Session 2

Control Flow -- Part 2

Dictionaries

Functions

8.20.19

Link to Jupyter Notebooks:
https://mybinder.org/v2/gh/data-voyage-solutions/oag-session-mats/master

Context

```
import pandas as pd

folder_names = ['2015-07','2015-08','2015-09','2015-10','2015-11','2015-12']
file_names = ['empty_full_results','bikeshare_nyc_raw']

for folder in folder_names:
    for file in file_names:
        if file == 'bikeshare_nyc_raw':
#             data = pd.read_csv(folder + '/' + file + '.csv', delim_whitespace=True)
            data = pd.read_csv(folder + '/' + file + '.csv', sep='\\t', engine='python')
        elif file == 'empty_full_results':
            data = pd.read_csv(folder + '/' + file + '.csv')
        else: pass

        data['union_id'] = data['dock_id']
        data['union_id'] = data['union_id'].astype(str)
        data['union_id'] = data['union_id'] + "-" + folder

        data.to_csv(folder + '/' + file + '_vf' + '.csv')

```

01

Wrap-up Prev. Session

45 min.

02

Dictionaries

1 hour

03

Functions

1 hour

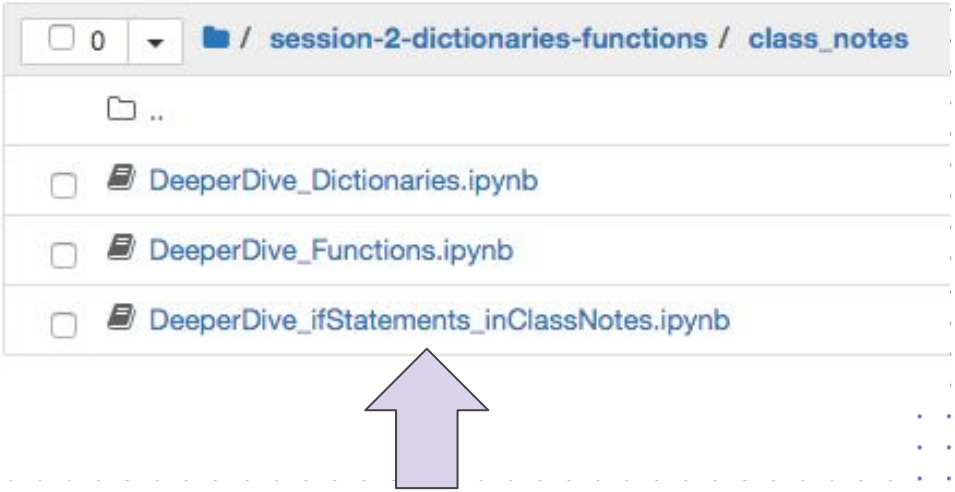
04

Session 2 Practice

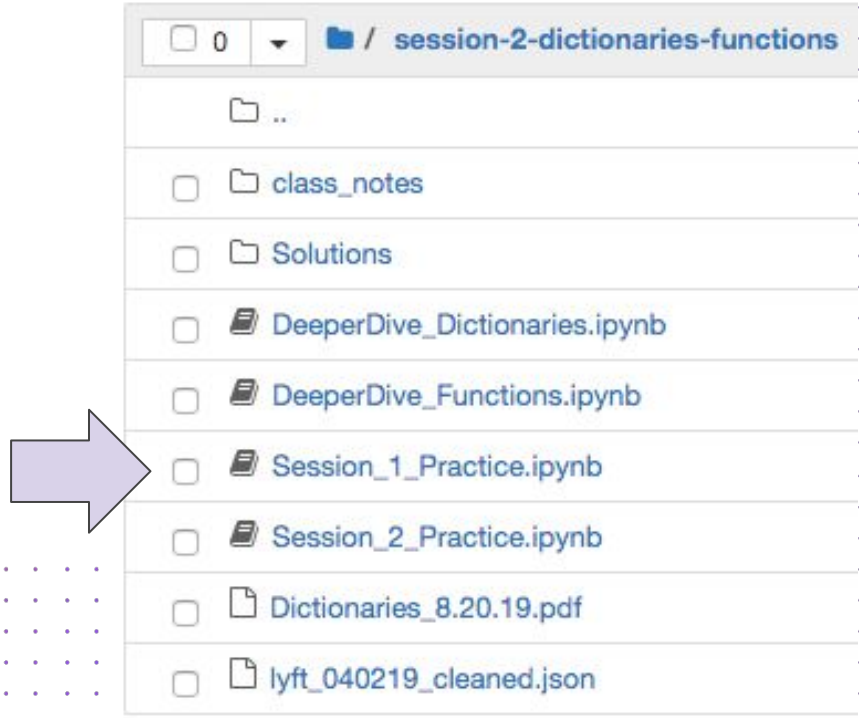
Remaining time

Pick up from last session

if Statements



Practice Session 1 Material



Dictionaries

Data Structures

Dictionaries

- Collections of **key/value pairs**
- Defined by curly brackets { }
- Slicing uses **keys**
- **Order** is not preserved
    - (well, for 3.6+ it is: https://stackoverflow.com/questions/39980323/are-dictionaries-ordered-in-python-3-6)

Python

```
1 params = {"parameter1" : 1.0, "parameter2" : 2.0,
2 "parameter3" : 3.0,}
3 print(type(params))
4 print(params)
```

DeeperDive_Dictionaries_Functions

Dictionaries

- A dictionary is a collection of **key-value pairs.**
- A key-value pair is a set of values **associated with each other.**
- A dictionary is **accessed by key** (not position)
- A **key is unique** and *must* be immutable.

Dictionaries

Below is a phone directory, which is a great example of a dictionary.

Why is that?

```
#----------------------------#
# Luna           | 444 - 4444
# Tee            | 123 - 4567
# Ada Lovelace   | 101 - 0101
#----------------------------#
```

Nested Dictionaries

Here is an example of a nested dictionary:

```
nested_example = {'info': {42: 1, type(''): 2}, 'spam': [1,2,3,'four']}
```

If we wanted to access the value associated with the key 42, you would use the syntax below:

```
print(nested_example['info'][42]) # fetches 1
```

Nested Dictionaries

- You can nest a dictionary inside another dictionary.
- For example, if you have several users for a website, each with a unique username, you can use the usernames as the keys in a dictionary.
- You can then store information about each user by using a dictionary as the value associated with their username.

Functions

Let's Consider a Repetitive Program…

Consider a program that prints a $5 shipping charge for products on a website:

```
print("You've purchased a Hanging Planter.")
print("Thank you for your order. There will be a $5.00 shipping charge for this order.")

# 10 minutes later...
print("You've purchased a Shell Mirror.")
print("Thank you for your order. There will be a $5.00 shipping charge for this order.")

# 5 minutes later...
print("You've purchased a Modern Shag Rug.")
print("Thank you for your order. There will be a $5.00 shipping charge for this order.")
```

What if there are 1,000 orders?

Seeing Functions in Action

So we *define* the function, then we can call the function by pairing its name with the parenthesis: `print_order()`.

```
def print_order():
 print("Thank you for your order. There will be a $5.00 shipping charge for this order.")

print("You've purchased a Hanging Planter.")
print_order()

print("You've purchased a Shell Mirror.")
print_order()

print("You've purchased a Modern Shag Rug.")
print_order()
```

Functions

We can write a **function** to print the order.

A function is simple — it's a reusable piece of code. We only define it once. Later, we can use its name as a shortcut to run that whole chunk of code.

- Functions are defined using the def syntax.
  - `def` stands for "define."
- In this case, we're *defining* a function named `function_name()`.

```
def function_name():
    # What you want the function to do.

# Call the function by name to run it:
function_name()
```

Pro tip: Don't forget the `()`, and be sure to indent!