Data-structures Cheat Sheet

Reference: https://docs.python.org/3/tutorial/datastructures.html

Prepared By: Shehab Abdel-Salam

Lists

1. `len(list)`: Returns the number of items in the list.

```
my_list = [1, 2, 3]
print(len(my_list))  # Output: 3
```

2. `list.append(x)`: Adds an item to the end of the list.

```
my_list.append(4)
print(my_list)  # Output: [1, 2, 3, 4]
```

3. `list.pop([i])`: Removes and returns the item at the given position in the list. If no index is specified, removes and returns the last item in the list.

```
my_list.pop()  # Output: 4
print(my_list)  # Output: [1, 2, 3]
```

4. `list.sort()`: Sorts the list in ascending order.

```
my_list.sort()
print(my_list)  # Output: [1, 2, 3]
```

5. `list.extend(iterable)`: Extends the list by appending all the items from the iterable.

```
my_list.extend([4, 5])
print(my_list)  # Output: [1, 2, 3, 4, 5]
```

6. `list.insert(i, x)`: Inserts an item at a given position.

```
my_list.insert(1, 'a')
print(my_list)  # Output: [1, 'a', 2, 3, 4, 5]
```

Dictionaries

1. `len(dict)`: Returns the number of items in the dictionary.

```
my_dict = {'a': 1, 'b': 2}
print(len(my_dict))  # Output: 2
```

2. `dict.get(key[, default])`: Returns the value for key if key is in the dictionary, else default.

```
print(my_dict.get('a'))  # Output: 1
print(my_dict.get('c', 3))  # Output: 3
```

3. `dict.keys()`: Returns a new view of the dictionary's keys.

```
print(my_dict.keys())  # Output: dict_keys(['a', 'b'])
```

4. `dict.values()`: Returns a new view of the dictionary's values.

```
print(my_dict.values())  # Output: dict_values([1, 2])
```

5. `dict.items()`: Returns a new view of the dictionary's items (key, value pairs).

```
print(my_dict.items())  # Output: dict_items([('a', 1),
('b', 2)])
```

6. `dict.update([other])`: Updates the dictionary with the key-value pairs from other, overwriting existing keys.

```
my_dict.update({'b': 3, 'c': 4})
print(my_dict)  # Output: {'a': 1, 'b': 3, 'c': 4}
```

7. `del dict[key]`: Removes the item with the specified key.

```
del my_dict['a']
print(my_dict)  # Output: {'b': 3, 'c': 4}
```

Tuples

1. `len(tuple)`: Returns the number of items in the tuple.

```
my_tuple = (1, 2, 3)
print(len(my_tuple))  # Output: 3
```

2. `tuple.count(x)`: Returns the number of times x appears in the tuple.

```
print(my_tuple.count(2))  # Output: 1
```

3. `tuple.index(x[, start[, end]])`: Returns the index of the first occurrence of x in the tuple.

```
print(my_tuple.index(2))  # Output: 1
```

Sets

1. `len(set)`: Returns the number of items in the set.

```
my_set = {1, 2, 3}
print(len(my_set))  # Output: 3
```

2. `set.add(x)`: Adds an element to the set.

```
my_set.add(4)
print(my_set)  # Output: {1, 2, 3, 4}
```

3. `set.update(iterable)`: Updates the set, adding elements from all the iterables.

```
my_set.update([5, 6])
print(my_set)  # Output: {1, 2, 3, 4, 5, 6}
```

4. `set.remove(x)`: Removes an element from the set. Raises `KeyError` if the element is not present.

```
my_set.remove(6)
print(my_set)  # Output: {1, 2, 3, 4, 5}
```

5. `set.union(*others)`: Returns a new set with elements from the set and all others.

```
other_set = {5, 6, 7}
print(my_set.union(other_set))  # Output: {1, 2, 3, 4,
5, 6, 7}
```

6. `set.intersection(*others)`: Returns a new set with elements common to the set and all others.

```
another_set = {4, 5, 6}
print(my_set.intersection(another_set))  # Output: {4,
5}
```