Week 3 - Summary

Columbia Business School

Dictionaries (1/2)

- Dictionaries are lists where you can label values and create multiple levels of hierarchy. They are used to represent rows of a table.
- Curly brackets '{}' in a syntax indicate a dictionary
- Values in a dictionary could be a string, number, a list or even a dictionary

Columbia Business School

Dictionaries (2/2)

To create a dictionary:

dictionaries.py — ~/Desktop/code

dictionaries.py

```
1  # Dictionaries let you label values using strings
2  stock = {"name": "Microsoft",
3           "ticker": "MSFT",
4           "index": "NASDAQ"}
```

To access something out of a dictionary:

```
print(f"{stock['name']}'s stock ticker is {stock['ticker']}")
```

Columbia Business School

Adding a New Key and Value Pair

Key names are basically strings and you can use spaces while adding them.

```
stock['open price'] = 108.25
stock['close price'] = 106.03
print(stock)
```

```
(base) Mattans-MacBook-Pro:code mattan$ python dictionaries.py
Microsoft's stock ticker is MSFT
Name: Mattan
Height: 5'10"
Shoe Size: 10.5
Hair: Brown
Eyes: Brown
{'name': 'Microsoft', 'ticker': 'MSFT', 'index': 'NASDAQ', 'open
price': 108.25, 'close price': 106.03}
(base) Mattans-MacBook-Pro:code mattan$
```

Columbia Business School

Adding a New Key and Value Pair

Key names are case-sensitive

```
stock['open price'] = 108.25
stock['close price'] = 106.03
print(stock)
print(stock['Open Price'])
```

```
(base) Mattans-MacBook-Pro:code mattan$ python dictionaries.py
Microsoft's stock ticker is MSFT
Name: Mattan
Height: 5'10"
Shoe Size: 10.5
Hair: Brown
Eyes: Brown
{'name': 'Microsoft', 'ticker': 'MSFT', 'index': 'NASDAQ', 'open
price': 108.25, 'close price': 106.03}
Traceback (most recent call last):
  File "dictionaries.py", line 25, in <module>
    print(stock['Open Price'])
KeyError: 'Open Price'
(base) Mattans-MacBook-Pro:code mattan$
```

Columbia Business School

Selecting a Subset of Users

```
brown_eyed_users = []
for user in users:
    if user['eyes'] == 'Brown':
        brown_eyed_users.append(user)

print(brown_eyed_users)
```

```
[{'name': 'Mattan', 'height': '5\'10"', 'shoe size': 10.5, 'hair'
: 'Brown', 'eyes': 'Brown'}, {'name': 'Lisa', 'height': 64, 'shoe
 size': 6.5, 'hair': 'Black', 'eyes': 'Brown', 'favorite movies':
 ['Crazy Rich Asians', 'Avengers', 'Lord of the Rings']}]
```

Columbia Business School

Role of Functions in Python

- Only one output
- List/dictionary with multiple values
- Multiple inputs allowed

A function removes any person dependencies

Columbia Business School

Creating a Function

Result

```
(base) Mattans-MacBook-Pro:code mattan$ python functions.py
New York
(base) Mattans-MacBook-Pro:code mattan$ python functions.py
(base) Mattans-MacBook-Pro:code mattan$ python functions.py
New York
(base) Mattans-MacBook-Pro:code mattan$
```

Code

functions.py

```
1  def get_city(address):
2      return address.split(', ')[1]
3
4  print(get_city("3022 Broadway, New York, NY 10027, USA"))
```

Columbia Business School

Functions Reducing Redundancy

Result

```
(base) Mattans-MacBook-Pro:code mattan$ python functio
New York
NY
True
False
(base) Mattans-MacBook-Pro:code mattan$
```

Longer Code

```
def is_divisible(number, divisor):
    if number % divisor == 0:
        return True
    else:
        return False
```

Result in both cases is the same

Result

```
(base) Mattans-MacBook-Pro:code mattan$ python functions.py
New York
NY
True
False
(base) Mattans-MacBook-Pro:code mattan$
```

Concise Code

```
def is_divisible(number, divisor):
    return number % divisor == 0
```

Columbia Business School

More About Functions

Too many or too few arguments in a function result in an error

Python maintains two versions of documentation on functions – 2.7.16 (some users still have Python 2) and 3.7.4 (recent version)

Columbia Business School

Functions: Optional Arguments

Optional argument #1

```
def greet(name="You"):
    return f"Hey {name}!"

print(greet("Mattan"))
```

Optional argument #2

```
def greet(name="You"):
    return f"Hey {name}!"

print(greet("Mattan"))
print(greet())
```

Same result

```
(base) Mattans-MacBook-Pro:code mattan$ python functions.py
New York
NY
True
False
ANANAB

This will work as well
Hey Mattan!
```

Columbia Business School

Function Gotchas

Any variable created inside of a function is not available outside of a function until you return it at the end

```
38  # Don't let variables sneak into your function
39  word = "jelly"
40
41  def reverse(text):
42      return word[::-1]
43
44  print(reverse("python"))
```

Columbia Business School

Importing a Function

functions.py fizzbuzz.py

```
import functions

# FizzBuzz Challenge
for number in range(1, 101):
    if number % 3 == 0 and number % 5 == 0:
        print("FizzBuzz")
    elif number % 3 == 0:
        print("Fizz")
    elif number % 5 == 0:
        print("Buzz")
    else:
        print(number)
```

Columbia Business School

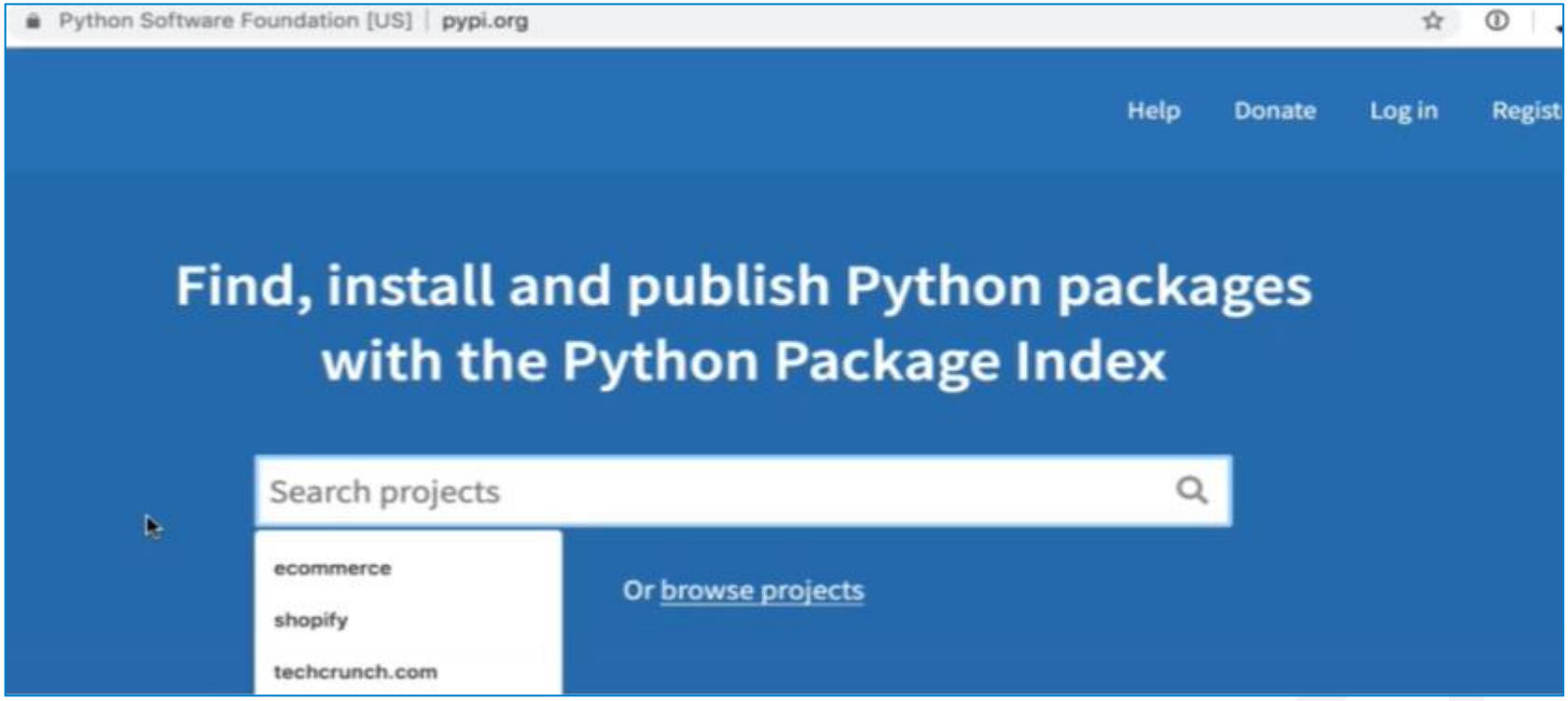
The Python Standard Library

The Python standard library is a set of python files that you can import in any of your files. It is a collection of functions that have already been written and can be imported instantly.

Columbia Business School

Third Party Libraries

You can download libraries created by other developers



Columbia Business School

Importing a Library

reading_csvs.py

```
import pandas

data = pandas.read_csv('census.csv')

print(data)
```

Code

Result

```
(base) Mattans-MacBook-Pro:code mattan$ python reading_csvs.py
      FIPS  ... 2017 Unemployment rate (Population 16 years and over)
0    10100  ...                                                 1.8
1    10140  ...                                                10.1
2    10180  ...                                                 4.4
3    10220  ...                                                 5.1
4    10300  ...                                                 5.9
5    10420  ...                                                 6.6
6    10460  ...                                                 9.3
7    10500  ...                                                12.6
8    10540  ...                                                 8.8
9    10580  ...                                                 5.6
10   10620  ...                                                 8.0
```