PAGOD KA NA BA SA PAULIT-ULIT NA...

SAAN?

KAHIT SAAN!

IKAW BAHALA!

INSPIRATATION

O DI KAYA...

Mas mahaba pa yung oras magdecide kesa sa travel time? Yung tipong nasa area na pero wala pa ring final decision?

Lagi na lang nauuwi sa iisang spot?

INSPIRATATION

Travel

TANGO

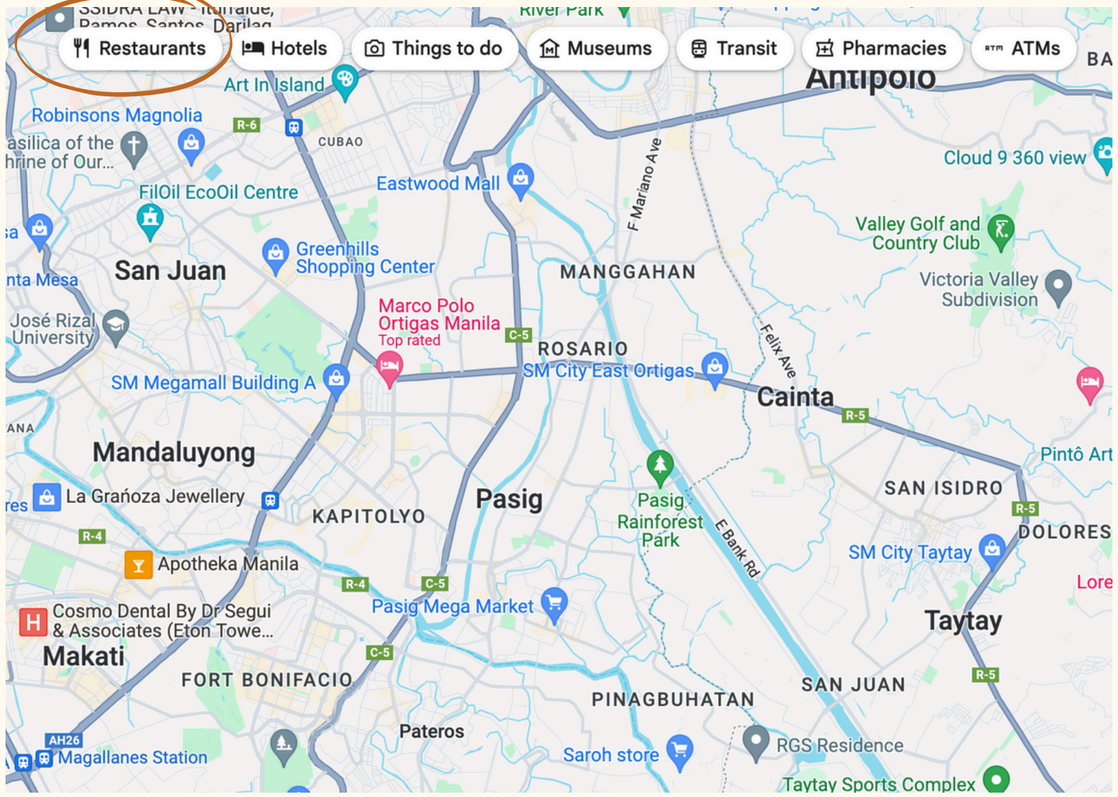
Macua, Nevado, Zapata

GOAL

provide a user-friendly solution that empowers people to easily discover and enjoy new or familiar locations, making outings more enjoyable and hassle-free

DISCOVER MORE

Google Maps



OVERVIEW

01 ASK FOR LOCATION

02 USE GOOGLE MAPS TO FILTER OPTIONS

03 EXTRACT RELEVANT DETAILS

04 STORE INFO IN DICTIONARIES

05 CONVERT TO DATAFRAMES.

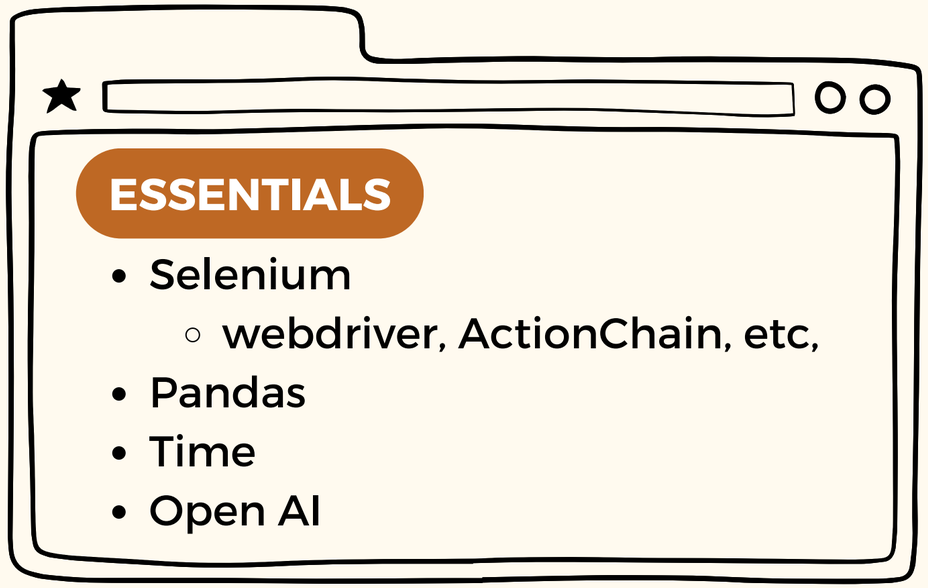
06 ASK FOR TIME AND PREFERENCES

07 CREATE PROMPT FOR CHATGPT

08 GENERATE ITINERARY

THE CODES

exploring the code, key functions, and challenges, leading to an efficient solution



ESSENTIALS

- Selenium
  - webdriver, ActionChain, etc,
- Pandas
- Time
- Open AI

```
#Fetch and set up version 0.28 of the OpenAI Python library
!pip install openai==0.28
```

```
import time
import pandas as pd
import openai
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver import ActionChains
from selenium.webdriver.support.ui import We
from selenium.webdriver.common.actions.whee
from selenium.webdriver.support import expec
from selenium.common.exceptions import Timed
```

Input Function

USER'S LOCATION

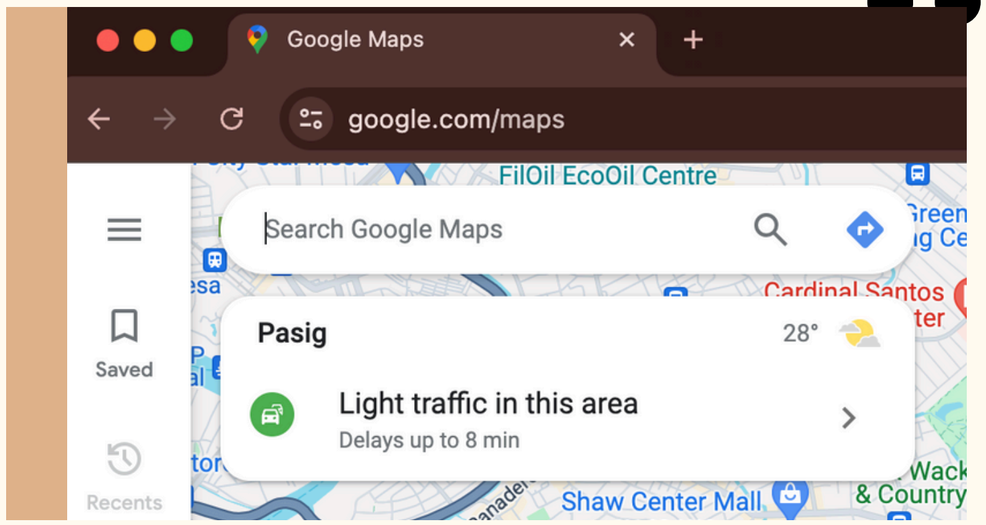
```
:  #Ask for the location

   user_input = input("Enter the location to search on Google Maps (be as specific
```

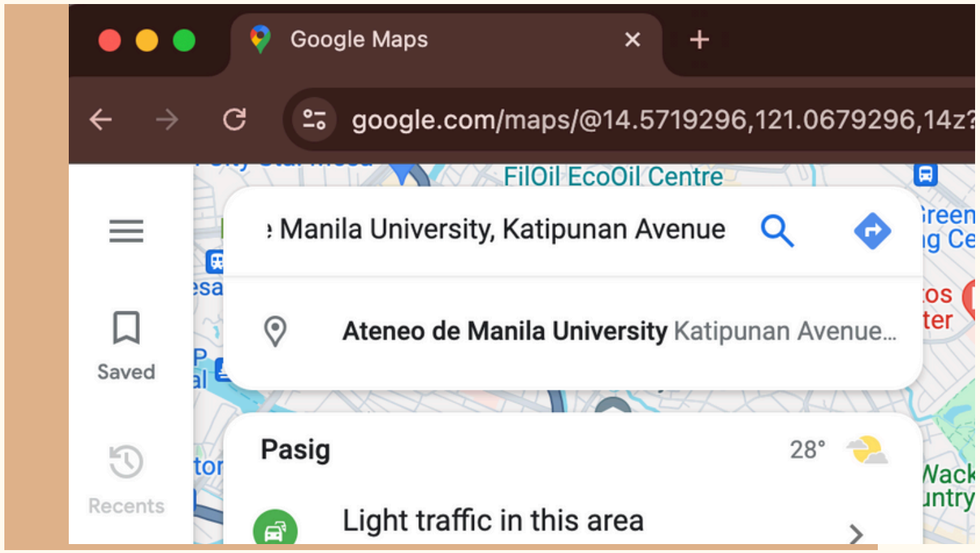
- Specificity - Landmarks, Detailed Address (eg. Ateneo de Manila University, Katipunan Avenue, Quezon City)

1

Browser, Google Maps and Time Sleep

```
# Initialize the browser
browser = webdriver.Chrome()

# Navigate to Google Maps
browser.get('https://www.google.co

# Wait for the page to load
time.sleep(3)
```

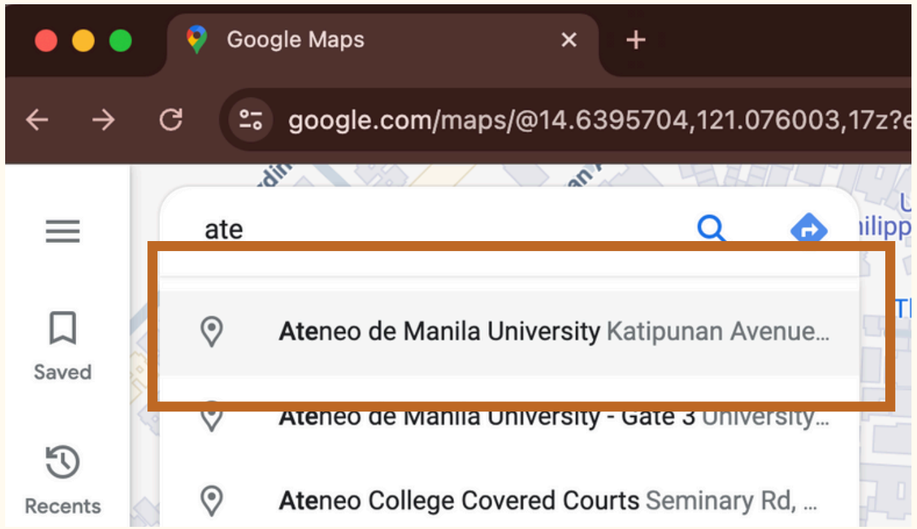




2

Searching Location

```
# Find the search bar and enter the user input
search_bar = browser.find_element(By.XPATH, '/
search_bar.send_keys(user_input)

# Wait for the suggestions to load and appear
time.sleep(2)
```



3

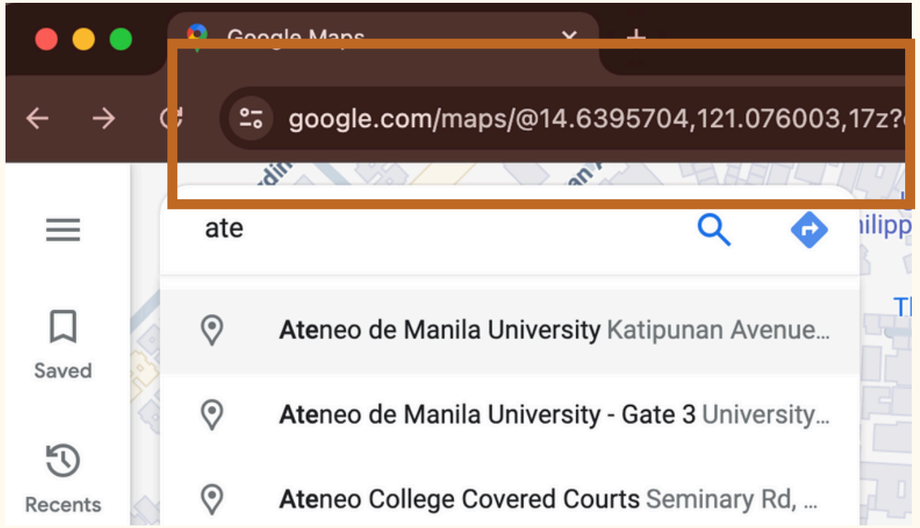
Locating First Suggestion

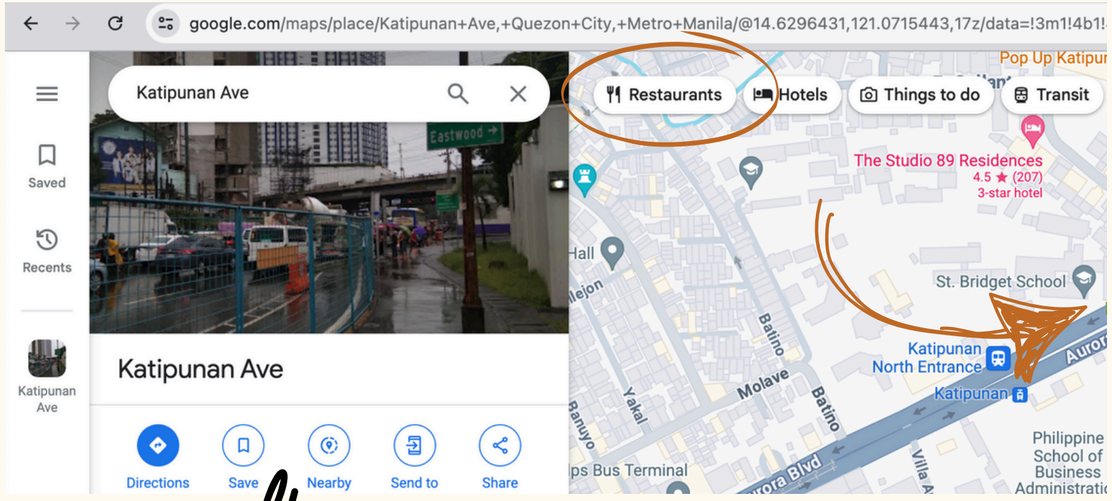
```
# Wait for the suggestions to load and appear
time.sleep(2)

# Find and click the first suggestion
try:
    first_suggestion = browser.find_element(By.CLASS_NAME, '
    first_suggestion.click()
except:
    time.sleep(1)
# Wait for the place to load and appear
time.sleep(2)
```

4

Saving the Browser Address

```
# store the address for future use (if there is)

loc_address = browser.find_element(By.CSS_SELECTOR, 'span.DkEa
```



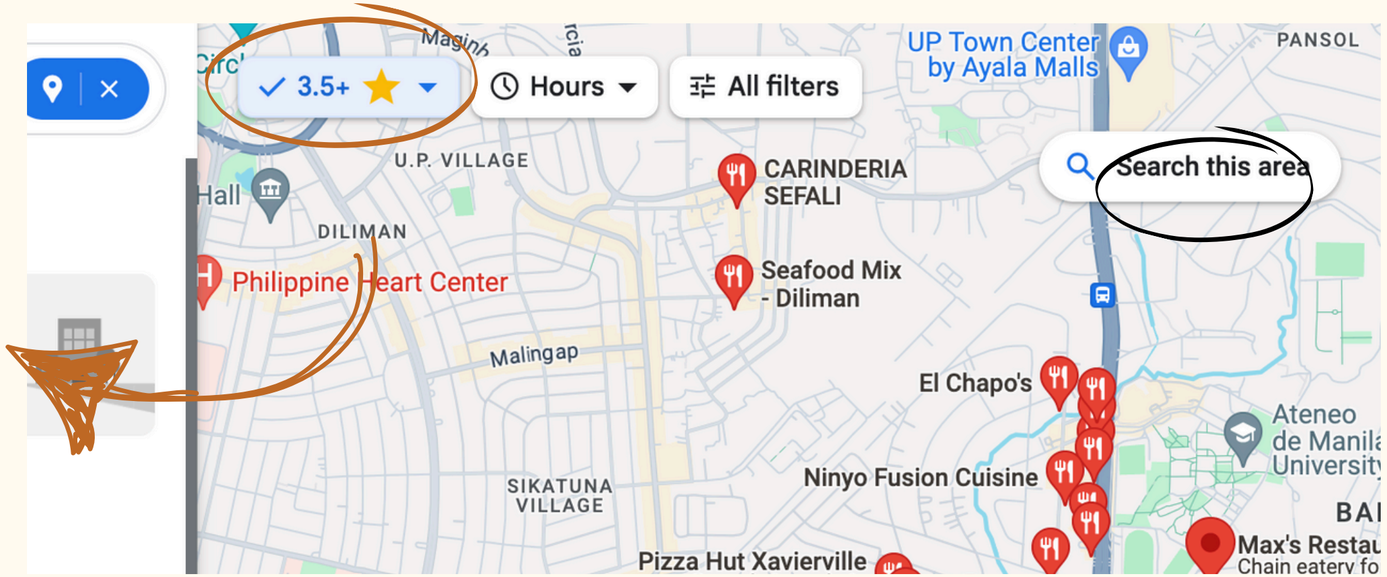


clicking category

```python
# Click Restaurant option
restaurants_option = browser.find_element(By.XPATH, '//button[@aria-
restaurants_option.click()

# Wait for all restaurants to load and appear
time.sleep(2)
```

filtering

```python
#filter 3.5*only
rating = browser.find_element(By.XPATH, '//*[@id="ass stive-chips"]/div/div/div/div[1]/div/div/
rating.click()

time.sleep(2)

rating = browser.find_element(By.XPATH, '//*[@id="action-menu"]/div[5]')
rating.click()
time.sleep(2)

#Zoom out function so that the search area button works
zoom_out = browser.find_element(By.XPATH, '//*[@id="widget-zoom-out"]/div')
zoom_out.click()

time.sleep(1)

#Search Area button
search_area = browser.find_element(By.XPATH, '//*[@id="search-this-area"]/div/button/span')
search_area.click()
```



DICTIONARY AND SCROLLING

```python
# Initialize a dictionary to store restaurant details
restaurants_dict = {}

# Scroll and gather restaurant details
while True:
    # Get all restaurant elements
    restaurant_list = browser.find_elements(By.CLASS_NAME, 'Nv2PK')
    action = ActionChains(browser)
    scroll_origin = ScrollOrigin.from_element(restaurant_list[len(restaurant_list)-1])
    action.scroll_from_origin(scroll_origin, 0, 2000).perform()
    time.sleep(1)
    new_restaurant_list = browser.find_elements(By.CLASS_NAME, 'Nv2PK')
    if len(new_restaurant_list) == len(restaurant_list):
        break
```

*initially used a for loop but did not capture all elements

While Loop

continuously retrieve elements until no new results

Find Element

get all the restaurant as basis for scrolling and looping

Action Chain and Scroll Origin

scroll continuous based on a given origin

```python
#clicks each element then extracts correspond
for i in range(len(restaurant_list)):
    action = ActionChains(browser)
    scroll_origin = ScrollOrigin.from_element
    action.scroll_from_origin(scroll_origin,
    time.sleep(1)
    action.move_to_element(restaurant_list[i]
    restaurant_list[i].click()
    time.sleep(3)
```

clicking each to EXTRACT

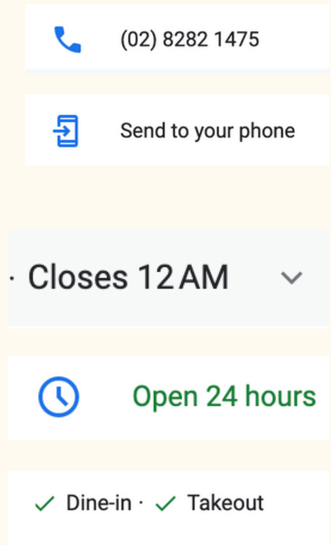
filtering based on REVIEWS

```python
try:
    review_element = WebDriverWait(browser, 5).until(
        EC.visibility_of_element_located((By.CSS_SELECTOR, 's
    )
        # Extract the number of reviews from the text
except:
    continue

    # Extract the number of reviews from the text
num_of_reviews = int(review_element.text[1:-1].replace(',', '
if num_of_reviews <= 65:
    continue
```

(531)

getting the INFO

name, address, contact number, operating hours and rating



Name

```python
#Restaurant Name
restaurant_name = browser.find_element(By.CSS_SELECTOR, '.a5H0ec').find_element(By.XPATH, './..').text
```

Rating

```python
#Restaurant Rating
restaurant_rating_element = WebDriverWait(browser, 2).until(
    EC.visibility_of_element_located((By.XPATH, '//*[@id="QA0Szd"]/div/div/div[1]/div[3]/div/div[1]/div/div/div[
)
restaurant_rating = restaurant_rating_element.text
```

Address

```python
#Restaurant Address
restaurant_address = browser.find_element(By.CLASS_NAME, 'Io6YTe.fontBodyMedium.kR99db').text
```

Contact Number

```python
#Restaurant Contact Number
try:
    restaurant_contact = browser.find_element(By.CSS_SELECTOR, 'button[data-item-id^="phone:"] .Io6YTe.fontBodyMed
except:
    restaurant_contact = "N/A"
```

Operating Hours

```python
#Restaurant Hours
try:
    restaurant_hours = browser.find_element(By.CSS_SELECTOR, 'span.ZDu9vd').text
    if "Closes" in restaurant_hours:
        restaurant_hours = restaurant_hours.replace('\u202f', ' ').strip()[7:]

except:
    try:
        restaurant_hours = browser.find_element(By.CSS_SELECTOR, 'div[aria-label="Open 24 hours"]').text
    except:
        restaurant_hours = "N/A"
```

storing in the

DICTIONARY

```
restaurants_dict[restaurant_name] = {'Rating': restaurant_rating, 'Address': restaurant_address, 'Contact': restaurant_contact, 'Operating
```

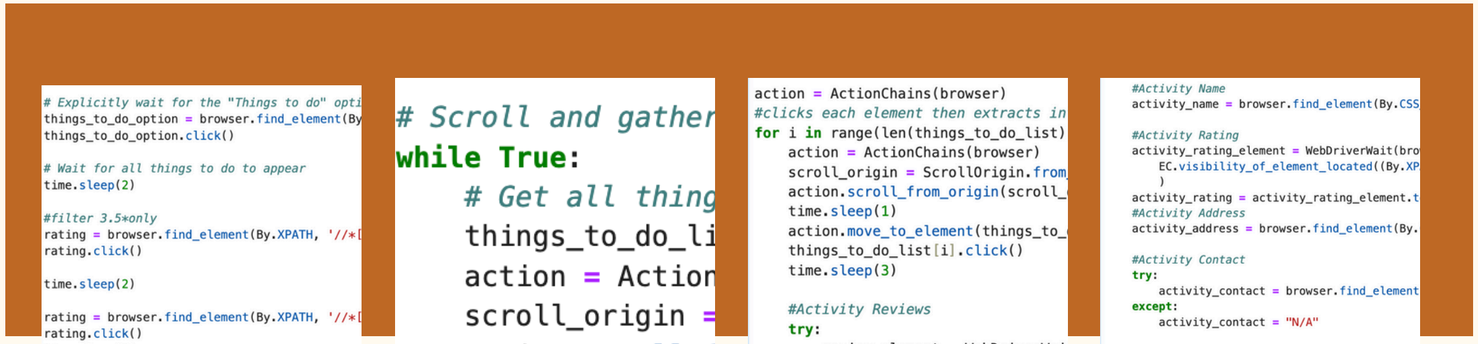
NAVIGATING BACK

```
#Go back to the previous page (location view)
back = browser.find_element(By.XPATH, '//*[@id="searchbox"]/div[2]/div/button/span')
back.click()

# Wait for the page to navigate back
time.sleep(2)
```

Same Process for

THINGS TO DO

*just different variable names



```
# Explicitly wait for the "Things to do" opti
things_to_do_option = browser.find_element(By
things_to_do_option.click()

# Wait for all things to do to appear
time.sleep(2)

#filter 3.5*only
rating = browser.find_element(By.XPATH, '//*[
rating.click()

time.sleep(2)

rating = browser.find_element(By.XPATH, '//*[
rating.click()
time.sleep(3)

#Zoom out function so that the search area bu
zoom_out = browser.find_element(By.XPATH, '//
zoom_out.click()

time.sleep(2)

#Search Area button
search_area = browser.find_element(By.XPATH,
search_area.click()

time.sleep(2)
```

Clicking and Filtering

```
# Scroll and gather
while True:
    # Get all thing
    things_to_do_li
    action = Action
    scroll_origin =
    action.scroll_f
    time.sleep(3)
    new_things_to_d
    if len(new_thin
        break
```

While Loop and Scrolling

```
action = ActionChains(browser)
#clicks each element then extracts in
for i in range(len(things_to_do_list)
    action = ActionChains(browser)
    scroll_origin = ScrollOrigin.from
    action.scroll_from_origin(scroll_
    time.sleep(1)
    action.move_to_element(things_to_
    things_to_do_list[i].click()
    time.sleep(3)

    #Activity Reviews
    try:
        review_element = WebDriverWai
            EC.visibility_of_element_
            )
            # Extract the number of r
    except:
        continue

        # Extract the number of revie
    num_of_reviews = int(review_eleme
    if num_of_reviews <= 100:
        continue
```

For Loop and Further Filtering

```
    #Activity Name
    activity_name = browser.find_element(By.CSS

    #Activity Rating
    activity_rating_element = WebDriverWait(bro
        EC.visibility_of_element_located((By.XP
        )
    activity_rating = activity_rating_element.t
    #Activity Address
    activity_address = browser.find_element(By.

    #Activity Contact
    try:
        activity_contact = browser.find_element
    except:
        activity_contact = "N/A"

    #Activity Hours
    try:
        activity_hours = browser.find_element(B
        if "Closes" in activity_hours:
            activity_hours = activity_hours.rep

    except:
        try:
            activity_hours = browser.find_eleme
        except:
            activity_hours = "N/A"

    things_to_do_dict[activity_name] = {'Rating
```

Extracting Info and Storing

navigates back twice to clear search

```
browser.back()
browser.back()

time.sleep(2)
search_bar.clear()
```

type in and click earch bar; use if-else for precision (loc address based on first suggestion vs user input)

```
# Type in the search bar to search for shopping malls
if loc_address.count(',') > 1:
    search_bar.send_keys("Shopping Malls near " + loc_address)

else:
    search_bar.send_keys("Shopping Malls near " + user_input)

time.sleep(1)

# Press the search button
search = browser.find_element(By.XPATH, '//*[@id="searchbox-searchbutton"]/span')
search.click()

time.sleep(3)
```

> 1000 reviews to avoid irrelevant results

```
    if num_of_reviews <= 1000:
```

shopping mall

SAME AS OTHERS, EXCEPT:

- Manual searching vs. button clicks
- Number of reviews considered

Going back and MAKING DATAFRAMES

```
# Close the browser
browser.quit()
```


{'Kanto Freestyle Breakfast La Salle': {'Rating': '4.4', 'Address': '2466 Leon Guinto St, Malate, Manila, 1004 Metro Manila', 'Contact': '0919 478 8055', 'Operating Hours': 'Open 24 hours'}, "Zark's Burgers": {'Rating': '4.2', 'Address': '2464 Taft Ave, Malate, Manila, 1000 Metro Manila', 'Contact': '(02) 3392 1211', 'Operating Hours': 'Closes 10 PM'}, 'El Poco Cantina': {'Rating': '4.7', 'Address': '945 Estrada St, Malate, Manila, 1004 Metro Manila', 'Contact': '0920 980 0945', 'Operating Hours': 'Closes 10:30 PM'}, 'Ababu': {'Rating': '3.9', 'Address': '931 Estrada St, Malate, Manila, 1004 Metro Manila', 'Contact': 'N/A', 'Operating Hours': 'Closes 11:30 PM'}, 'Arabic house': {'Rating': '4.9', 'Address': '912 P.Ocampo St., Malate, Manila, 1104 Metro Manila', 'Contact': '0915 633 1309', 'Operating Hours': 'N/A'}, 'Chomp Chomp': {'Rating': '4.3', 'Address': '2nd Floor, Bellago Residence, 2450 Leon Guinto St, Malate, Manila, 1004 Metro Manila', 'Contact': '(02) 8775 7028', 'Operating Hours': 'Closes 10 PM'}, 'Angrydobo': {'Rating': '3.7', 'Address': '2456, 1004 Taft Ave, Malate, Manila, 1004 Metro Manila', 'Contact': '0968 619 4971', 'Operating Hours': 'Closes 9 PM'}, 'Tinuhog Ni Benny Barbecue House': {'Rating': '4.4', 'Address': '879 Dagonoy St, Malate, Manila, 1004 Metro Manila', 'Contact': '(02) 8253 1072', 'Operating Hours': 'Closes 2:30 AM'}}
{'Manila Zoo': {'Rating': '3.7', 'Address': 'Adriatico St, Malate, Manila, 1004 Metro Manila', 'Contact': '0909 836 6911', 'Operating Hours': 'Closes 6 PM'}, 'Upside Down Museum': {'Rating': '4.3', 'Address': 'Boom Na Boom Grounds CCP Complex, Roxas Blvd, Pasay, 1300 Metro Manila', 'Contact': '(02) 8551 5


```
# Convert the restaurants dictionary to a DataFrame and add a new column for type
df_restaurants = pd.DataFrame.from_dict(restaurants_dict, orient='index')
df_restaurants.reset_index(inplace=True)
df_restaurants.rename(columns={'index': 'Name'}, inplace=True)
df_restaurants['Type'] = 'Restaurant'
```

Name	Rating	Address	Con
usion Cuisine	4.7	66 Esteban Abada St, Quezon City, 1108 Metro M...	(02) 8
Aurora Blvd.	4.3	991 Aurora Blvd, Project 3, Quezon City, 1102 ...	
aw Katipunan	4.5	The Pop Up, Katipunan Avenue corner, Xaviervil...	0917 772
rsian Kitchen	4.2	75 Xavierville Ave, Quezon City, 1108 Metro Ma...	(02) 7506
ger + Burrito	4.3	299 Katipunan Ave, Quezon City, 1108 Metro Manila	(02) 8282
atipunan Ave	4.2	291 Katipunan Ave, Quezon City, Metro Manila	(02) 8 9
su Katipunan	4.5	Unit 3A, 3rd floor The Oracle Hotel & Residenc...	0977 855
Greenwich	4.4	A. BONIFACIO AVENUE, RIVERBANKS CENTER, BARANG...	
ers Roasters	4.4	303 Katipunan Ave, Quezon City, 1108 Metro Manila	(02) 3435

ask for

INFO

Any specific location and how may attractions to visit. If none is given, then the locations would be randomized.

```
# Asks if there are specific locations the person wants to visit and how many attractions to visit
# If not given enough specific locations, it will perform a random search
no_attraction = int(input("How many attractions do you want to visit?"))
while no_attraction > len(df_things_to_do):
    print("Not enough locations to support the number")
    no_attraction = int(input("How many attractions do you want to visit?"))
```

Check if the number of locations they want to visit is more than the provided locations

```
list_attractions = []
specific_attraction = ''

# Loop to gather specific attractions requested by the user
while  len(list_attractions) < no_attraction:
    specific_attraction = input('Any specific attractions to visit? (quit if done)')

    if specific_attraction.upper() == 'QUIT':
        break

    if not df_things_to_do[df_things_to_do['Name'] ==  specific_attraction].empty:

        # Extract details of the specific things to do and add to list
        row = df_things_to_do[df_things_to_do["Name"] == specific_attraction].to_dict()
        key = list(row['Name'].keys())[0]
        attraction = "Name: " + row['Name'][key] + "\nRating: " + row['Rating'][key] + "\nAddress: "
        list_attractions.append(attraction)

        # Remove selected things to do from DataFrame
        df_things_to_do = df_things_to_do.drop(index = key)

    else:
        print('Location is not in the dataframe')
```

Asks the user for any specific location gets the data, and removes it from the dataframe

Cleaning the SPECIFIED LOCATIONS

```
#Subtract the number of specific locations to the number of locations wanted
no_attraction = no_attraction - len(list_attractions)

# Randomly sample remaining things to do from DataFrame
rows = df_things_to_do.sample(n=no_attraction)
```

RANDOMIZING AND GETTING THE DETAILS OF EACH RANDOMLY SELECTED LOCATIONS

```
# Loop to gather randomly selected things to do
for key in rows.get('Name').keys():
    attraction = "Name: " + rows['Name'][key] + "\nRating: " + rows['Rating'][key] + "\nAddress: " + rows['Address'][key] + "\nContact: " + ro

    # Combine specific and randomly selected things to do into one list
    list_attractions.append(attraction)
```

Note: This process is done for all restaurants, attractions, and shopping malls. This code above only shows the code for attractions

```python
# Setting the OpenAI API key for authentication
openai.api_key = 'sk-proj-SfijTyxZbOoiin8ugrOoT3BlbkFJeeadSi5R3X5F5DUTx9I7'

def generate_itinerary(restaurants, attractions, shopping_malls, time1, time2):
    # Prompt for generating the itinerary
    prompt = ("Create a one-day itinerary lasting from " +time1+ " to " +time2+ " using all of
        "Restaurants:\n\n" + "\n\n".join(restaurants) + "\n\n"
        "Attractions:\n\n" + "\n\n".join(attractions) + "\n\n"
        "Shopping Malls:\n\n" + "\n\n".join(shopping_malls) + "\n\n")

    # Generate itinerary and get response using OpenAI's ChatGPT model
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "user", "content": prompt}
        ],
        max_tokens=1000,
        temperature=0.7
    )
    # Return the generated itinerary
    return print(response['choices'][0]['message']['content'].strip())

# Collect input for trip start and end times
time1 = input("What time does the trip start? (00:00 24 hour format)")
time2 = input("What time does the trip end? 00:00 24 hour format)")


generate_itinerary(list_restaurants, list_attractions, list_shopping_mall, time1, time2)
```

Connecting to CHATGPT

- Uses the openAI API
- Creates a defined function

EXECUTION

final result

```
generate_itinerary(list_restaurants, list_attractions, list_shopping_mall, time1, time2)
```

```
What time does the trip start? (00:00 24 hour format) 10:00
What time does the trip end? 00:00 24 hour format) 18:00
10:00 AM - Start your day by visiting The Dessert Museum in Pasay. Indulge in a sweet experience and take some Instagram-worthy photos.

12:00 PM - Head over to Ababu for a delicious lunch. Try their specialties and enjoy the cozy atmosphere.

2:00 PM - After lunch, take a leisurely stroll along Manila Baywalk Dolomite Beach. Enjoy the scenic views of the bay and relax by the water.

4:00 PM - Make your way to Victory Pasay Mall for some shopping. Explore the shops and find some souvenirs to take home.

6:00 PM - End your day with a hearty dinner at Kenny Rogers Roasters. Enjoy a delicious meal before heading back to your accommodation.
```

LIMITATIONS/RECOMMENDATIONS

01 Slow and breaks down ocasionally—consider batch processing, optimize scroll distances, and minimize time sleep

02 Data heavily relies on Google Maps—enhance validation and cross-checking.

03 Not as comprehensive in scheduling—need features for distance, shortest travel time without compromising runtime

04 Limited customization/user interaction—introduce advanced filtering (price, etc.), user profiles, and save preferences/history for predictive machine learning

THANK YOU!

MACUA, NEVADO, ZAPATA,