Tuples and Dictionaries

In this lesson, we will learn about two important data structures in Python: tuples and dictionaries.

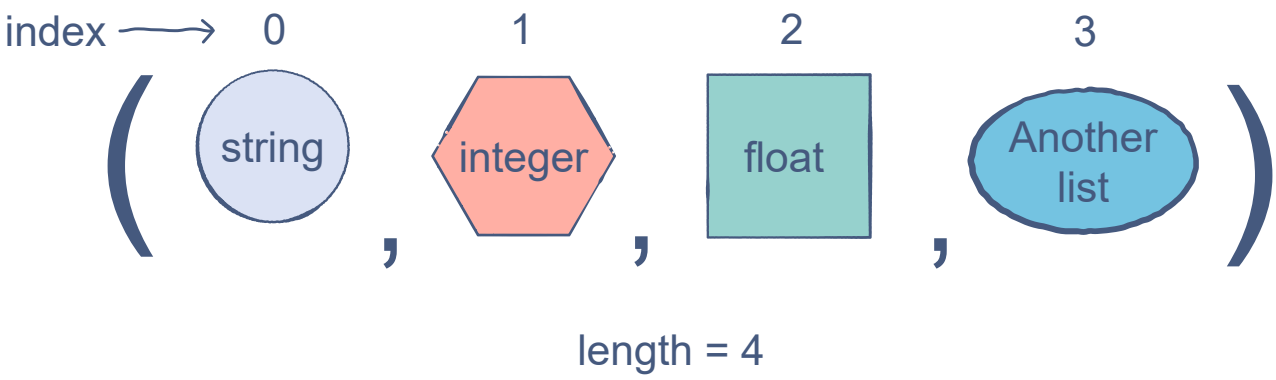
WE'LL COVER THE FOLLOWING

Tuples

A tuple is very similar to a list, except its contents cannot be changed. In other words, a tuple is **immutable**. However, it can contain mutable elements like a list. These elements can be altered.

> 💡 We can't add or delete elements from them. Furthermore, it isn't possible to append another tuple to an existing tuple.

The contents of a tuple are enclosed in parentheses, `()`. They are also ordered, and hence, follow the linear index notation.



Creating

Tuples can be created similar to lists. All the indexing operations apply to it as well:

```
car = ("Ford", "Raptor", 2019, "Red")
print (car)

# Length
print (len(car))

# Indexing is done using square brakcers
print (car[1])
```

We can also create a *tuple of tuples*:

```
hero1 = ("Superman", "Clark Kent")
hero2 = ("Aquaman", "Arthur Curry")
awesome_team = (hero1, hero2)
print (awesome_team)
```

Merging tuples

Tuples can be merged using the `+` operator:

```
hero1 = ("Superman", "Clark Kent")
hero2 = ("Aquaman", "Arthur Curry")
awesome_team = hero1 + hero2
print (awesome_team)
```

```
print (awesome_team)
```

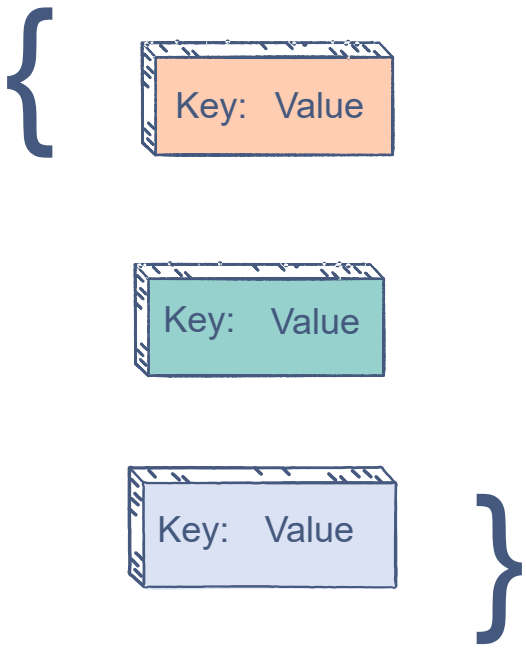
Dictionary

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

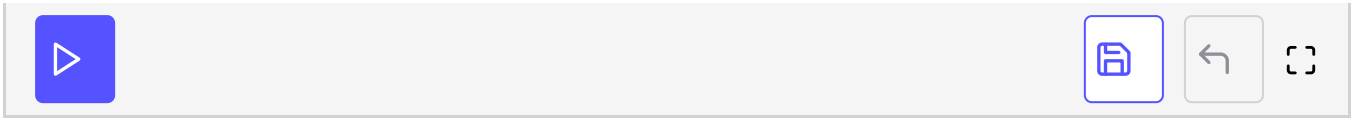


Creating a dictionary

Let's create a dictionary in Python with string-integer as key-value pairs:

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

phone_book = {"Batman": 428426,
              "Superman": 28918398,
```

```
                "Flash": 8198318}
print (phone_book)
```



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

All keys must be unique. However, the same value can be mapped to multiple keys.

We can see from the `phone_book` example how dictionaries can organize data in a meaningful way.

It is easy to tell a superhero's phone number because the pair is stored together as we shall see next.

Accessing values

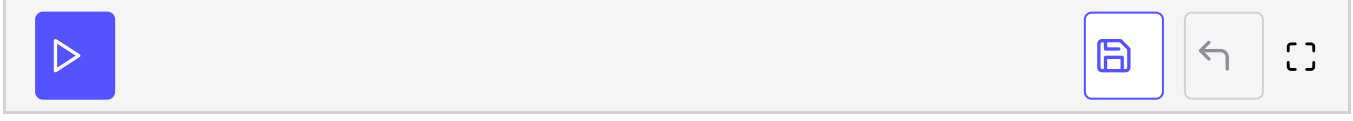
For many, this is where a dictionary has an edge over a list or a tuple. Since there are no linear indices, we do not need to keep track of where values are stored.

Instead, we can access a value by enclosing the name of the key in square brackets, `[]`. Alternatively, we can use the `get()` method as follows:

```
dictionaryName.get(key)
```

Accessing data this way is more meaningful than the integer indices we use for tuples and lists.

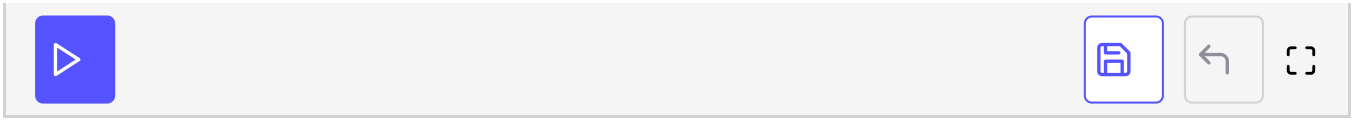
```
phone_book = {"Batman": 428426,
              "Superman": 28918398,
              "Flash": 8198318}

print (phone_book["Batman"])
print (phone_book.get("Flash"))
```



Adding entries

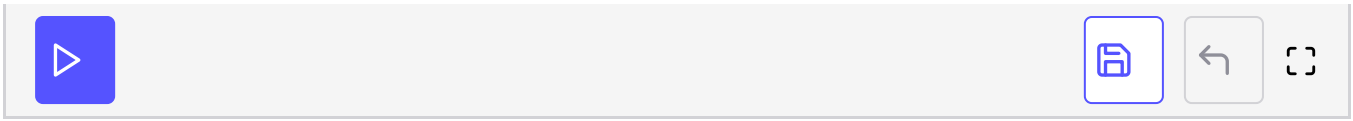
We can add new entries in a dictionary by simply assigning a value to a key. Python automatically creates the entry.

```
phone_book = {"Batman": 428426,
              "Superman": 28918398,
              "Flash": 8198318}

print (phone_book)
phone_book["Aquaman"] = 4209211
print (phone_book)
```



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

```
phone_book = {"Batman": 428426,
              "Superman": 28918398,
              "Flash": 8198318}

print (phone_book)
phone_book["Batman"] = 4209211
print (phone_book)
```



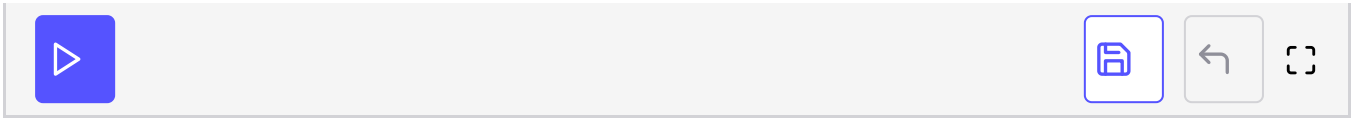
Checking existing entries

If you want to check that a certain key exists in a dictionary, you can use an `if` statement

```
if key in dictionary:
  do something
```

```
phone_book = {"Batman": 428426,
              "Superman": 28918398,
              "Flash": 8198318}

if "Batman" in phone_book:
  print ("Gotham is saved.")
else:
  print ("Gotham is in danger.")

if "Aquaman" in phone_book:
  print ("Atlantis is saved.")
```

```
else:
  print ("Atlantis is in danger.")
```



In the next lesson, we will learn about Python packages.