**#Signtings.py**

#Displays the menu

def display\_menu():

    print("Help")

    print("===")

    print("The following commands are recognised.")

    #Formatiing the menu to show align

    print('Display help {:>20}  wildlife> help'.format(''))

    print('Display animal species in a city   wildlife> species Cairns')

    print('Display venonmous species wildlife> species Cairns venomous')

    print('Display animal sightings in a city wildlife> sightings Cairns 1039')

    print('Exit the application {:>12}  wildlife> exit\n'.format(''))

#Filters out the main fucntion

def main():

    display\_menu()

    while(True):

        comm = input().split(' ')

        print("\n")

        if(comm[0]=="exit"):

            exit(0)

        elif(comm[0]=="help"):

            display\_menu()

        elif(comm[0]=="species"):

            if len(comm)>2:

                display\_species(filter\_venomous(search\_species(comm[1])))

            else:

                display\_species(search\_species(comm[1]))

        elif(comm[0]=="sightings"):

            display\_sightings(search\_sightings(comm[2],comm[1]))

        else:

            print("ERROR: Unrecognized Command. Please check and try again.")

            display\_menu()

#Displays  species input static

def search\_species(city):

    output= [ {"Species":{"AcceptedCommonName":"dolphin", "PestStatus":"Nil"}}, {"Species":{"AcceptedCommonName":"snake","PestStatus":"Venomous"}} ]

    return output

#Displays search species

def display\_species(species\_list):

    index = 1

    for species in species\_list:

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

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

        print(f"Species {index}:")

        print(f"  Accepted Common Name: {common\_name}")

        print(f"  Pest Status: {pest\_status}")

        print()

        index += 1

#Displays the dearch sightings using the taxonoid city

def search\_sightings(taxonid,city):

    sightings = [{"properties":{"StartDate":"1999-11-15","LocalityDetails":"Tinaroo"}}]

    return sightings

#Displays the sightings

def display\_sightings(sightings):

    for sighting in sightings:

        print(f"Date: {sighting['properties']['StartDate']}")

        print(f"Location: {sighting['properties']['LocalityDetails']}")

        print("------------------------\n")

# filter\_venomous method

def filter\_venomous(species\_list):

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

main()

**#nominations.py**

import requests

def gps\_coordinate(city):

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

    x=input("Enter the city name:")

    if x in url:

     print(url[x])

    response = requests.get(url)

    getcord = response.json()

    a=getcord[lat][long]

    return a

def gps(city):

    x=gps\_coordinate(city)

    print(x)

**#wildlife.py**

import requests

#Get species list from the api using cordinate radius with a response json file

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

    base\_url = "https://apps.des.qld.gov.au/species/?op=getspecieslist"

    lat, lon = coordinate["latitude"], coordinate["longitude"]

    params = {

        "kingdom": "animals",

        "circle": f"{lat},{lon},{radius}"

    }

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

    data = response.json()

    if "SpeciesSightingSummariesContainer" in data:

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

    else:

        return None

#Gets the Surveys of Species using Coordinate,radius and taxonoid

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

    base\_url = "https://apps.des.qld.gov.au/species/?op=getsurveysbyspecies"

    lat, lon = coordinate["latitude"], coordinate["longitude"]

    params = {

        "taxonid": str(taxonid),

        "circle": f"{lat},{lon},{radius}"

    }

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

    data = response.json()

    return data.get("features", [])

#Searches sightings using taxonoid,cordinate and radius

def search\_sightings(taxonid, coordinate, radius):

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

    incidental\_sightings = [s for s in surveys if s.get("properties", {}).get("SiteCode") == "INCIDENTAL"]

    return incidental\_sightings

#Searches the sightings from the properties with a start date

def earliest(sightings):

    return min(sightings, key=lambda x: x.get("properties", {}).get("StartDate"), default=None)

#Searches the Sightings with the properties and a start dates

def sort\_by\_date(sightings):

    return sorted(sightings, key=lambda x: x.get("properties", {}).get("StartDate"))

#Searches the display sightings by sorting the date& location

def display\_sightings(sightings):

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

    for sighting in sorted\_sightings:

        print(f"Date: {sighting.get('properties', {}).get('StartDate')}, Location: {sighting.get('properties', {}).get('LocalityDetails')}")

#Inputs ni it with Latitude and Longitude

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

    coordinate = {"latitude": -16.92, "longitude": 145.777}

    radius = 100000

    taxonid = 860

#list of species

species\_list = get\_species\_list(coordinate, radius)

    print("Species List:", species\_list)

    #surveys by species

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

    print("Surveys:", surveys)

    # For incidental sightings

incidental\_sightings = search\_sightings(taxonid, coordinate, radius)

    print("Incidental Sightings:", incidental\_sightings)

    # Display and sort these sightings if any

if incidental\_sightings:

    display\_sightings(incidental\_sightings)

else:

    print("No incidental sightings found.")

Git hub link: https://github.com/GhubU121/Ass-P1COIT.git