**Analysing and displaying relevant compound information through text/camera input**

**Final Project Report**

A logo for a university

Description automatically generated

**Jeremy Nathanael Gunawan ( 2802522960 )**

**L1BC**

**Jude Joseph Lamug Martinez MCS**

**Algorithm and Programming**

**Binus International University**

**Jakarta**

**Table of Contents**

**Project Specification...........................................................................................3**

**Introduction.........................................................................................................3**

**Inspiration......................................................................................................... 4**

**Solution Design...................................................................................................4 Discussion..........................................................................................................12**

**Limitations.........................................................................................................12**

**Use Case Diagram.............................................................................................13 Activity Diagram...............................................................................................14 Class Diagram.................................................................................................. 15 Modules Used....................................................................................................15 Evidence of Working Program........................................................................18**

**Source Code and Video Demonstration......................................................... 26**

**Lessons Learned................................................................................................27**

**References..........................................................................................................27**

# **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:

A screenshot of a computer

Description automatically generated

**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 :**

**A black background with white text

Description automatically generated**

**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:**

**A screenshot of a computer program

Description automatically generated**

**Figure #4 the dictionary displayed for molecular descriptor output in chem\_data\_output.py:**

**A screen shot of a computer program

Description automatically generated**

**Figure #5 datalookup.py modules and library :**

**A screenshot of a computer

Description automatically generated**

**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 :**

**A computer screen shot of a code

Description automatically generatedA computer screen shot of a program

Description automatically generated**

**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.**

**A screen shot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedFigure #8 and #9 datalookup,py data return:**

**Figure #10 camera.py modules and library:**

**A black screen with white text

Description automatically generated**

**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:

**A screenshot of a computer program

Description automatically generated**

**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:**

**A screen shot of a computer program

Description automatically generated**

**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:**

**A screenshot of a computer program

Description automatically generated**

**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:   
A computer screen shot of a program code

Description automatically generated**

**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:**

**A screen shot of a computer program

Description automatically generated**

**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 :  
A screen shot of a computer code

Description automatically generated**

**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:**

**A computer screen shot of a program

Description automatically generated**

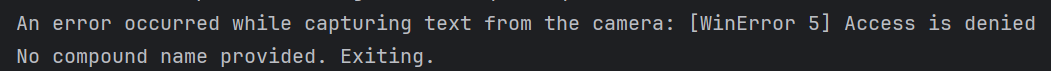
**Discussion**

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 mana.

**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:



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”.

**Use-case Diagram**

A diagram of a diagram

Description automatically generatedA diagram of a diagram

Description automatically generated

**A diagram of a software process

Description automatically generated with medium confidence** Activity Diagram

**A diagram of a process

Description automatically generated**

**A screenshot of a computer

Description automatically generatedClass\_diagram**

**Modules and libraries:**

**Code snippet of the modules imported:**

**Module snippet #1 ( chem\_analyzer\_main.py ):**

**A screenshot of a computer

Description automatically generated**

**Module snippet #2 ( chem\_data\_output.py ):**

**A screenshot of a computer

Description automatically generated**

**Module snippet #3 ( datalookup.py ):**

**A screenshot of a computer

Description automatically generated**

**Module snippet #4 ( camera.py ):**

**A screen shot of a computer

Description automatically generated**

**Module snippet #5 ( save\_chem\_Data.py ):**

**A screenshot of a computer

Description automatically generated**

**Module snippet #6 ( structure3dmodel.py ):**

**A screenshot of a computer

Description automatically generated**

**‘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):**

**A screenshot of a computer program

Description automatically generated**

**Code Snippet #2 (chem\_analyzer\_main.py):**

**A computer screen with text on it

Description automatically generated**

**Code Snippet #3 (chem\_analyzer\_main.py):**

**A screenshot of a computer program

Description automatically generated**

**Code Snippet #4 (chem\_analyzer\_main.py):**

**A screen shot of a computer program

Description automatically generated**

**Code Snippet #5 (chem\_analyzer\_main.py):**

**A screenshot of a computer program

Description automatically generated**

**chem\_data\_output.py code snippets**

**Code Snippet #1 (chem\_data\_output.py):**

**A screenshot of a computer program

Description automatically generated**

**Code Snippet #2 (chem\_data\_output.py):**

**A screenshot of a computer program

Description automatically generated**

**datalookup.py code snippets**

**Code Snippet #1 (datalookup.py):**

**A screenshot of a computer program

Description automatically generated**

**Code Snippet #2 (datalookup.py):**

**A screenshot of a computer

Description automatically generated**

**Code Snippet #3 (datalookup.py):**

**A screenshot of a computer program

Description automatically generated**

**Code Snippet #4 (datalookup.py):**

**A computer screen shot of a program

Description automatically generated**

**camera.py code snippets**

**Code Snippet #1 (camera.py):**

**A screenshot of a computer

Description automatically generated**

**Code Snippet #2 (camera.py):**

**A computer screen shot of a program code

Description automatically generated**

**save\_chem\_Data.py code snippets**

**Code Snippet #1 (save\_chem\_Data.py):**

**A screenshot of a computer program

Description automatically generated**

**structure3dmodel.py code snippets**

**Code Snippet #1 (save\_chem\_Data.py):**

**A screenshot of a computer program

Description automatically generated**

**Code Snippet #2 (save\_chem\_Data.py):**

**A computer code on a black background

Description automatically generated**

**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**](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/>

‌

‌

‌

‌