Cheatsheets / **Learn Python 3**

Dictionaries

Python dictionaries

A python dictionary is an unordered collection of items. It contains data as a set of key: value pairs.

```
my_dictionary = {1: "L.A. Lakers", 2: "Houston Rockets"}
```

Syntax of the Python dictionary

The syntax for a Python dictionary begins with the left curly brace (`{`), ends with the right curly brace (`}`), and contains zero or more `key : value` items separated by commas (`,`). The `key` is separated from the `value` by a colon (`:`).

```
roaster = {"q1": "Ashley", "q2": "Dolly"}
```

Dictionary value types

Python allows the *values* in a dictionary to be any type – string, integer, a list, another dictionary, boolean, etc. However, *keys* must always be an immutable data type, such as strings, numbers, or tuples.

In the example code block, you can see that the keys are strings or numbers (int or float). The values, on the other hand, are many varied data types.

```
dictionary = {
  1: 'hello',
  'two': True,
  '3': [1, 2, 3],
  'Four': {'fun': 'addition'},
  5.0: 5.5
}
```

Accessing and writing data in a Python dictionary

Values in a Python dictionary can be accessed by placing the key within square brackets next to the dictionary. Values can be written by placing key within square brackets next to the dictionary and using the assignment operator (`=`). If the key already exists, the old value will be overwritten. Attempting to access a value with a key that does not exist will cause a `KeyError` .

To illustrate this review card, the second line of the example code block shows the way to access the value using the key `"song"` . The third line of the code block overwrites the value that corresponds to the key `"song"` .

```
my_dictionary = {"song": "Estranged", "artist": "Guns N' Roses"}
print(my_dictionary["song"])
my_dictionary["song"] = "Paradise City"
```

Merging Dictionaries with the `.update()` Method in Python

Given two dictionaries that need to be combined, Python makes this easy with the `.update()` function.

For `dict1.update(dict2)` , the key-value pairs of `dict2` will be written into the `dict1` dictionary.

For keys in *both* `dict1` and `dict2` , the value in `dict1` will be overwritten by the corresponding value in `dict2` .

```
dict1 = {'color': 'blue', 'shape': 'circle'}
dict2 = {'color': 'red', 'number': 42}

dict1.update(dict2)

# dict1 is now {'color': 'red', 'shape': 'circle', 'number': 42}
```

get() Method for Dictionary

Python provides a `.get()` method to access a `dictionary` value if it exists. This method takes the `key` as the first argument and an optional default value as the second argument, and it returns the value for the specified `key` if `key` is in the dictionary. If the second argument is not specified and `key` is not found then `None` is returned.

```
# without default
{"name": "Victor"}.get("name")
# returns "Victor"

{"name": "Victor"}.get("nickname")
# returns None

# with default
{"name": "Victor"}.get("nickname", "nickname is not a key")
# returns "nickname is not a key"
```

The `.pop()` Method for Dictionaries in Python

Python dictionaries can remove key-value pairs with the `.pop()` method. The method takes a key as an argument and removes it from the dictionary. At the same time, it also returns the value that it removes from the dictionary.

```
famous_museums = {'Washington': 'Smithsonian Institution', 'Paris': 'Le
Louvre', 'Athens': 'The Acropolis Museum'}
famous_museums.pop('Athens')
print(famous_museums) # {'Washington': 'Smithsonian Institution', 'Paris':
'Le Louvre'}
```

Dictionary accession methods

When trying to look at the information in a Python dictionary, there are multiple methods that access the dictionary and return lists of its contents.

`.keys()` returns the keys (the first object in the key-value pair), `.values()` returns the values (the second object in the key-value pair), and `.items()` returns both the keys and the values as a tuple.

```
ex_dict = {"a": "anteater", "b": "bumblebee", "c": "cheetah"}
```

```
ex_dict.keys()
# ["a","b","c"]

ex_dict.values()
# ["anteater", "bumblebee", "cheetah"]

ex_dict.items()
# [("a","anteater"),("b","bumblebee"),("c","cheetah")]
```

Related Courses

Course

Learn Python 3

Learn the latest and greatest version of the most popular programr