Dictionaries

This lesson highlights the key features of the dictionary data structure.

We'll cover the following

Structure

Compared to a list or tuple, a dictionary has a slightly more complex structure.

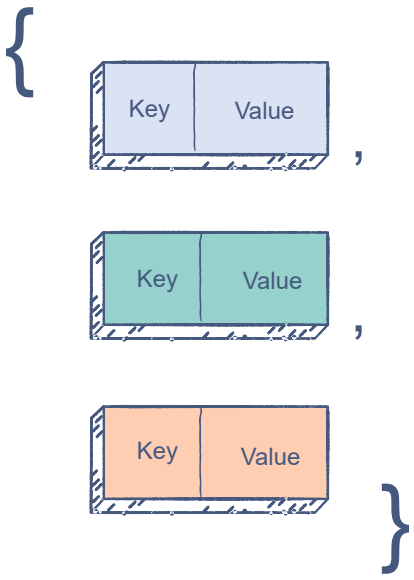
When we think of a dictionary, what pops up is the image of a vast book containing words with their meanings.

In simpler terms, information is stored in pairs of words and meanings. Python's dictionary data structure follows the same structure.

> A **dictionary** stores **key-value** pairs, where each unique key is an **index** which holds the value associated with it.

Dictionaries are **unordered** because the entries are not stored in a linear structure.

In Python, we must put the dictionary's content inside curly brackets, `{}`:



A key-value pair is written in the following format:

```
key:value
```

Creating a Dictionary

Let's create an empty dictionary and a simple `phone_book` using the dictionary data structure:

```
empty_dict = {}  # Empty dictionary
print(empty_dict)

phone_book = {"Batman": 468426,
              "Cersei": 237734,
              "Ghostbusters": 44678}
print(phone_book)
```

The dictionary above could also be written in the same line. Dividing it into lines, only increases readability.

The `dict()` Constructor

As the name suggests, the `dict()` constructor can be used to construct a dictionary. Don't worry about the term "constructor" for now. Think of it as an operation which gives us a dictionary.

If our keys are simple strings without special characters, we can create entries in

the constructor. In that case, values will be assigned to keys using the `=` operator.

A popular practice is to create an empty dictionary and add entries later.

Let's refactor the `empty_dict` and `phone_book` examples to make them work with `dict()`:

```
empty_dict = dict()  # Empty dictionary
print(empty_dict)

phone_book = dict(Batman=468426, Cersei=237734, Ghostbusters=44678)
# Keys will automatically be converted to strings
print(phone_book)

# Alternative approach
phone_book = dict([('Batman', 468426),
                   ('Cersei', 237734),
                   ('Ghostbusters', 44678)])
print(phone_book)
```

The **keys** and **values** can have any of the basic data types or structures we've studied.

Two keys can have the same value. However, it is crucial that all keys are **unique**.

We can see from the `phone_book` example how dictionaries can organize data in a meaningful way. It is easy to tell a character's phone number because the pair is stored together.

Accessing Values

For many, this is where a dictionary has an edge over a list or a tuple. Since there are no linear indices, we do not need to keep track of where values are stored.

Instead, we can access a value by enclosing its **key** in square brackets, `[]`. This is more meaningful than the integer indices we use for tuples and lists.

Alternatively, we can use the `get()` method as follows:

```
a_dictionary.get(key)
```

Let's build upon our previous example:

```
phone_book = {"Batman": 468426,
              "Cersei": 237734,
              "Ghostbusters": 44678}
print(phone_book["Cersei"])
print(phone_book.get("Ghostbusters"))
```

We've understood the purpose of dictionaries and how we can store data in them. In the next lesson, we'll explore their functionality further.