

CSSD 1102 Group Project: Screening Similar Drug Structures

Project Overview

In this project, your team will implement a software to filter out potential drug candidates with similar structures by making programmatic PUG-REST requests to the online PubChem database. You will use Python and the PubChem database to perform compound searches, data filtering, and reporting, simulating real-world scenarios that chemists and bioinformaticians encounter.

Learning Outcomes

By completing this project, you will:

1. Understand basic cheminformatics concepts, such as compound identification (CID), chemical structure similarity, and molecular connectivity.
2. Develop skills in computational data handling, including querying compound databases, filtering chemical data, and sorting results by frequency.
3. Implement algorithms to detect and remove duplicate chemical structures based on connectivity, refining compound lists for analysis.
4. Apply Lipinski's rule of five to filter compounds, simulating early-stage screening in drug discovery.
5. Collaborate effectively in a team environment to design, code, and test a software program.
6. Improve problem-solving abilities by applying programming techniques to solve chemical challenges.

Project Tasks

- Retrieve the compound ID (CID) for a given substance name from the PubChem database.
- Identify compounds with similar chemical structures.
- Filter out compounds with identical connectivity to remove duplicates.
- Sort the remaining compounds by frequency of occurrences.
- Exclude non-drug-like structures based on Lipinski's rule of five.
- Generate a report with the final results.
- Visualize the top 10 results.
- Manipulate the data (e.g., resort the result) obtained to answer questions.

Teamwork

Similar to the CSSD 1101 group project, this is a collaborative assignment. You may form a new group or continue with your previous team. All teamwork guidelines remain the same as outlined in CSSD 1101. It is essential to make meaningful contributions to the project. Examples of unsatisfactory participation include, but are not limited to:

- Being unresponsive to teammates.
- Failing to actively engage in project activities.
- Not completing assigned tasks to a satisfactory standard and not seeking assistance or informing the team when issues arise.

While collaboration and support among group members are encouraged, free riding is not tolerated, as it can significantly impact the group's overall performance. **If your team reports you for a lack of participation or inadequate contributions, you will receive zero points for the project.** If you have concerns about non-participating group members affecting your team's performance, please reach out to the teaching team promptly for support.

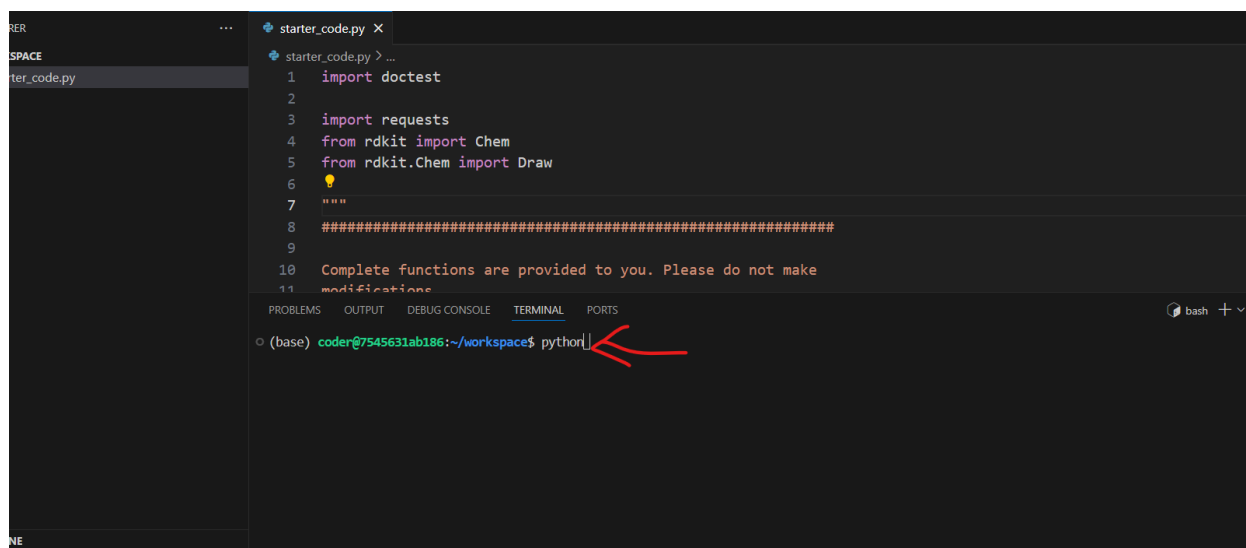
Important: According to Senate policy, no assessments will be accepted after **December 3rd**. This deadline is firm and cannot be extended for any reason.

Implementation Instructions

Set up environment

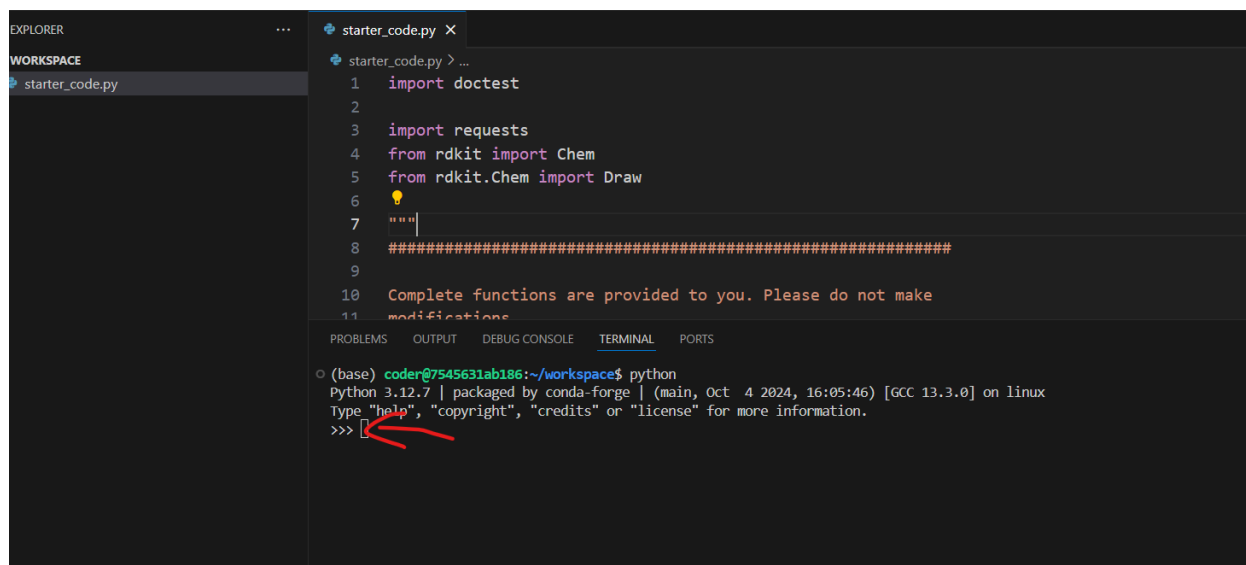
A few external modules are required for this project. You can work on the lab computers (you are supposed to work on the project during the lab time) where all necessary modules have been installed, use the provided workspace on PrairieLearn (You can find the workspace in the question to submit your code), or install those on your own laptop locally. Please note that the workspace on PL may be able to save your file. But if something weird happens in the workspace and you need to reset your workspace, you will lose all your work. So please remember to **keep a local version of your code**.

For the workspace, if you want to enter the interactive mode, just enter python in the window at the bottom and press enter:



The screenshot shows the VS Code interface. The Explorer sidebar on the left shows a file named 'starter_code.py'. The main editor area displays the contents of 'starter_code.py', which includes imports for 'doctest', 'requests', 'rdkit.Chem', and 'rdkit.Chem.Draw', followed by a docstring template. Below the editor, the TERMINAL panel is active, showing a shell prompt '(base) coder@7545631ab186:~/workspace\$ python|' with a red arrow pointing to the prompt.

Then you can put what you want to try after the prompt:



This screenshot shows the same VS Code interface, but the terminal now displays the output of the 'python' command. It shows the Python version (3.12.7) and environment details. A red arrow points to the 'Python 3.12.7' line in the terminal output.

The workspace uses the vscode IDE. You can search online for how to work with this IDE.

Please note that there might be conflict if multiple people working on the same Python script. Each of you can create a Python script and implement your functions there. But when you submit the code, the autograder will only grade code from starter_code.py

If you want to install the modules, you can follow the instructions in previous labs on how to install third-party libraries (e.g., CSSD 1101 lab 4 handout, CSSD 1102 lab 2 handout). Please note that no support will be provided by the teaching team for this option. You can find a list of modules to be installed here:

Module Name	Command
numpy	pip install numpy
pillow	pip install pillow
matplotlib	pip install matplotlib
RDKit	pip install rdkit
sklearn	pip install scikit-learn
seaborn	pip install seaborn
mordred	pip install mordred
PyMol	pip install ipymol
pypdb	pip install pypdb
requests	pip install requests

Note: on a windows machine, you should instead use the command with `./pip.exe` after changing to the correct directory. For example: `./pip.exe install numpy`

ChemPub

Recall that in lab 2 task 7, you wrote a function to generate a URL. With the URL, you can make a request to the ChemPub. In this project, you will need to figure out how to generate the following component of a URL so that you can make a valid query to the ChemPub database:

- Input
- Operation
- Output
- (optional) Options

This is another exercise in how to read documentation. You may find the **official documentation** here:

- <https://pubchem.ncbi.nlm.nih.gov/docs/pug-rest>

Other useful links:

- https://chem.libretexts.org/Courses/Intercollegiate_Courses/Cheminformatics/01%3A_Introduction/1.08%3A_Programmatic_Access_to_the_PubChem_Database
- <https://pubchem.ncbi.nlm.nih.gov/docs/pug-rest-tutorial#section=By-Fast-Synchronous-Structure-Search>

For your convenience, we included a few queries as hints to this project:

- <https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/water/cids/txt>

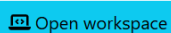
- https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/fastsimilarity_2d/cid/962/cids/txt
- https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/fastidentity/cid/962/cids/json?identity_type=same_connectivity
- <https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/962,910,10003,512452/property/XLogP,CanonicalSMILES/csv>

Note: You can find other properties in the official documentation.

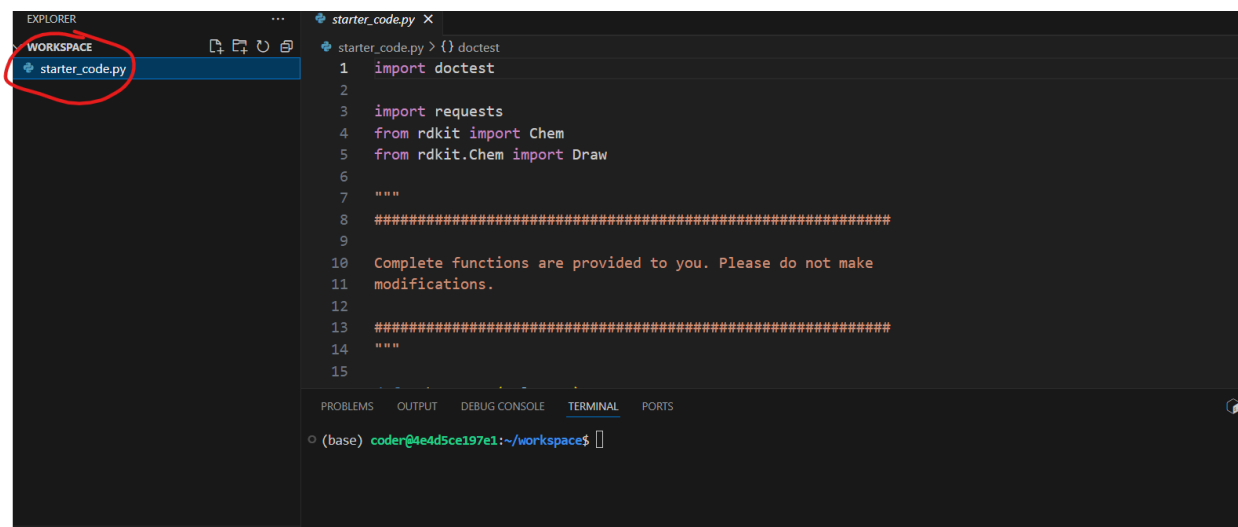
You can make queries for multiple compounds, as shown in one of the examples above. However, if you include too many compounds in a single query, it may result in a timeout. To avoid this, consider breaking your list into smaller chunks and running queries for fewer compounds at a time. Please note that these queries can take some time to complete, so please be patient while you wait for the results.

Starter Code Description

You can find the starter code in the relevant question on PrairieLearn. Click on the Workspace:

 Open workspace

After you open it, click on starter_code.py



Modules

The script imported multiple modules. Doctest is a built-in module and the others are third-party modules that you need to install. You are free to import any other built-in modules, but you are not allowed to include other third-party modules that are not installed in the workspace. You are also free to use any built-in functions.

Completed Functions

We provided two functions for you. Please read the function design recipe to learn what they do.

- `make_query`
 - You may get 503 errors. Please read the documentation what it represents. Usually if you just rerun your code, it will be fine. It may happen when there are too many queries and the server has too many traffic and dropped your request. Because of this, **DO NOT TRY TO COMPLETE THE WORK AT THE LAST MINUTE as extensions CANNOT be granted for this project.**
- `visualize_multiple_smiles`

Helper Functions

- You are free to add as many helper functions as you need. Please remember to include them if they are in the functions to be autograded.
- Please include them in the designated places in the script. If you did not modify the `starter_code`, you can add them under line 54.

Functions to be implemented

You need to implement the following functions. Please read the function design recipe for what they are.

- `generate_url`
- `name_to_cid`
- `find_similar_structures`
- `remove_query_cid`
- `sort_by_frequency`
- `get_chunks`

- `remove_non_drug`
- `cid_to_smiles`
- `sort_and_retrieve_property`

Run the script

After you implement the above functions, run the script. Please note that `mol_name` is `water` in this example, so the results you will see are the results for water. You will need to replace `water` with another compound if you want to see the result of another compound. Please note that all the examples in the doctest provided are also the results of `water`. If you see discrepancies, you can check your result and the result of the provided examples.

If things are working properly, you will see the following output (if your sorting results in a different order, then you are likely to see different results):

The CID of the compound `water` is `962`

There are `63` similar structures

After removing compounds with identical connectivity, `46` compounds left

After sorting by frequency, the top `10` compounds are

`57422081`

`59102850`

`58422532`

`171528`

`23560202`

`78060557`

`44144404`

`158956564`

`72499223`

`22247451`

After removing non-drug-like structures, `13` compounds left.

The top 10 compounds are

58422532

171528

137349153

58750500

157350

57634601

961

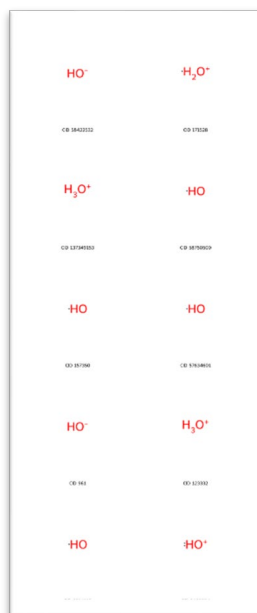
123332

6914119

5460554

You will see a csv file generated in where you save this script: water.csv . You can compare your results with the csv file that we provided.

You will also see a picture generated. You can click on the generated file, e.g., water.png and you will see this:



Rubrics

You are allowed to reuse your code in previous labs. However, you are NOT allowed to copy code for labs from students outside of your team.

CLO4-Participation (1 pt in total):

- Submit the csv reports for the following compounds:
 - Nicotine
 - Enzalutamide
 - Alpelisib
 - Crizotinib
 - Axitinib
 - Vemurafenib
 - Vandetanib
 - Dacomitinib
 - Avapritinib
 - Selumetinib
 - Pexidartinib
- Submit the Visualization of the top 10 compounds of each drug (sorted by weight).

CLO5 evaluation (5 pts in total)

- Function implementation (3 pts total)
 - generate_url (0.1 pt)
 - name_to_cid (0.2 pt)
 - find_similar_structures (0.25 pt)
 - remove_query_cid (0.5 pt)
 - sort_by_frequency (0.5 pt)
 - get_chunks (0.25 pt)
 - remove_non_drug (1 pt)
 - cid_to_smiles (0.2 pt)
- Answer questions (2 pts in total)
 1. (1 pts) Of the final 10 compounds that are sorted based on molecular weight/mass (sorting in the descending order, which is from the heaviest to the lightest), which compound is the most polar (refer to xLog P slide in lecture) for each compound. If there are ties, please provide all of them.
 2. (0.5 pts) Based on your answer in the previous question, determine (for all drugs) the types of non-carbon and non-hydrogen heteroatoms in these SMILES, like, nitrogen, oxygen, fluorine, chlorine, bromine, iodine, phosphorous, and sulfur.

3. (0.5 pts) Provide the URL to determine the CIDs of compounds with an exact mass of 4112.1187318 g/mol in JSON format.

CLO7 evaluation (1 pt in total)

- Teamwork

Submission

Please submit all documents to PrairieLearn:

https://ca.prairielearn.com/pl/course_instance/9337/assessment/62117

Please follow the instructions on PrairieLearn to submit your files (e.g., what to submit, file name, file type)

For the coding questions, after you finish working on the question. Please go back to the PL and click on “Save & Grade”. It will grade the `starter_code.py` file in your workspace and they will be graded with the autograder. It will also upload all the generated csv files and all the png files for CLO4-Participation. We will manually grade them later.

For short answer questions and peer evaluation, please remember to click “Save” for the platform to record your submissions.