Dictionaries: Takeaways

by Dataquest Labs, Inc. - All rights reserved © 2018

Syntax

DICTIONARIES

- Creating an empty dictionary:

```
scores = {}
```

- Adding key-value pairs to a dictionary:

```
scores["Tom"] = 70

scores["Sue"] = 80
```

- Selecting a value from a dictionary using the key:

```
print(scores["Tom"]) # Returns 70
```

- Modifying an existing value in a dictionary:

```
scores["Tom"] = 90

scores["Tom"] = scores["Tom"] + 5
```

Concepts

- A dictionary is a data structure that contains key-value pairs. While lists are keyed by integer index values (0 to n-1, where n is the length of the list), we can key a dictionary by any arbitrary unique value:

```
dict_ex = {}

dict_ex[50] = "Hey!"

dict_ex["A"] = 500
```

- Unlike lists, dictionaries have no inherent order to the values. Dictionaries are useful whenever we want the key to be something unique that we care about (e.g. keys: book titles, values: number of pages).

- One powerful use case for dictionaries is counting unique values. Let's say we want to understand the number of times each value in the following list occurs:

```
pantry = ["apple", "orange", "grape", "apple", "orange"]
```

- To count the unique values, we can create an empty dictionary that's keyed by the unique value, use a for loop to iterate over the list, and then increment the dictionary for every instance of a given key.

```
for item in pantry:
    if item in pantry_counts:
        pantry_counts[item] = pantry_counts[item] + 1
    else:
        pantry_counts[item] = 1
```

Resources

- Python Documentation: Dictionaries

Takeaways by Dataquest Labs, Inc. - All rights reserved © 2018