itida
هيئة تنمية صناعة تكنولوجيا المعلومات

egyptfwd
initiative

UDACITY

fwd initiative

Project-Bike Share Data
Walk-through

Agenda

Project Overview

Code Walkthrough

1

Project Details

- Technical requirements
- The Datasets
- Statistics Computed
- The Files

2

Workspace & Submission

3

Interactive Experience

The get_filter function

4

Data loading & Statistics Output

6 functions

5

Interactive Raw Data display

display_raw_data(city)

itida

Agenda

Project Overview

Code Walkthrough

1

Project Details

- Technical requirements
- The Datasets
- Statistics Computed
- The Files

2

Workspace & Submission

3

Interactive Experience

The get_filter function

4

Data loading & Statistics Output

6 functions

5

Interactive Raw Data display

display_raw_data(city)

itida

Project Overview

Using Python to explore data related to bike share systems for three major cities in the United States—Chicago, New York City, and Washington. You will write code to:

1. **Import** the data
2. **Answer** interesting questions about it by **computing descriptive statistics**.
3. You will also **write a script** that **takes in raw input** to create an **interactive experience** in the terminal to present these statistics.

Project Details

What Software Do I Need to complete this project locally?:

1. You should have **Python 3**, **NumPy**, and **pandas** installed using **Anaconda**
2. A text editor, like **Sublime** or **Atom**.
3. A terminal application

Project Details

The Datasets:

1. Randomly selected data for the ***first six months of 2017*** are provided for all three cities. All three of the data files contain the same core six (6) columns:
   a. Start Time (e.g., 2017-01-01 00:07:57)
   b. End Time (e.g., 2017-01-01 00:20:53)
   c. Trip Duration (in seconds - e.g., 776)
   d. Start Station (e.g., Broadway & Barry Ave)
   e. End Station (e.g., Sedgwick St & North Ave)
   f. User Type (Subscriber or Customer)
2. The **Chicago** and **New York City** files also have the following two columns:
   a. Gender
   b. Birth Year

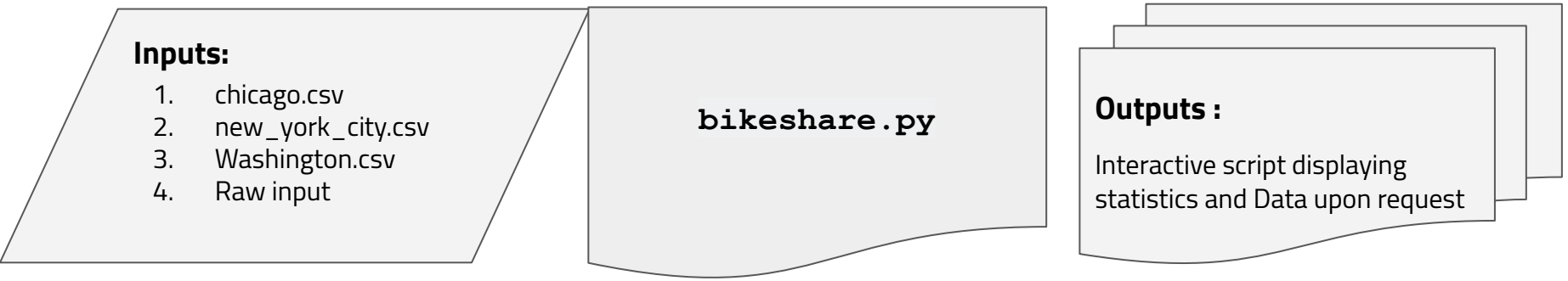
Project Details

Statistics Computed:

1. **Popular times** of travel (i.e., occurs most often in the start time):
    a. most common month
    b. most common day of week
    c. most common hour of day
2. **Popular stations** and trip:
    a. most common start station
    b. most common end station
    c. most common trip from start to end (i.e., most frequent combination of start station and end station)
3. **Trip duration**:
    a. total travel time
    b. average travel time
4. **User info**:
    a. counts of each user type
    b. counts of each gender (**only available for NYC and Chicago**)
    c. earliest, most recent, most common year of birth (only available for NYC and Chicago)

Project Details

The Files:



All four of these files are zipped up in the Bikeshare file in the resource tab

itida
هيئة تنمية صناعة تكنولوجيا المعلومات

Agenda

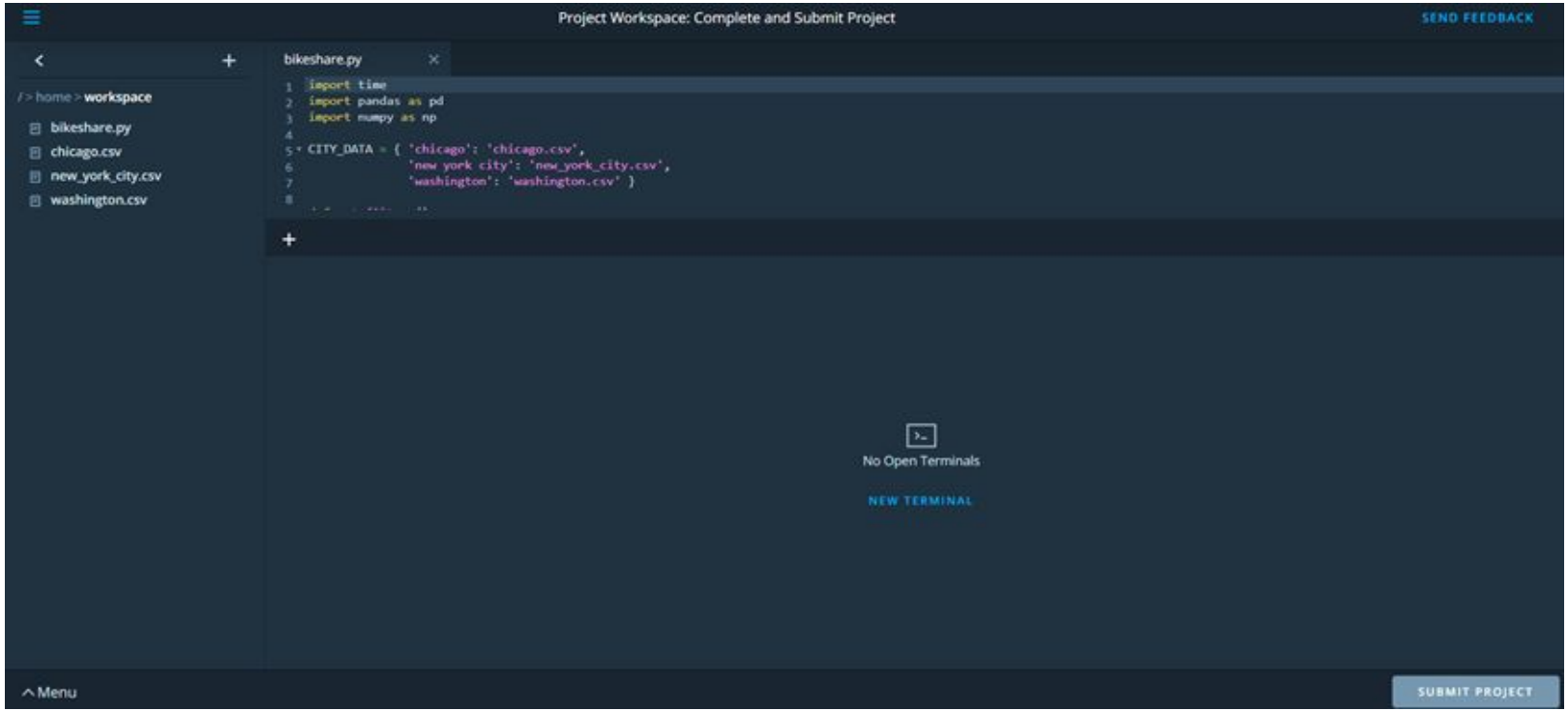
Project Overview

Code Walkthrough

1. **Project Details**
   - Technical requirements
   - The Datasets
   - Statistics Computed
   - The Files
2. **Workspace & Submission**
3. **Interactive Experience**
   The get_filter() function
4. **Data loading & Statistics Output**
   6 functions
5. **Interactive Raw Data display**
   display_raw_data(city)

itida

Workspace & Submission



itida

Workspace & Submission

Before You Submit::

1. **Check the Rubric**:

   Your project will be evaluated by a Udacity reviewer according to this Project Rubric. Be sure to **review it thoroughly before you submit**. Your project "**meets specifications**" only if it meets specifications **in all the criteria**.

2. **Gather Submission Materials:**
   a. `bikeshare.py:` Your code
   b. `readme.txt:` If you refer to other websites, books, and other resources to help you in solving tasks in the project, make sure that you document them in this file

Agenda

Project Overview

Code Walkthrough

1

Project Details

- Technical requirements
- The Datasets
- Statistics Computed
- The Files

2

Workspace & Submission

3

Interactive Experience

The get_filter function

4

Data loading & Statistics Output

6 functions

5

Interactive Raw Data display

display_raw_data(city)

itida

Interactive Experience

There are four questions that will change the answers:

1. **The City input**: Would you like to see data for Chicago, New York, or Washington?
2. **TimeFrame input:** Would you like to filter the data by month, day, or not at all?
3. **Month input (If they chose month):** Which month - January, February, March, April, May, or June?
4. **Day input (If they chose day):** Which day - Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, or Sunday?
5. After filtering the dataset, users will see the statistical result of the data, and ***choose to start again or exit***.

itida

Interactive Experience

Remember:

Any time you ask users for input, there is a chance they **may not enter what you expect**, so your code should **handle unexpected input well without failing**. You need to **anticipate raw input errors** like:

1. Using improper upper or lower case
2. Typos
3. Users misunderstanding what you are expecting.

Use the tips provided in the sections of the Scripting lesson in this course to make sure your code does not fail with an execution error due to unexpected raw input.

itida

The `get_filter()` function

Inputs:
Raw input (City - Timeframe - Which month /Which day)

`Bikeshare.py`

Outputs :

`return(city, month, day)`

Script Setting Up:

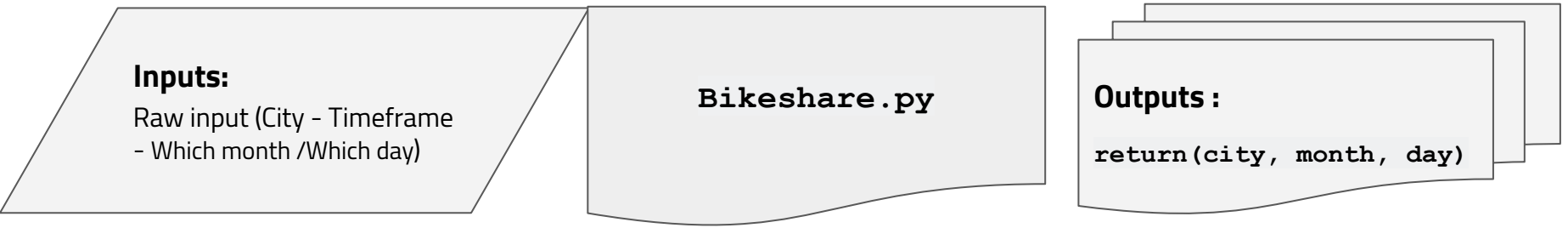
```
import time
import pandas as pd
import numpy as np
```

Importing the required libraries at the top of the script as per the best practices.

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

Assigning a dictionary to map the city names' to the corresponding file name path in the file system to access later within the code.

The `get_filter()` function



Function Purpose:

```
"""
Asks user to specify a city, month, and day to analyze.
Returns:
    (str) city - name of the city to analyze
    (str) month - name of the month to filter by, or "all" to apply no month filter
    (str) day - name of the day of week to filter by, or "all" to apply no day filter
"""
```

- This function is a bit lengthy and can be divided into two or three functions if you would like; (`get_city_filters()`, def `get_time_filters()`) One to return the city, and the one to return a time filters of month and day, or only month or only day or the whole time span.

itida
هيئة تنمية صناعة تكنولوجيا المعلومات

The `get_filter()` function (1)

User Input for City:

- Ask the user: which city to display data about?
- **USER INPUT COLLECTION:**
  - Easy City selection process to minimize the risk of erroneous input potential.
  - In case of not valid input, prompt the user again to enter valid input until the input is accepted.
- **Pseudo code**:

```
def get_filters():
    print('Hello! Let\'s explore some US bikeshare data!')

    # get user input for city (chicago, new york city, washington).
    city_selection = input('To view the available bikeshare data, kindly type:\n The letter
(a) for\
    Chicago\n The letter (b) for New York City\n The letter (c) for Washington\n  ').lower()
```

itida

The `get_filter()` function (1)

User Input for City:

- **USER INPUT VALIDATION:**
  - We have now a city selection as one of the following letters (a, or b, or c) each stands for a specific city.
  - We have a possibility that a user may hit a wrong choice mistakenly so in order to handle this we can use a while loop to help reenter the selection step again:
- **Pseudo code**: *That's one approach.. keep on for a second approach using a while, try, except statement.*

```
while city_selection not in the set of choices of 'a','b','c':
      print('That\'s invalid input.') # tell the user that the input is not right.

      # Ask again with the same above code
      city_selection = input('To view the available bikeshare data, kindly type:\n The letter (a) for Chicago\n The letter (b) for New York City\n The letter (c) for Washington:\n ').lower()
```

The `get_filter()` function (1)

User Input for City:

- **USER INPUT VALIDATION:**
- **Pseudo code**:

```
while True:
        try:
            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  ').lower()

            # Terminate the loop once getting a right answer
            if city_selection in the set of choices of 'a','b','c':
                Terminate the loop
        except:
            print('That\'s invalid input.')
```

The `get_filter()` function (1)

User Input for City:

- **USER INPUT ASSIGNMENT:**
  - As a result, any potential error is now handled and the rest will be to map the selection to the appropriate city
  - This can be done by `if` and `elif` statements
- **Pseudo code**:

```
if city_selection is equal to "a":
        set the variable name  (city) to 'chicago'  # repeat this to each choice.
```

- The same logic can be applied to the time frame selection which is much more involved.
- It's perfectly okay to stop here and `return city`

The `get_filter()` function (2)

User Input for Time filters:

- **USER INPUT COLLECTION (1st stage):**
  - First what're the expected value for time filters: **('month', 'day', 'both', 'none')**, for **none** the whole dataframe will be analyzed and for other choices the data will be filtered to reflect the appropriate time window.
- **Pseudo code**:

```
time_frame = input('\n\n Would you like to filter {}\'s data by month, day, both, or not at all? type month or day or both or none: \n'.format(city.title())).lower()
```

The `get_filter()` function (2)

User Input for Time filters:

- **USER INPUT VALIDATION (1st stage):**
  - Write a while loop to check for the valid choice ('month', 'day', 'both', 'none'); (that's for example you might have a different naming of the values).
  - The purpose of the while loop will be the same as in the city selection part, is to catch invalid answers and ask for input again
- **Pseudo code**:

```
while time_frame not in (choices):
  print(warning message that the choice is not valid)
  Ask for the time_frame input again
```

- You can also use a `try` , `except` block as in we did in the city selection part

The `get_filter()` function (2)

User Input for Time filters:

- **USER INPUT ASSIGNMENT (1st stage):**
  - After ensuring the collection of a valid input, move to the next step of acting on the input. *This step is crucial as you have two options*
  - Either the user didn't specify a `time_frame` in which case you would go ahead and **assign** the values of the **month** and **day** to **all** (the case of the **none** choice).
  - This can be done by `if` and `elif` statements
- **Pseudo code**:

```
If the selected time_frame equals 'none':
    print('\n Filtering for {city} for the 6 months period \n')
    Set month to 'all' for all months
    Set day to 'all' for all days
```

itida

The `get_filter()` function (2)

User Input for Time filters:

- **USER INPUT COLLECTION (2nd stage):**
  - The second possibility is that the user has selected either **day**, or **month** or **both**. In such case you would ***ask for input again*** according to each case. So, for each case you have to tailor the elif statement, for example
  - **Note**: You can write a list of the months named months for example and also one for days just before the if statement. This will make your code a bit less redundant in the upcoming lines.

The `get_filter()` function (2)

Collection Validation Assignment

```
If the selected time_frame equals 'none':
     print('\n Filtering for {city} for the 6 months period \n')
     Set month to 'all' for all months
     Set day to 'all' for all days

elif time_frame == 'both`:
   # Collect User input about the month;
    month_selection = user input for the month by asking to write the month name
   # Then again validate the selection with While loop for the values;
    while month_selection not in (values for months):
        print(warning message that the choice is not valid)
        month_selection = asking for user input again with the same code as above
    # Assign the variable month to the month selection.
    Set month to month_selection
    # Follow the same process as above for the day selection below.
    Collect, validate  & Assign day to day_selection
```

itida

The `get_filter()` function (2)

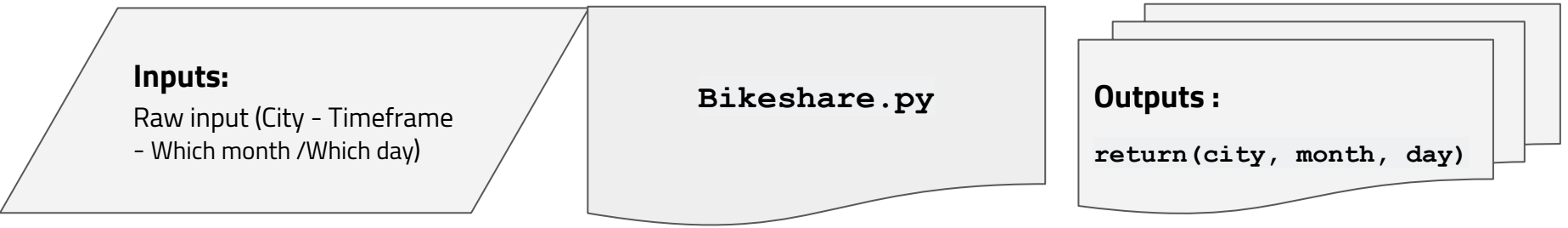

```
elif time_frame = 'month':
    # follow the process for the month as in the case of 'both'
    Set month to month_selection

    # in this case day will be set to all
    Set day to 'all'
-------------------------------------------------------------------------------------------
-----------------------------

elif time_frame = 'day':
     # follow the same sequence of getting and validating the day from the user
    Set day to day_selection

    # Set the month variable to all
     Set month to 'all'
```

The `get_filter()` function



Outputs:

Now, you can set the return statement: `return(city, month, day)`

- **Don't forget**, in your script after you are done writing the `get_filters()` function you call it and assign the result to the variable names that will be used as input for the `load_data()` function like this:

```
filtered_values = get_filters()
city, month, day = filtered_values
```

- Also Take EXTRA CARE of the INDENTATION

Agenda

Project Overview

Code Walkthrough

1

Project Details

- Technical requirements
- The Datasets
- Statistics Computed
- The Files

2

Workspace & Submission

3

Interactive Experience

The get_filter function

4

Data loading & Statistics Output

6 functions

5

Interactive Raw Data display

display_raw_data(city)

itida

Data loading & Statistics Output (Google is your friend)

The `load_data(city, month, day)`:

- Refer to the Concept no 9 "Practice Problem #3" in the classroom.
- This function should return **`df`** which is a dataframe.
- **Don't forget** to call and assign a variable **`df`** to the output

```
load_data(city, month, day)
df = load_data(city, month, day)
```

- Also note that in this function you will need to extract month and day of week from Start Time to create new columns as per the classroom `df['day_of_week'] = df['Start Time'].dt.weekday_name` . But if you work locally use `df['day_of_week'] = df['Start Time'].dt.day_name()` instead.

itida

Agenda

Project Overview

Code Walkthrough

1

Project Details

- Technical requirements
- The Datasets
- Statistics Computed
- The Files

2

Workspace & Submission

3

Interactive Experience

The get_filter function

4

Data loading & Statistics Output

6 functions

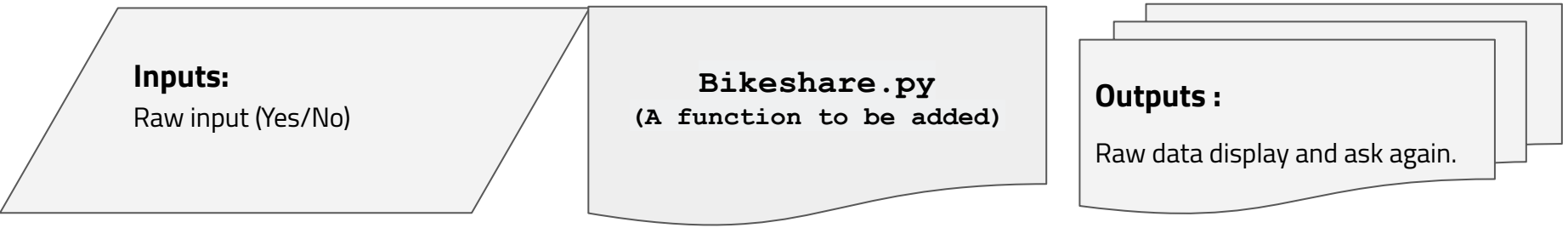
5

Interactive Raw Data display

display_raw_data(city)

itida

Interactive Raw Data display



The `display_raw_data(city)` function:

Your script also needs to prompt the user whether they would like to see the raw data. If the user answers 'yes,' then the script should print 5 rows of the data at a time, then ask the user if they would like to see 5 more rows of the data. The script should continue prompting and printing the next 5 rows at a time until the user chooses 'no,' they do not want any more raw data to be displayed. **(From here you can extract your docstring for the function)**

Interactive Raw Data display

The `display_raw_data(city)` function:

1. Ask for input:

```
# You can print a message like
 print('\n Raw data is available to check... \n')

 # Then we follow a similar process of getting user input and taking action base on it, using the input function
 display_raw = May you want to have a look on the raw data? Type yes or no
```

Interactive Raw Data display

The `display_raw_data(city)` function:

2. **Display 5 rows** and **Loop** to repeat asking if user still needs more input **Using the argument** `chunksize` in the call to `pd.read_csv`

```
    while display_raw == 'yes':
        try:
            looping over the chunks in the pd.read_csv() function with a for loop (chunksize=5):
                print(chunk) # repeating the question
                display_raw = May you want to have a look on the raw data? Type yes or no
                if display_raw != 'yes':
                    print('Thank You')
                    terminate the for loop
            terminate the while loop
        except KeyboardInterrupt:
            print('Thank you.')
display_raw_data(city)
```

itida

I know it's a lot of work and mental exercising but enjoy building your pythonista coding Brain!!