

# Readme File

## Group D2 F

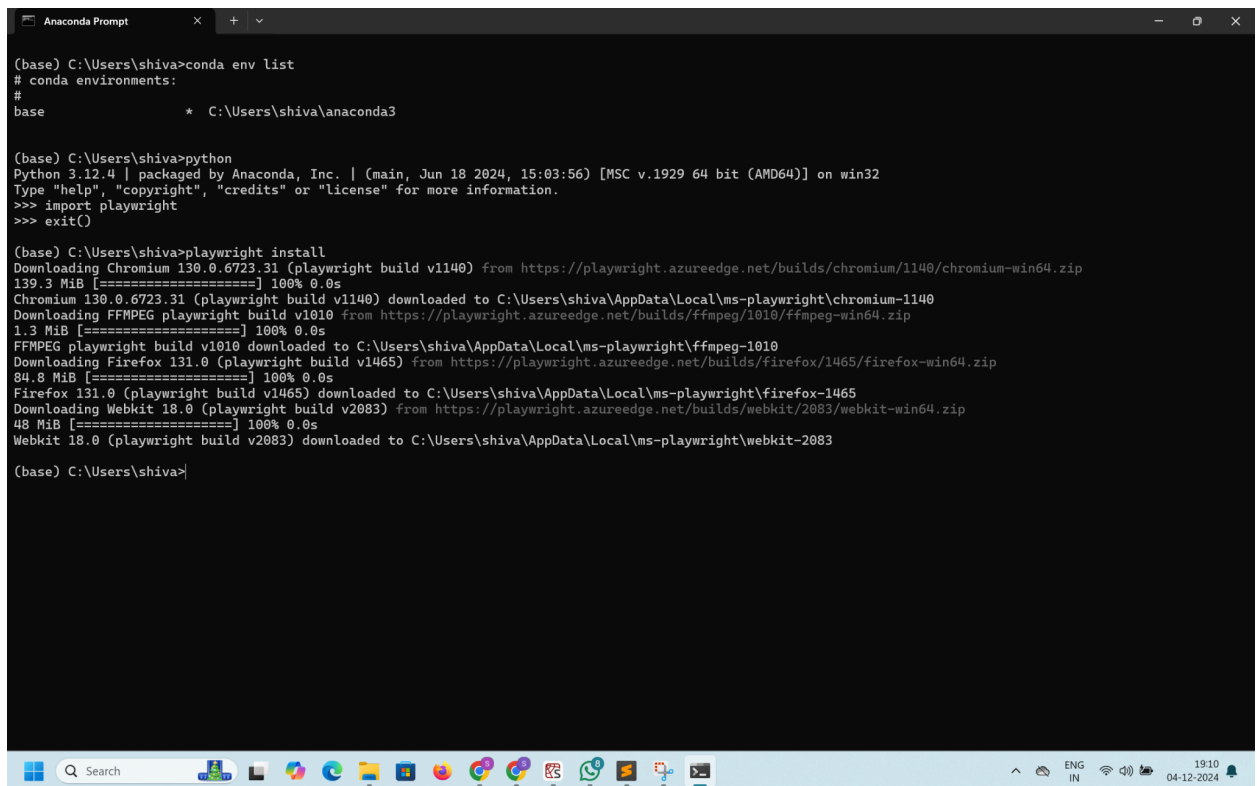
**Members** - Eric Kumara (ekumara@andrew.cmu.edu)  
Mohini Madhur (mmadhur@andrew.cmu.edu)  
Shivani Kale ([sskale@andrew.cmu.edu](mailto:sskale@andrew.cmu.edu))  
Jeevana Desai (jeevanad@andrew.cmu.edu)  
Shruti Ujlan (sujlan@andrew.cmu.edu)

**Product/Application Name** - WanderWise

**Code Demo** - [https://youtu.be/dmi\\_RV5Ax0](https://youtu.be/dmi_RV5Ax0)

## Installation Steps:

1. Install Anaconda
2. Install playwright library using *pip install playwright* in the Anaconda prompt
3. Run command *playwright install* to install the latest build and drivers of the playwright library



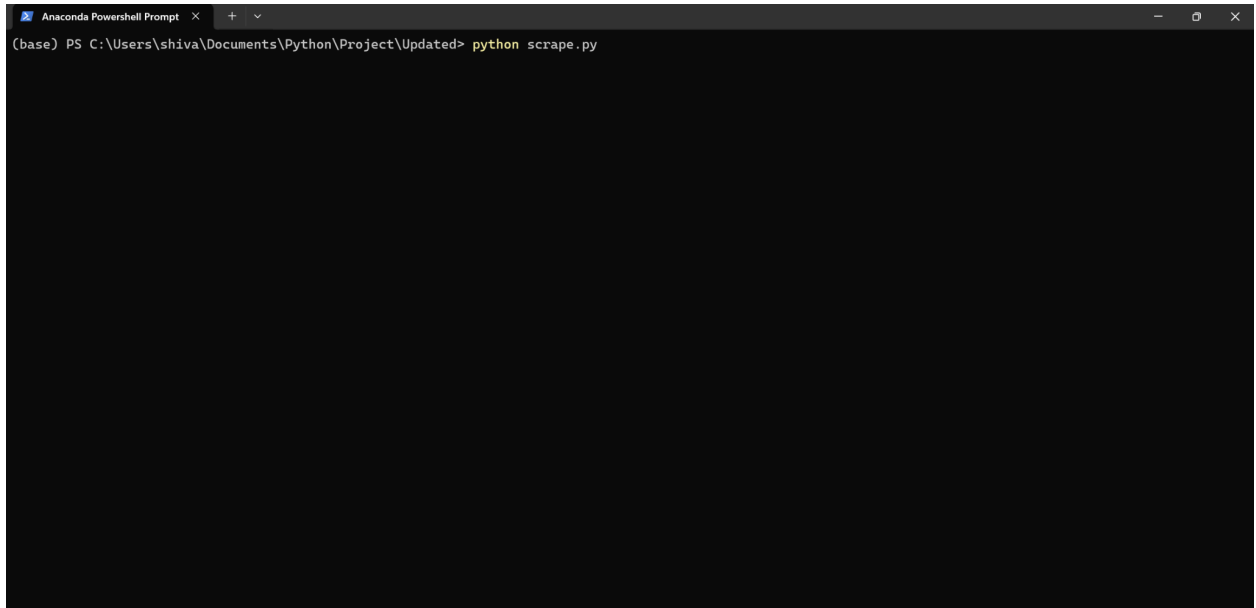
```
(base) C:\Users\shiva>conda env list
# conda environments:
#
base                * C:\Users\shiva\anaconda3

(base) C:\Users\shiva>python
Python 3.12.4 | packaged by Anaconda, Inc. | (main, Jun 18 2024, 15:03:56) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import playwright
>>> exit()

(base) C:\Users\shiva>playwright install
Downloading Chromium 130.0.6723.31 (playwright build v1140) from https://playwright.azureedge.net/builds/chromium/1140/chromium-win64.zip
139.3 MiB [=====] 100% 0.0s
Chromium 130.0.6723.31 (playwright build v1140) downloaded to C:\Users\shiva\AppData\Local\ms-playwright\chromium-1140
Downloading FFMPEG playwright build v1010 from https://playwright.azureedge.net/builds/ffmpeg/1010/ffmpeg-win64.zip
1.3 MiB [=====] 100% 0.0s
FFMPEG playwright build v1010 downloaded to C:\Users\shiva\AppData\Local\ms-playwright\ffmpeg-1010
Downloading Firefox 131.0 (playwright build v1465) from https://playwright.azureedge.net/builds/firefox/1465/firefox-win64.zip
84.8 MiB [=====] 100% 0.0s
Firefox 131.0 (playwright build v1465) downloaded to C:\Users\shiva\AppData\Local\ms-playwright\firefox-1465
Downloading Webkit 18.0 (playwright build v2083) from https://playwright.azureedge.net/builds/webkit/2083/webkit-win64.zip
48 MiB [=====] 100% 0.0s
Webkit 18.0 (playwright build v2083) downloaded to C:\Users\shiva\AppData\Local\ms-playwright\webkit-2083

(base) C:\Users\shiva>
```

4. Open the Anaconda Powershell Prompt to run the code



5. Run command `python scrape.py` to run the application as shown in above image. Ensure that the directory path is the path where all the code files are saved/run from. **The code only runs in the Terminal or IDLE.** It is not compatible with Spyder because Spyder automatically uses the async library whereas the implementation uses the sync library.

#### Source Code breakup:

1. **scrape.py** - Main input file to run, this file imports other scraping python code files as modules and executes the scraping and recommendation functions.
2. **google\_flight.py** - This is the code file that scrapes google flights to get airline name, flight time in minutes and cost.
3. **kayak\_flight.py** - This is the code file that scrapes Kayak website to scrape flight data and get airline name, flight time in minutes and cost.
4. **airbnb.py** - This code file scrapes Airbnb website for hotel/stay recommendations to obtain Name of the Stay, Rating and Pricing
5. **expedia.py** - This code file scrapes Expedia website for hotel/stay recommendations to obtain Name of the Stay, Rating and Pricing
6. **booking\_scrape.py** - This code file scrapes Booking.com website for hotel/stay recommendations to obtain Name of the Stay, Rating and Pricing

## Inputs:

For origin city and destination city, please input city names from the following values because Kayak.com cannot accept custom input and requires specific formats.

```
dict_keys(['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix', 'Philadelphia', 'San Antonio', 'San Diego', 'Dallas', 'San Jose', 'Austin', 'Jacksonville', 'Fort Worth', 'Columbus', 'Indianapolis', 'Charlotte', 'San Francisco', 'Seattle', 'Denver', 'Washington', 'Boston', 'El Paso', 'Nashville', 'Detroit', 'Oklahoma City', 'Portland', 'Las Vegas', 'Memphis', 'Louisville', 'Baltimore', 'Milwaukee', 'Albuquerque', 'Tucson', 'Fresno', 'Mesa', 'Sacramento', 'Kansas City', 'Atlanta', 'Omaha', 'Colorado Springs', 'Raleigh', 'Miami', 'Long Beach', 'Virginia Beach', 'Oakland', 'Minneapolis', 'Tulsa', 'Tampa', 'Arlington', 'New Orleans', 'Wichita', 'Cleveland', 'Bakersfield', 'Aurora', 'Anaheim', 'Honolulu', 'Santa Ana', 'Riverside', 'Corpus Christi', 'Lexington', 'Henderson', 'Stockton', 'St. Paul', 'Cincinnati', 'St. Louis', 'Pittsburgh', 'Greensboro', 'Lincoln', 'Anchorage', 'Plano', 'Orlando', 'Irvine', 'Newark', 'Durham', 'Chula Vista', 'Toledo', 'Fort Wayne', 'St. Petersburg', 'Laredo', 'Jersey City', 'Chandler', 'Madison', 'Lubbock', 'Scottsdale', 'Reno', 'Buffalo', 'Gilbert', 'Glendale', 'North Las Vegas', 'Winston-Salem', 'Chesapeake', 'Norfolk', 'Fremont', 'Garland', 'Irving', 'Hialeah', 'Richmond', 'Boise', 'Spokane', 'Baton Rouge', 'Tacoma'])
```

The user is prompted to give below inputs which are used to scrape the data:

1. **Origin City** - User has to input the city name alone. For example, if the city is 'New York City', please enter only 'New York' as input
2. **Destination City** - User has to input the name of the city where they wish to fly or stay.
3. **CheckIn Date** - The date the user plans to fly or check into a hotel/stay in MM/DD/YYYY format
4. **CheckOut Date** - The date the user plans to fly back or check out of a hotel/stay in MM/DD/YYYY format
5. **Number** - Number of people that are travelling to obtain proper costs, this has to be an integer.

```
Anaconda Powershell Prompt x + v
(base) PS C:\Users\shiva\Documents\Python\Project\Updated> python scrape.py
Enter the Origin city (Please write just the city name. Ex:For New York City, just enter New York):
Boston
Enter the Destination city (Please enter the city name only):
Pittsburgh
Enter checkin date of journey in MM/DD/YYYY format:
12/20/2024
Enter the checkout date in MM/DD/YYYY format:
01/03/2025
Enter the number of people travelling:
4
```

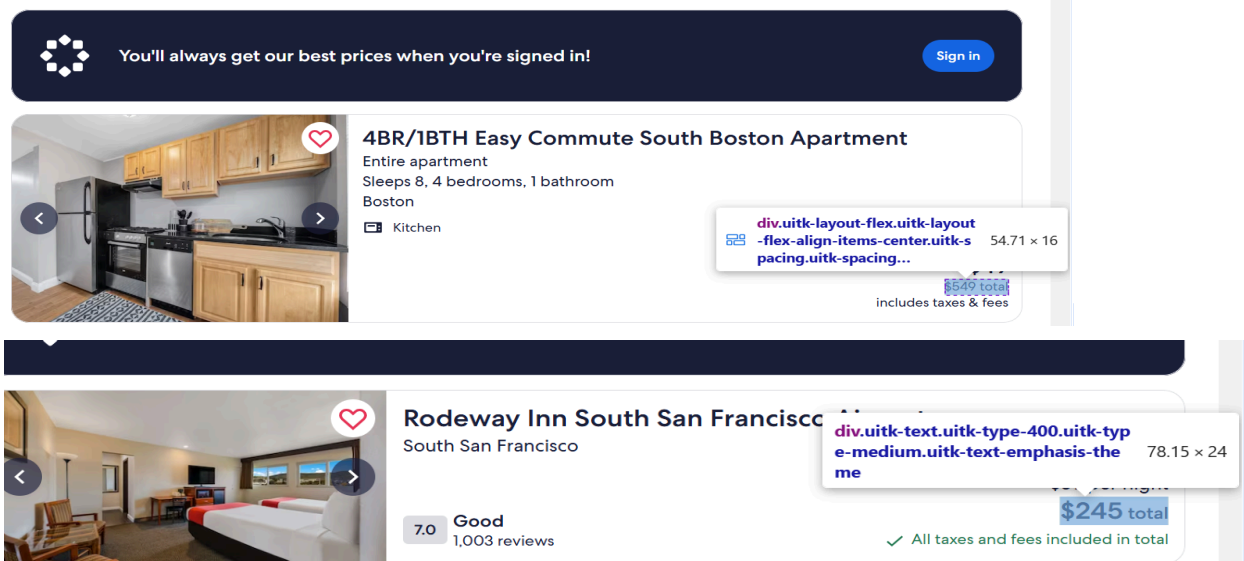
## Output:

Post scraping the user will be asked to submit his preference about rating/cost/rating per dollar and flight and stay recommendations will be displayed accordingly to the user as seen in the demo.

### Code Edge cases:

Scraping data from dynamic websites has proven to be challenging. **Note that due to the nature of dynamic scraping, encountering errors will be normal.** The code works completely fine except for a few edge cases:

- The real time scraping gives either a limited number of outputs to write to the file, or does not work; in which case, no data is written to the scraped file.
- Some websites have rate limiting implemented which hinders scraping and gives an error when scraping which also leads to no data written in file, giving errors when plotting and running the code. (See screenshot below). A work around is to use a VPN
- Scraping is only valid for current websites of Google Flight, Expedia, Airbnb and Kayak. Scraping with Dynamic Websites requires hard coding the identifiers of div boxes, and it could change. For example, in Black Friday, the UI of Expedia for California changed so the total price entry is in a different div box.**



- Due to unstable internet connection, scrapping for a particular input set might not work on the first try. It is recommended to run the code again or ensure the code is run with proper and fast internet connection.

- e. In case blocked from scrapping, connect to a different WiFi network and then run the code (The scraping did not work after a few tries on a home network, but ran perfectly fine when joined the CMU WiFi network).

```
Anaconda Powershell Prompt
12/20/2024
Enter the checkout date in MM/DD/YYYY format:
01/03/2025
Enter the number of people travelling:
4
Scraping Google Flight
Done
Scraping Kayak
Done
Scraping Airbnb
Done
Scraping Expedia
https://www.expedia.com/Hotel-Search?destination=Pittsburgh&d1=2025-01-03&startDate=2024-12-20&d2=2025-01-03&endDate=2025-01-03&adults=4&rooms=1&theme=&userIntent=&semdtl=&useRewards=false&sort=PRICE_LOW_TO_HIGH&children=&latLong=&mapBounds=&spaDialog=
Done
Scraping Booking.com
Done
Traceback (most recent call last):
  File "C:\Users\shiva\Documents\Python\Project\Updated\scrape.py", line 164, in <module>
    expedia['Cost'] = expedia['Cost'].astype('int64')
    ~~~~~^~~~~~
  File "C:\Users\shiva\anaconda3\Lib\site-packages\pandas\core\generic.py", line 6643, in astype
    new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
    ~~~~~^~~~~~
  File "C:\Users\shiva\anaconda3\Lib\site-packages\pandas\core\internals\managers.py", line 430, in astype
    return self.apply(
    ~~~~~^~~~~~
  File "C:\Users\shiva\anaconda3\Lib\site-packages\pandas\core\internals\managers.py", line 363, in apply
    applied = getattr(b, f)(**kwargs)
    ~~~~~^~~~~~
  File "C:\Users\shiva\anaconda3\Lib\site-packages\pandas\core\internals\blocks.py", line 758, in astype
    new_values = astype_array_safe(values, dtype, copy=copy, errors=errors)
    ~~~~~^~~~~~
  File "C:\Users\shiva\anaconda3\Lib\site-packages\pandas\core\dtypes\astype.py", line 237, in astype_array_safe
    new_values = astype_array(values, dtype, copy=copy)
    ~~~~~^~~~~~
  File "C:\Users\shiva\anaconda3\Lib\site-packages\pandas\core\dtypes\astype.py", line 182, in astype_array
    values = _astype_nansafe(values, dtype, copy=copy)
    ~~~~~^~~~~~
  File "C:\Users\shiva\anaconda3\Lib\site-packages\pandas\core\dtypes\astype.py", line 133, in _astype_nansafe
    return arr.astype(dtype, copy=True)
    ~~~~~^~~~~~
TypeError: int() argument must be a string, a bytes-like object or a real number, not 'NoneType'
(base) PS C:\Users\shiva\Documents\Python\Project\Updated>
```