# COIT20245 Introduction to Programming

Assignment 2 Project

Group Members

Priyanka Narsareddynoor(12267145)

Ashish Vardhan Malyala (12264442)

Koushik Reddy Mupiddi (12264186)

# 

# Table Of Contents

Task 1……………………………………………………………3-4

Task 2…………………………………………………………….5-7

Task3……………………………………………………………..8-10

Task 4…………………………………………………………..11-12

Task 5…………………………………………………………..13-14

Task 6…………………………………………………………..15-16

Task 7…………………………………………………………..17-18

Task8……………………………………………………...……19-20

Task9…………………………………………………………...21-22

Task10………………………………………………………….23-24

Sighting.py……………………………………………………24-27

Nominatim.py………………………………………………..27-28

Wildlife.py……………………………………………………..28-29

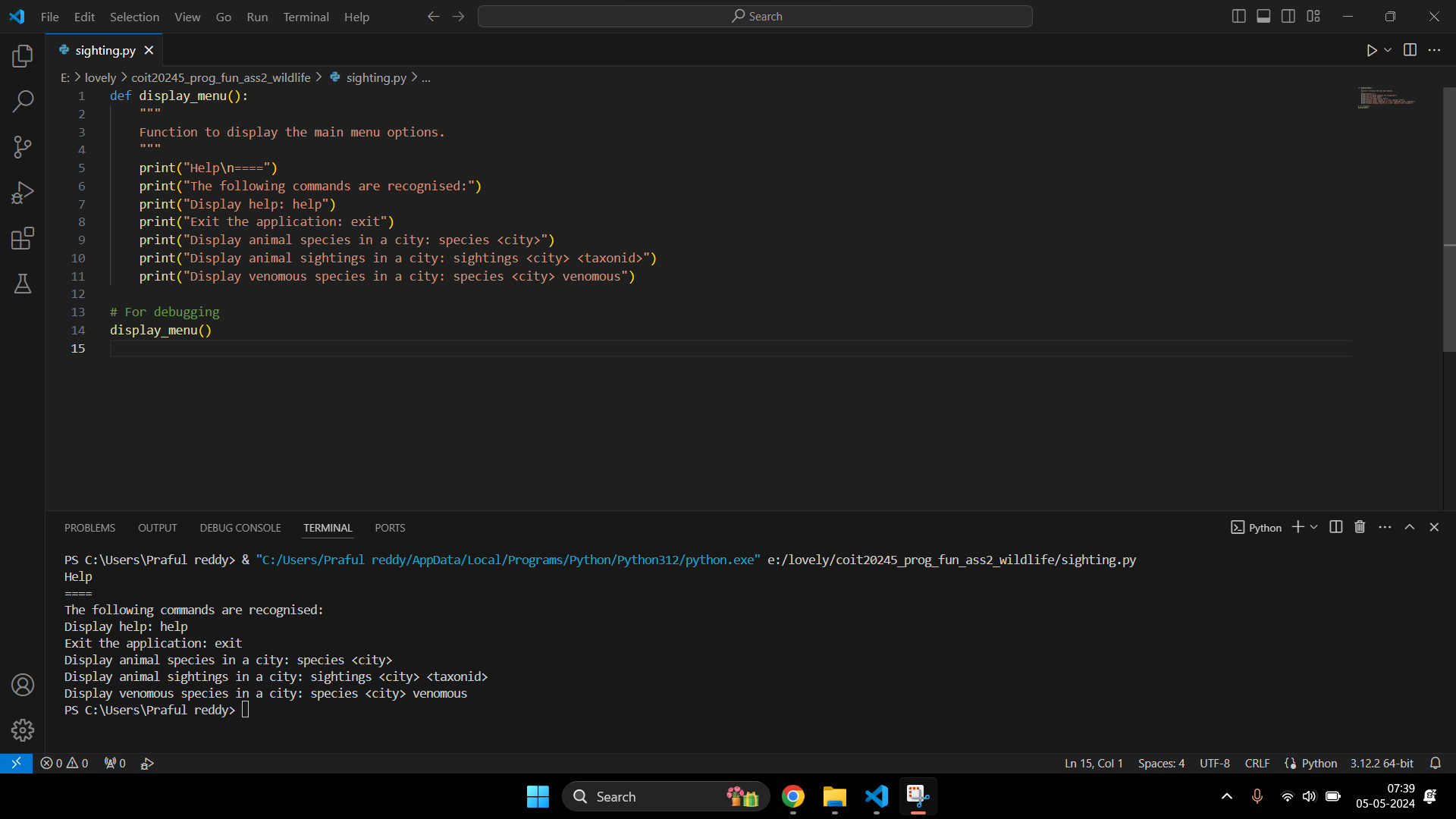
Github link………………………………………………………...29

Drive link…………………………………………………………...29

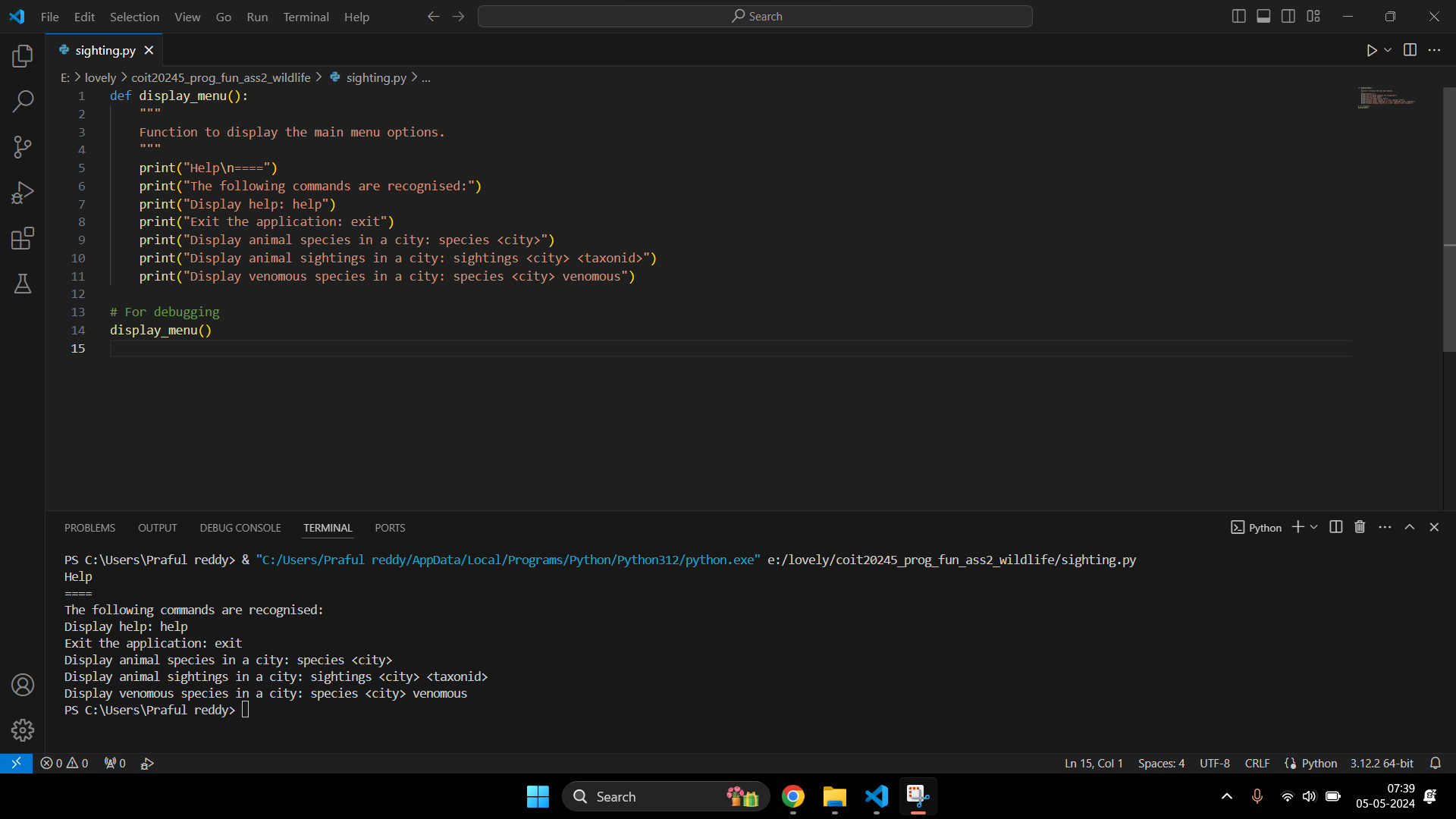
# **Task1**

## Screenshots:

The display\_menu() function that uses print to display the menu.



The display\_menu() function after debugging the code.



## About the Function used:

> Function Name: display\_menu()

> Purpose: Function to display the main menu options.

> Parameters: None

> Returns: It returns the statements given in the ‘print()’ function.

> Exception: None.

> Example Calls:

Help

====

The following commands are recognised:

Display help: help

Exit the application: exit

Display animal species in a city: species <city>

Display animal sightings in a city: sightings <city> <taxonid>

Display venomous species in a city: species <city> venomous

# 

# 

# 

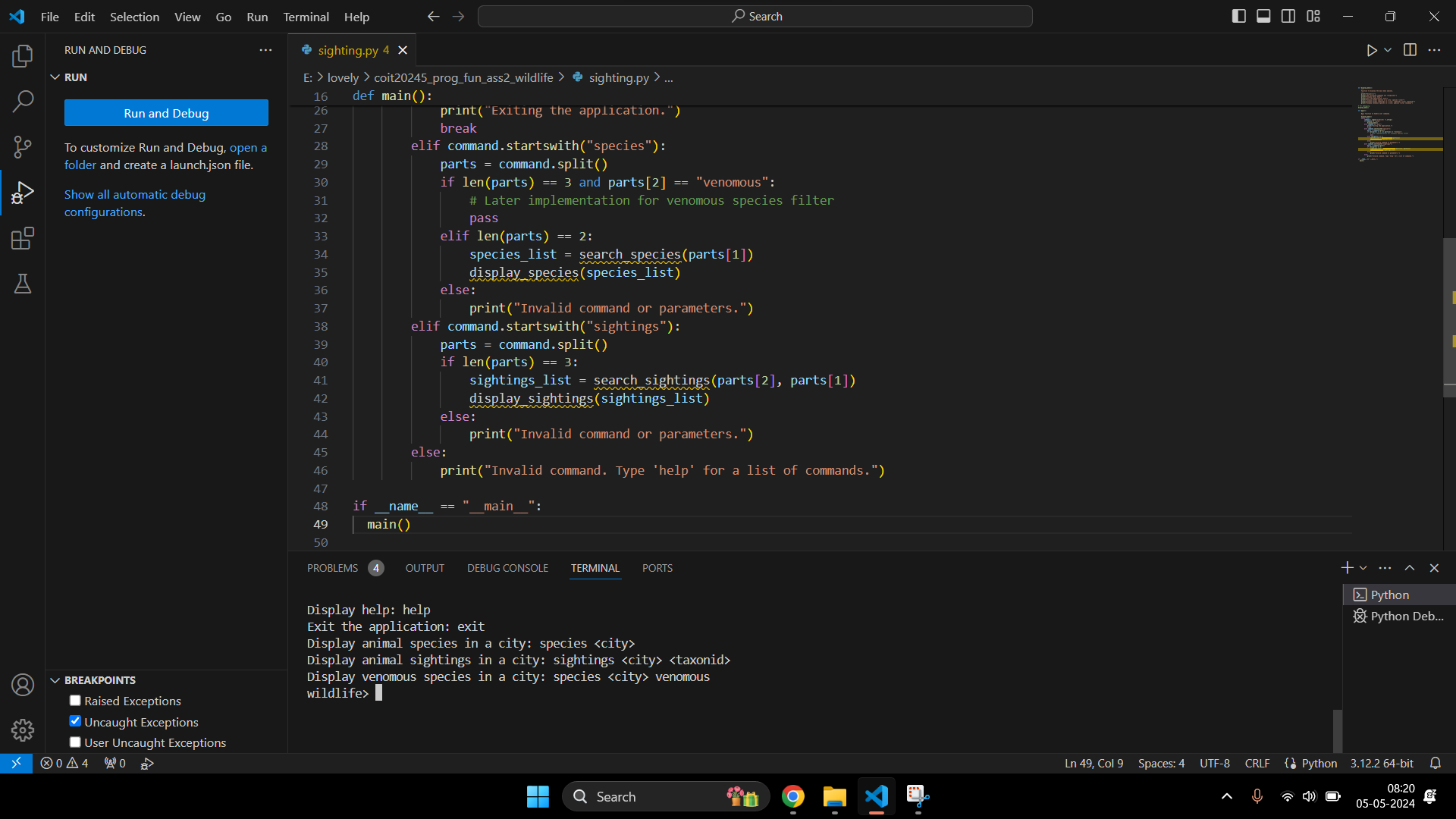
# 

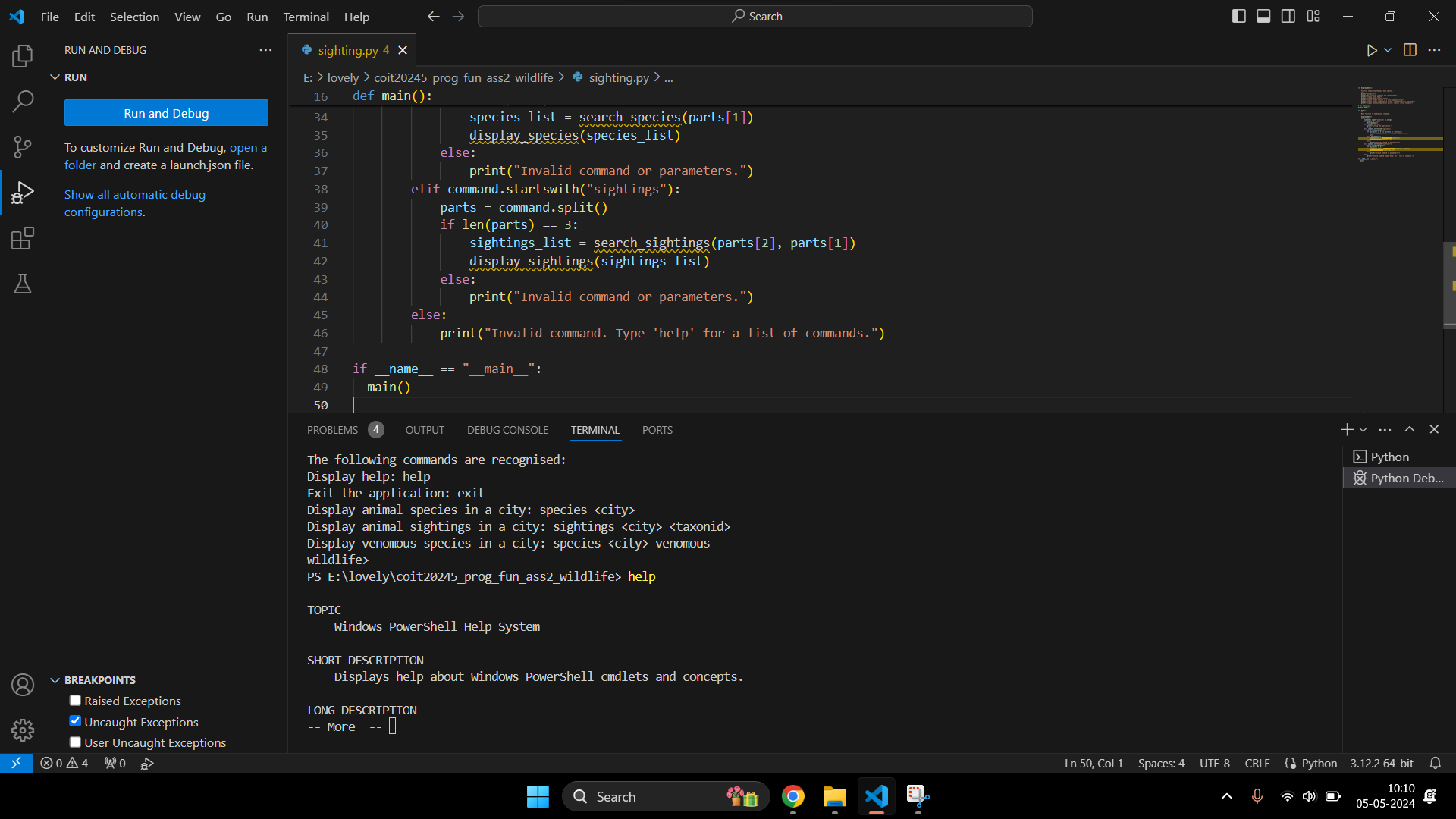
# 

# **Task 2**

## Screenshots:

The main() function that displays the help menu and then repeatedly prompts the user to input their command.





Input: help

Output: Task 1 output will be displayed

Input: exit

Output: Exiting the application. (then the program terminates)

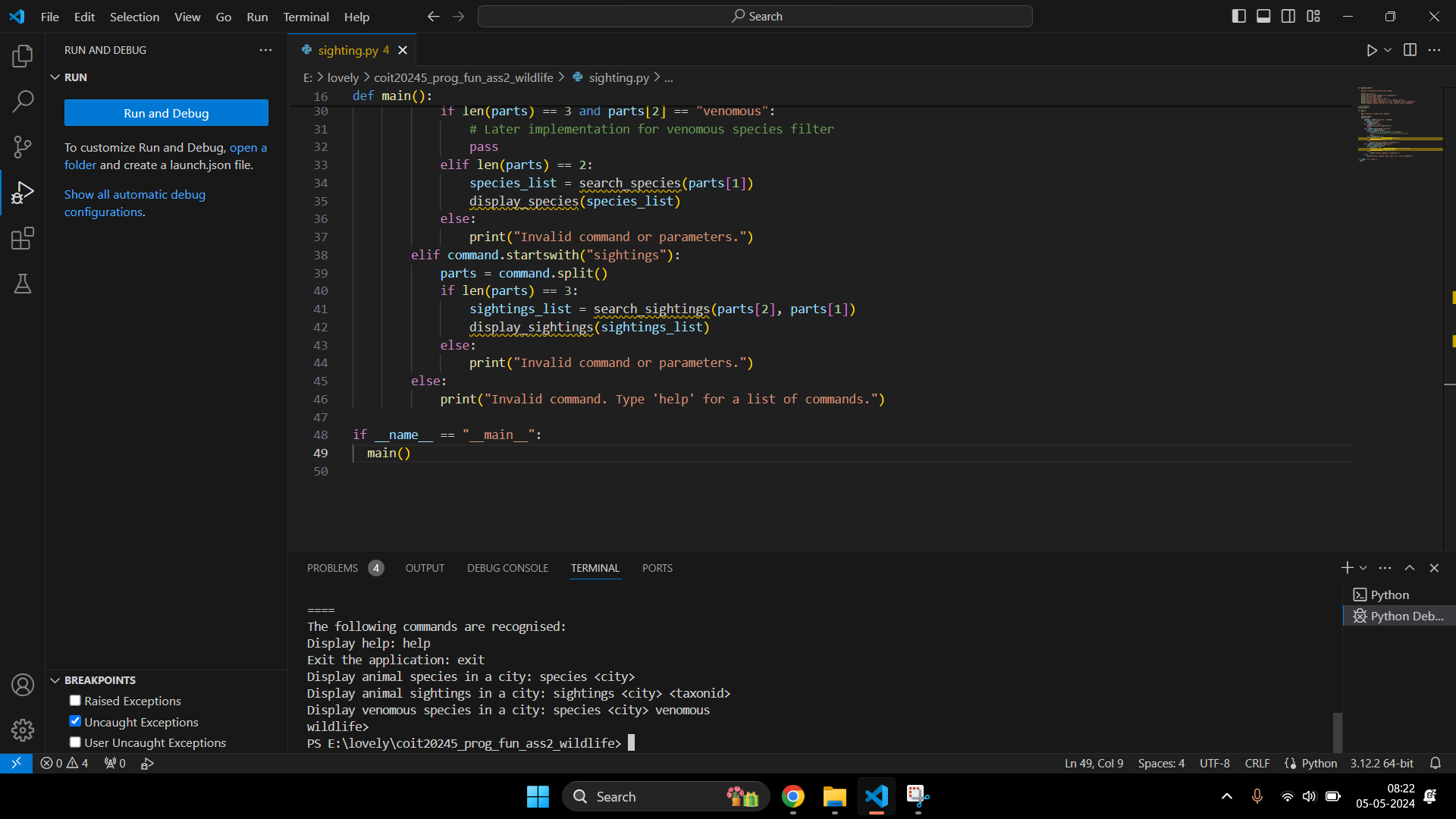
Input: species Cairns

Output: species listing will be displayed (Task3).

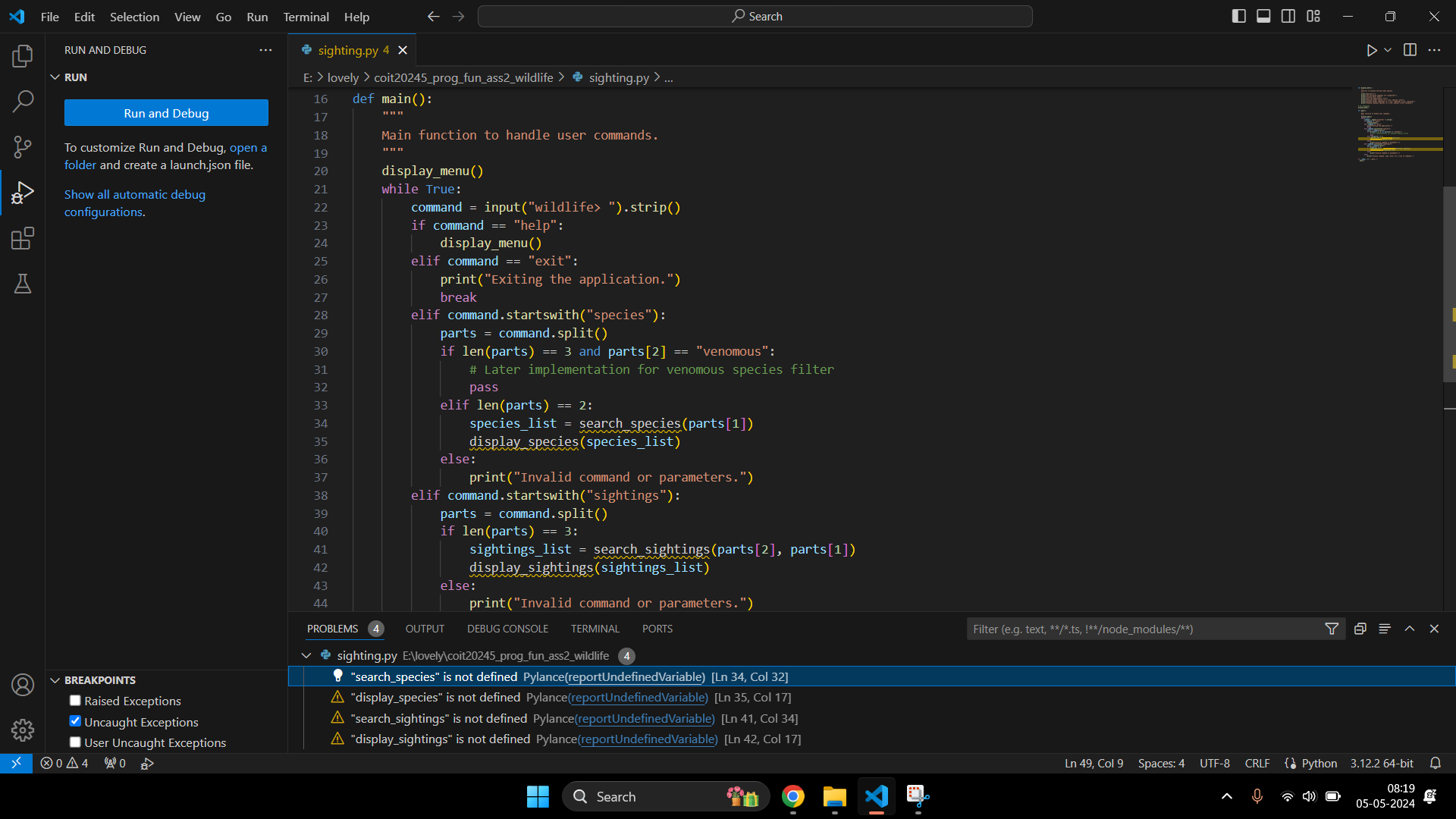
Input: sightings Cairns 1039

Output: sightings listing (Task 4)

The main() function that displays after debugging of code.



## Problems occurred:



## About the Function used:

> Function Name: main()

> Purpose: Function to create commands.

> Parameters: None

> Returns: None

> Exception: It excepts some on-defined functions such as search\_species, display\_species, search\_sighting and display\_sighting.

> Example Calls:

main()

wildlife> help

Displays help menu.

wildlife> exit

Exits the application

# 

# 

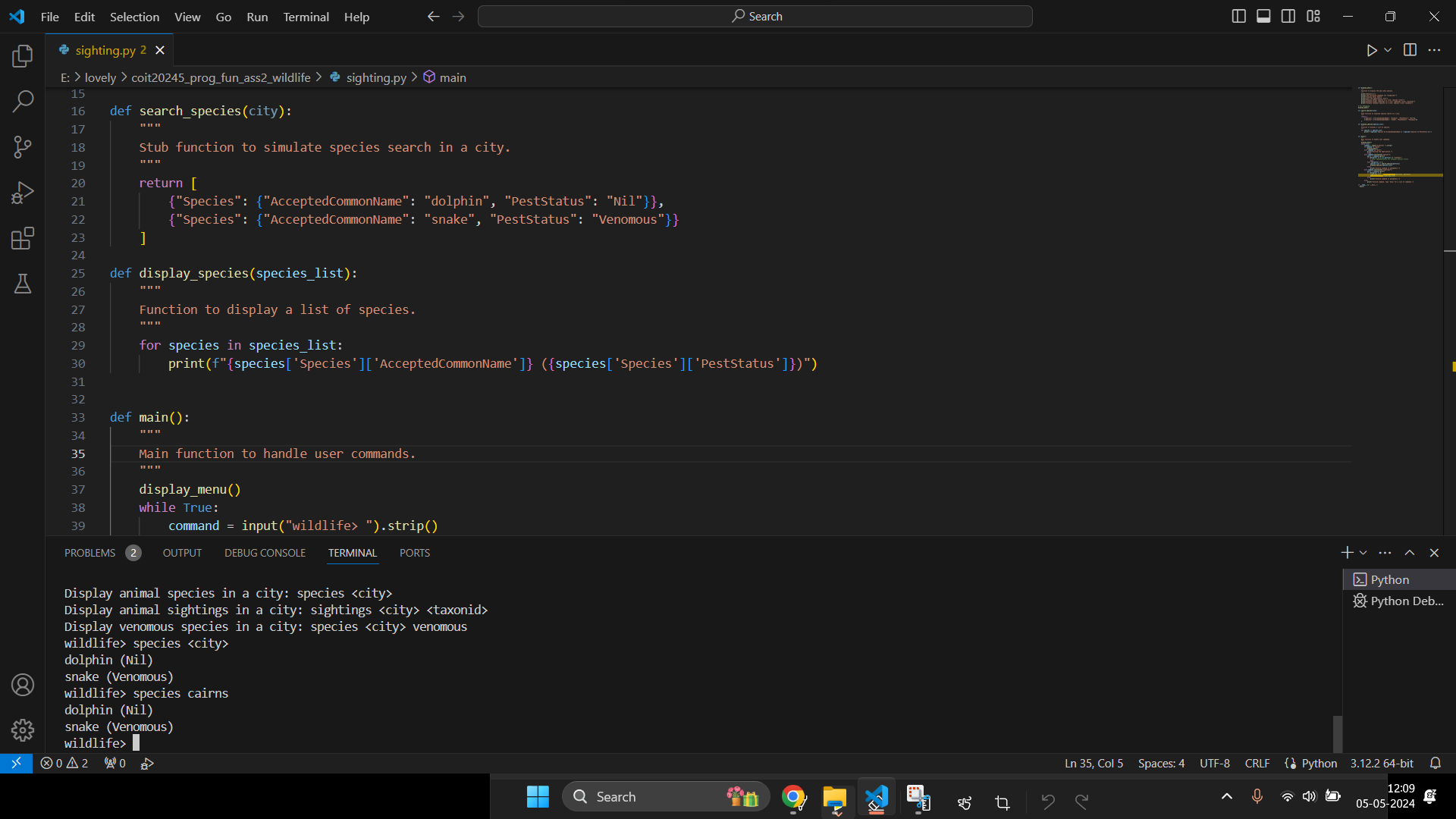
# 

# 

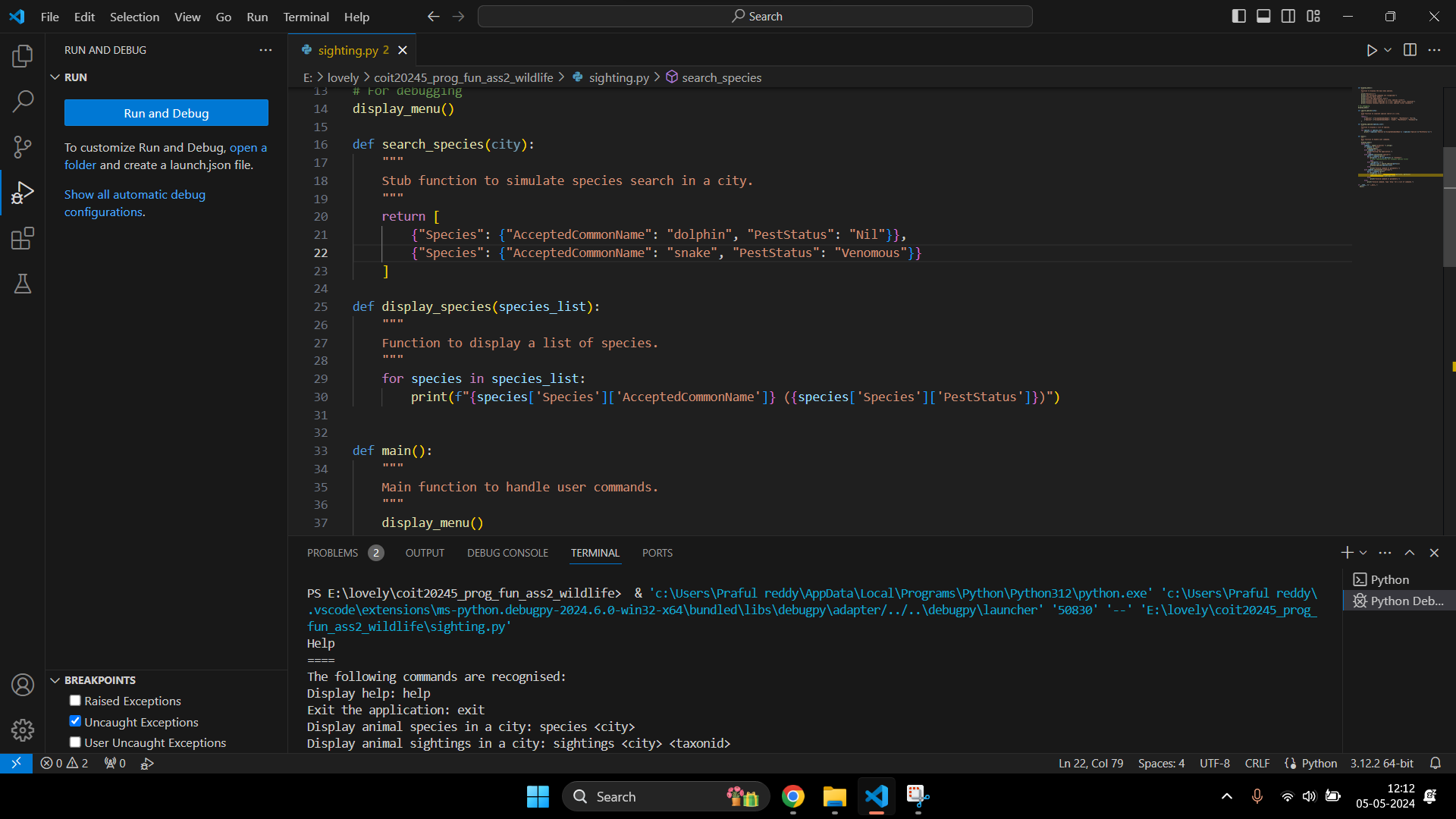
# **Task 3**

## Screenshots:

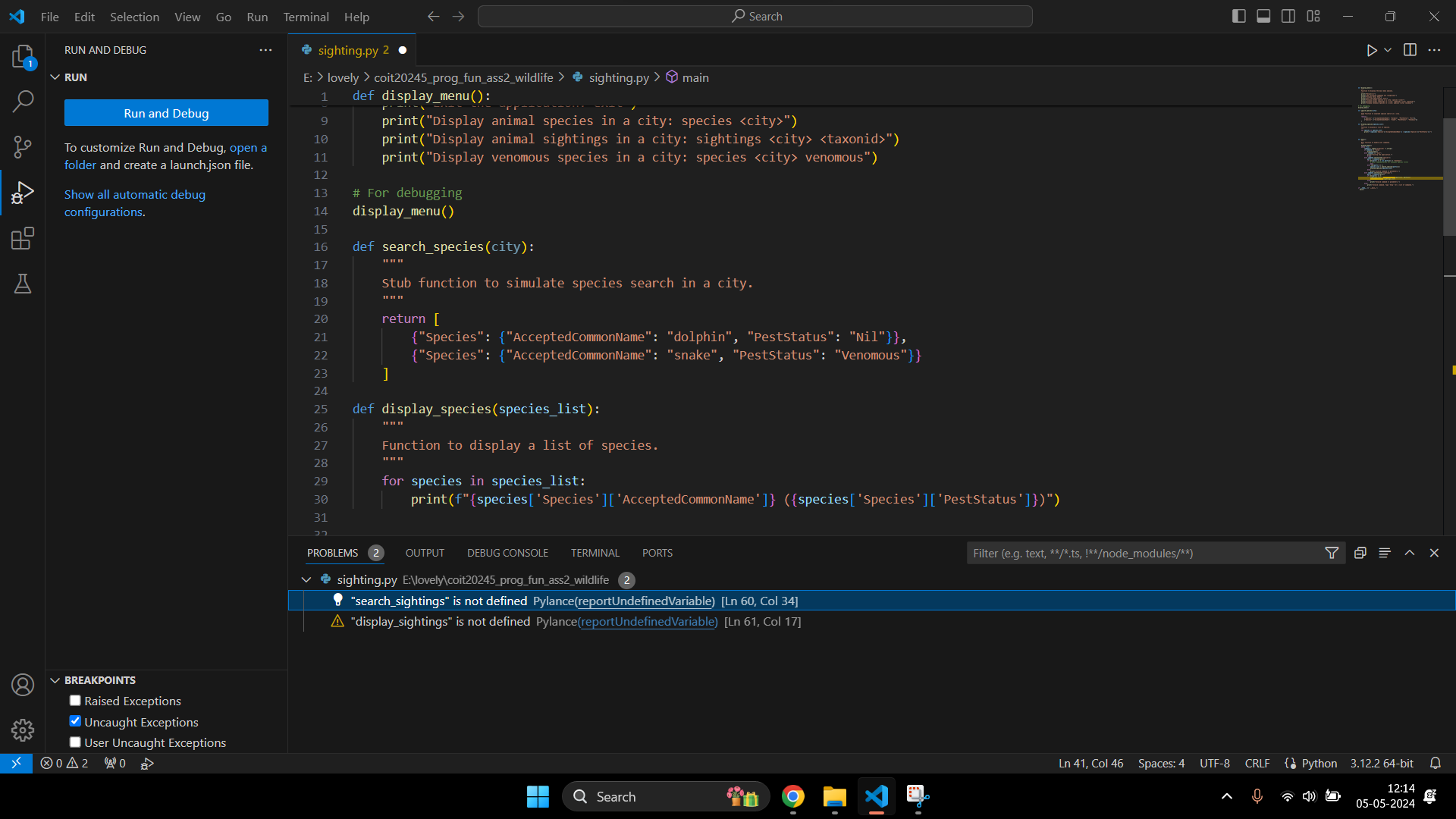
Update display\_menu() by adding a few functions.



Output after debugging the functions.



## Problems occurred:

****

## About the Function used:

> Function Name: search\_species(city)

> Purpose: Function to search different types of species in the city

> Parameters: city (str): The city where the species are to be searched.

> Returns: list: A list of dictionaries, each containing species information

> Exception: None

> Example Calls:

search\_species("Cairns")

[{"Species": {"AcceptedCommonName": "dolphin", "PestStatus": "Nil"}},

{"Species": {"AcceptedCommonName": "snake", "PestStatus": "Venomous"}}]

> Function Name: display\_species(city)

> Purpose: Displays a list of species to the user.

> Parameters: city (str): species\_list (list): A list of dictionaries containing species information.

> Returns: None

> Exception: None

> Example Calls:

display\_species([{"Species": {"AcceptedCommonName": "dolphin", "PestStatus": "Nil"}}])

Species: dolphin, Pest Status: Nil

# 

# 

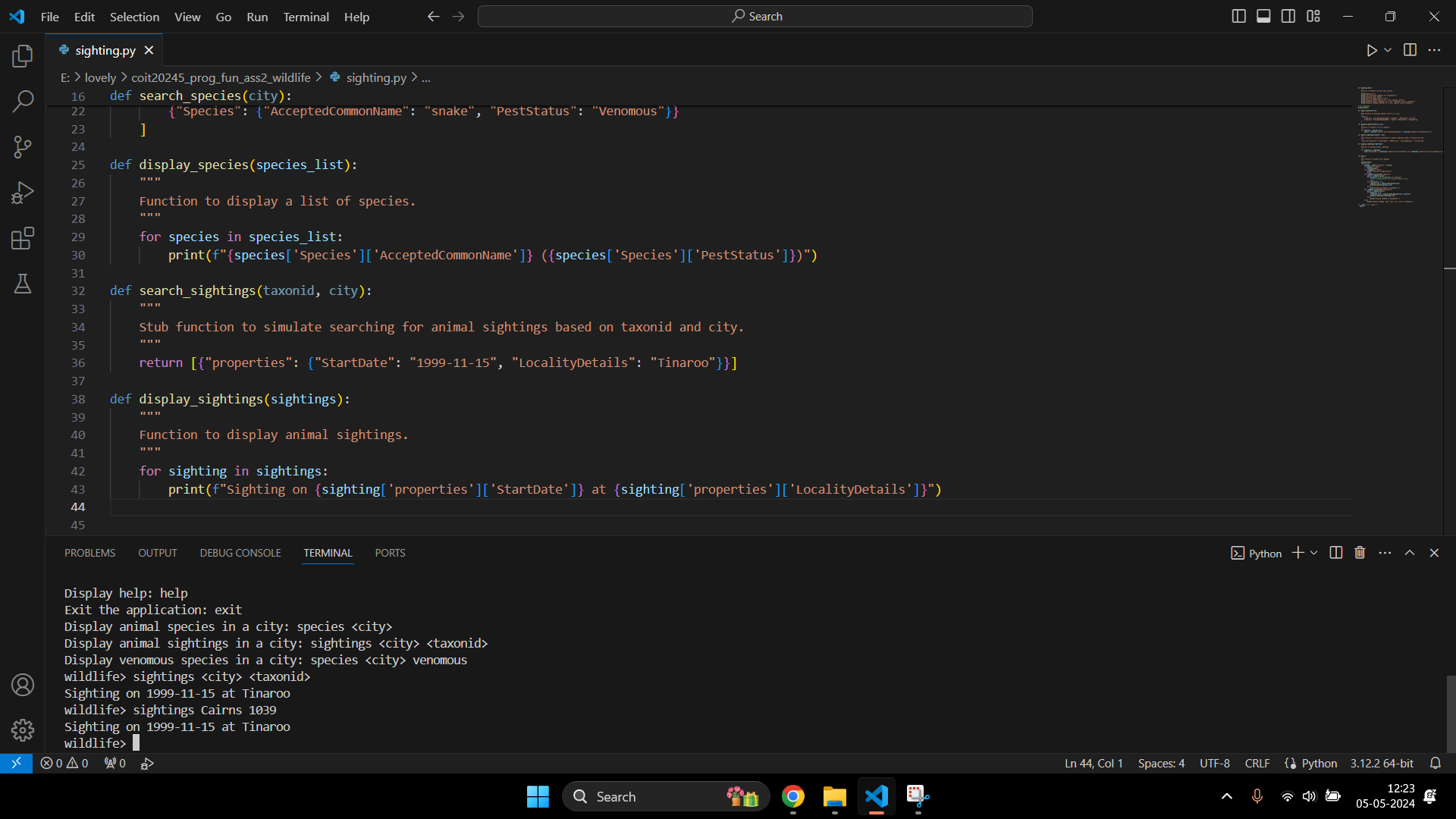
# 

# 

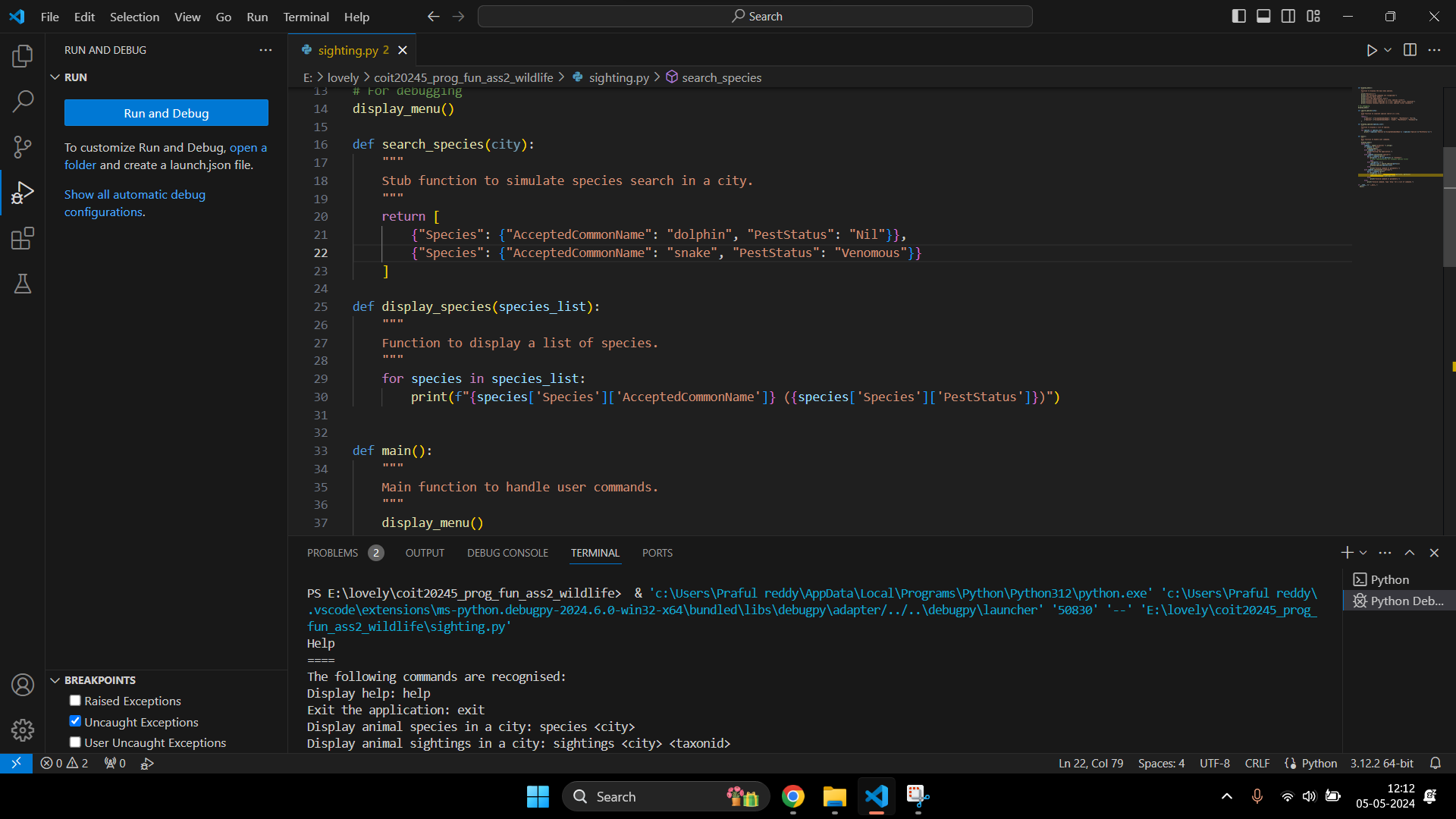
# **Task 4**

## Screenshots:

Update display\_menu() by adding a few functions.

****

Output after debugging the functions.

****

## About the Function used:

> Function Name: search\_sightings(taxonid, city)

> Purpose: Searches for animal sightings based on TaxonID and city. Returns a stubbed list of sightings.

> Parameters: taxonid (int): The identifier for the species.

city (str): The city where the sightings are to be searched.

> Returns: list: A list of dictionaries, each containing sighting details.

> Exception: None

> Example Calls:

search\_sightings(1039 , " Cairns ")

[{"properties": {" StartDate ": " 1999-11-15", "Locality Details": "Tinaroo" }} ]

> Function Name: display\_sightings(sightings)

> Purpose: Displays a list of animal sightings.

> Parameters: sightings (list): A list of dictionaries containing sighting details.

> Returns: None

> Exception: None

> Example Calls:

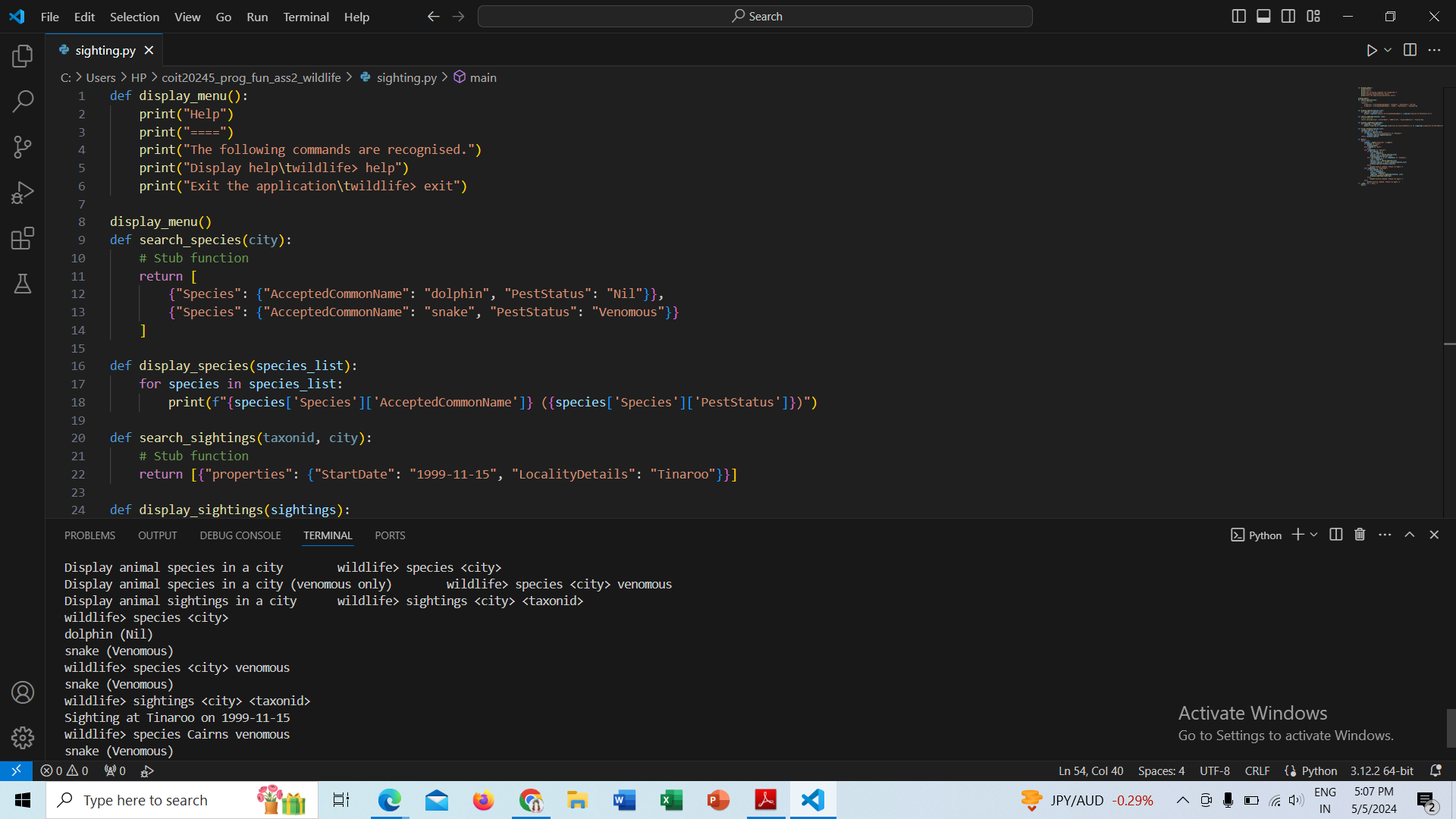
display\_sightings([{ "properties" : {"Start Date": "1999-11-15", "Locality Details": "Tinaroo"}}])

Sighting: Tinaroo, Date: 1999-11-15

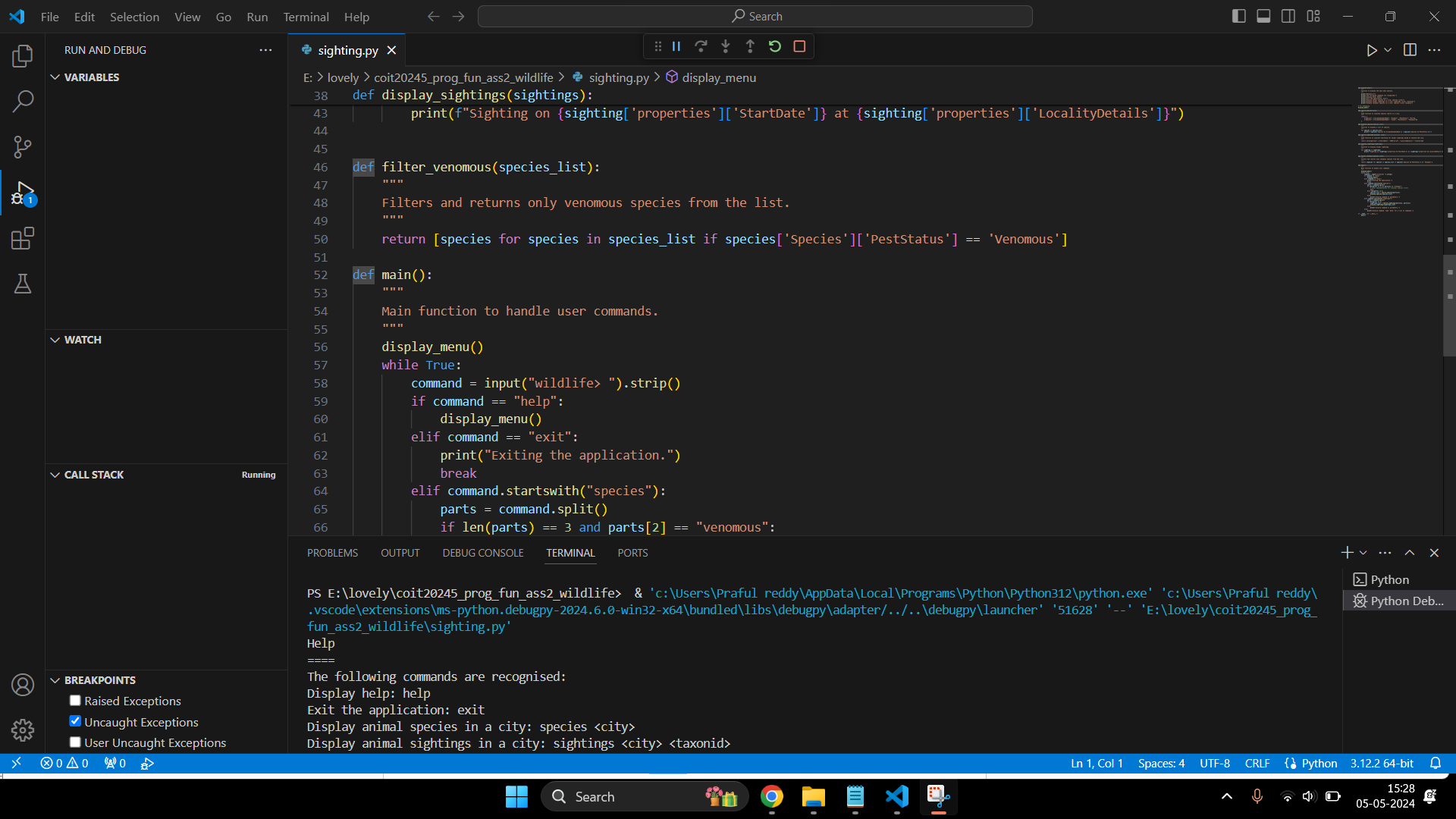
# **Task 5**

## Screenshots:

Update display\_menu() by adding a few functions.



Output after debugging the functions.



## About the Function used:

> Function Name: filter\_venomous(species\_list)

> Purpose: Filters and returns only venomous species from a list of species.

> Parameters: species\_list (list): A list of dictionaries containing species details.

> Returns: list: A filtered list of dictionaries for venomous species only.

> Exception: None

> Example Calls:

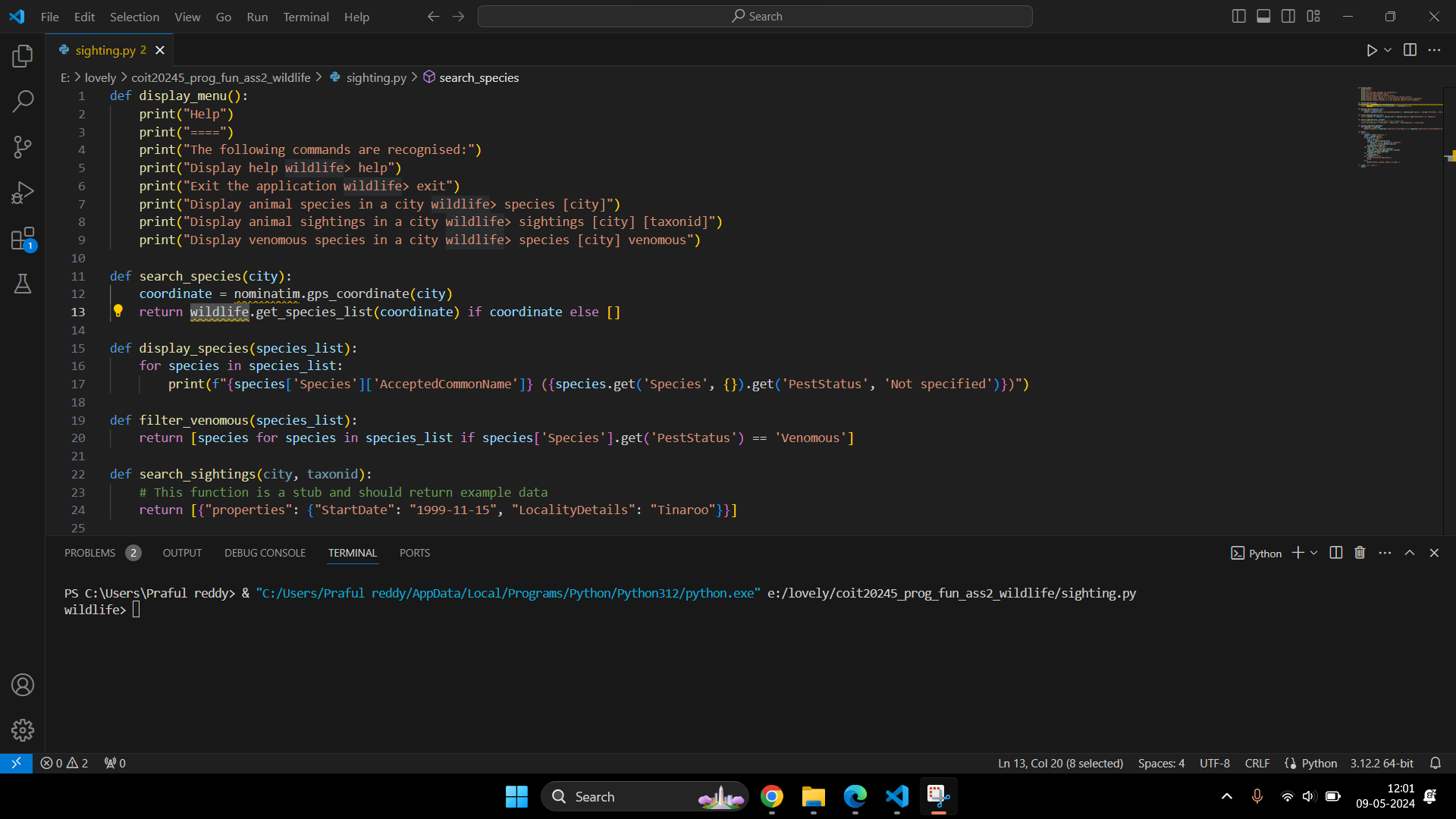
filter\_venomous([{"Species": {"AcceptedCommonName": "snake", "PestStatus": "Venomous"}}])

[{"Species": {"AcceptedCommonName": "snake", "PestStatus": "Venomous"}}]

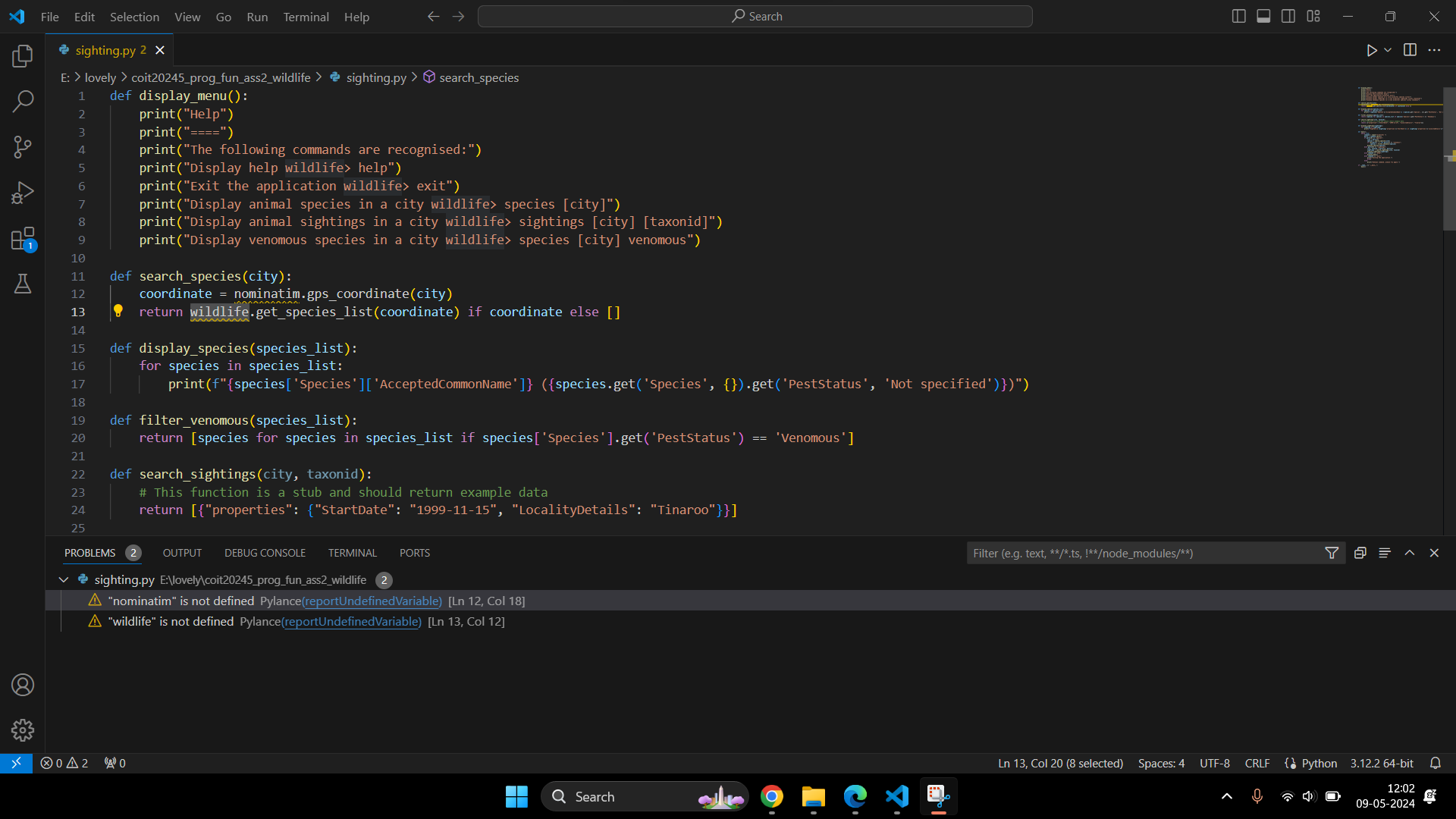
# **Task 6**

# Screenshots:

Updated search\_sightings(taxonid, city) function in sighting.py file.



Problems occurred while importing and updating the function.



> Function Name: gps(city)

> Purpose: Gets GPS coordinates for the given city name.

> Parameters: city (str): The city name.

> Returns: dict: The latitude and longitude of the city.

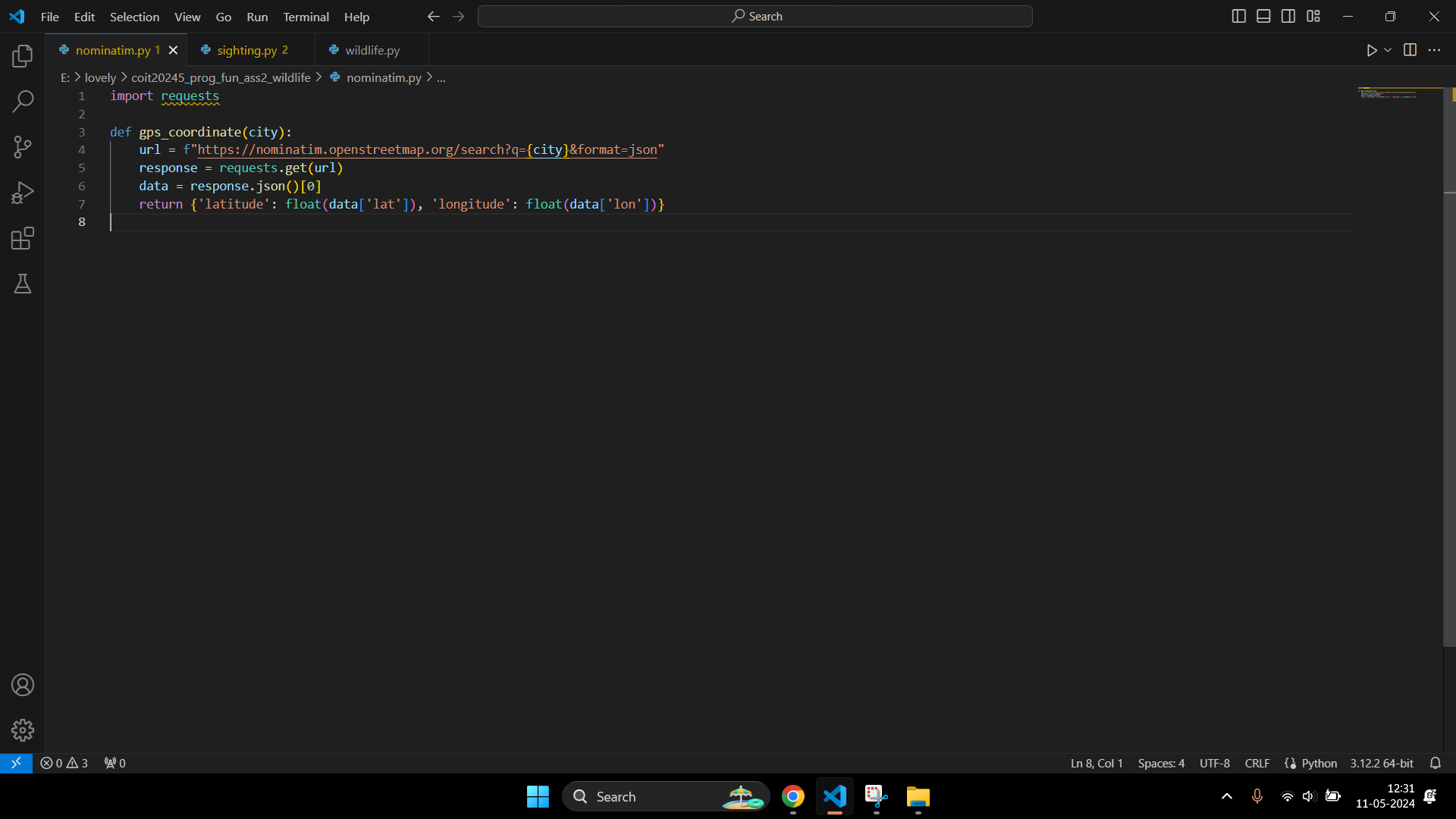
> Exception: None

> Example Calls: None

# **Task 7**

## Screenshots:

A new file nominatim.py has been created.



> Function Name: gps\_coordinate(city)

> Purpose: Fetches the latitude and longitude for a given city using the Nominatim geocoding service.

> Parameters: city (str): The city name to geocode.

> Returns: dict: A dictionary with keys 'latitude' and 'longitude'

> Exception: None

> Example Calls:

base\_url = 'https://nominatim.openstreetmap.org/search'

params = {'q': city, 'format': 'json'}

response = requests.get(base\_url, params=params)

data = response.json()

if data:

first\_result = data[0]

coordinates = {

'latitude': float(first\_result['lat']),

'longitude': float(first\_result['lon'])

}

return coordinates

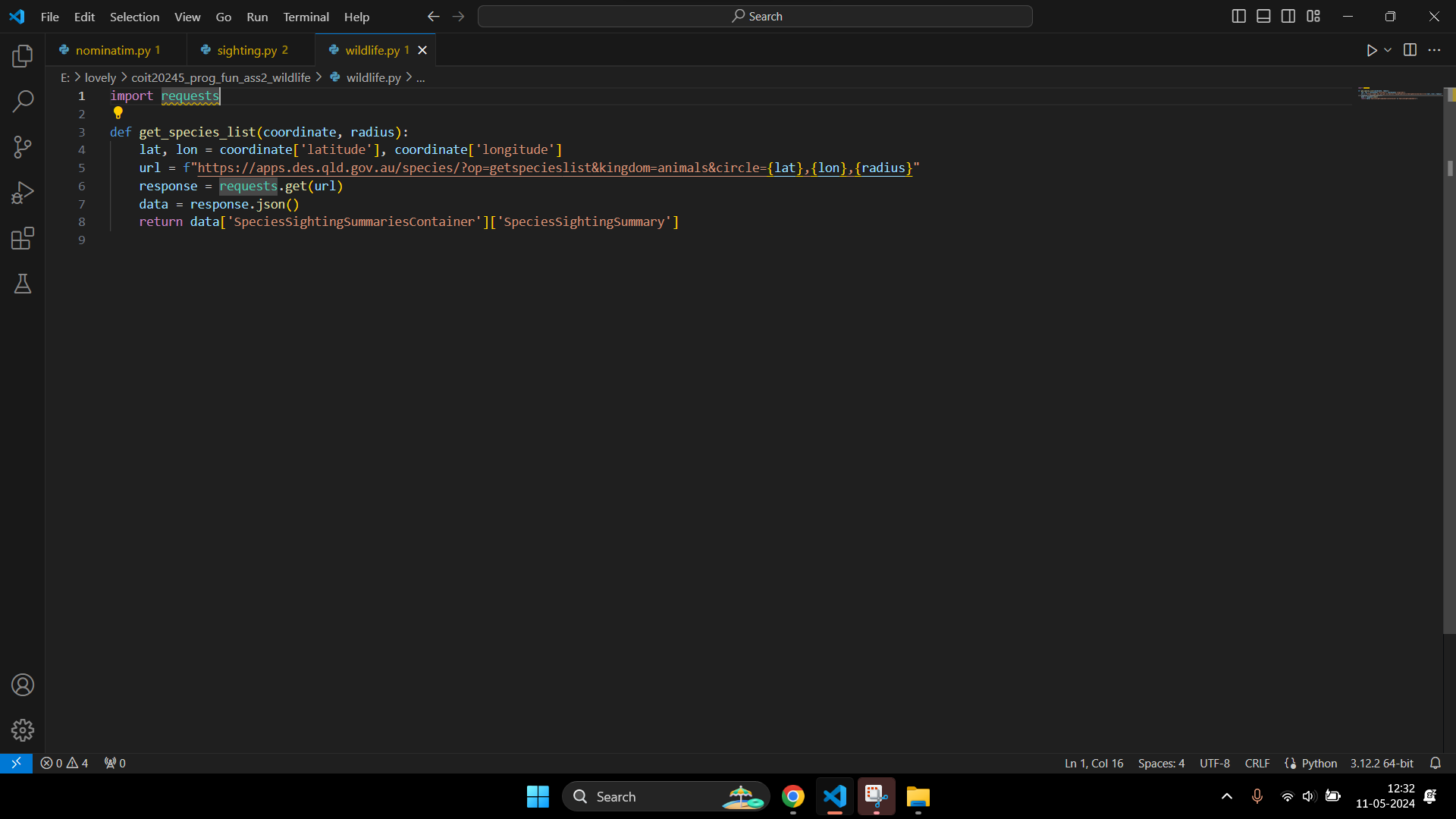
else:

return None

# **Task 8**

## Screenshots:

A new file wildlife.py has been created.

****

> Function Name: get\_species\_list(coordinate, radius)

> Purpose: Retrieves a list of species in an area defined by a circle with a given radius around a coordinate.

> Parameters: coordinate (dict): A dictionary with keys 'latitude' and 'longitude'.

radius (int): The radius of the search area in meters.

> Returns:list: A list of species dictionaries.

> Exception: None

> Example calls:

base\_url = 'https://apps.des.qld.gov.au/species/'

params = {

'op': 'getspecieslist',

'kingdom': 'animals',

'circle': f"{coordinate['latitude']},{coordinate['longitude']},{radius}"

}

response = requests.get(base\_url, params=params)

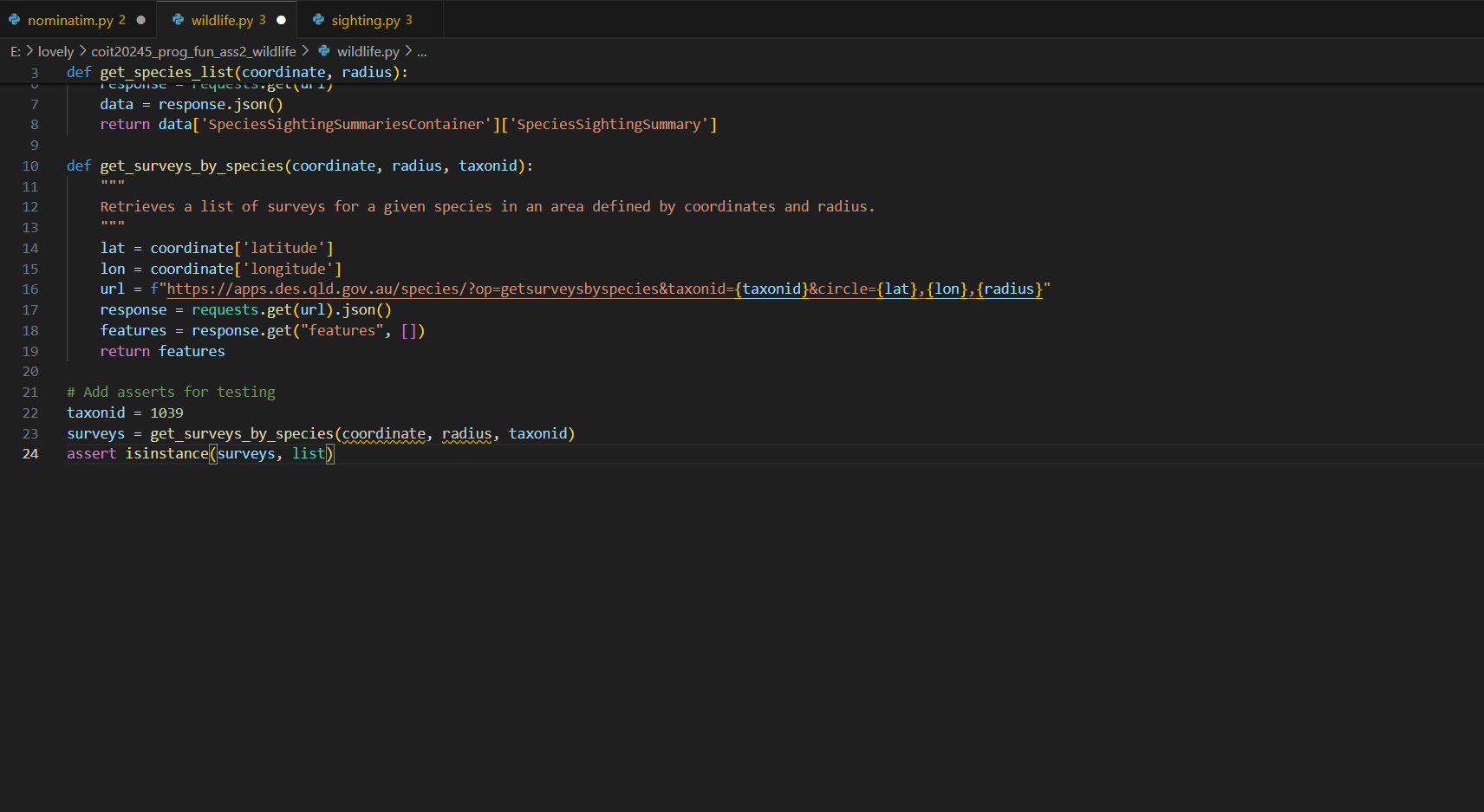
data = response.json()

return data["SpeciesSightingSummariesContainer"]["SpeciesSightingSummary"]

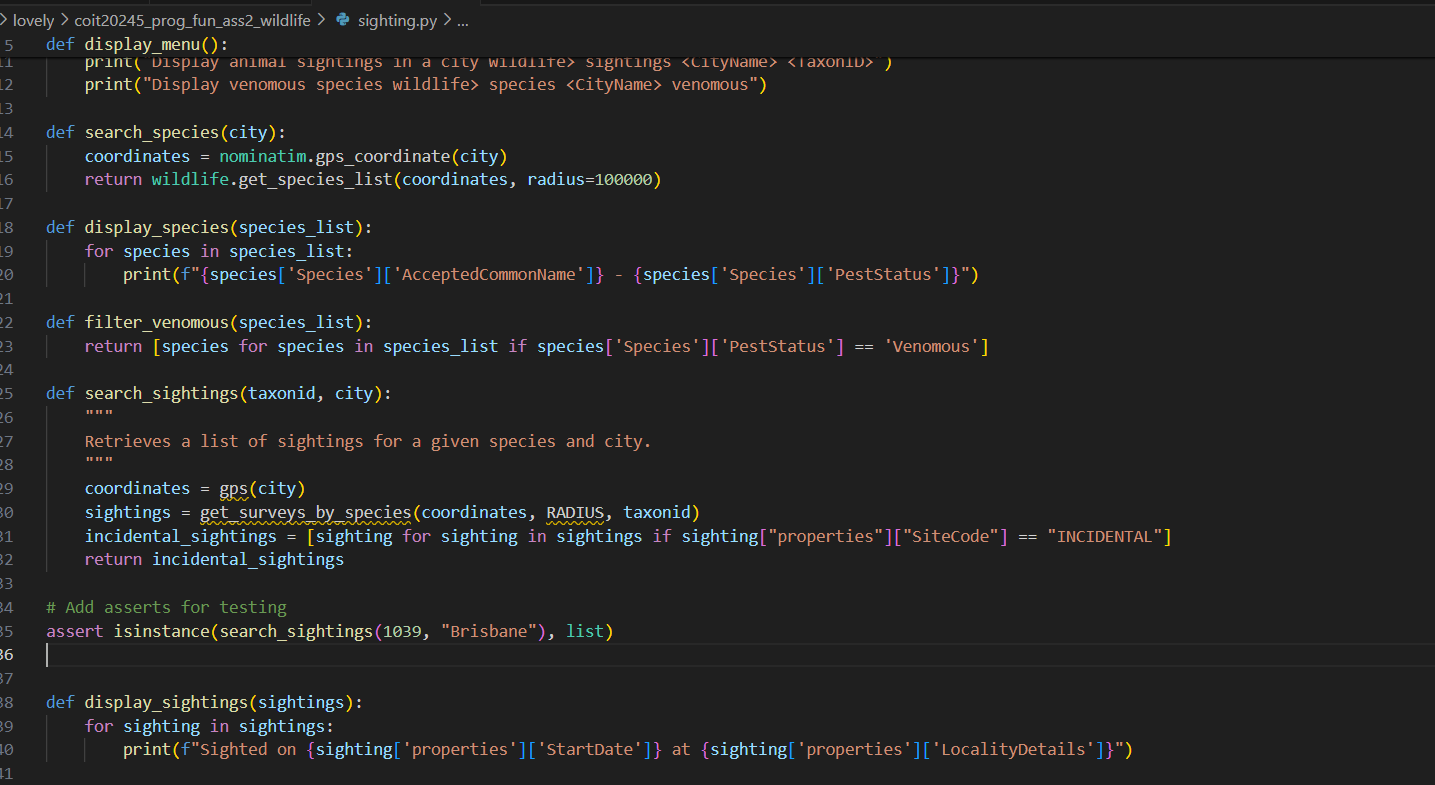
# **Task 9**

## Screenshots:

Added the ‘get\_surveys\_by\_species()’ function in the wildlife.py file.

****

Updated the ‘search\_sightings()’ function in the sighting.py file.



> Function Name: get\_surveys\_by\_species(coordinate, radius, taxonid):

> Purpose: Retrieve a list of surveys for a particular species in an area.

> Parameters: coordinate (tuple): A tuple of latitude and longitude (lat, lon).

radius (int): The radius (in meters) around the coordinate to search for surveys.

taxonid (int): The taxon ID of the species.

> Returns: list: A list of dictionaries containing survey details.

> Exception: None

> Example Calls:

url = f"https://apps.des.qld.gov.au/species/?op=getsurveysbyspecies&taxonid={taxonid}&circle={coordinate[0]},{coordinate[1]},{radius}"

response = requests.get(url)

if response.status\_code == 200:

data = response.json()

return data['features']

else:

response.raise\_for\_status()

> Function Name: search\_sightings(taxonid, city):

> Purpose: Search for animal sightings by taxon ID and city.

> Parameters: taxonid (int): The taxon ID of the species.

city (str): The city to search for sightings.

> Returns: list: A list of filtered sightings.

> Exception: None

> Example Calls:

# Dummy coordinate and radius for the example. Ideally, these should be determined based on the city.

coordinate = (-16.92, 145.777)

radius = 100000

surveys = get\_surveys\_by\_species(coordinate, radius, taxonid)

# Filter surveys by SiteCode 'INCIDENTAL'

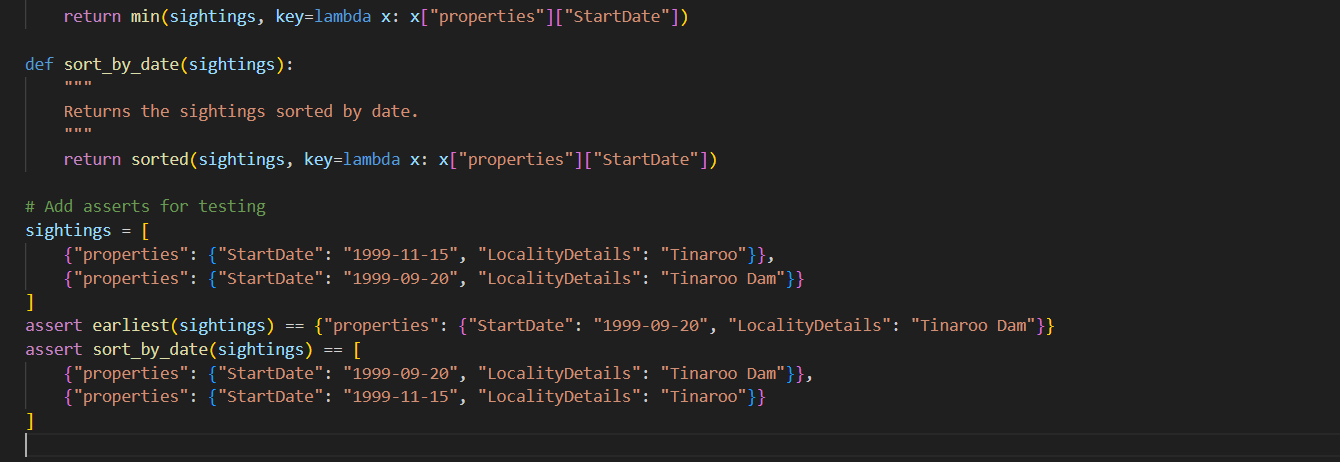
filtered\_surveys = [survey for survey in surveys if survey['properties'].get('SiteCode') == 'INCIDENTAL']

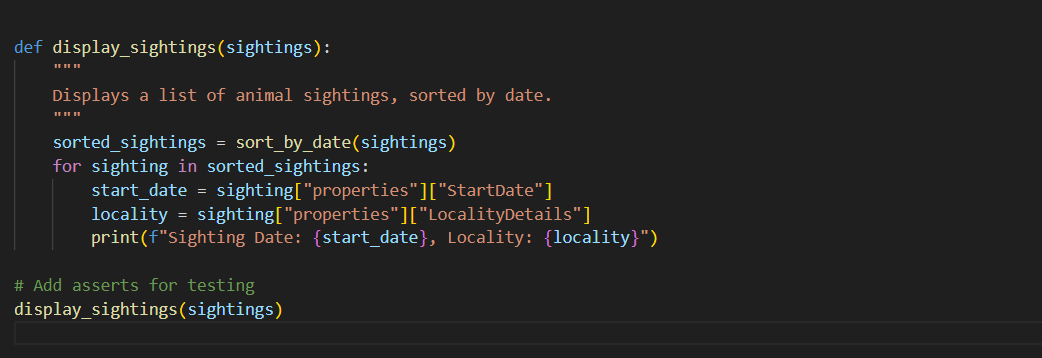
return filtered\_surveys

# **Task 10**

## Screenshots:

Added the ‘sort\_by\_date()’ function.



Updated the ‘display\_sighting()’ function.

> Function Name: earliest(sightings)

> Purpose: Return the sighting with the earliest start date.

> Parameters: sightings (list): A list of dictionaries containing sighting details.

> Returns: dict: The sighting with the earliest start date.

> Exception: None

> Example Calls:

if not sightings:

return None

return min(sightings, key=lambda sighting: datetime.strptime(sighting['properties']['StartDate'], "%Y-%m-%d"))

> Function Name: sort\_by\_date(sightings)

> Purpose: Return sightings sorted by start date.

> Parameters: sightings (list): A list of dictionaries containing sighting details.

> Returns: list: A list of sightings sorted by start date.

> Exception: None

> Example Calls:

return sorted(sightings, key=lambda sighting: datetime.strptime(sighting['properties']['StartDate'], "%Y-%m-%d"))

# **Sighting.py**

## Code:

from nominatim import gps\_coordinate

from wildlife import get\_species\_list, get\_surveys\_by\_species

# Constants

RADIUS = 100000

def display\_menu():

"""

Displays the menu to the user.

"""

print("Help")

print("====")

print("The following commands are recognised.")

print("Display help wildlife> help")

print("Exit the application wildlife> exit")

print("Display animal species in a city wildlife> species <city>")

print("Display animal sightings in a city wildlife> sightings <city> <taxonid>")

print("Display venomous species wildlife> species <city> venomous")

def gps(city):

"""

Retrieves GPS coordinates for a given city.

"""

return gps\_coordinate(city)

def search\_species(city):

"""

Retrieves a list of species for a given city.

"""

coordinates = gps(city)

species\_list = get\_species\_list(coordinates, RADIUS)

return species\_list

def search\_sightings(taxonid, city):

"""

Retrieves a list of sightings for a given species and city.

"""

coordinates = gps(city)

sightings = get\_surveys\_by\_species(coordinates, RADIUS, taxonid)

incidental\_sightings = [sighting for sighting in sightings if sighting["properties"]["SiteCode"] == "INCIDENTAL"]

return incidental\_sightings

def filter\_venomous(species\_list):

"""

Filters the list of species to include only venomous species.

"""

return [species for species in species\_list if species["Species"]["PestStatus"] == "Venomous"]

def display\_species(species\_list):

"""

Displays a list of species.

"""

for species in species\_list:

common\_name = species["Species"]["AcceptedCommonName"]

pest\_status = species["Species"]["PestStatus"]

print(f"Species: {common\_name}, Pest Status: {pest\_status}")

def display\_sightings(sightings):

"""

Displays a list of animal sightings, sorted by date.

"""

sorted\_sightings = sort\_by\_date(sightings)

for sighting in sorted\_sightings:

start\_date = sighting["properties"]["StartDate"]

locality = sighting["properties"]["LocalityDetails"]

print(f"Sighting Date: {start\_date}, Locality: {locality}")

def earliest(sightings):

"""

Returns the sighting with the earliest start date.

"""

return min(sightings, key=lambda x: x["properties"]["StartDate"])

def sort\_by\_date(sightings):

"""

Returns the sightings sorted by date.

"""

return sorted(sightings, key=lambda x: x["properties"]["StartDate"])

def main():

"""

Main function to handle user input and commands.

"""

display\_menu()

while True:

command = input("wildlife> ").strip()

if command == "help":

display\_menu()

elif command == "exit":

break

elif command.startswith("species "):

parts = command[len("species "):].strip().split()

if len(parts) == 1:

city = parts[0]

species\_list = search\_species(city)

display\_species(species\_list)

elif len(parts) == 2 and parts[1] == "venomous":

city = parts[0]

species\_list = search\_species(city)

venomous\_species = filter\_venomous(species\_list)

display\_species(venomous\_species)

else:

print("Invalid command format. Use: species <city> [venomous]")

elif command.startswith("sightings "):

parts = command[len("sightings "):].strip().split()

if len(parts) == 2:

city, taxonid = parts

sightings = search\_sightings(taxonid, city)

display\_sightings(sightings)

else:

print("Invalid command format. Use: sightings <city> <taxonid>")

else:

print("Unrecognized command. Type 'help' for a list of commands.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

# **Nominatim.py**

## Code:

import requests

def gps\_coordinate(city):

"""

Retrieves GPS coordinates for a given city using the Nominatim web service.

"""

url = f"https://nominatim.openstreetmap.org/search?q={city}&format=json"

response = requests.get(url).json()

if response:

latitude = float(response[0]['lat'])

longitude = float(response[0]['lon'])

return {'latitude': latitude, 'longitude': longitude}

return None

# Uncomment to test the function

# assert gps\_coordinate("Brisbane") == {'latitude': -27.4689682, 'longitude': 153.0234991}

# **Wildlife.py**

## Code:

import requests

def get\_species\_list(coordinate, radius):

"""

Retrieves a list of species in an area defined by coordinates and radius.

"""

lat = coordinate['latitude']

lon = coordinate['longitude']

url = f"https://apps.des.qld.gov.au/species/?op=getspecieslist&kingdom=animals&circle={lat},{lon},{radius}"

response = requests.get(url).json()

species\_list = response.get("SpeciesSightingSummariesContainer", {}).get("SpeciesSightingSummary", [])

return species\_list

def get\_surveys\_by\_species(coordinate, radius, taxonid):

"""

Retrieves a list of surveys for a given species in an area defined by coordinates and radius.

"""

lat = coordinate['latitude']

lon = coordinate['longitude']

url = f"https://apps.des.qld.gov.au/species/?op=getsurveysbyspecies&taxonid={taxonid}&circle={lat},{lon},{radius}"

response = requests.get(url).json()

features = response.get("features", [])

return features

# Uncomment to test the functions

# assert get\_species\_list({'latitude': -27.4689682, 'longitude': 153.0234991}, 100000)

# assert get\_surveys\_by\_species({'latitude': -27.4689682, 'longitude': 153.0234991}, 100000, 1039)

# **Github link:**

<https://github.com/ashishvardhan29/COIT20245-Assignment-2>

# **Drive link:**

* <https://drive.google.com/drive/folders/1t6f71EOGlF7Uf8lBoSUs0p8FzSriYC5P?usp=drive_link>
* <https://drive.google.com/drive/folders/1A4FNXSNBNJGLijoQAOKoFyboC0YsE7T6?usp=drive_link>