Dictionary Operations

In this lesson, we'll learn about more operations that help us use a dictionary to its full potential.

We'll cover the following

Adding/Updating Entries

We can add new entries in a dictionary by simply assigning a value to a key. Python automatically creates the entry.

If a value already exists at this key, it will be updated:

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

phone_book["Godzilla"] = 46394  # New entry
print(phone_book)

phone_book["Godzilla"] = 9000  # Updating entry
print(phone_book)
```

Removing Entries

To delete an entry, we can use the `del` keyword:

```
phone_book = {"Batman": 468426,
              "Cersei": 237734,
```

```
                "Ghostbusters": 44678}
print(phone_book)


del phone_book["Batman"]
print(phone_book)
```

If we want to use the deleted value, the `pop()` or `popitem()` methods would work better:

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

cersei = phone_book.pop("Cersei")
print(phone_book)
print(cersei)

# Removes and returns an arbitrary pair as a tuple
lastAdded = phone_book.popitem()
print(lastAdded)
```

Length of a Dictionary

Similar to lists and tuples, we can calculate the length of a dictionary using `len()`:

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

Checking Key Existence

The `in` keyword can be used to check if a key exists in a dictionary:

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

Copying Contents

To copy the contents of one dictionary to another, we can use the `update()` operation:

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

second_phone_book = {"Catwoman": 67423, "Jaime": 237734, "Godzilla": 37623}

# Add secondphone_book to phone_book
phone_book.update(second_phone_book)
print(phone_book)
```

Dictionary Comprehension

Python also supports dictionary comprehensions, which work very similar to list comprehensions. We'll be creating new key-value pairs based on an existing dictionary.

However, to iterate the dictionary, we'll use the `dict.items()` operation which turns a dictionary into a list of `(key, value)` tuples.

Here's a simple example where the keys of the original dictionary are squared and `'!'` is appended to each string value:

```
houses = {1: "Gryffindor", 2: "Slytherin", 3: "Hufflepuff", 4: "Ravenclaw"}
new_houses = {n**2: house + "!" for (n, house) in houses.items()}
print(houses)
print(new_houses)
```

That brings us to the end of our discussion on dictionaries. If you want to explore them further, check out the official documentation by Python.

In the next lesson, we'll tackle **sets**.