictionary by referring to its key name, inside square brackets:

```
x = dict["model"]              // Return the value of the key
```

2. You can change the value of a specific item by referring to its key name:

   ```
   dict["year"] = 2018
   ```

3. You can loop through a dictionary by using a `for` loop. When looping through a dictionary, the return value are the *keys* of the dictionary, but there are methods to return the *values* as well.

   ```
   for x in dict:
         print(x)         // Print all key names in the dictionary

   for x in dict:
         print(dict[x])   // Print all values of the dictionary

   for x, y in dict.items():
         print(x, y)       // Print both keys and value of the dictionary
   ```

4. Adding an item to the dictionary is done by using a new index key and assigning a value to it:

```
dict["color"] = "red"
      print(dict)             // Add new key and value to dictionary
```

5. The `pop()` method removes the item with specified key name:

```
dict.pop("model")       // Removes model key/value pair in dictionary
```