itida
هيئة تنمية صناعة تكنولوجيا المعلومات

egyptfwd
initiative

UDACITY

fwd initiative

Support Webinar

Agenda

1. First Project Tips
2. Second Project Tips
3. Questions

Agenda

1. First Project Tips
2. Second Project Tips
3. Questions

The First Function

```
CITY_DATA = { 'chicago': 'chicago.csv',
              'new york city': 'new_york_city.csv',
              'washington': 'washington.csv' }
```

City

get_filters()

Raw User input

Month:
- January
- February
- March
- April
- May
- June
- **All**

Day:
- Saturday
- Sunday
- Monday
- Tuesday
- Wednesday
- Thursday
- Friday
- **All**

Output:
- **City** that matches one in the `CITY_DATA dictionary`
- **Month** that is a string of the month name or the string “all”
- **Day** that is a string of the day name or the string “all”

load_data(city, month, day)

Filtered Dataframe

df

The Last Function

Refer to discourse - Topic name - Problem Showing the sample data if requested - Bike Share project

https://nfpdiscussions.udacity.com/t/problem-showing-the-sample-data-if-requested-bike-share-project/26827

Common Errors

1. **Syntax error:**

```
  File "bikeshare.py", line 98
    def load_data(city, month, day):
      ^
SyntaxError: invalid syntax
```

2. Indentation error:

```
IndentationError: unexpected indent
```

3. Key error:

KeyError: ‘new york’

4. Name error

NameError: name ‘popular_hour’ is not defined

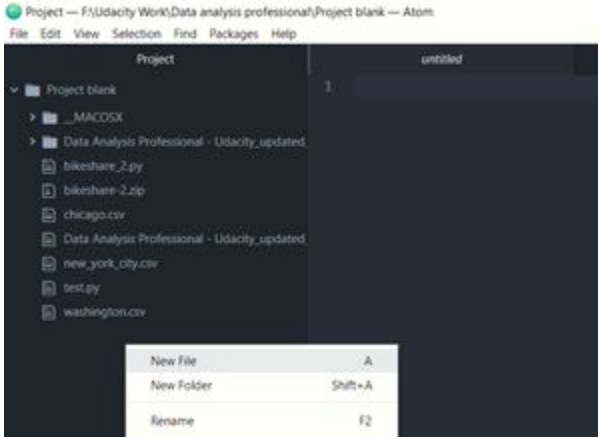
itida

Focus on The Process before the Product

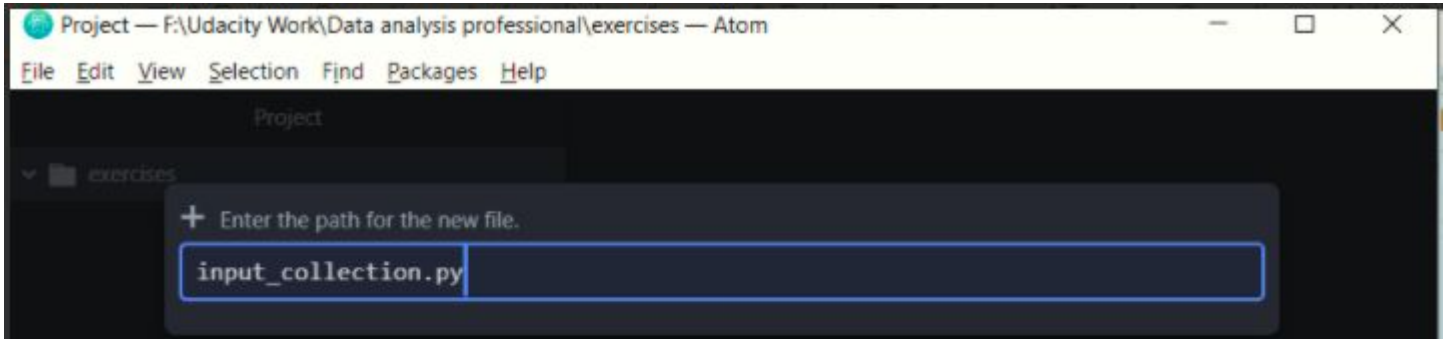
Build gradually …..
Test frequently

Focus on The Process before the Product

1. **Open new file for building and testing by right clicking on the left panel where the files of the project resides**



2. **Name it**



Focus on The Process before the Product

3. Copy the import statements and the dictionary given to you in the template file to the new one.

bikeshare.py | testing.py | testing_2.py | test_os_city.py | session.py | test_os

```
1  import time
2  import pandas as pd
3  import numpy as np
4
5  CITY_DATA = { 'chicago': 'chicago.csv',
6                'new york city': 'new_york_city.csv',
7                'washington': 'washington.csv' }
8
```

4. Start building the first function

```
import time
import pandas as pd
import numpy as np

CITY_DATA = { 'chicago': 'chicago.csv',
              'new york city': 'new_york_city.csv',
              'washington': 'washington.csv' }

def get_filters():
    print('Hello! Let\'s explore some US bikeshare data!')
    # get user input for city (chicago, new york city, washington). HINT: Use a while loop to handle invalid inputs
    city_selection = input('To view the available bikeshare data, type:\n (a) for Chicago\n (b) for New York City\n (c) for Washington\n  ').low
```

itida

Focus on The Process before the Product

5. Test your code after each step or functionality you add to make sure you are on the right track. Testing is done by calling the function and printing its output if it returns an output.

It can be done using the main() function block as well.

```
import time
import pandas as pd
import numpy as np

CITY_DATA = { 'ch': 'chicago.csv',
              'ny': 'new_york_city.csv',
              'w': 'washington.csv' }

def get_filters():
    print('Hello! Let\'s explore some US bikeshare data!')
    # get user input for city (chicago, new york city, washington). HINT: Use a while loop to handle invalid inputs
    city = input('To view the available bikeshare data, type:\n (ch) for Chicago\n (ny) for New York City\n (w) for Washington\n  ').lower()

    # validate city input
    while city not in CITY_DATA.keys():
        print('invalid input')
        city = input('To view the available bikeshare data, type:\n (ch) for Chicago\n (ny) for New York City\n (w) for Washington\n  ').lower()

    return city

# testing
city  = get_filters()
print(city)
```

```
def main():
    city = get_filters()
    print(city)

if __name__ == '__main__':
    main()
```

itida
هيئة تنمية صناعة تكنولوجيا المعلومات

Focus on The Process before the Product

6. Keep adding new lines of code and testing after each addition going gradually from one function to the next and always keep the testing part at the bottom of the script.

```
import time
import pandas as pd
import numpy as np

CITY_DATA = { 'chicago': 'chicago.csv',
              'new york city': 'new_york_city.csv',
              'washington': 'washington.csv' }

def get_filters():

def load_data(city, month, day):

def main():
    city, month, day = get_filters()
    df = load_data(city, month, day)
    print(df.head())
if __name__ == '__main__':
    main()
```

itida

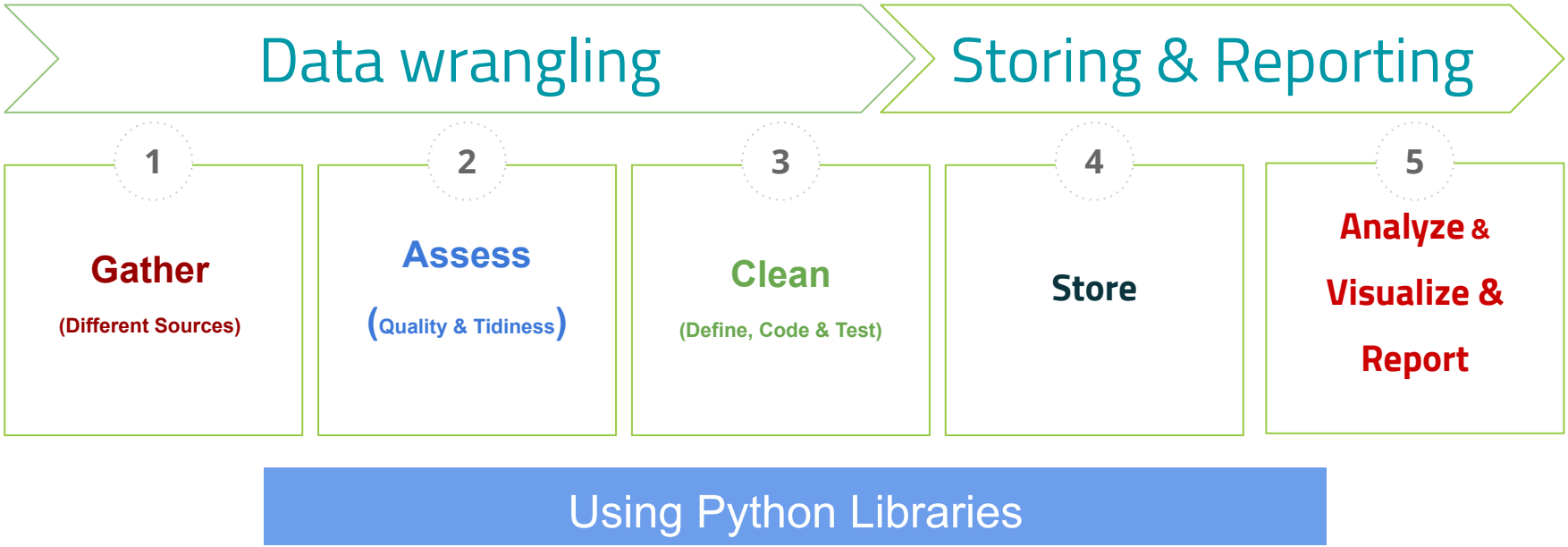
References For Helpful Discourse Topics (Project 1)

1. First function step by step.
2. Load data and time stats
3. Testing scenario
4. Washington error
5. Different approaches to display raw data

Agenda

1. First Project Tips
2. Second Project Tips
3. Questions

Second Project



itida

Data Gathering

Inputs(Data sources):

Additional Data via the Twitter API

`wrangle_act.ipynb`

Outputs (DataFrames):

Tweet_json.txt

Tips & Tricks on API Querying:

```
consumer_key = '**************************'
consumer_secret = '**************************'
access_token = '**************************'
access_secret = '**************************'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

itida

Data Gathering

Inputs(Data sources):

Additional Data via the Twitter API

`wrangle_act.ipynb`

Outputs (DataFrames):

Tweet_json.txt

Tips & Tricks on API Querying:

1. Experimenting to extract one tweet's id information after creating an API object.

```
exp_tweet = api.get_status(archive.tweet_id[1000], tweet_mode = 'extended')
content = exp_tweet._json
print(Content)
```

2. Checking the `keys` of the test tweet through `content.keys()`
3. Then Getting the `retweet_count` and `favorite_count` for the test tweet. There are two ways to do that:
   a. `exp_tweet.retweet_count`, - `exp_tweet.id`, - `exp_tweet.favorite_count`
   b. `content['retweet_count']` , `content['id']`, `content['favorite_count']`
   c. `content['user']['followers_count']`

itida

Data Gathering

Inputs(Data sources):

Additional Data via the Twitter API

`wrangle_act.ipynb`

Outputs (DataFrames):

Tweet_json.txt

Tips & Tricks on API Querying:

```
errors = []
if not os.path.isfile('tweet_json.txt'):
    # create the file and write on it
    with open ('tweet_json.txt', 'w') as file:
        for tweet_id in archive['tweet_id']:
            try:
                status = api.get_status(tweet_id, wait_on_rate_limit=True,  wait_on_rate_limit_notify=True, tweet_mode =
'extended')
                json.dump(status._json, file)
                file.write('\n')
            except Exception as e:
                print("Error on tweet id {}".format(tweet_id) + ";" + str(e))
                errors.append(tweet_id)
```

itida

Reading `tweet_json.txt`

Refer to discourse - Topic name - Data wrangling project, reading text file

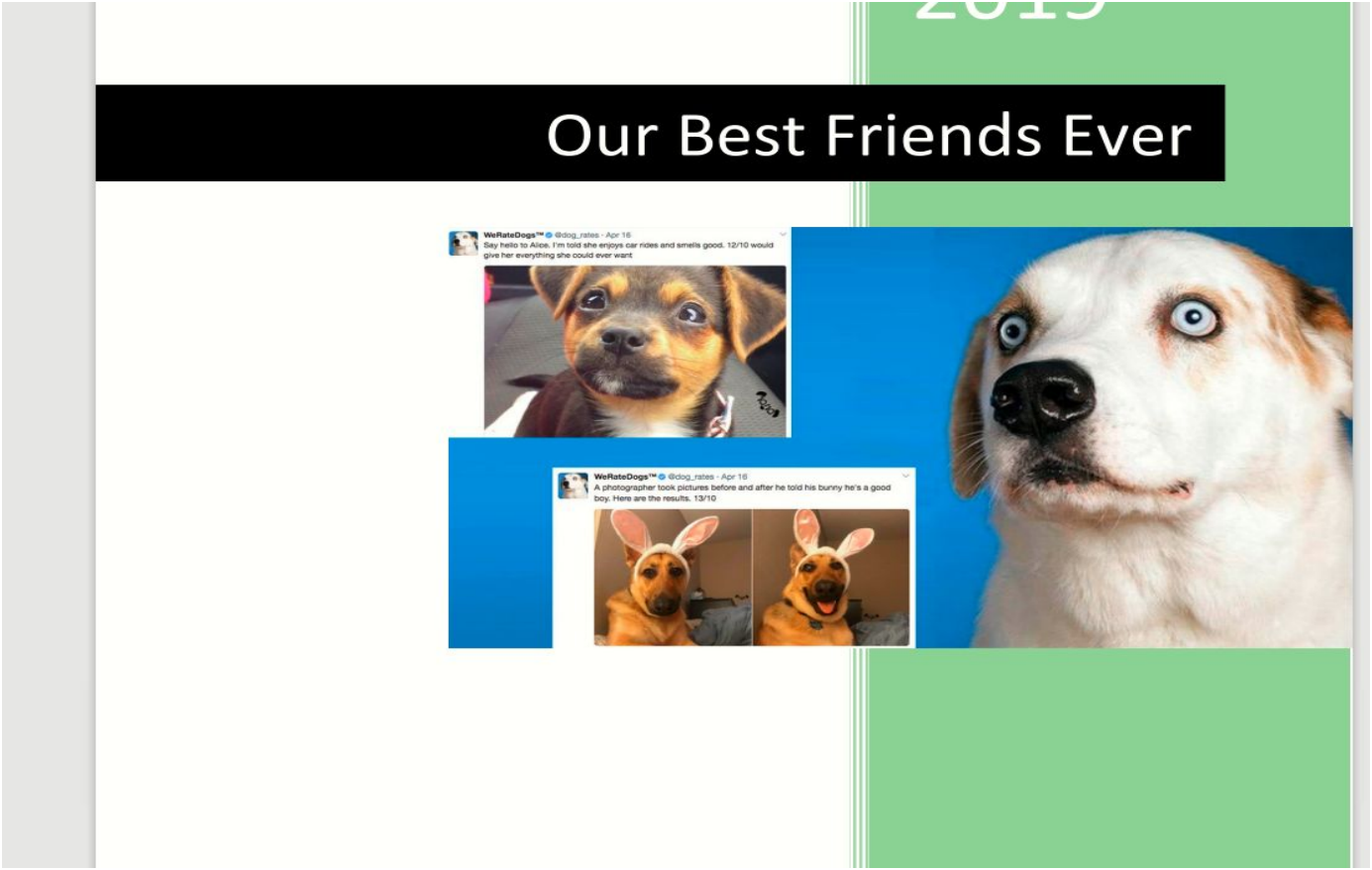
https://nfpdiscussions.udacity.com/t/data-wrangling-project-reading-text-file/31708

Dealing with Dog Stages

Refer to discourse - Topic name - Pd.melt Inquiry

https://nfpdiscussions.udacity.com/t/pd-melt-inquiry/23200

Example of report cover for the Wrangle act report



Example of report content for the Wrangle act report

WeRateDogs Insights for Dogs' Lovers

WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog. These ratings almost always have a denominator of 10. The numerators, though? Almost always greater than 10. 11/10, 12/10, 13/10, etc. Why? Because "**they're good dogs Brent**." If you are a dogs' lover like me; the insights gleaned from this twitter account data introduces some interesting facts about dogs and dogs' lovers behaviour towards their posts that mostly features dogs' photos in it.

1. How is the interaction with the account's posts doing over time?

The interaction is usually defined in terms of favorites and retweets. In the line plot below, there's an evident upward trend in the average monthly count of retweets and favorites attracted by WeRateDogs posts over the time of the analysis. People engagement with the posts is a great indicator of appeal of the content of the posts.

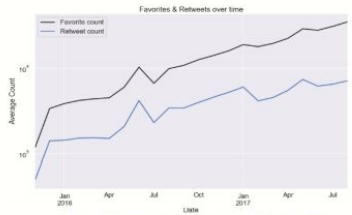


Figure 1 - The average monthly retweets and favorites count

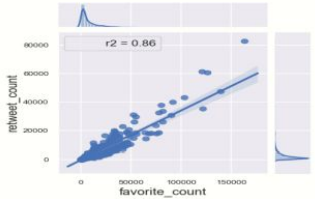
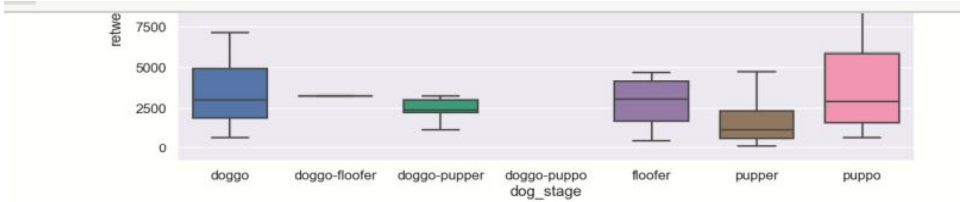


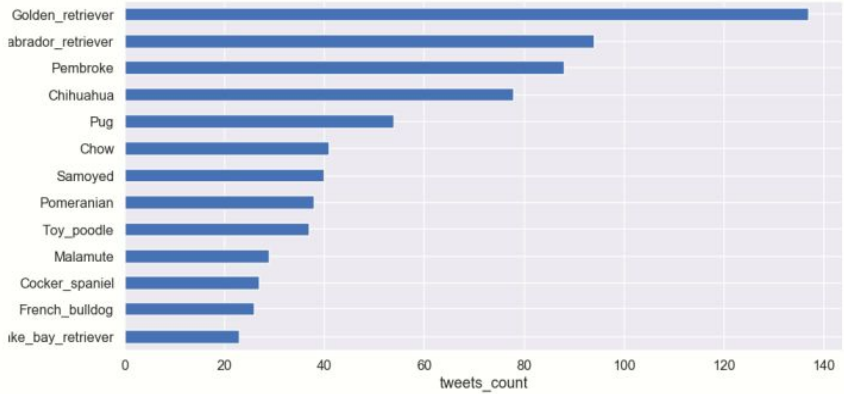
Figure 2 - Association between the favorite counts and retweets count

There's also an apparent association between the retweets and favorites counts that is illustrated in Figure – 2 that features a considerably high coefficient of determination of 0.86.

2. But, hold on for a second, this might be due to larger number of posts tweeted out by WeRateDogs not the interesting and engaging content!



c) The dogs breed also has its impacts on the engagement; The most frequent dog breed featured in the tweets are as follows:



Example of report content for the Wrangle effort report

Data Wrangling Report

By Osama Hamdy Osman

April 2019

As an assignment for the Udacity Data Analyst Nanodegree; This Report illustrates the main steps involved in the data-wrangling of Twitter account "WeRateDogs".

Data Gathering

In this step, collecting data takes place. For this project, there were three main sources for the data to deal with:

1. Twitter_archive_enhanced.csv file, this file was delivered by email and downloaded manually to our working directory and then imported into our working environment using Pandas function "pd.read_csv".
2. Image_prediction.tsv is the second file that has been hosted on a webpage and downloaded from its relevant URL using the Requests library get function and pd.read_csv pandas' function. This file encompassed image predictions for the dogs' breeds obtained through a neural network on most of the tweets in the archive file
3. The final dataset was gathered from twitter REST API via the Tweepy library by querying the API to obtain extra information pertinent to the tweets' ids in the first file, e.g. retweets count and favorite count aspects

Data Assessment

In this step, we investigate our imported datasets both visually and programmatically for quality and tidiness issues.

1. The visual assessment done on spreadsheet application like excel and then the programmatic assessment is conducted in Jupiter notebook.
2. Missing data were addressed first then messy structures were addressed to facilitate the tackling of the rest of the quality issues that fall in the buckets of

References For Helpful Discourse Topics (Project 2)

1. **Gathering For image_prediction dataset.**
2. **Gathering Additional Data via the Twitter API**
3. **Reading the twitter_json.txt**
4. **Accessing your keys and tokens**
5. **How to deal with the alternative files for API data**
    - **Read this whole topic**
    - https://nfpdiscussions.udacity.com/t/data-wrangling-json-i-couldnt-get-the-data-from-twitter-api/42359/2

References For Helpful Discourse Topics

1. **Assessment**
    - Image predictions tidiness issues
    - Classifying the issue as quality or tidiness
    - Example -1 on assessing the data
    - Example -2 on Assessing the name column
2. **Cleaning Archive dataset**
    - adjusting the number of rows in all dataframes
    - Using the dataframes to get rid of retweets and replies
    - The rating numerator issue
    - replacing the string "none" with empty string
    - dog types issue
    - Name column

References For Helpful Discourse Topics

Cleaning Image prediction

Assessment of the dataset

image prediction transformation

Reports structure

Documenting the wrangling effort

Example on the visualizations

General

Comprehensive topic

Questions