

# Cheat Sheet: Python Data Structures Part-2

## Dictionaries

| Package/Method        | Description   | Code Example  |
|-----------------------|---|---|
| Creating a Dictionary | A dictionary is a built-in data type that represents a collection of key-value pairs. Dictionaries are enclosed in curly braces {}.   | Example:<br><br><pre>dict_name = {} #Creates an empty dictionary person = { "name": "John", "age": 30, "city": "New York"}</pre>  |
| Accessing Values      | You can access the values in a dictionary using their corresponding keys.   | Syntax:<br><br><pre>Value = dict_name["key_name"]</pre><br>Example:<br><br><pre>name = person["name"] age = person["age"]</pre>   |
| Add or modify         | Inserts a new key-value pair into the dictionary. If the key already exists, the value will be updated; otherwise, a new entry is created.  | Syntax:<br><br><pre>dict_name[key] = value</pre><br>Example:<br><br><pre>person["Country"] = "USA" # A new entry will be created. person["city"] = "Chicago" # Update the existing value for the same key</pre> |
| del                   | Removes the specified key-value pair from the dictionary. Raises a <code>KeyError</code> if the key does not exist.   | Syntax:<br><br><pre>del dict_name[key]</pre><br>Example:<br><br><pre>del person["Country"]</pre>  |
| update()              | The <code>update()</code> method merges the provided dictionary into the existing dictionary, adding or updating key-value pairs.   | Syntax:<br><br><pre>dict_name.update({key: value})</pre><br>Example:<br><br><pre>person.update({"Profession": "Doctor"})</pre>  |
| clear()               | The <code>clear()</code> method empties the dictionary, removing all key-value pairs within it. After this operation, the dictionary is still accessible and can be used further. | Syntax:<br><br><pre>dict_name.clear()</pre><br>Example:<br><br><pre>grades.clear()</pre>  |
| key existence         | You can check for the existence of a key in a dictionary using the <code>in</code> keyword  | Example:<br><br><pre>if "name" in person:     print("Name exists in the dictionary.")</pre>   |
| copy()                | Creates a shallow copy of the dictionary. The new dictionary contains the same key-value pairs as the original, but they remain distinct objects in memory.                       | Syntax:<br><br><pre>new_dict = dict_name.copy()</pre><br>Example:<br><br><pre>new_person = person.copy() new_person = dict(person) # another way to create a copy of dictionary</pre>                           |
| keys()                | Retrieves all keys from the dictionary and converts them into a list. Useful for iterating or processing keys using list methods.   | Syntax:<br><br><pre>keys_list = list(dict_name.keys())</pre><br>Example:<br><br><pre>person_keys = list(person.keys())</pre>  |
| values()              | Extracts all values from the dictionary and converts them into a list. This list can be used for further processing or  | Syntax:   |

|         |  |   |
|---------|--|---|
|         | analysis.  | <pre>values_list = list(dict_name.values())</pre> <p>Example:</p> <pre>person_values = list(person.values())</pre>    |
| items() | Retrieves all key-value pairs as tuples and converts them into a list of tuples. Each tuple consists of a key and its corresponding value. | <p>Syntax:</p> <pre>items_list = list(dict_name.items())</pre> <p>Example:</p> <pre>info = list(person.items())</pre> |

## Sets

| Package/Method | Description   | Code Example   |
|----------------|---|--|
| add()          | Elements can be added to a set using the `add()` method. Duplicates are automatically removed, as sets only store unique values.  | <p>Syntax:</p> <pre>set_name.add(element)</pre> <p>Example:</p> <pre>fruits.add("mango")</pre>                                   |
| clear()        | The `clear()` method removes all elements from the set, resulting in an empty set. It updates the set in-place.   | <p>Syntax:</p> <pre>set_name.clear()</pre> <p>Example:</p> <pre>fruits.clear()</pre>   |
| copy()         | The `copy()` method creates a shallow copy of the set. Any modifications to the copy won't affect the original set.   | <p>Syntax:</p> <pre>new_set = set_name.copy()</pre> <p>Example:</p> <pre>new_fruits = fruits.copy()</pre>                        |
| Defining Sets  | A set is an unordered collection of unique elements. Sets are enclosed in curly braces `{}`. They are useful for storing distinct values and performing set operations.                 | <p>Example:</p> <pre>empty_set = set() #Creating an Empty Set fruits = {"apple", "banana", "orange"}</pre>                       |
| discard()      | Use the `discard()` method to remove a specific element from the set. Ignores if the element is not found.  | <p>Syntax:</p> <pre>set_name.discard(element)</pre> <p>Example:</p> <pre>fruits.discard("apple")</pre>                           |
| issubset()     | The `issubset()` method checks if the current set is a subset of another set. It returns True if all elements of the current set are present in the other set, otherwise False.         | <p>Syntax:</p> <pre>is_subset = set1.issubset(set2)</pre> <p>Example:</p> <pre>is_subset = fruits.issubset(colors)</pre>         |
| issuperset()   | The `issuperset()` method checks if the current set is a superset of another set. It returns True if all elements of the other set are present in the current set, otherwise False.     | <p>Syntax:</p> <pre>is_superset = set1.issuperset(set2)</pre> <p>Example:</p> <pre>is_superset = colors.issuperset(fruits)</pre> |
| pop()          | The `pop()` method removes and returns an arbitrary element from the set. It raises a `KeyError` if the set is empty. Use this method to remove elements when the order doesn't matter. | <p>Syntax:</p> <pre>removed_element = set_name.pop()</pre> <p>Example:</p> <pre>removed_fruit = fruits.pop()</pre>               |

|                |   |   |
|----------------|---|---|
| remove()       | Use the `remove()` method to remove a specific element from the set. Raises a `KeyError` if the element is not found. | <p>Syntax:</p> <pre>set_name.remove(element)</pre> <p>Example:</p> <pre>fruits.remove("banana")</pre>   |
| Set Operations | Perform various operations on sets: `union`, `intersection`, `difference`, `symmetric difference`.                    | <p>Syntax:</p> <pre>union_set = set1.union(set2) intersection_set = set1.intersection(set2) difference_set = set1.difference(set2) sym_diff_set = set1.symmetric_difference(set2)</pre> <p>Example:</p> <pre>combined = fruits.union(colors) common = fruits.intersection(colors) unique_to_fruits = fruits.difference(colors) sym_diff = fruits.symmetric_difference(colors)</pre> |
| update()       | The `update()` method adds elements from another iterable into the set. It maintains the uniqueness of elements.      | <p>Syntax:</p> <pre>set_name.update(iterable)</pre> <p>Example:</p> <pre>fruits.update(["kiwi", "grape"])</pre>   |



**Skills** Network