

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 {}.	<div>Example:</div> <pre>1. 1 2. 2 1. dict_name = {} #Creates an empty dictionary 2. person = { "name": "John", "age": 30, "city": "New York"}</pre> <div>Copied!</div> <div>Syntax:</div> <pre>1. 1 1. Value = dict_name["key_name"]</pre> <div>Copied!</div>
Accessing Values	You can access the values in a dictionary using their corresponding keys.	<div>Example:</div> <pre>1. 1 2. 2 1. name = person["name"] 2. age = person["age"]</pre> <div>Copied!</div> <div>Syntax:</div> <pre>1. 1 1. dict_name[key] = value</pre> <div>Copied!</div>
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.	<div>Example:</div> <pre>1. 1 2. 2 1. person["Country"] = "USA" # A new entry will be created. 2. person["city"] = "Chicago" # Update the existing value for the same key</pre> <div>Copied!</div> <div>Syntax:</div> <pre>1. 1 1. del dict_name[key]</pre> <div>Copied!</div>
del	Removes the specified key-value pair from the dictionary. Raises a <code>KeyError</code> if the key does not exist.	<div>Example:</div> <pre>1. 1 1. del person["Country"]</pre> <div>Copied!</div> <div>Syntax:</div> <pre>1. 1 1. dict_name.update({key: value})</pre> <div>Copied!</div>
update()	The <code>update()</code> method merges the provided dictionary into the existing dictionary, adding or updating key-value pairs.	<div>Example:</div> <pre>1. 1 1. person.update({"Profession": "Doctor"})</pre> <div>Copied!</div> <div>Syntax:</div> <pre>1. 1 1. dict_name.clear()</pre> <div>Copied!</div>
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.	<div>Syntax:</div> <pre>1. 1 1. dict_name.clear()</pre> <div>Copied!</div>

		<div>Example:</div> <div><div>1. 1</div><div>1. grades.clear()</div></div> <div>Copied!</div>
key existence	You can check for the existence of a key in a dictionary using the <code>in</code> keyword	<div>Example:</div> <div><div>1. 1</div><div>2. 2</div><div>1. if "name" in person: 2. print("Name exists in the dictionary.")</div></div> <div>Copied!</div> <div>Syntax:</div> <div><div>1. 1</div><div>1. new_dict = dict_name.copy()</div></div> <div>Copied!</div>
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.	<div>Example:</div> <div><div>1. 1</div><div>2. 2</div><div>1. new_person = person.copy() 2. new_person = dict(person) # another way to create a copy of dictionary</div></div> <div>Copied!</div> <div>Syntax:</div> <div><div>1. 1</div><div>1. keys_list = list(dict_name.keys())</div></div> <div>Copied!</div>
keys()	Retrieves all keys from the dictionary and converts them into a list. Useful for iterating or processing keys using list methods.	<div>Example:</div> <div><div>1. 1</div><div>1. person_keys = list(person.keys())</div></div> <div>Copied!</div> <div>Syntax:</div> <div><div>1. 1</div><div>1. values_list = list(dict_name.values())</div></div> <div>Copied!</div>
values()	Extracts all values from the dictionary and converts them into a list. This list can be used for further processing or analysis.	<div>Example:</div> <div><div>1. 1</div><div>1. person_values = list(person.values())</div></div> <div>Copied!</div> <div>Syntax:</div> <div><div>1. 1</div><div>1. items_list = list(dict_name.items())</div></div> <div>Copied!</div>
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.	<div>Example:</div> <div><div>1. 1</div><div>1. info = list(person.items())</div></div> <div>Copied!</div>

Sets

Package/Method	Description	Code Example
add()	Elements can be added to a set using the <code>'add()'</code> method. Duplicates are automatically removed, as sets only store unique values.	<div>Syntax:</div> <div><div>1. 1</div><div>1. set_name.add(element)</div></div> <div>Copied!</div>

		<p>Example:</p> <pre>1. 1 1. fruits.add("mango")</pre> <p>Copied!</p> <p>Syntax:</p> <pre>1. 1 1. set_name.clear()</pre> <p>Copied!</p>
clear()	The <code>'clear()'</code> method removes all elements from the set, resulting in an empty set. It updates the set in-place.	<p>Example:</p> <pre>1. 1 1. fruits.clear()</pre> <p>Copied!</p> <p>Syntax:</p> <pre>1. 1 1. new_set = set_name.copy()</pre> <p>Copied!</p>
copy()	The <code>'copy()'</code> method creates a shallow copy of the set. Any modifications to the copy won't affect the original set.	<p>Example:</p> <pre>1. 1 1. new_fruits = fruits.copy()</pre> <p>Copied!</p> <p>Example:</p> <pre>1. 1 2. 2 1. empty_set = set() #Creating an Empty 2. Set fruits = {"apple", "banana", "orange"}</pre> <p>Copied!</p> <p>Syntax:</p> <pre>1. 1 1. set_name.discard(element)</pre> <p>Copied!</p>
Defining Sets	A set is an unordered collection of unique elements. Sets are enclosed in curly braces <code>'{'</code> . They are useful for storing distinct values and performing set operations.	<p>Syntax:</p> <pre>1. 1 1. set_name.discard(element)</pre> <p>Copied!</p>
discard()	Use the <code>'discard()'</code> method to remove a specific element from the set. Ignores if the element is not found.	<p>Example:</p> <pre>1. 1 1. fruits.discard("apple")</pre> <p>Copied!</p> <p>Syntax:</p> <pre>1. 1 1. is_subset = set1.issubset(set2)</pre> <p>Copied!</p>
issubset()	The <code>'issubset()'</code> 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>Example:</p> <pre>1. 1 1. is_subset = fruits.issubset(colors)</pre> <p>Copied!</p> <p>Syntax:</p> <pre>is_superset = set1.issuperset(set2)</pre>
issuperset()	The <code>'issuperset()'</code> 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>Example:</p> <pre>1. 1 1. is_superset = colors.issuperset(fruits)</pre> <p>Copied!</p> <p>Syntax:</p> <pre>1. 1 1. removed_element = set_name.pop()</pre>
pop()	The <code>'pop()'</code> method removes and returns an arbitrary element from the set. It raises a <code>'KeyError'</code> if the set is empty. Use this method to remove elements when the order doesn't matter.	

remove()

Use the `'remove()'` method to remove a specific element from the set. Raises a `'KeyError'` if the element is not found.

Set Operations

Perform various operations on sets: `'union'`, `'intersection'`, `'difference'`, `'symmetric difference'`.

update()

The `'update()'` method adds elements from another iterable into the set. It maintains the uniqueness of elements.



Skills Network

© IBM Corporation. All rights reserved.

Copied!

Example:

```
1. 1
1. removed_fruit = fruits.pop()
```

Copied!

Syntax:

```
1. 1
1. set_name.remove(element)
```

Copied!

Example:

```
1. 1
1. fruits.remove("banana")
```

Copied!

Syntax:

```
1. 1
2. 2
3. 3
4. 4

1. union_set = set1.union(set2)
2. intersection_set = set1.intersection(set2)
3. difference_set = set1.difference(set2)
4. sym_diff_set = set1.symmetric_difference(set2)
```

Copied!

Example:

```
1. 1
2. 2
3. 3
4. 4

1. combined = fruits.union(colors)
2. common = fruits.intersection(colors)
3. unique_to_fruits = fruits.difference(colors)
4. sym_diff = fruits.symmetric_difference(colors)
```

Copied!

Syntax:

```
1. 1
1. set_name.update(iterable)
```

Copied!

Example:

```
1. 1
1. fruits.update(["kiwi", "grape"])
```

Copied!