Dictionaries

SHARP SIGHT

What you'll learn

- What dictionaries are

- Why we use dictionaries

- Working with dictionaries
  - creating dictionaries
  - adding/removing items

- A few important dictionary methods

The basic structure of dictionaries

What are dictionaries?

- Dictionaries hold key-value pairs

- Dictionaries are similar to lists
  - dictionaries hold values, much like lists
  - but, dictionaries use "keys" instead of numeric indexes

- Other names for dictionaries:
  - dict, hash, hash table

- We use dictionaries to create a relationship between values and keys
  - we want to store a value and we want a ‘key’ that can access it

How to create dictionaries

- Dictionaries are created using curly brackets, **{ }**

- Key-value pairs are put inside the brackets, separated by a colon
  - these pairs are sometimes called items

- Note: you can also create an empty dictionary

```
parking_spots = { 101:'Ferrari' }
```

key value

Dictionary values are like the values of a list

```
parking_spots = { 101:'Ferrari' }
```

Much like a list, the values of a dictionary are just "items" that we want to store

Dictionary keys are like the indexes of a list, but more flexible

```
parking_spots = { 101:'Ferrari' }
```

The keys of a dictionary are very similar to the integer indexes of a list

… but there are important differences !

Dictionary keys can be any immutable object

```
parking_spots = { 101:'Ferrari' }
```

The keys of a dictionary can be any "immutable object

So the keys of a dictionary can be integers, strings, or tuples

ADD MULTIPLE ITEMS TO A DICTIONARY WITH COMMAS

```
parking_spots = { 101:'Ferrari', 102:'Bugatti'}
```

Here, we've separated two key-value pairs with a comma

We can add many key-value pairs this way

There are some restrictions on keys

- No duplicate keys
  - dictionary keys must be unique
  - i.e., no two values can have the same key

- Keys can be any immutable type
  - string
  - integer
  - boolean
  - float
  - tuple
  - (and others)

A simple example of a dictionary

DICTIONARY EXAMPLE: PARKING SPOTS

```
parking_spots = { 101:'Ferrari'
                 ,102:'Bugatti'
                 ,103:'Porsche'
                 ,104:'Mclaren'
                }
```

Here, we're creating a dictionary with 4 key-value pairs

Dictionary Example: parking spots

This is a visual representation
of the resulting dictionary

```
parking_spots = { 101:'Ferrari'
                 ,102:'Bugatti'
                 ,103:'Porsche'
                 ,104:'Mclaren'
                }
```

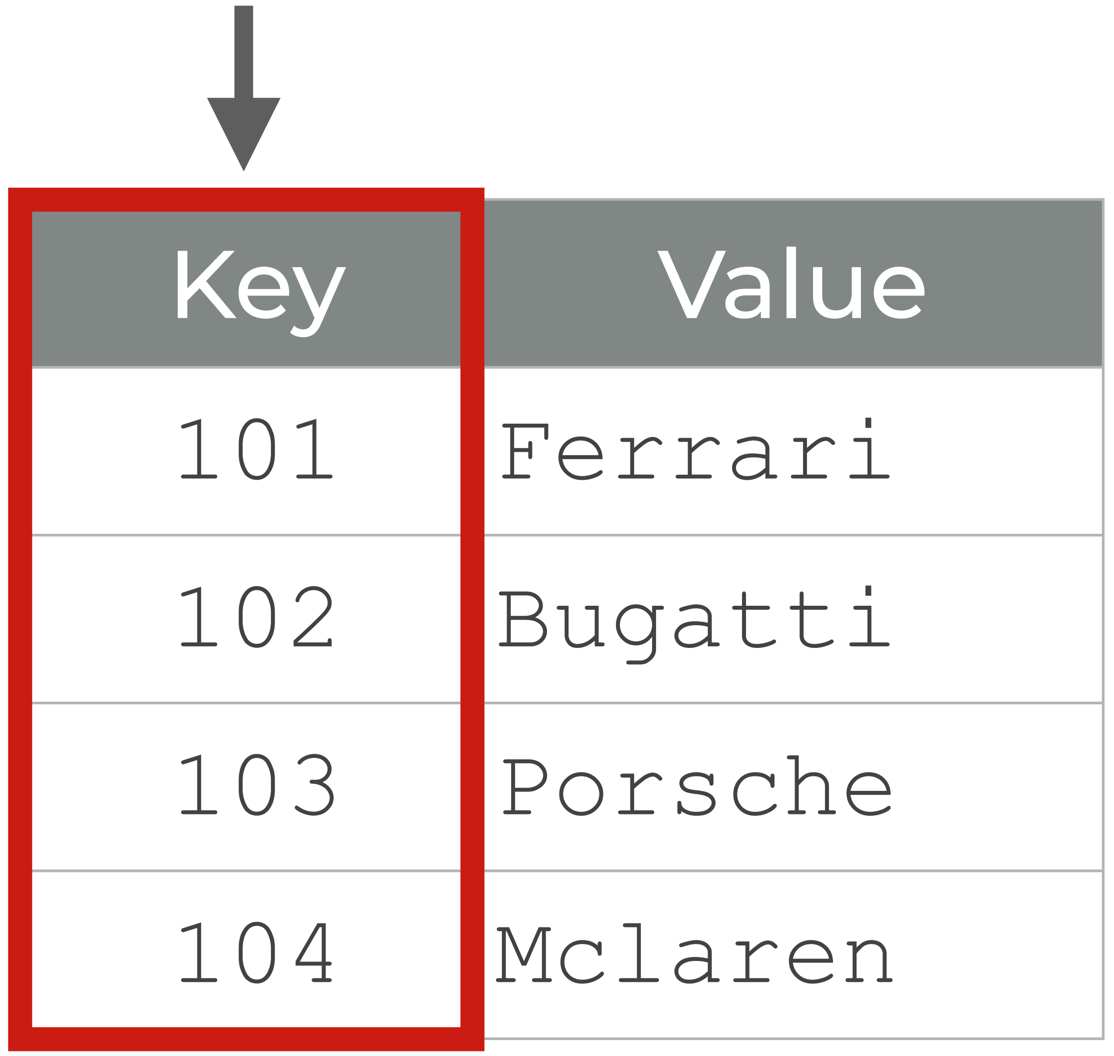
Key	Value
101	Ferrari
102	Bugatti
103	Porsche
104	Mclaren

Dictionary Example: parking spots

There are 4 unique keys

... these are the unique "parking spots" of our data structure

```
parking_spots = { 101:'Ferrari'
                 ,102:'Bugatti'
                 ,103:'Porsche'
                 ,104:'Mclaren'
                }
```



Key	Value
101	Ferrari
102	Bugatti
103	Porsche
104	Mclaren

Dictionary Example: parking spots

Every key has an associated value
... i.e., for every parking spot, there is
a car "parked" in that spot

```
parking_spots = { 101:'Ferrari'
                 ,102:'Bugatti'
                 ,103:'Porsche'
                 ,104:'Mclaren'
                }
```

Key	Value
101	Ferrari
102	Bugatti
103	Porsche
104	Mclaren

GETTING VALUES FROM A DICTIONARY

Dictionary indexing basics

- Use "bracket notation" to retrieve values
  - This is very similar to how we get values from other structures

- Dictionaries are indexed by *key*
  - This is different from how we index strings, lists, etc

- To get a specific value, you need to provide the key

Syntax: how to get a single item from a dictionary

To retrieve an item from a dictionary, first type the name of the dict, followed by brackets

```
your_dictionary[key-to-retrieve]
```

Inside of the brackets, type the *key* associated with the value you want to get

Example: how to get a single item from a dictionary

Specify the dictionary name, and the key associated with the value you want to get

```
parking_spots[103]
```

Key	Value
101	Ferrari
102	Bugatti
103	Porsche
104	Mclaren

Adding and removing values from a dictionary

Adding and removing values

- Add values using "bracket" notation
  - Provide the key in brackets
  - Use the equal sign to assign a value

- Delete values using the `del` operator

Delete items with the `del` operator

```
parking_spots = { 101:'Ferrari'
                 ,102:'Bugatti'
                 ,103:'Porsche'
                 ,104:'Mclaren'
                }

del parking_spots[103]
```

Key	Value
101	Ferrari
102	Bugatti
104	Mclaren

This code deletes the record associated with key `103`

Add an item by providing a new key in brackets, and a value

```
parking_spots = { 101:'Ferrari'
                 ,102:'Bugatti'
                 ,103:'Porsche'
                 ,104:'Mclaren'
                }

parking_spots[105] = 'Tesla'
```

Key	Value
101	Ferrari
102	Bugatti
103	Porsche
104	Mclaren
105	Tesla

This code adds the new value 'Tesla' which is associated with key 105

Some important dictionary methods

Important Dictionary Methods

- Dictionaries have several useful methods for retrieving data and dictionary-manipulation
  - Use these using "dot" notation after the name of the dictionary

Method	What it does
`keys()`	retrieve all keys
`values()`	retrieve all values
`items()`	retrieve all items (key-value pairs returned as tuples)
`clear()`	delete all items from dictionary

When to use dictionaries

When to use dictionaries

- Store things that are "paired" together
  - when two things are related to one another
  - Example: State abbreviation <—> state name

- Need to retrieve data based on an ‘ID’, identifier, or key
  - when one item is used to look up another

Recap

Recap of what we learned

- What dictionaries are, and why we use them
  - dictionaries contain key-value pairs
  - what keys are
- Working with dictionaries
  - creating dictionaries
  - adding/removing items from dictionaries
- Important dictionary methods