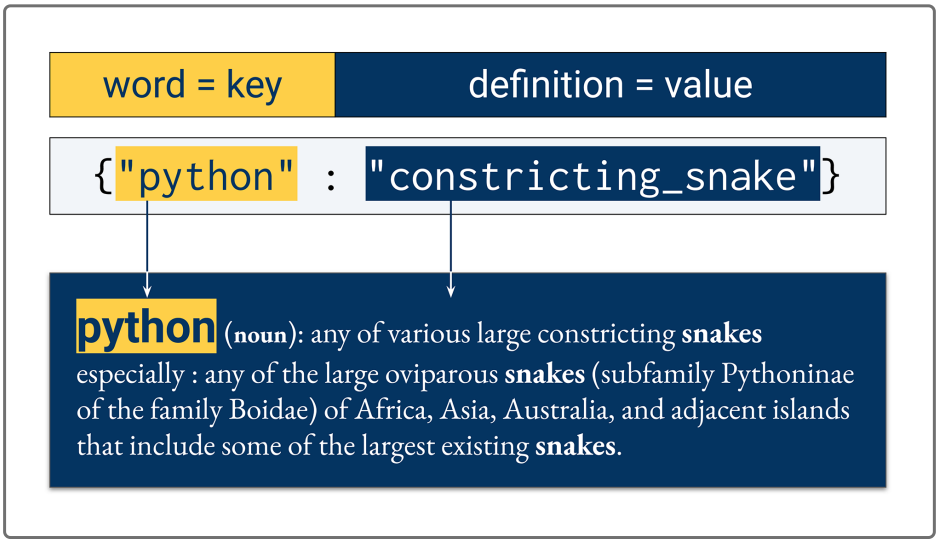
3.2.7 Data Structures: Dictionaries

Now that you're familiar with two common data structures, lists and tuples, Tom would like you to learn how data is stored and retrieved from a third data structure: dictionaries. Real-world data is stored in dictionaries rather than lists or tuples. There are multiple ways to retrieve data from a dictionary, and Tom wants to make sure you are familiar with the different methods available. Some may come in handy for the elections analysis, while other methods you may need to use in upcoming projects.

A **dictionary** is an object that stores a collection of data. An object in Python and other computer programming languages is data stored in the computer's memory.

A Python dictionary has a **key** and a **value**, or **key-value pairs**. Very similar to a dictionary that contains definitions, the words in the dictionary would be considered the keys, and the definitions of those words would be the values.



Key-value pairs are enclosed in a set of curly braces, `{}`. The syntax for a dictionary is the key followed by a colon, which is followed by a value: `{key:value}`.

When we want to find a definition in a regular dictionary, we look up the word. Likewise, in Python, when we want to get a specific value from the dictionary, we look up the key.

If there's more than one element or key-value pair in the dictionary they are separated by commas, like this: `{key1:value1, key2:value2}`.

There are two key rules for dictionaries:

1. Values in a dictionary can be objects of any type: integers, floating-point decimals, strings, Boolean values, datetime values, and lists.
2. Keys must be immutable objects, like integers, floating-point decimals, or strings. Keys cannot be lists or any other type of mutable object.

Create a Dictionary

During the data analysis process, you may be given a dictionary and have to retrieve data from the dictionary. Occasionally, you may need to initialize a dictionary in which you will store data as you collect it from other sources, like lists, tuples, or CSV files (as we will encounter later in this module).

To initialize or create an empty dictionary, we use the following syntax: `my_dictionary = {}`. Or you can create a dictionary with the built-in Python

`dict()` method, `my_dictionary = dict()`.

Let's create a dictionary with the counties as keys and the number of registered voters of each county as values.

Create the empty counties dictionary by typing the following code in the Python interpreter and pressing Enter.

```
>>> counties_dict = {}
```

Next, add the county "Arapahoe" to the dictionary as the key and the number of registered voters for Arapahoe as the values for this key.

IMPORTANT

> The standard format for creating a key in a dictionary is to place the key between single or double quotes and inside brackets.

```
>>> counties_dict["Arapahoe"] = 422829
```

Let's see how this looks. Type `counties_dict` on the next line and press Enter.

```
>>> counties_dict
{'Arapahoe': 422829}
```

Repeat this process two more times to add the counties "Denver" and "Jefferson" with their respective number of registered voters.

```
>>> counties_dict["Denver"] = 463353
>>> counties_dict["Jefferson"] = 432438
```

When we enter `counties_dict` and press Enter, the output in the interpreter should look like this:

```
>>> counties_dict
{'Arapahoe': 422829, 'Denver': 463353, 'Jefferson': 432438}
```

Congratulations on creating a dictionary in Python!

REWIND

Remember, when typing integers in Python, do not use the thousands separator (comma) in the number.

Get the Length of a Dictionary

Similar to lists and tuples, the length of a dictionary can be found by using the `len()` function. To get the length of `counties_dict`, run the following code:

```
>>>  len(counties_dict)
3
>>>
```

Get All Keys and Values

To get all the keys and values printed to the screen, simply print the dictionary name. Or, we can use the `items()` method on the dictionary. This will return a list of tuples where the first element in each tuple is the key of the dictionary, and the second element in each tuple is the value corresponding to that key.

If we add the `items()` method to the end of `counties_dict`, we'll get this output:

```
>>> counties_dict.items()
dict_items([('Arapahoe', 422829), ('Denver', 463353), ('Jefferson', 432438)])
>>>
```

In the output, the information inside the `dict_items([])` is what is known as a **view object**. A view object will give us a snapshot of what is in the dictionary. In this output, we can see each key and their respective values.

NOTE

> You cannot use list indexing with the `items()` method.

Get All the Keys

To get only the keys from a dictionary, add the `keys()` method at the end of the dictionary, like this:

```
>>> counties_dict.keys()
dict_keys(['Arapahoe', 'Denver', 'Jefferson'])
>>>
```

The `keys()` method will return a view object that contains the keys of the dictionary as a list.

Get All the Values

To retrieve only the values from a dictionary, add the `values()` method to the end of the dictionary, like this:

```
>>> counties_dict.values()
dict_values([422829, 463353, 432438])
>>>
```

Just like the `items()` and `keys()` methods, the `values()` method will return a view object that contains the values of the dictionary as a list.

Get a Specific Value

There are two methods you can use to get a specific value from a dictionary.

The first method is to use the `get()` method. With the `get()` method, we pass a key inside the parentheses to "get" the value of that key.

Let's "get" the value or the number of registered voters, in Denver County. Type the following code and press Enter.

```
>>> counties_dict.get("Denver")
463353
```

Retake

To get the number of registered voters in Arapahoe County, we can also wrap the key in brackets using this format: `dictionary_name[key]`.

When we enter a key to retrieve its value, we must type the key within single or double-quotes. Without quotes, we will get the following `NameError`.

```
>>> counties_dict[Arapahoe]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Arapahoe' is not defined
```

Using single or double quotes will satisfy the requirement.

```
>>> counties_dict['Arapahoe']
422829
>>> counties_dict["Arapahoe"]
422829
```

Lists of Dictionaries

Sometimes Python dictionaries have the same keys associated with different values, which are written in this format:

`[{key1:value1, key2:value2}, {key1:value3, key2:value4}]`.

This is referred to as a **list of dictionaries** because each dictionary, `{}`, is wrapped in brackets.

Let's create a list of dictionaries where the keys are "county" and "registered_voters," and each county and its corresponding registered voters are the values for those keys.

First, create an empty list called `voting_data`.

```
>>> voting_data = []
```

Then add, or append, each dictionary to the `voting_data` list with the following code and press Enter:

```
>>> voting_data.append({"county":"Arapahoe", "registered_voters": 422829})
>>> voting_data.append({"county":"Denver", "registered_voters": 463353})
>>> voting_data.append({"county":"Jefferson", "registered_voters": 432438})
```

When we type `voting_data` and press Enter, we get the following output:

```
>>> voting_data
[{'county': 'Arapahoe', 'registered_voters': 422829}, {'county': 'Denver', '
```

Take a look at this code. This is a **list of dictionaries** because each element in the list is a dictionary. The keys are "county" and "registered_voters."

The format of a CSV file is like a list of dictionaries, where the headers (column names) "county" and "registered_voters" are the keys, and the values are the data in the rows. If you added this data to a CSV file it would look like this.

	A	B
1	county	registered_voters
2	Arapahoe	422829
3	Denver	463353
4	Jefferson	432438

We can use list methods to:

1. Get the length of the `voting_data` list of dictionaries.
2. Use indexing and slicing to get one or more dictionaries.
3. Use the `append()`, `insert()`, and `remove()` methods to add and remove one or more dictionaries.
4. Change a value for one of the keys in the list of dictionaries.

Retake

NOTE

> For more information, refer to the documentation on the following topics:
>
> - **Dictionary keys (https://docs.python.org/3.7/faq/design.html#why-must-dictionary-keys-be-immutable)**
> - **View objects (https://docs.python.org/3.7/library/stdtypes.html#dictionary-view-objects)**
> - **Built-in Python functions such as dict() (https://docs.python.org/3.7/library/functions.html)**