`

Analysing and displaying relevant compound information through text/camera input

Final Project Report

Jeremy Nathanael Gunawan (2802522960)

L1BC

Jude Joseph Lamug Martinez MCS

Algorithm and Programming

Binus International University

Jakarta

`

Table of Contents

`

Abstract

This document serves as a documentation report for the Algorithm and Programming's Final Project. It demonstrates a student's proficiency utilizing the programming language, Python, through the creation of a program made by the student's incorporation of the language's fundamental data structures. The report made will focus on a particular program that outputs molecular descriptors by inputting the name of the compound name.

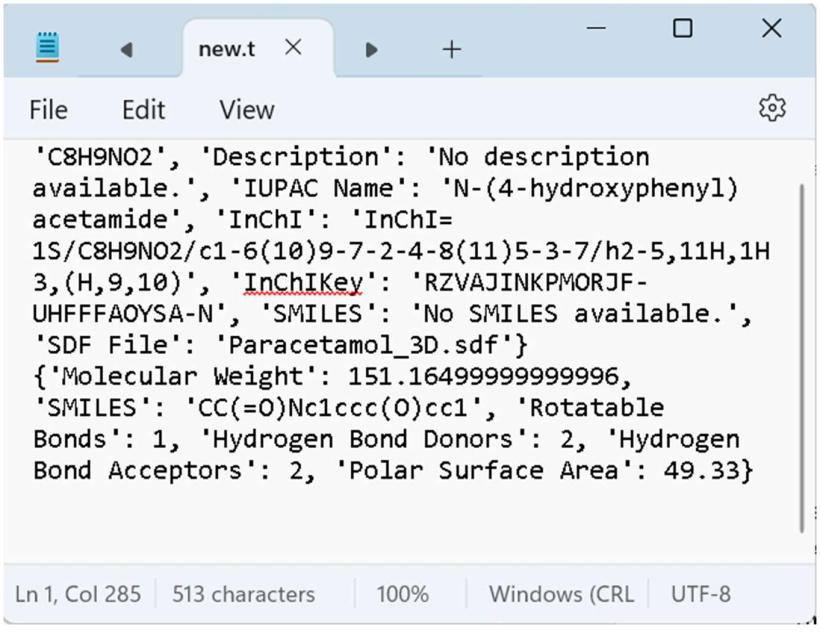
Project specification

Introduction

The final project tasks students to push their knowledge of Python outside what they have been taught in classes. It will determine how fluent and capable utilizing the language and to solve real-world problems through programming. Specifically, the project challenges the support for students who are learning chemical compounds and requires a quick way to view the structures of the compound and additionally specific information (such as SMILES, IUPAC, hydrogen bonds, polar surface area, etc.). It will be a niche help for drug discovery, cheminformatics, and materials science.

All the data will similarly look like the figure below. Based from the database.

Figure 1:



`

Sources of Inspiration

The idealisation comes from my deep aspiration to pursue medicinal chemistry- although I am a computer science student. The specification to create a program that tackles real-world challenges made me think to use my knowledge that supports a field even if I will not be in it. The quick display of molecular descriptors will surely aid in education in relation to biology and chemistry, as data comes to prominence in these majors especially when needing to handle with prediction, calculations, and structures to easily be seen without difficulties. In addition, I added camera features as I was inspired by my senior, Almira Farahiyah Shafiqa Rana, as she used OCR and machine learning to create a hangman game for the deaf.

Solution Design

This is a simple chemical compound analyser that utilizes different modules and library in order to facilitate the objective of the program. Firstly, each module file .py must be made first.

Figure #2 chem_data_output.py modules :

```
1  from rdkit import Chem
2  from rdkit.Chem import Draw, Descriptors, AllChem
```

chem_data_output.py, comprises of the import of RDKit library and the analyze_compound function. This function utilizes the compound_name as input. It will then load in the 3D molecular structure, generate 3D conformer, save the 2D structure as well, and finally display molecular descriptors through the use of a dictionary. To go further into depth, the 'Chem' functions in handling the chemical structure, 'Draw' generates the structures, 'Descriptors' calculates the molecular properties of the compound, and finally 'AllChem' handles with the manipulation of the molecule's data and also the conformer generation. 'Chem.SDMolSupplier' in this module functiosn to load the SDF file obtained from another module of this program. mol = Chem.SDMolSupplier(sdf_filename)[0], the 0 here functions to fetch the first molecule to be displayed.

`

Figure #3 chem_data_output.py the function that obtains the molecular descriptors:

```
# 3D conformer, based from databases
AllChem.EmbedMultipleConfs(mol, numConfs=1)

# Obtain properties
# Molecular weight
mol_weight = Descriptors.MolWt(mol)
# SMILES string
smiles = Chem.MolToSmiles(mol)
# Number of rotatable bonds
rotatable_bonds = Descriptors.NumRotatableBonds(mol)
# Number of hydrogen bond
hydrogen_bond_donors = Descriptors.NumHDonors(mol)
# Number of hydrogen bond acceptors
hydrogen_bond_acceptors = Descriptors.NumHAcceptors(mol)
# Polar surface area
polar_surface_area = Descriptors.TPSA(mol)
```

Figure #4 the dictionary displayed for molecular descriptor output in chem_data_output.py:

```
    return {
        "Molecular Weight": mol_weight,
        "SMILES": smiles,
        "Rotatable Bonds": rotatable_bonds,
        "Hydrogen Bond Donors": hydrogen_bond_donors,
        "Hydrogen Bond Acceptors": hydrogen_bond_acceptors,
        "Polar Surface Area": polar_surface_area,
    }
xcept Exception as e:
```

Figure #5 datalookup.py modules and library :

`

```python
# Import
# This import is necessary for requesting HTTP requests to download files which will be re
import requests
# For searching in the PubChem database
import pubchempy as pcp

# Function to obtain data of chemical compound from PubChem database
def lookup_compound_data(compound_name):  2 usages
```

datalookup.py, contains the libraries pubchempy and the module ‘requests’. This module functions similarly to the previous module mentioned (chem_data_output.py). It will also show the molecular descriptors of the compound obtained through retrieval of the chemical data by requesting through HTTP request with the module ‘requests’ accessing the PubChempy database by the library ‘pubchempy’.

Figure #6 and #7 datalookup.py data retrieval request :

```python
# Obtain relevant information based from the compound name
compound_results = pcp.get_compounds(compound_name, namespace='name')
# Display this message if unable to find chemical name in PubChem
if not compound_results:
    # The output
    print(f"Compound '{compound_name}' not found in PubChem.")
    return None

# Compound variable
compound = compound_results[0]
```

```python
# If-condition when unable to obtain smiles
if smiles == "No SMILES available/found.":
    cid = compound.cid
    # Requests for SDF file
    sdf_url = f"https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/{cid}/sdf"
    sdf_response = requests.get(sdf_url)

    # If request for SDF file is successful,
    # Status code 200 is when it is successful
    if sdf_response.status_code == 200:
        # Variable classification
        sdf_filename = f"{compound_name}_3D.sdf"
        # Save SDF file
        with open(sdf_filename, "wb") as sdf_file:
            sdf_file.write(sdf_response.content)
```

The figure above shows the function that requests and retrieves the data, pcp.get_compounds. Each modules are also provided with simple error handling to notify users if an error is encountered. compound_results[0] here functions that it will select the first compound based from the search and as a result of the search. The figures below shows the similar process to chem_data_output.py on returning the chemical properties/relevant datas.

Figure #8 and #9 datalookup,py data return:

```python
# Retrieve compound details

# Molecular weight
molecular_weight = compound.molecular_weight
# Molecular formula
molecular_formula = compound.molecular_formula


# Other further information regarding the molecule, each has its own message when failed to

# Description, however the code tends to be quite useless for now. This code line will stil
# Outputs a message when failed or unable to obtain description
description = compound.description if hasattr(compound, 'description') and compound.descrip

# IUPAC name
iupac_name = compound.iupac_name if hasattr(compound, 'iupac_name') else "No IUPAC name ava

# Chemical identifiers
# InChi and InChiKey
inchi = compound.inchi if hasattr(compound, 'inchi') else "No InChI available/found."
inchikey = compound.inchikey if hasattr(compound, 'inchikey') else "No InChIKey available/f

# SMILES ( Simplified Molecular Input Line Entry System ). The output tends to be unreliabl
smiles = compound.smiles if hasattr(compound, 'smiles') else "No SMILES available/found."
```

```python
#         Returns the compound inf
return {
    "Molecular Weight": molecular_
    "Formula": molecular_formula,
    "Description": description,
    "IUPAC Name": iupac_name,
    "InChI": inchi,
    "InChIKey": inchikey,
    "SMILES": smiles,
    "SDF File": sdf_filename,
}
```

`

Figure #10 camera.py modules and library:

```
# Importing the OpenCv and pytesseract
import cv2
import pytesseract

# End of importing
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR'
```

Note: it is extremely important to implement the code line there after the comment "# End of importing". This is needed by the module/program to identify the path where they can run the OCR software.

camera.py, utilizes pytesseract library and OpenCv library to handle image processing functions. This module will capture the image taken by the camera and extract the text from the image. It will allow the user to make use of their device's built-in camera if they need to handle a lot of chemical structures. cv2 is used for capturing the video from the camera and pytesseract is used for extracting the text from the capture. The function, def compound_capture() is the only def function crucial for the program. cv2.VideoCapture(0) will utilize the default camera or the first camera of the device.

Figure #11 camera.py code snippet:

```
# Indefinitely loops that displays the camera frame
while True:
    # Capture frame from camera and will also indicate whether reading the frame w
    ret, frame = cap.read()

    # Show what the camera sees in a popup window, named "Camera" instead of frame
    cv2.imshow( winname: "Camera", frame)

    # Validates if user presses the valid key, 's' to capture the frame
    if cv2.waitKey(1) & 0xFF == ord('s'):
        # Save the current frame as 'captured_frame.jpg'
        cv2.imwrite( filename: "captured_frame.jpg", frame)

        # Use pytesseract to extract text from the saved image
        compound_name = pytesseract.image_to_string("captured_frame.jpg").strip()

        # Display the text it read from the frame
        print(f"Extracted text: {compound_name}")

    # Break the loop if the user presses 'q'
    elif cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

The figure above shows ret, frame = cap.read(); where .read() captures a frame as image by the user if activated and 'frame' will contain the data of the image. ret indicates if it was successful. The cv2.imshow followed by "Camera", frame has a simple function which names the popup window as "Camera" and the frame will be the data. if cv2.waitKey(1) & 0xFF == ord('s'): is used to capture a frame of the video capture. cv2.imwrite("captured_frame.jpg", frame) this will save the frame as

`

“captured_frame.jpg” to compare if desired. pytesseract.image_to_string("captured_frame.jpg").strip() will extract the text from the image to a string and .strip() provides better performance as it will erase empty spaces after the name of the compound which actually causes problems when searching in the database. The print functionality there will simply display the extracted text and, finally, the last remaining code lines of the figure functions for the user to break the camera loop. As this is because this module is made so that closing the camera popup window (clicking the x button) will simply open the camera popup window back in a loop indefinitely until the user breaks it by pressing “s”. cap.release() and cv2.destroyAllWindows() stops the camera.

Figure #12 save_chem_Data.py module and code snippet:

```python
# For external operations
import os
# For moving files
import shutil

#Function to save data to a file in a folder
def save_data_to_file(folder_name, data, filename):  2 usages

    try:
        # Basically checks the existence of the folder. If folder does not exist,
        os.makedirs(folder_name, exist_ok=True)

        # The file is saved to a .txt file
        # Variable for path of .txt file in folder
        text_file_path = os.path.join(folder_name, filename)
        # Open the .txt file and writes data to the file
        with open(text_file_path, 'w') as f:
            # Calls a function to write the data in the file
            f.write(data)
        #     Feedback that saving the data was a success
        print(f"Data successfully saved to {text_file_path}")
    #     Error handling
    except Exception as e:
        print(f"Error saving data to file: {e}")
```

save_chem_Data.py is a simple module that has the module ‘OS’. Generally, it will save the result of the analysis as a text file in a folder if it exists and will save it to a default folder made if user enters a wrong input. ‘os’ allows the program to interact with the operating system. In here, ‘data’ in the def function will be the content of file which is the analysis data. os.makedirs(folder_name, exist_ok=True) is an error handling that checks if the folder already exists which prevents errors for the later part of the module. os.path.join connects the folder name and filename specified. with open(text_file_path, 'w') as f: f.write(data) will write the analysis data to the file.

Figure #13 structure3dmodel.py module and software file path:

`

```
import os
# For opening the images
from PIL import Image
# To run external operations such as having to open software
import subprocess
# Import End

# File path for the Jmol software
# Variable is done with full capitalisation to help make it easier to find the path varia
JMOL_PATH = r'D:\downloadtemp\jmol-16.3.5\Jmol.jar'
```

Note : It is important to specify the path of the jmol software file path such as the one shown above. The format is a variable or it could be implemented into the def function, however the format is r'{file_path}'. Java is also required to run the software.

The structure3dmodel.py module handles with the displaying of the 3d structure and 2d structure. It has two def functions in this module- display_image() and open_jmol(). The figure below shows the code snippet for both these functions.

Figure #14 structure3dmodel.py def functions:

```
# Function to display 2D structure by image
def display_image(compound_name):  2 usages
    # Naming the image by the name of chemical compound followed by _2D + extension
    image_filename = f"{compound_name}_2D.png"
    # Checks if image exists then will open
    if os.path.exists(image_filename):
        # Opens image with Image.open and display with img.show()
        img = Image.open(image_filename)
        img.show()
    else:
        # Prints if it does not exist
        print(f"2D image {image_filename} not found.")

# Function to open Jmol software
def open_jmol(compound_name):  2 usages
    # Name policy similar to the one in the def display_image
    sdf_filename = f"{compound_name}_3D.sdf"
    if os.path.exists(sdf_filename):
        # It provides feedback for the action if it is happening
        print(f"Opening Jmol for 3D structure: {compound_name}")
        # It instructs to open jmol software by running the "java -jar" and then followed by
        jmol_display = f'java -jar {JMOL_PATH} {sdf_filename}'
        # Opens jmol with the sdf file import and uses the system's shell to run the softwar
        subprocess.run(jmol_display, shell=True)
    else:
        # Simple error handling if sdf is not found
        print(f"SDF file {sdf_filename} not found.")
```

display_image functions to display the image of the chemical's 2d structure generated and makes use of 'os' to determine if the file exists (os.path.exists(image_filename) and opens with both

`

Image.open(image_filename and img.show(). On the other hand, open_jmol will open an external software used to display 3d structure of chemical properties. The jmol software will require the file path of the .exe file and java. The software can be opened with java through the code ‘java -jar’.

Figure #15 chem_analyzer_main.py modules:

chem_analyzer_main.py × chem_data_output.py datalookup.py camera.py save_chem_Data.py

```
# Importing the modules into the main py file
import os
from structure3dmodel import open_jmol, display_image
from datalookup import lookup_compound_data
from chem_data_output import analyze_compound
from save_chem_Data import save_data_to_file
from camera import compound_capture

```

Finally, chem_analyzer_main.py, is the main .py that imports all the personalized modules. All the def functions have already been imported to the main .py file here. As the name suggests, it will run and accomplish as the main program. In this part, how the program will run will be told in chronological order. Firstly, def main() will start the program. It will prompt the user to input the chemical compound’s name between two options, 1 (manual typing input of the chemical compound) and 2 (camera capture). If user chooses option 2, it will initiate the def function ‘compound_capture()’ from camera module. In both cases, the input will be known as the variable ‘compound_name’. The variable will be used to in the initiation of the lookup_compound_data() and analyze_compound() to lookup the name of the compound and retrieve the molecular descriptors. Afterwards, it will return the molecular descriptors by the codes in the figure below.

Figure #16 chem_analyzer_main.py molecular descriptor output :

```
# Display compound data
print(f"\nCompound Data:")
for key, value in compound_data.items():
    print(f"{key}: {value}")

# Analyze compound properties using RDKit
analysis_results = analyze_compound(compound_name)

# If analysis function fails, (error handling)
if not analysis_results:
    print("Error in process of analyzing the compound using RDKIT. Exiting program. Ple
    return

# Display compound analysis result
print("\nCompound Analysis Results:")
for key, value in analysis_results.items():
    print(f"{key}: {value}")
```

The main .py file also contains choices for the users to save the analysis data of the chemical compounds. They are given two options, to save or not to save the chemical data. If yes, however, it will prompt the user to save into an existing folder or create a new folder if they desire to save it- a

`

wrong input will automatically save it to a folder that is made as an error handling folder if user chooses an option outside of the option. Lastly, saving or not saving the chemical data will display both 2d and 3d structures of the chemical structure. It is saved as the last part to prevent error as it tends to erase the analysis data to be displayed.

Figure #17 chem_analyzer_main.py save options:

```
# Ask user if they want to save the results
save_choice = input("Do you want to save the results? (yes/no): ").strip().lower()

if save_choice == 'yes':
    # Prompt user for folder selection
    create_or_use_folder = input(
        "Do you want to use an existing folder or create a new one? (existing/new): ").

    # Set the folder based on user input
    if create_or_use_folder == 'existing':
        folder_name_input = input("Enter the existing folder name: ").strip()
        if os.path.exists(folder_name_input):
            folder_name = folder_name_input
        else:
            print("Folder does not exist. Saving to default folder 'saved_untitled_comp
            folder_name = "saved_untitled_compound_data"

    elif create_or_use_folder == 'new':
        folder_name = input("Enter the new folder name: ").strip()
        # Create the new folder
        os.makedirs(folder_name, exist_ok=True)

    else:
        print("Invalid choice. Saving to default folder 'saved_untitled_compound_data'.
        folder_name = "saved_untitled_compound_data"
```

`

Discussion

The chemical compound analyser program eases users to access chemical compounds and obtain relevant molecular descriptors without any difficulty and to save time. The algorithm that comes to prominence are the different personalized modules that will need to function otherwise it will not work. Data retrieval and analysis – bot RDKit and PubChempy - are strong aspects in this program. It will aid in working with chemical data with efficiency and effectiveness. The image processing libraries are also essential, OpenCV and pytesseract. In addition, to the modules that we needed 'OS' 'subprocess' 'pillow'. The run time and performance can vary due to the HTTP requests to the database which will require internet connection as the database is an online repository.

The processing libraries, OpenCV and pytesseract, allows capturing/extracting texts from the frame captured as data by the video capture. This provides an alternative for inputting the compound name which lessens the need to type the name manually. Furthermore, the implementation of 'OS' and 'subprocess', and 'Pillow' allows file manipulation, running external software, and perform image handling.

In terms of data structure, the chem analyser program comprises of dictionaries for storing data, managing data, and displaying the analysis data. Key-value pairs is used in the main module to return the data. This makes it easy to understand how the program works and is a versatile way of handling data.

The run-time will vary depending on the connectivity. HTTP requests to the PubChempy online repository database is being accessed by the program for the search and retrieval of the chemical data. Internet connectivity will be required and utilizing HTTP requests is a simpler way than to start from source to access certain file databases.

Limitations

Some flaws are still present that is experienced when using the program. It is extremely hard to capture texts using the OCR depending on the camera quality. Majority of the time, it will not be able to detect the text by camera. The only way for it to be functional and well is to repetitively capture the frame from the video capture over and over until the user breaks it and confirms the text. However, that solution/alternative will be part of the main root cause of the lag to the frame rate and performance of the device as it will constantly capture over and over. Therefore, it is best to implement a button to capture over and over only by pressing the button. This may make it hard to focus the text when the user needs to move their fingers to press the button.

Figure #18 Error camera capture:

```
An error occurred while capturing text from the camera: [WinError 5] Access is denied
No compound name provided. Exiting.
```

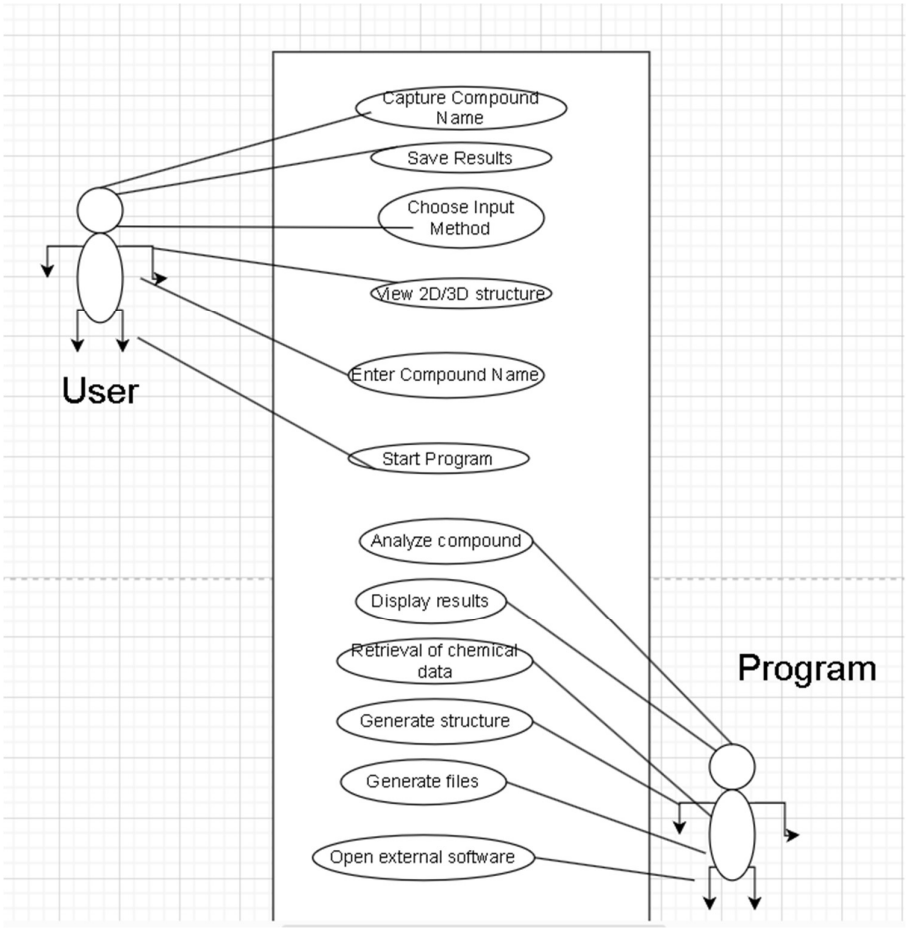
Due to not utilizing the best API, it will be harder to display a more complete information. The use of APIs may cause problems which displays red texts in the terminal. This goes the same for

`

handling with huge data files. The program may crash or load for a very long time. Some bond planes may interfere with one another which the RDKit will experience problems generating a conformer. Furthermore, the privacy and security of the system may hinger the program's functionality. Currently, due to saving and moving of many data files, the program has experienced being unable to capture and extract text due to, possibly, securities- such as "###Error 5 Access is denied".

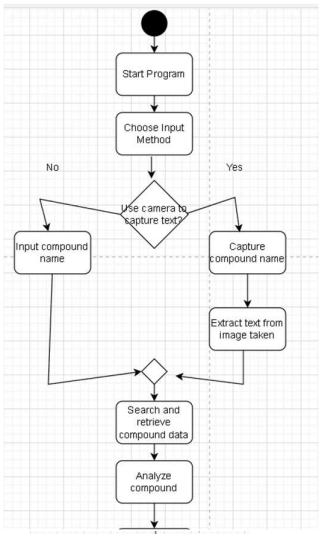
Connectivity to the internet becomes a necessity unfortunately as a file database is not being used here. Testing it with a file database downloaded have shown errors, thus the program will make use of HTTP requests to the PubChempy for the search and retrieval of data. The HTTP request may provide a failure response if connectivity is low. The time for the program to finish its analysis will depend on the connectivity and how large the data is as a result.

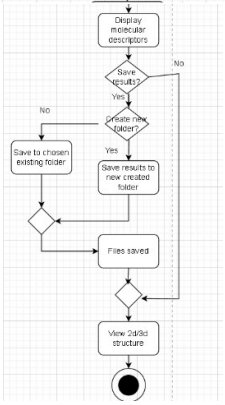
Use-case Diagram



`

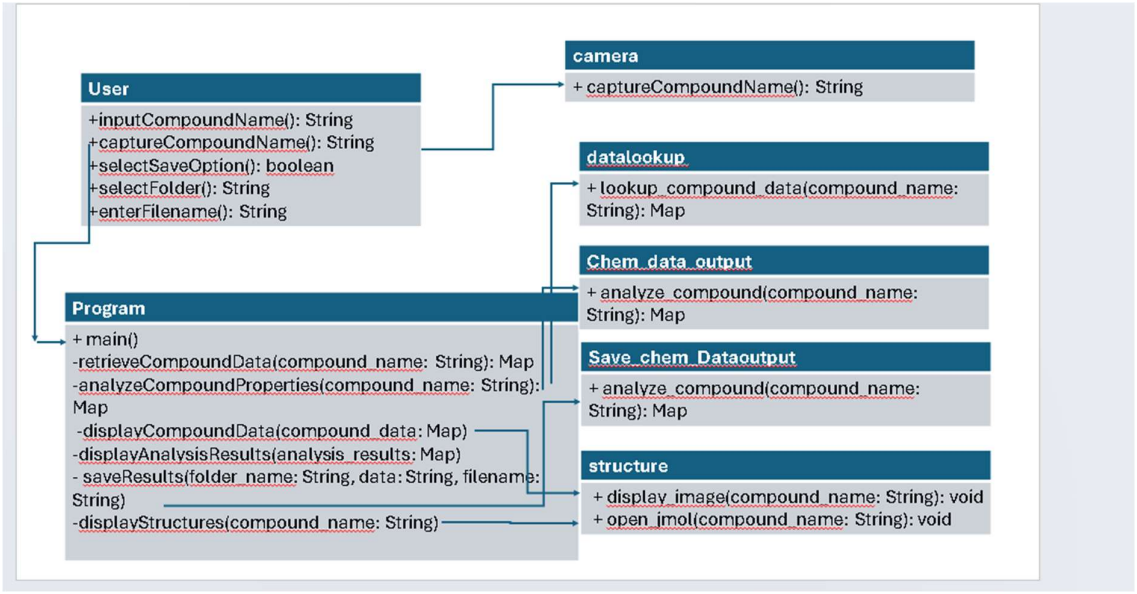
Activity Diagram





`

Class_diagram

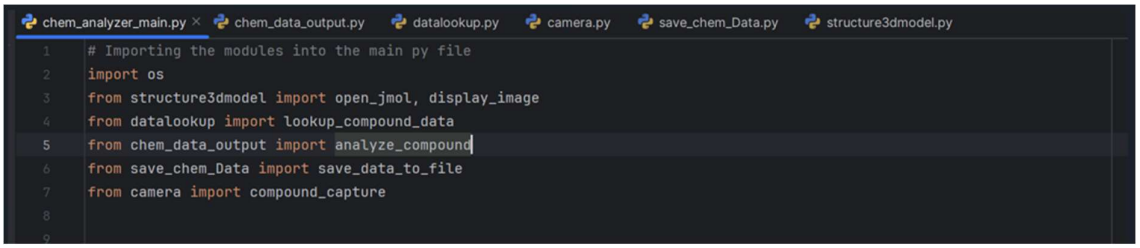


`

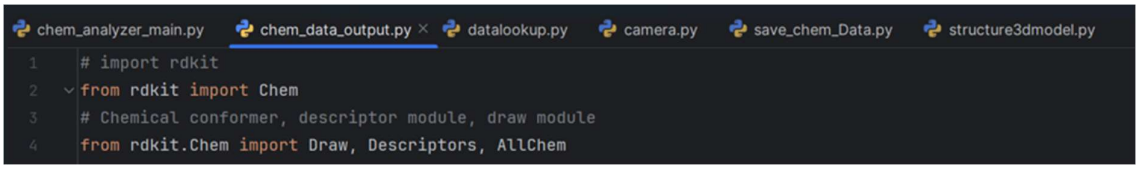
Modules and libraries:

Code snippet of the modules imported:

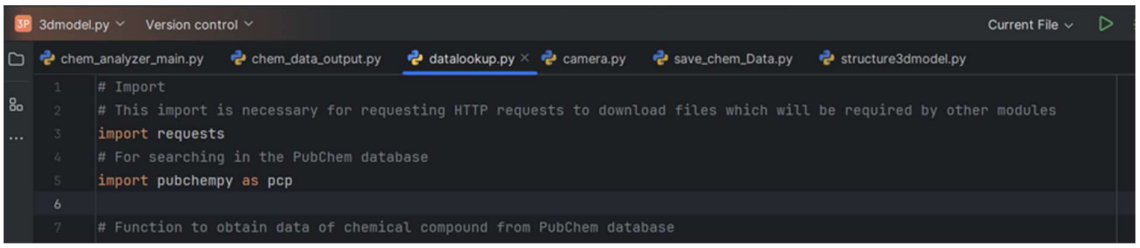
Module snippet #1 (chem_analyzer_main.py):



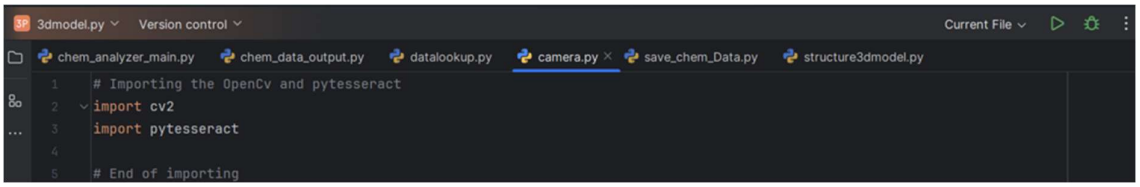
Module snippet #2 (chem_data_output.py):



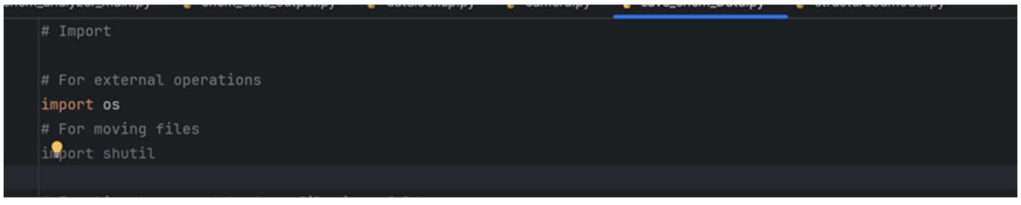
Module snippet #3 (datalookup.py):



Module snippet #4 (camera.py):

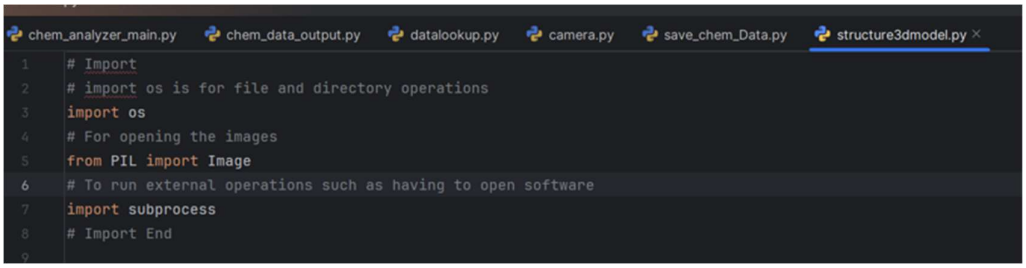


Module snippet #5 (save_chem_Data.py):



Module snippet #6 (structure3dmodel.py):

`



'OpenCV'

OpenCV (Open Source Computer Vision Library) is an open-source software library for computer vision, machine learning, and image/video processing. The library features image/video analysis (object recognition, facial detection, and tracking). It is compatible with multiple platforms and programming languages.

'OS'

Operating System (OS) is a built-in Python module that interacts with the operating system to interact with files (File and Directory Operations), environment variable management, and system command operation.

'RDKit'

RDKit (rdkit) is an open-source Cheminformatics library for cross platforms and programming languages (Python, C++, C#). It provides utilities to manipulate and to perform analysis of chemical structures. In addition, the library is well-known for its ability to generate file formats (SMILES, SDF, PDB), displaying molecular structures in 2D/3D, molecular calculations, and predictions; commonly known for its utilization in drug discovery, chemical research, chemical data analysis, etc.

'subprocess'

Subprocess is a Python module that runs external commands, scripts, and programs- such as running .exe files like notepad.

'PIL'

pillow (PIL) is open-source library for image processing. It provides utilization to open, manipulate, and save images in various file formats. The library allows multiple applications- such as resizing, cropping, rotating, flipping, adding shapes and also text to images, etc.

'Pytesseract'

Pytesseract is a library that features optical character recognition (OCR) tools. It allows the extraction of text from images, documents, and PDFs. Its application is commonly known for data extraction, image processing operations, etc.

'requests'

requests is a library that involves making HTTP requests. Its applications range from sending HTTP requests and obtaining responses to interacting with websites, APIs, and online operations.

'PubChemPy'

PubChemPy (pubchem) is a library to interact with the PubChem database, where it is an online repository comprising chemical information, and a chemical toolkit. It allows searching and the

`

retrieval chemical data online through their database, and manipulating chemical data- such as compound properties, structural information, etc. This provides easy-to-access data for the applications in chemistry, pharmacology, and bioinformatics.

Made/personalized modules:

chem_analyzer_main.py : The main function of the program .py file that imports all the personalized modules.

datalookup.py : datalookup.py is responsible for the search and retrieval compound data by accessing the PubChem database.

chem_data_output.py : chem_data_output.py is responsible with the analysis and presenting the results of the chemical compound through RDKIT.

save_chem_Data.py : Saves the retrieved and analyzed data to a file through the use of OS.

camera.py : camera.py captures the text from the image taken with the use of OpenCv and pytesseract.

structure3dmodel.py : The structure3dmodel.py module provides the functionalities to display 2D and 3D structures of chemical compounds with Pillow and subprocess.

Evidence of Working Program

Source Code:

https://github.com/JJ-wqr/Algoprog_Final.git

chem_analyzer_main.py code snippets

Code Snippet #1 (chem_analyzer_main.py):

`

chem_analyzer_main.py | chem_data_output.py | datalookup.py | camera.py | save_chem_Data.py | structure3dmodel.py

```python
# Importing the modules into the main py file
import os
from structure3dmodel import open_jmol, display_image
from datalookup import lookup_compound_data
from chem_data_output import analyze_compound
from save_chem_Data import save_data_to_file
from camera import compound_capture


# End of importing

def main():  1 usage
    print("Welcome to the Chemical Compound Information Program!")

    # Options for user to choose between inputting the text manually or to use camera
    # It will loop until user inputs valid choice
    while True:
        print("Choose an option:")
        print("1. Enter compound name manually")
        print("2. Use camera to capture compound name")
        choice = input("Enter your choice (1/2): ").strip()

        # Breaks if user correctly inputs
        if choice == '1':
            # Obtain user input and will strip whitespaces
            compound_name = input("Enter the compound name: ").strip()
            break
```

Code Snippet #2 (chem_analyzer_main.py):

```python
        elif choice == '2':
            # Initiate the camera function to obtain input (the chemical compound name)
            compound_name = compound_capture()
            break
        else:
            # Display this message if input is not a valid choice
            print("Invalid choice. Please enter 1 or 2.")

    # If it is not a compound that exists or is unable to read, then display this message.
    if not compound_name:
        print("No compound name provided. Exiting.")
        return

    # Obtains compound information/data from PubChem
    compound_data = lookup_compound_data(compound_name)

    # If unable to find compound data
    if not compound_data:
        print("Compound not found. Exiting program. Please run the program again!")
        return

    # Display compound data
    print(f"\nCompound Data:")
    for key, value in compound_data.items():
```

Code Snippet #3 (chem_analyzer_main.py):

`

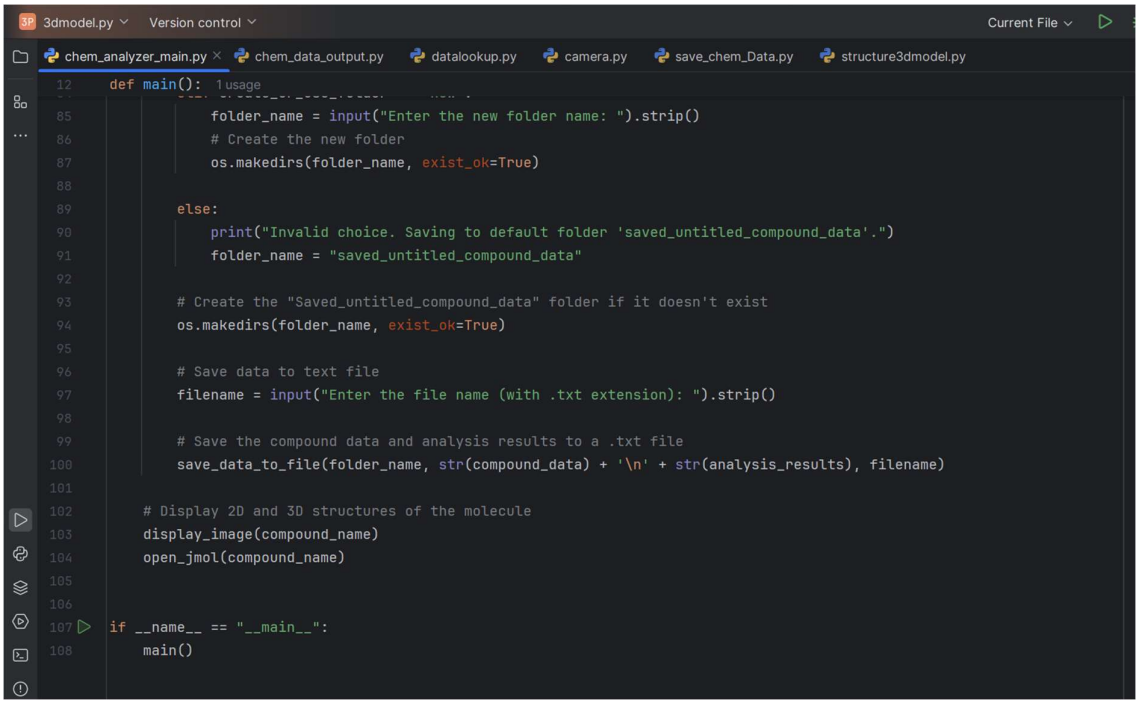
```
for key, value in compound_data.items():
    print(f"{key}: {value}")

# Analyze compound properties using RDKit
analysis_results = analyze_compound(compound_name)

# If analysis function fails, (error handling)
if not analysis_results:
    print("Error in process of analyzing the compound using RDKIT. Exiting program. Please run the program again!")
    return

# Display compound analysis result
print("\nCompound Analysis Results:")
for key, value in analysis_results.items():
    print(f"{key}: {value}")

# Ask user if they want to save the results
save_choice = input("Do you want to save the results? (yes/no): ").strip().lower()

if save_choice == 'yes':
    # Prompt user for folder selection
    create_or_use_folder = input(
        "Do you want to use an existing folder or create a new one? (existing/new): ").strip().lower()

    # Set the folder based on user input
```

Code Snippet #4 (chem_analyzer_main.py):

```
# Set the folder based on user input
if create_or_use_folder == 'existing':
    folder_name_input = input("Enter the existing folder name: ").strip()
    if os.path.exists(folder_name_input):
        folder_name = folder_name_input
    else:
        print("Folder does not exist. Saving to default folder 'saved_untitled_compound_data'.")
        folder_name = "saved_untitled_compound_data"

elif create_or_use_folder == 'new':
    folder_name = input("Enter the new folder name: ").strip()
    # Create the new folder
    os.makedirs(folder_name, exist_ok=True)

else:
    print("Invalid choice. Saving to default folder 'saved_untitled_compound_data'.")
    folder_name = "saved_untitled_compound_data"

# Create the "Saved_untitled_compound_data" folder if it doesn't exist
os.makedirs(folder_name, exist_ok=True)

# Save data to text file
filename = input("Enter the file name (with .txt extension): ").strip()

# Save the compound data and analysis results to a .txt file
```

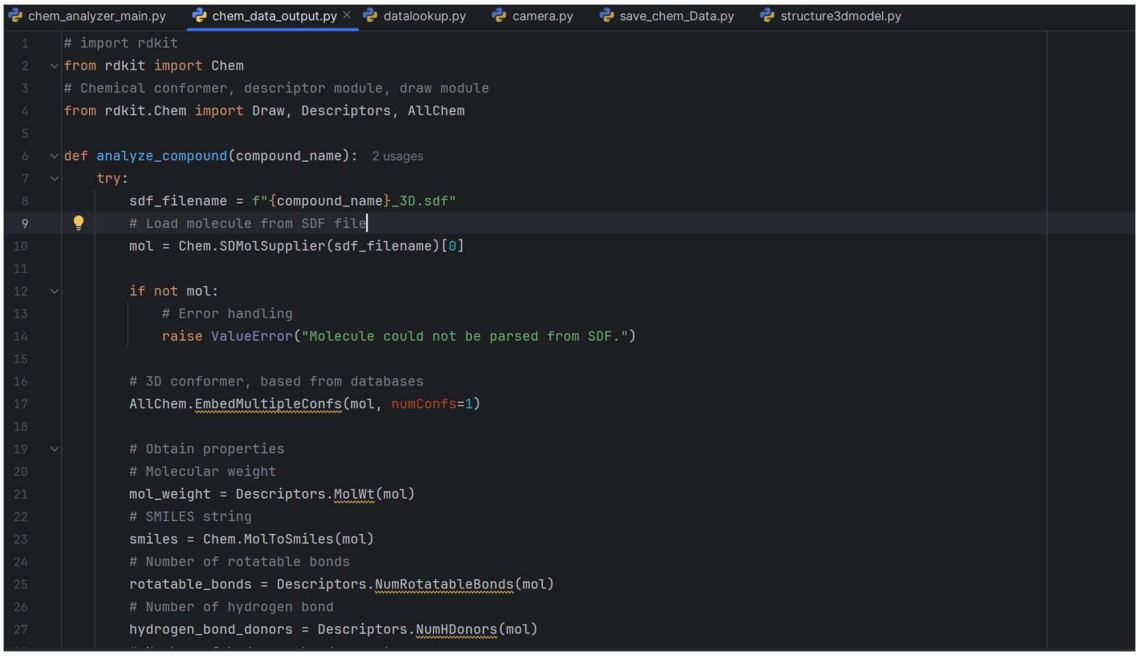
Code Snippet #5 (chem_analyzer_main.py):

`



chem_data_output.py code snippets

Code Snippet #1 (chem_data_output.py):



Code Snippet #2 (chem_data_output.py):

`

```
def analyze_compound(compound_name):  2 usages
        hydrogen_bond_acceptors = Descriptors.NumHAcceptors(mol)
        # Polar surface area
        polar_surface_area = Descriptors.TPSA(mol)

        # Generates 2D structure
        img = Draw.MolToImage(mol)
        # Image save name format
        img.save(f"{compound_name}_2D.png")
        # Provides feedback
        print(f"2D structure saved as {compound_name}_2D.png")

        # Outputs molecular information
        return {
            "Molecular Weight": mol_weight,
            "SMILES": smiles,
            "Rotatable Bonds": rotatable_bonds,
            "Hydrogen Bond Donors": hydrogen_bond_donors,
            "Hydrogen Bond Acceptors": hydrogen_bond_acceptors,
            "Polar Surface Area": polar_surface_area,
        }
    except Exception as e:
        # Error handling
        print(f"Error analyzing compound: {e}")
    #     Return nothing when error
    return None
```

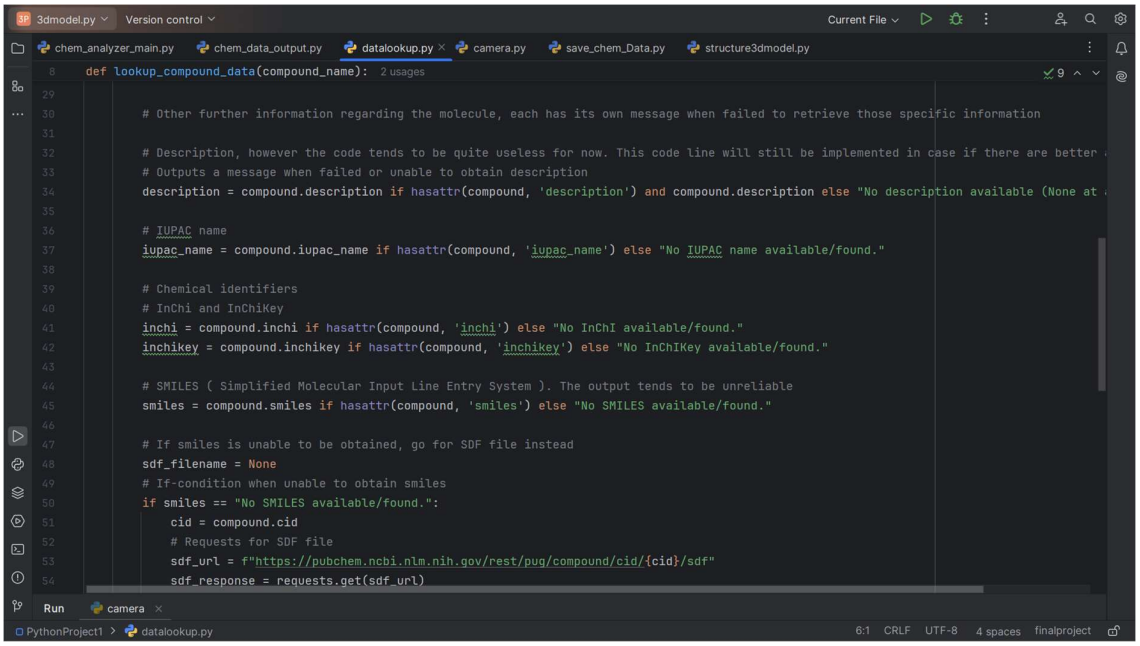
datalookup.py code snippets

Code Snippet #1 (datalookup.py):

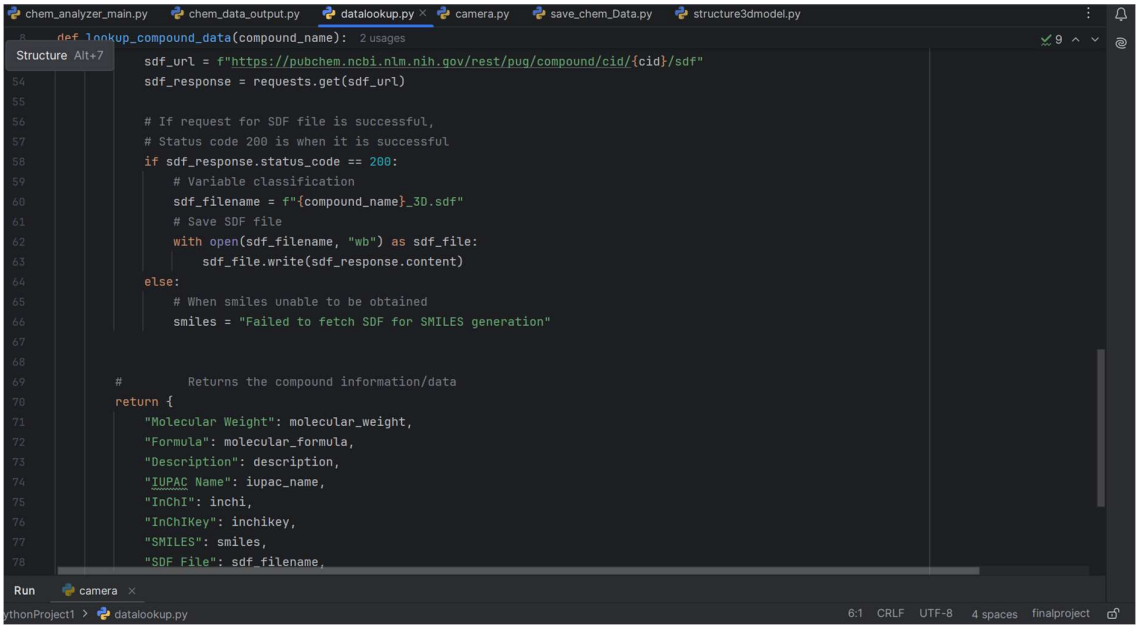
```
# Import
# This import is necessary for requesting HTTP requests to download files which will be required by other modules
import requests
# For searching in the PubChem database
import pubchempy as pcp

# Function to obtain data of chemical compound from PubChem database
def lookup_compound_data(compound_name):  2 usages

    try:
        # Obtain relevant information based from the compound name
        compound_results = pcp.get_compounds(compound_name, namespace: 'name')
        # Display this message if unable to find chemical name in PubChem
        if not compound_results:
            # The output
            print(f"Compound '{compound_name}' not found in PubChem.")
            return None

        # Compound variable
        compound = compound_results[0]

        # Retrieve compound details

        # Molecular weight
        molecular_weight = compound.molecular_weight
        # Molecular formula
        molecular_formula = compound.molecular_formula
```

Code Snippet #2 (datalookup.py):

`



Code Snippet #3 (datalookup.py):



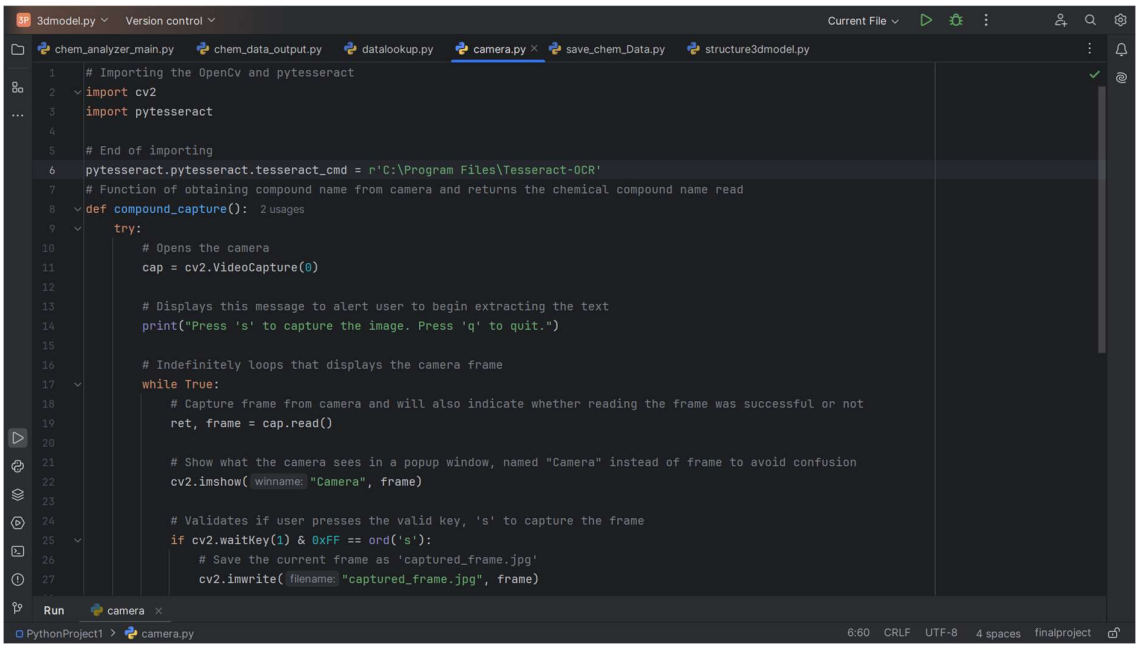
Code Snippet #4 (datalookup.py):

`

```
        #         Returns the compound information/data
        return {
            "Molecular Weight": molecular_weight,
            "Formula": molecular_formula,
            "Description": description,
            "IUPAC Name": iupac_name,
            "InChI": inchi,
            "InChIKey": inchikey,
            "SMILES": smiles,
            "SDF File": sdf_filename,
        }
    # Error handling
    except Exception as e:
        print(f"Error fetching data for {compound_name}: {e}")
        return None
```

camera.py code snippets

Code Snippet #1 (camera.py):



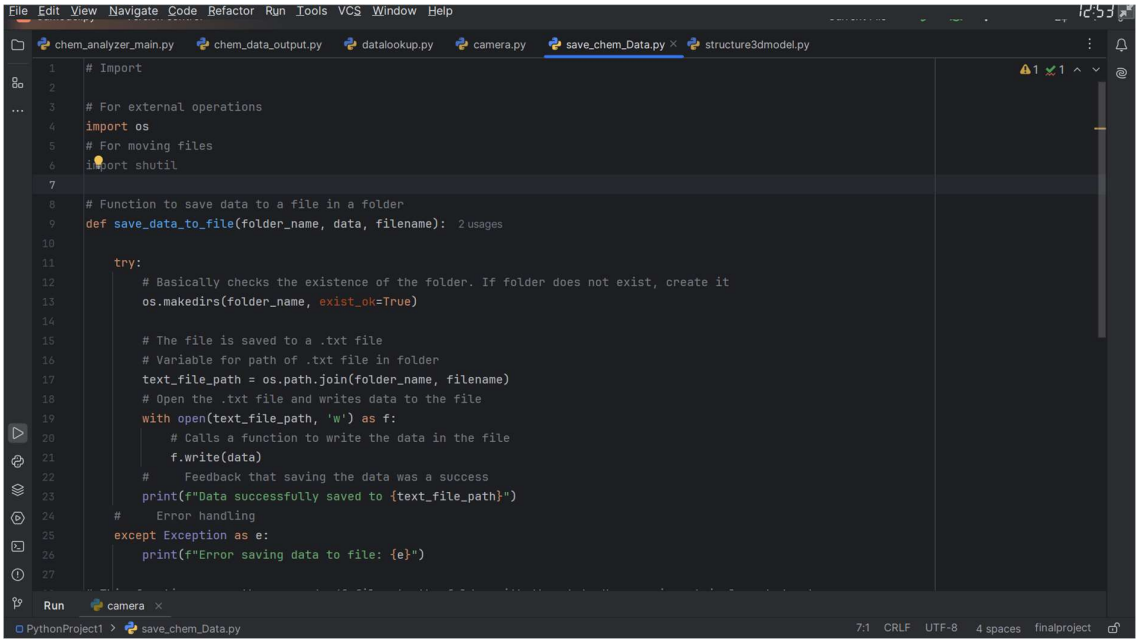
Code Snippet #2 (camera.py):

`

```
                # Use pytesseract to extract text from the saved image
                compound_name = pytesseract.image_to_string("captured_frame.jpg").strip()

                # Display the text it read from the frame
                print(f"Extracted text: {compound_name}")

            # Break the loop if the user presses 'q'
            elif cv2.waitKey(1) & 0xFF == ord('q'):
                break

        # Exits camera and closes it
        cap.release()
        cv2.destroyAllWindows()

    # Error handling
    except Exception as e:
        # If an error occurs, display this message
        print(f"An error occurred while capturing text from the camera: {e}")
```

save_chem_Data.py code snippets

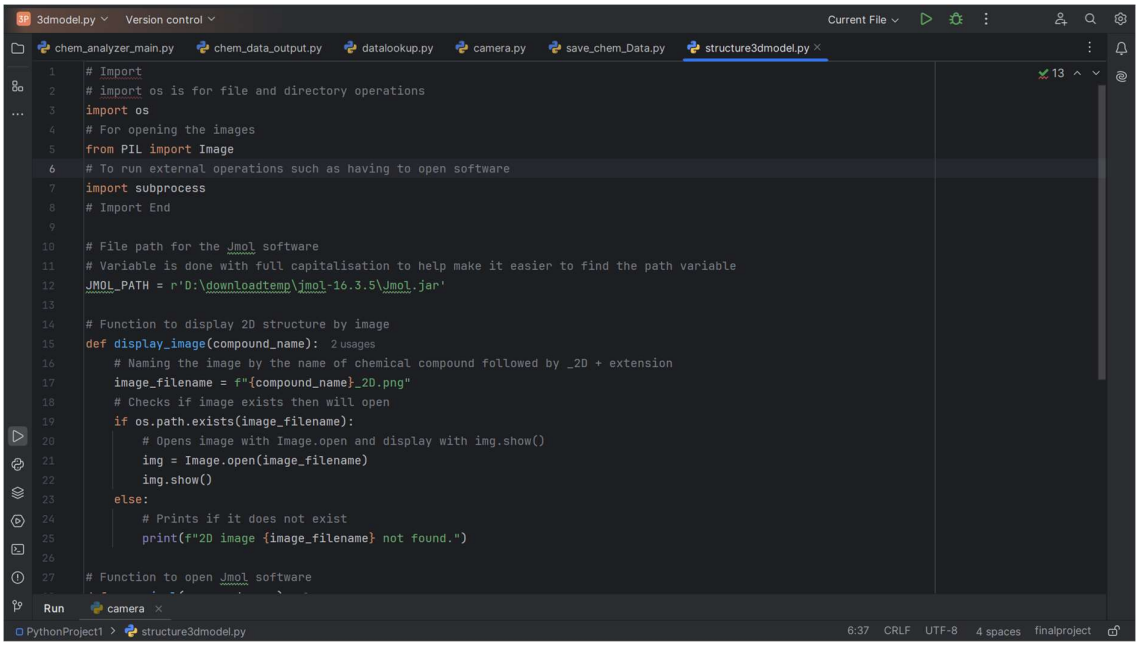
Code Snippet #1 (save_chem_Data.py):



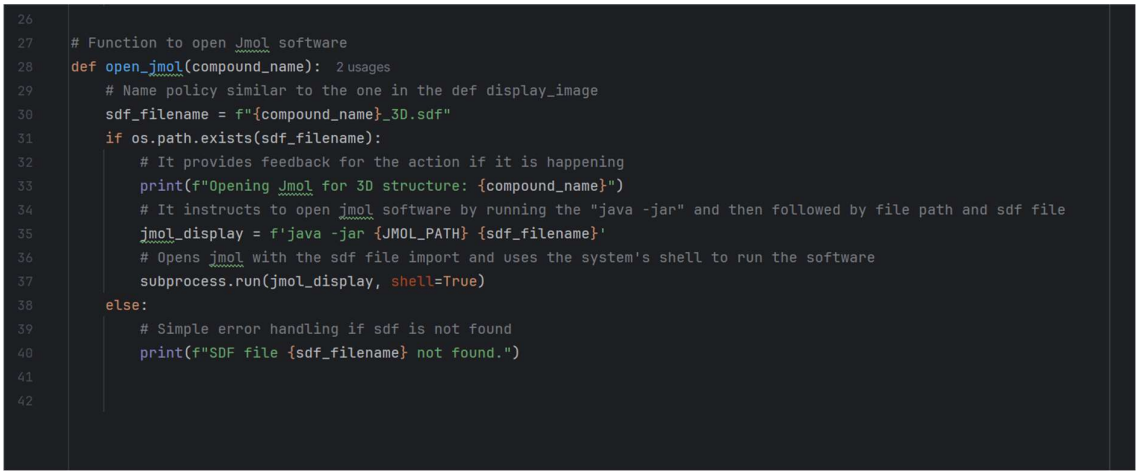
structure3dmodel.py code snippets

Code Snippet #1 (save_chem_Data.py):

`



Code Snippet #2 (save_chem_Data.py):



Video of working program (the camera function is unable to be tested due to Windows access however when using the same code in a newly made .py file it works. This is possibly due to the laptop reset I did): https://youtu.be/Uzx8L3GM9rc?si=i_V6dy6Boz8rRtob

`

Lesson Learned

I learned that, even though Python is a high-programming language, it still has indefinite possibilities its applications can have to the real-world. Even if it is a high-programming language, I had little to no experience utilizing programming languages. In addition, this made me realise that programming is versatile and is applicable in many ways even if you do not have a profession in that department or, in other words, supporting other fields of expertise through computational solutions.

References

Bento, A.P., Hersey, A., Félix, E. *et al.* An open source chemical structure curation pipeline using RDKit. *J Cheminform* **12**, 51 (2020). https://doi.org/10.1186/s13321-020-00456-1

Hähnke, V.D., Kim, S. & Bolton, E.E. PubChem chemical structure standardization. J Cheminform 10, 36 (2018). https://doi.org/10.1186/s13321-018-0293-8

Jan Jensen. (2019, November 5). *Introduction to RDKit Part 1*. YouTube. https://www.youtube.com/watch?v=ERvUf_lNopo&list=PLzfVULc1l7TdDSIfgDm12v21Y8G_B6uh

Bajorath J. (2018). Foundations of data-driven medicinal chemistry. *Future science OA*, *4*(8), FSO320. https://doi.org/10.4155/fsoa-2018-0057

PhD Gil. (2024, May 27). *1. Database (NTP ICE, ECHA biocide, Pubchem, pubchempy)*. YouTube. https://www.youtube.com/watch?v=qofi49V9gcM&list=PL49ip_eZtzYiTrTxVKSbYxoAsU3tJAS0e&index=2

GeeksforGeeks. (2024, November 7). *OpenCV tutorial in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/opencv-python-tutorial/

JayMartMedia. (2024, February 25). *How to use Tesseract OCR in a Python script (pytesseract)* [Video]. YouTube. https://www.youtube.com/watch?v=HNCypVfeTdw

How to Write to a text .txt file in Python! Processing Lists, and Outputting Data! (n.d.). Www.youtube.com. https://www.youtube.com/watch?v=6jsCbjQB3y0

Schafer, C. (2019). Python Requests Tutorial: Request Web Pages, Download Images, POST Data, Read JSON, and More. In *YouTube*. https://www.youtube.com/watch?v=tb8gHvYlCFs

Kenneth Stover. (2019, January 24). *Installing Jmol on Windows*. YouTube. https://www.youtube.com/watch?v=lMecyuIGrJk

GeeksforGeeks. (2024, August 21). Python subprocess module. GeeksforGeeks. https://www.geeksforgeeks.org/python-subprocess-module/