**Lists**

* **List** is a collection which is ordered.
* Lists are mutable (changeable) .
* Allows duplicate members
* Brackets used to represent: []
* Lists are like arrays but can store different type of data and also can grow dynamically.
* list.append(elem) -- adds a single element to the end of the list. Common error: does not return the new list, just modifies the original.
* list.insert(index, elem) -- inserts the element at the given index, shifting elements to the right.
* list.extend(list2) adds the elements in list2 to the end of the list. Using + or += on a list is similar to using extend().
* list.index(elem) -- searches for the given element from the start of the list and returns its index. Throws a ValueError if the element does not appear (use "in" to check without a ValueError).
* list.remove(elem) -- searches for the first instance of the given element and removes it (throws ValueError if not present)
* list.sort() -- sorts the list in place (does not return it).
* list.reverse() -- reverses the list in place (does not return it)
* list.pop(index) -- removes and returns the element at the given index. Returns the rightmost element if index is omitted (roughly the opposite of append()).

**Tuples**

* Collection of items which is ordered.
* Tuples are immutable (unchangeable) .
* Brackets used to represent: ()
* Only difference between tuples and lists are that lists can be changed.
* Tuples are faster than lists as they are immutable.

count(x) Returns the number of items x

index(x) Returns the index of the first item that is equal to x

**Sets**

* Collection of Unordered and Unindexed items.
* Sets are mutable (changeable).
* Does not take duplicate Values.
* Sets are unordered, so you cannot be sure in which order the items will appear.
* Brackets used to represent: { }.

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

**Dictionaries**

* Key: Value Pair in Python
* 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.
* Brackets used to represent: {}.

clear()

Removes all the elements from the dictionary

copy()

Returns a copy of the dictionary

fromkeys()

Returns a dictionary with the specified keys and values

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