**Lists**Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

**List Items**

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.

Creating a list:  
myList = [1, "a", 2, "b", 3, "c"]

**List Methods:**

* **append()**: Adds an element to the end of the list.  
  Syntax: list.append(element)  
  myList.append(4)  
  print(myList) # Output: [1, "a", 2, "b", 3, "c", 4]
* **clear()**: Removes all elements from the list.  
  Syntax: list.clear()  
  myList.clear()  
  print(myList) # Output: []
* **copy()**: Returns a copy of the specified list.  
  Syntax: list.copy()  
  newList = myList.copy()  
  print(newList) # Output: [1, "a", 2, "b", 3, "c"]
* **count()**: Returns the number of elements with the specified value.  
  Syntax: list.count(value)  
  num = myList.count(1)  
  print(num) # Output: 1
* **extend()**: Adds elements of an iterable (e.g., list, tuple, set) to the end of the list.  
  Syntax: list.extend(iterable)  
  List2 = [4, "d"]  
  myList.extend(List2)  
  print(myList) # Output: [1, "a", 2, "b", 3, "c", 4, "d"]
* **index()**: Returns the position at the first occurrence of the specified value.  
  Syntax: list.index(element)  
  num = myList.index("b")  
  print(num) # Output: 3
* **insert()**: Inserts an element at the specified position.  
  Syntax: list.insert(position, element)  
  myList.insert(2, "Hello World")  
  print(myList) # Output: [1, "a", "Hello World", 2, "b", 3, "c"]
* **pop()**: Removes the element at the specified position.  
  Syntax: list.pop(position)  
  myList.pop(4)  
  print(myList) # Output: [1, "a", 2, "b", "c"]
* **remove()**: Removes the first occurrence of the element with the specified value.  
  Syntax: list.remove(element)  
  myList.remove("c")  
  print(myList) # Output: [1, "a", 2, "b", 3]
* **reverse()**: Reverses the order of the elements.  
  Syntax: list.reverse()  
  myList.reverse()  
  print(myList) # Output: ["c", 3, "b", 2, "a", 1]
* **sort()**: Sorts the list in ascending order by default.  
  Syntax: list.sort(reverse=True|False, key=myFunc)  
  newList = [3, 5, 2, 4, 1]  
  newList.sort()  
  print(newList) # Output: [1, 2, 3, 4, 5]
* **join()**: Joins all items in an iterable into one string with a specified separator.  
  Syntax: separator.join(iterable)  
  newList = ["a", "b", "c", "d", "e"]  
  print(" ".join(newList)) # Output: "a b c d e"

**Dictionaries**Dictionaries are used to store data values in key:value pairs.

**Dictionary Items**

Dictionary items are ordered, changeable, and do not allow duplicates.

Dictionary items are presented in key:value pairs, and can be referred to by using the key name.

Creating a dictionary:  
my\_dict = {"a": 1, "b": 2, "c": 3}

**Dictionary Methods:**

* **clear()**: Removes all elements.  
  Syntax: dictionary.clear()  
  my\_dict.clear()  
  print(my\_dict) # Output: {}
* **copy()**: Returns a copy of the dictionary.  
  Syntax: dictionary.copy()  
  new\_dict = my\_dict.copy()  
  print(new\_dict) # Output: {'a': 1, 'b': 2, 'c': 3}
* **fromkeys()**: Creates a new dictionary with specified keys and value.  
  Syntax: dict.fromkeys(keys, value)  
  keys = ("a", "b", "c")  
  new\_dict = dict.fromkeys(keys, 0)  
  print(new\_dict) # Output: {'a': 0, 'b': 0, 'c': 0}
* **get()**: Retrieves value of the specified key.  
  Syntax: dictionary.get(key)  
  value = my\_dict.get("a")  
  print(value) # Output: 1
* **items()**: Returns a list of tuples with key-value pairs.  
  Syntax: dictionary.items()  
  print(my\_dict.items()) # Output: [('a', 1), ('b', 2), ('c', 3)]
* **keys()**: Returns a list of dictionary keys.  
  Syntax: dictionary.keys()  
  print(my\_dict.keys()) # Output: ['a', 'b', 'c']
* **pop()**: Removes the specified key.  
  Syntax: dictionary.pop(key)  
  my\_dict.pop("b")  
  print(my\_dict) # Output: {'a': 1, 'c': 3}
* **popitem()**: Removes the last inserted key-value pair.  
  Syntax: dictionary.popitem()  
  my\_dict.popitem()  
  print(my\_dict) # Output: {'a': 1}
* **setdefault()**: Returns value of specified key; inserts key with specified value if key doesn't exist.  
  Syntax: dictionary.setdefault(key, value)  
  my\_dict.setdefault("d", 4)  
  print(my\_dict) # Output: {'a': 1, 'c': 3, 'd': 4}
* **update()**: Updates dictionary with specified key-value pairs.  
  Syntax: dictionary.update({key: value})  
  my\_dict.update({"e": 5})  
  print(my\_dict) # Output: {'a': 1, 'c': 3, 'd': 4, 'e': 5}
* **values()**: Returns a list of all values in the dictionary.  
  Syntax: dictionary.values()  
  print(my\_dict.values()) # Output: [1, 3, 4, 5]

**Sets**Sets are used to store multiple unique values in a single variable. A set is a collection which is unordered, unchangeable\*, and unindexed.

**Set Items**

Set items are unordered, unchangeable, and do not allow duplicate values.

Creating a set:  
my\_set = {1, 2, 3, 4}

**Set Methods:**

* **add()**: Adds an element to the set.  
  Syntax: set.add(element)  
  my\_set.add(5)  
  print(my\_set) # Output: {1, 2, 3, 4, 5}
* **clear()**: Removes all elements.  
  Syntax: set.clear()  
  my\_set.clear()  
  print(my\_set) # Output: set()
* **copy()**: Returns a copy of the set.  
  Syntax: set.copy()  
  new\_set = my\_set.copy()  
  print(new\_set) # Output: {1, 2, 3, 4}
* **difference()**: Returns a set with elements that are only in the first set.  
  Syntax: set.difference(set)  
  diff\_set = my\_set.difference({2, 3})  
  print(diff\_set) # Output: {1, 4}
* **difference\_update()**: Removes elements found in another set from the set.  
  Syntax: set.difference\_update(set)  
  my\_set.difference\_update({2, 3})  
  print(my\_set) # Output: {1, 4}
* **discard()**: Removes specified element.  
  Syntax: set.discard(element)  
  my\_set.discard(1)  
  print(my\_set) # Output: {2, 3, 4}
* **intersection()**: Returns a set of elements common to both sets.  
  Syntax: set.intersection(set)  
  common\_set = my\_set.intersection({3, 4, 5})  
  print(common\_set) # Output: {3, 4}
* **intersection\_update()**: Removes elements not common to both sets.  
  Syntax: set.intersection\_update(set)  
  my\_set.intersection\_update({3, 4, 5})  
  print(my\_set) # Output: {3, 4}

**Tuples**Tuples store multiple items in a single variable. Tuples are immutable, meaning they cannot be changed after creation.

Tuples are used to store multiple items in a single variable.

A tuple is a collection which is ordered and unchangeable.

**Tuple Items**

Tuple items are ordered, unchangeable, and allow duplicate values.

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

Creating a tuple:  
my\_tuple = (1, 2, 3, 4)

**Tuple Methods:**

* **count()**: Counts occurrences of a specified value.  
  Syntax: tuple.count(value)  
  count = my\_tuple.count(2)  
  print(count) # Output: 1
* **index()**: Returns the index of the specified value.  
  Syntax: tuple.index(value)  
  index = my\_tuple.index(3)  
  print(index) # Output: 2