|  |  |
| --- | --- |
| Sets | |
| setname.add() | Takes 1 parameter: element.  # adds a given element to a set |
| setname.clear() | Takes no parameters.  # empties the given set |
| newset = oldset.copy() | Takes no parameters.  # returns a new set that is a copy of the old set |
| setname.difference(set) | Takes 1 parameter: set, which should be the set you want to check for differences in.  # returns a set that is the difference between two sets. (E.g. SetA minus SetB = only SetA). If we have equal sets, it will return the null set. |
| setname.difference\_update(set) | Takes 1 parameter: set, which should be the set you want to check for differences in.  # updates current set with the difference between two sets. (Note: it will not create a new set). |
| setname.discard(element) | Takes 1 parameter: element.  # removes the specified item from the set |
| setname.intersection(set1,set2,set3,….) | Takes 1 or more parameters which should all be the setnames.  # returns a set that contains the similarity between two or more sets |
| setname.intersection\_update(set1,set2,set3,…) | Takes 1 or more parameters which should all be the setnames.  # removes the items that is not present in both sets (or in all sets if the comparison is done between more than two sets). |
| setname.isdisjoint(set) | It takes 1 parameter: set, which should be a setname. You can also pass an iterable (list, tuple, dictionary, and string) to disjoint(). isdisjoint() method will automatically convert iterables to set and checks whether the sets are disjoint or not.  # returns True if none of the items are present in both sets. Otherwise, it returns False |
| setname.issubset(set) | Takes 1 parameter: set, which should be a setname.  # returns True is all items in the set exists in the specified set, otherwise it returns False. |
| setname.issuperset() | Takes 1 parameter: set, which should be a setname.  # returns True if all items in the specified set exists in the original set, otherwise it returns False. |
| setname.pop() | Takes no parameters.  # removes a random item from the set and returns the removed item. |
| setname.remove(element) | Takes 1 parameter: element.  # removes the specified element from the set and updates the set. It doesn’t return any value. If the element doesn’t exist in set, there will be a KeyError. |
| setname.symmetric\_difference(set) | Takes 1 parameter: set, the set to check for matches in.  # returns the symmetric difference of two sets. The symmetric difference of two or more sets is a set of elements that are in all sets but not in their intersections. |
| setname.symmetric\_difference\_update() | Takes 1 parameter: set, the set to check for matches in.  # finds the symmetric difference of two sets and updates the set. |
| setname.union(set1,set2,set3…) | Takes 1 or more parameters which should all be the setnames.  Create and return a new set with all elements that are in this set, or in any of the specified set arguments. |
| setname.update(set) | Takes 1 parameter: set, The iterable insert into the current set  # updates the current set, by adding items from another set (or any other iterable). If an item is present in both sets, only one appearance of this item will be present in the updated set. |

**SETS, TUPLES, DICTS, AND LISTS**

* A = (3) # Here, A is an int D = {‘colour’: ‘blue’, ‘clothing’: ‘shirt’, ‘brand’: ‘Ford’} # Here, D is a dictionary
* A = (3,) # Here, A is a tuple
* B = ([1,2]) # Here, B is a list
* A = 3,4,5 # Here, A is a tuple
* A = (1,2,3) # Here, A is a tuple
* B = ([1,2],) # Here, B is a tuple

**Converting a tuple to a dictionary:** listoftuples = [(‘hello’, 7), (‘hi’, 10), (‘there’, 45), (‘at’, 23), (‘this’, 77)]

Dictionary = dict(listoftuples)

Print(Dictionary) = {'this': 77, 'there': 45, 'hi': 10, 'at': 23, 'Hello': 7}

**Some notes on sets:**

* Cant have duplicates
* Are unordered
* Mutable (this means you can change the values/elements of an existing set)
* A hash lookup is used to find an element in a set. This makes the ‘in’ operator (e.g. for x in range…) more efficient than lists.
* Sets can only contain hashable items. (E.g. if you try: set(([1],[2])) it will cause a TypeError).

|  |  |
| --- | --- |
| Lists | |
| listname.append(element) | Takes 1 parameter: element.  # adds a given element to a list |
| listname.clear() | Takes no parameters.  # empties the given list |
| newlist = oldlist.copy() | Takes no parameters.  # returns a new list that is a copy of the old list |
| listname.count(element) | Takes 1 parameter: element.  # returns the number of times the specified element appears in the list. |
| listname.extend(iterable) | Takes an iterable such as list, tuple, string, etc as its parameter.  # adds the specified list elements (or any iterable) to the end of the current list. |
| listname.index(element) | Takes 1 parameter: element  # returns the index of a given element in the list. If element is not in the list, ValueError will be raised. |
| listname.insert(position, element) | Takes two parameters:   * index - the index where the element needs to be inserted * element - this is the element to be inserted in the list   # inserts the specified value/element at the specified position. |
| listname.pop(1) | Has one parameter: the index number  # removes the item at the given index 1 from the list and returns the removed item. |
| listname.remove(element) | Has 1 parameter: element.  # takes a single element as an argument and removes it from the list. |
| listname.reverse() | Takes no parameters.  # reverses the order of a list. (e.g. list = 1234 🡪 list = 4321) |
| listname.sort() | Takes no parameters.  # sorts the list ascending by default. You can also do descending by doing listname.sort(reverse=True) |

**Some notes about lists:**

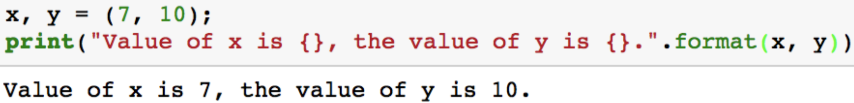
* EmptyList = []
* **Lists are mutable** (this means u **can** update/change the items in a list once it is created!)
* Has a defined order and will not change
* Can add new items to list. The new items get placed at end of list. (Some list methods can change the order however such as the following image:
* Allows duplicate values
* Lists are indexed. (First item in list has index[0], second item in list has index[1], etc.)

|  |  |
| --- | --- |
| Tuples | |
| t.count (element) | # returns the number of times a specified element/value appears in the tuple. |
| t.index () | # returns the index of the given element in the tuple. · If the element is not found, a ValueError exception is raised. |

**Graphical user interface, text, application

Description automatically generatedSome notes on tuples:** EmptyTuple = () **Or** EmptyTuple = tuple()

**Tuples are immutable** (this means u **cannot** update/change the items in a tuple once it is created!). It is possible however to take portions of an existing tuple to create a new tuple. (See image on the right as an example of how to update/change content of a tuple)

* **Printing the first index** at which a value occurs…. print(listname.index(‘value’))
* **Printing the number of times a value occurs** in a tuple…. print(listname.count(‘value’))
* Tuples are iterables (this means it **can be used in for loops)**
* Graphical user interface, text, application, chat or text message

  Description automatically generated**Tuple unpacking**? See image on the right. It is useful to create a tuple and then one by one you can input the values in order using the (.format)
* **Enumerate** function returns a tuple containing a count for every iteration (from start which by default is 0) and the values obtained from iterating over a sequence. See image on the right.

Text

Description automatically generatedTable

Description automatically generated

Table

Description automatically generatedText

Description automatically generated

Table

Description automatically generated

|  |  |
| --- | --- |
| Dicts/Dictionaries | |
| dictname.fromkeys(sequence[, value]) | Takes two parameters:   * sequence - sequence of elements which is to be used as keys for the new dictionary * value (Optional) - value which is set to each element of the dictionary   # returns a new dictionary with the given sequence of elements as the keys of the dictionary. If the value argument is set, each element of the newly created dictionary is set to the provided value. |
| dictname.clear() | Takes no parameters.  # deletes all the key-value pairs present in the dict and returns an empty dict. |
| newdict = olddict.copy() | # a new dictionary is created which is filled with a copy of the references from the original dictionary. If you set dict2 = dict1, you are making them refer to the same exact dict object, so when you mutate it, all references to it keep referring to the object in its current state. |
| dictname.get(key) | Takes one parameter: Key which should be the keyname.  # returns the value for the specified key if key is in the dictionary.  # returns None if the key is not found and value is not specified.  # returns value if the key is not found and value is specified. |
| dictname.items () | Takes no parameters.  # returns a view object. The view object contains the key-value pairs of the dictionary, as tuples in a list. |
| dictname.keys() | Takes no parameters.  # returns a view object. The view object contains the keys of the dictionary, as a list. The view object will reflect any changes done to the dictionary |
| dictname.pop(key[, default]) | # It takes two parameters:   * key - key which is to be searched for removal * default - value which is to be returned when the key is not in the dictionary   # removes and returns an element from a dictionary having the given key. |
| dictname.popitem () | Takes no parameters  # removes the item that was last inserted into the dictionary. In versions before 3.7, the popitem() method removes a random item |
| dictname.setdefault(key, value) | # It takes two parameters:   * Key – the keyname of the item you want to return the value from. * Value – This parameter is optional. If the key exists, this parameter has no effect. If the key does not exist, this value becomes the key’s value default value None. |
| dictname.update(iterable) | Parameters:   * method takes either a dictionary or an iterable object of key/value pairs (generally tuples). * If update() is called without passing parameters, the dictionary remains unchanged.   # updates the dictionary with elements from a dictionary object or an iterable object of key/value pairs.  # It doesn't return any value (returns None). |
| dictname.values() | Doesn’t take any parameters  # returns a view object that displays a list of all values in a given dictionary. |

Some notes on Dicts: Emptydictionary = {} **Or** Emptydictionary = dict()

* Are **unordered**
* **Mutable** (which means its values can change once a dictionary is created.
* Key value pairs are **separated with a comma** (E.g. key, value).
* **A colon also separates** the key from its value (E.g. key: value).