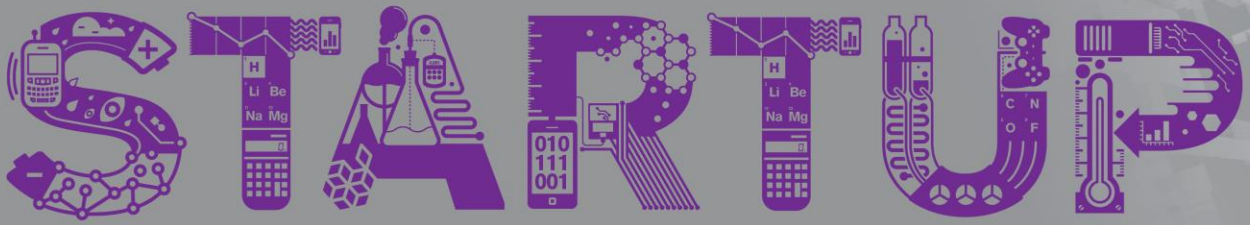
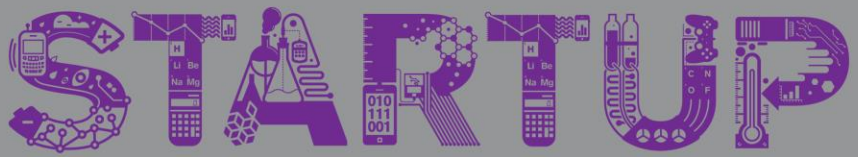
Bloomberg



Bloomberg



Python Lesson 9

Dictionaries

Review: Warm up

1) Create a list that contains at least three names.
2) Write a for loop that iterates over your list and prints out “Hello <NAME>” for each name in your list.

How could we store the following data?

Phone Numbers:

Tobias: 888-777-6666

Nico: 555-444-3333

Nelson: 222-111-9999

How could we store the following data?

Phone Numbers:
Tobias: 888-777-6666
Nico: 555-444-3333
Nelson: 222-111-9999

We could use variables:

```
tobias_phone = 8887776666
nico_phone = 5554443333
nelson_phone=2221119999
```

Problem: This could be a lot of variables.

How could we store the following data?

Phone Numbers:
Tobias: 888-777-6666
Nico: 555-444-3333
Nelson: 222-111-9999

We could use a list:

```
phone_numbers = [88877766, 5554443333, 2221119999]
```

Problem: How do we know whose is whose?

How could we store the following data?

Phone Numbers:
Tobias: 888-777-6666
Nico: 555-444-3333
Nelson: 222-111-9999

- Each phone number has two pieces: The person it belongs to and the number itself.
- Python has a data type to store data like this: **Dictionaries**

Dictionaries: Key-Value Pairs

Dictionaries store data in the form of **key-value pairs**.

- A **key** is like the name of the data.
  - In our phone book example, the key is the person's name.
- A **value** is the piece of data you want to associate to the name.
  - In the phone book, the value is the actual phone number.
- In order to look up the **value,** you need to use the **key.**

Anatomy of a Dictionary

Dictionaries have names, just like lists.

```
phone_numbers_dict = {
    'tobias': 8887776666,
    'nico': 5554443333,
    'nelson': 2221119999
}
```

Inside the curly brackets, a comma separated list of key-value pairs.

Key-Value pairs are written as Key: Value

Start and end with curly brackets

Keys must be unique!

Example

Create a dictionary that stores the following price data:

Oreos cost 2.75, Doritos cost 1.25, and Donuts cost 0.80.

```
snack_prices = {
    "oreos" : 2.75,
    "doritos" : 1.25,
    "donuts": 0.80
}
```

Getting Data Out of a Dictionary

In order to get data out of a dictionary, you need to use the key.

```
print(snack_prices["oreos"])
```

Editing Data in a Dictionary

To edit the value in a dictionary, you access it the same way:

```
snack_prices["oreos"] = 2.50
```

Adding Data to a Dictionary

You can add a new key-value pair to a dictionary at any time.

```
snack_prices["pringles"] = 2.30
```

Example

- Create a dictionary called `me` that stores your name, age, and favorite color.
- Using that dictionary, print out your favorite color.
- Add your favorite number to the dictionary.
- It’s your birthday! Increase your age by 1.

Dictionary Rules

- Keys can be strings or numbers.
- Values can be anything!
- Keys must be unique.
- Unlike lists, dictionaries do NOT care about the order of their keys.
  - { “a” : “hello”, “b” : “world”} is the same as {“b” : “world”, “a” : “hello” }
  - [1, 2] is NOT the same thing as [2, 1]

Example

- Create a dictionary called `student_grades`. Add the following data:
  - Tobias's grades are 96, 85, and 91.
  - Nico's grades are 95, 94, and 83.
- Use the names as keys. Store the grade data as lists.
- Use your dictionary to print out Tobias's and Nico's average grades.

Exercises

- Continue work on exercises

Additional Dictionary Properties

- You can loop through dictionaries:

```
for tasty_food in snack_prices:
    print(tasty_food, snack_prices[tasty_food])
```

- You can test if a key is in a dictionary:

```
if "oreos" in snack_prices:
    print("I love oreos!")
```

Python Explorer:

Dictionaries let us store data about each location.

```
location_data = {
  0 : { "name" : "empty"},
  1 : { "name" : "basket"},
  2 : { "name" : "empty"},
  3 : { "name" : "hole in the wall"},
  4 : { "name" : "locked door"},
}
```

Python Explorer:

We can then refer to that data later on to tell the user where they are!

```
def look(current_location):
  if current_location == 0:
    print("You are at the south end of the tunnel. It's walled off.")
  elif current_location == len(location_data) - 1:
    print("You are at the north end of the tunnel. There is a door locking you in.
  elif location_data[current_location]["name"] != "empty":
    print( "You are standing at a", location_data[current_location]["name"])
  else:
    print("The tunnel here is empty and dark")
```

Recap

- Dictionaries store pairs of data: keys and values.
- This lets us store more complicated data.
- It also lets us store data and look it up later without remembering lots of separate variable names.