Nathan Parker

toichify

Your go-to Stoichiometric solver.

A Python-powered logical system and GUI application designed to solve stoichiometric equations, by breaking down each step as you would write them on a piece of paper. Thus building the needed knowledge — all for free. Stoichify is open-source and aimed at chemistry students and teachers to learn and better their experience with the daunting complexity of stoichiometry.

GitHub: https://github.com/KingPr0o7/Stoichify

USER MANUAL

Built and Designed by **Nathan Parker**

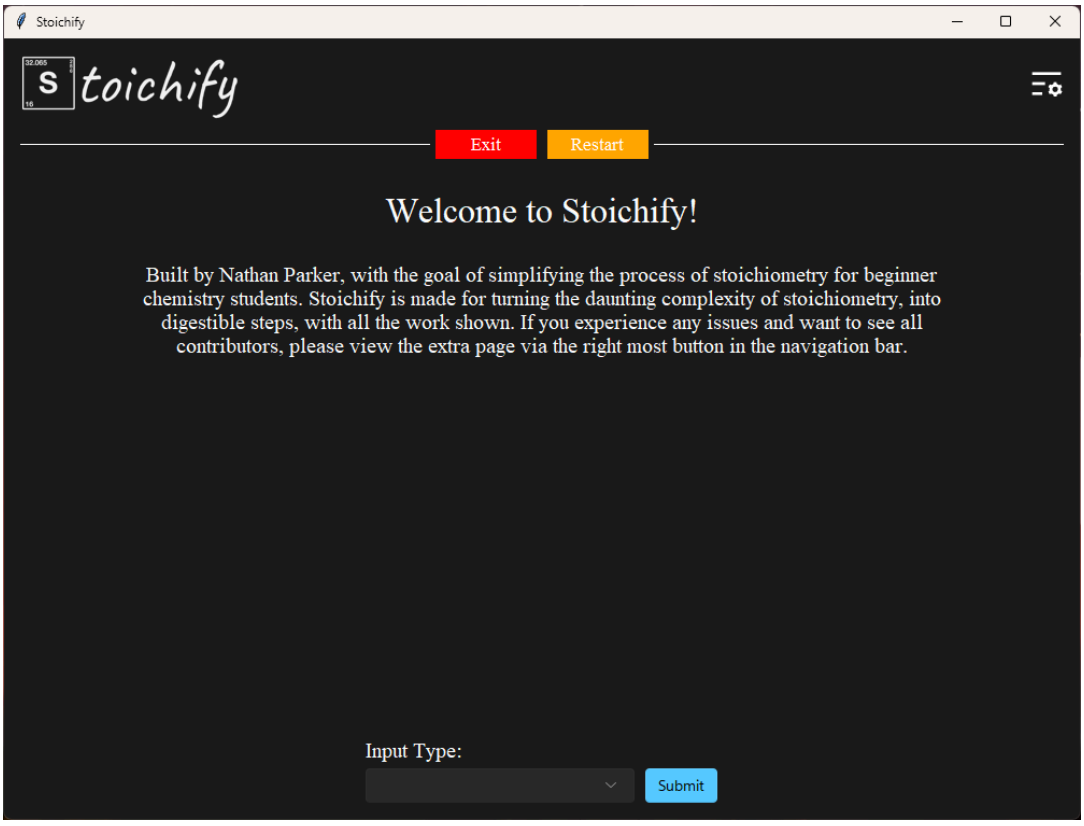
… with help from other contributors

Table of Contents

How to Use Stoichify

The main goal of this project was to make things as simple as possible. Thus, all aspects of the GUI and most methods if called-upon in a CLI-type of way have standard explanations (doc-strings, comments, and descriptions) to show exactly what's needed and being done. However, this user manual will only show how to use the GUI that I have built[1].

That being said, after you install Stoichify and have it running on your desktop, you'll be greeted with a window that looks like **Figure 1**:



(Figure 1)

[1] Might change if more community interest is accumulated.

Stage 1 (Choosing Type — Substance / Equation)

With Stage 1, you're presented with 5 intractable elements, which are:

1. Extra Page Button (Navigation Bar)
2. Exit Button (Navigation Bar)
3. Restart Button (Navigation Bar)
4. Input Selection Box (Input Panel)
5. Input Submit Button (Input Panel)

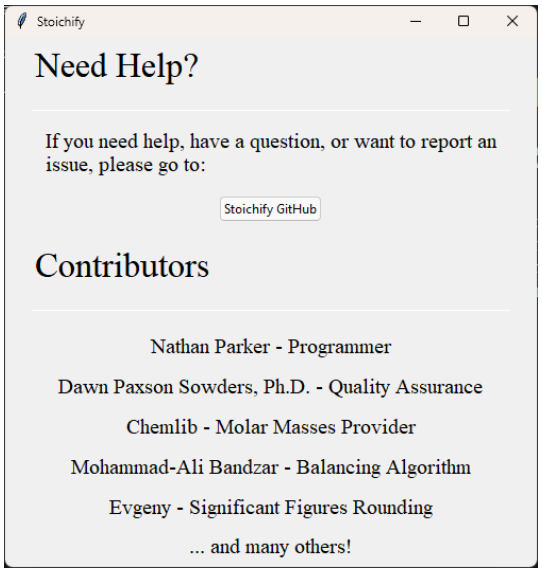
Below is **Figure 2** which highlights the above listed elements in their exact positions:

(Figure 2)

Any element located in the *Navigation Bar* will never **disappear** on you. The *Navigation Bar* is distinguished by lines next to the *Exit Button* and *Restart Button*. Anything after those lines are considered *Content Elements* and will change on you after each stage (called upon clicking the *Submit Button*). Unfortunately, I had no time to account for tabbing to elements, which will be added *eventually*.

Starting off with the two *Navigation buttons*, they consist of the *Exit Button* and *Restart Button*. In which, the *Exit Button* will simply close Stoichify, leading you back to your preferred coding editor[2]. With the *Restart Button*, it'll take you back to *Stage 1*, regardless of what stage you're currently at. It'll **never close or kill variable instances**.

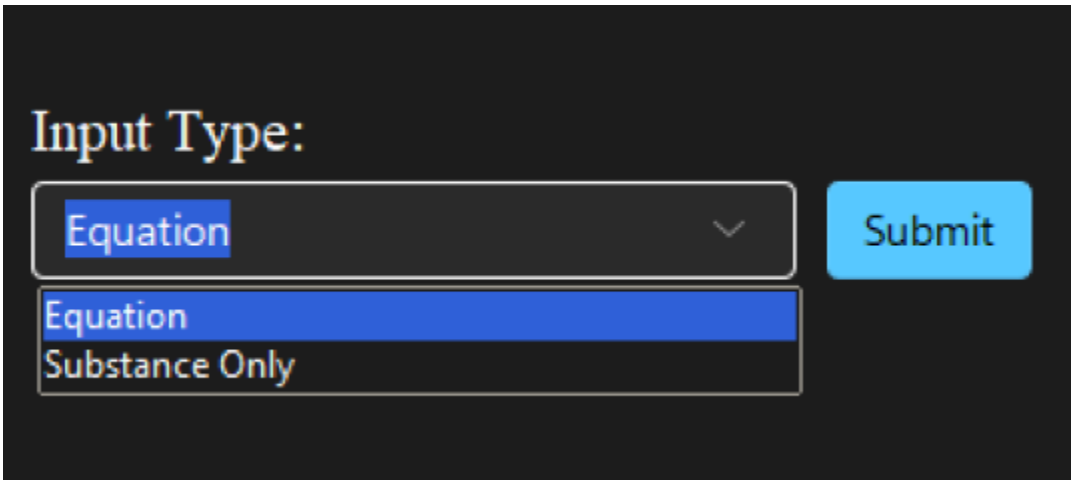
The '*Extra' Page Button* opens another window with a link to the Stoichify GitHub (in case of any issues) and a list of some top contributors, shown in **Figure 3**:



(Figure 3)

[2] May change if Stoichify is bundled differently.

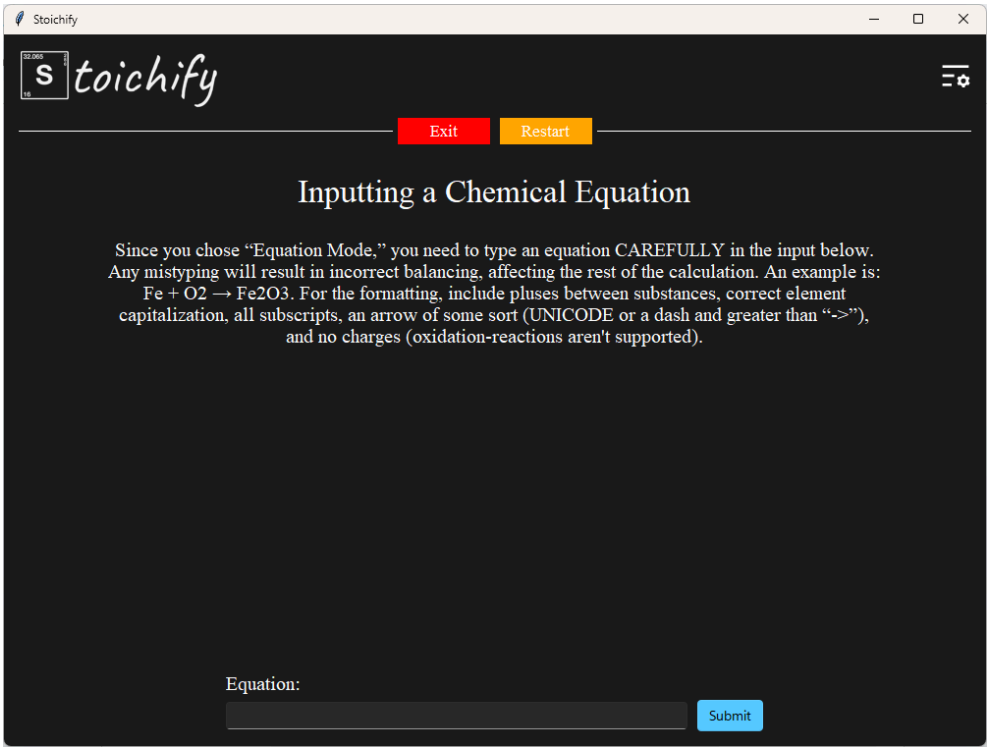
Getting onto the *Input Panel*, this is where you'll be interacting with the most. All required inputs are always located at the bottom of the screen. They change based on stage and this input type presented on *Stage 1*. There are two types of ways that Stoichify can handle, being *Equations* or a *Substance*. If you decide to choose an *Equation*, you'll be required to put in an equation in the desired format, whereas if you chose *Substance Only*, you'll need a substance to move on. This choice is presented in the dropdown (***Figure 4***) and upon submitting you'll be moved into *Stage 2* for that type:



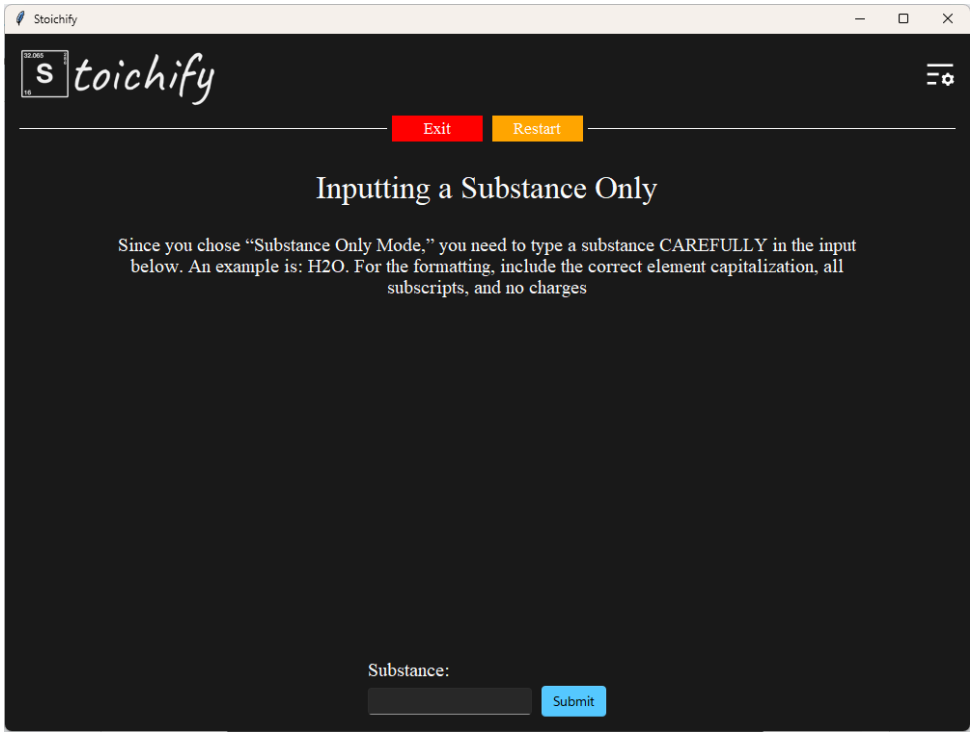
(Figure 4)

Stage 2 (Inputting Substance/Equation)

After selecting one of the types, either *Equation* (***Figure 5***) or *Substance Only* (***Figure 6***), you'll move on to *Stage 2*:



(Figure 5 — Equation Type)



(Figure 6 — Substance Type)

Follow the instructions and input into the *Input Entries*. Stoichify doesn't support charges which lead to oxidation-reactions that have a different balancing method. *Equations* allow for pasting of:

- Any arrow type from UNICODE in case your assignment/website uses them (99% likely).
- Any subscript included
- Any substance state (s, l, g, aq)
- Substances surrounded by: [] (complex; ignored)

Which covers most bases; however, if you're typing raw into Stoichify you can type regular numbers for subscripts and use the arrow "->" (dash and greater than sign) to represent the yield arrow (it'll be replaced). The same general rules **also apply to *Substances*.** Just then you don't have to type arrows, pluses, reactants, or products as you're only dealing with

one substance. If you mess up, errors will appear, allowing you to fix your mistakes[3]. Here are examples of equations and substances that'll pass the checks:

- SO2 + O2 -> SO3
- Fe + O2 -> Fe2O3
- $Fe_2 + O_2 \rightarrow Fe_2O_3$
- $F_2(g)$
- F2
- $H_2(g)$ + O(g) -> H2O(l)
- K4[Fe(SCN)6] + K2Cr2O7 + H2SO4 → Fe2(SO4)3 + Cr2(SO4)3 + CO2 + H2O + K2SO4 + KNO3
- Co2
- CO3

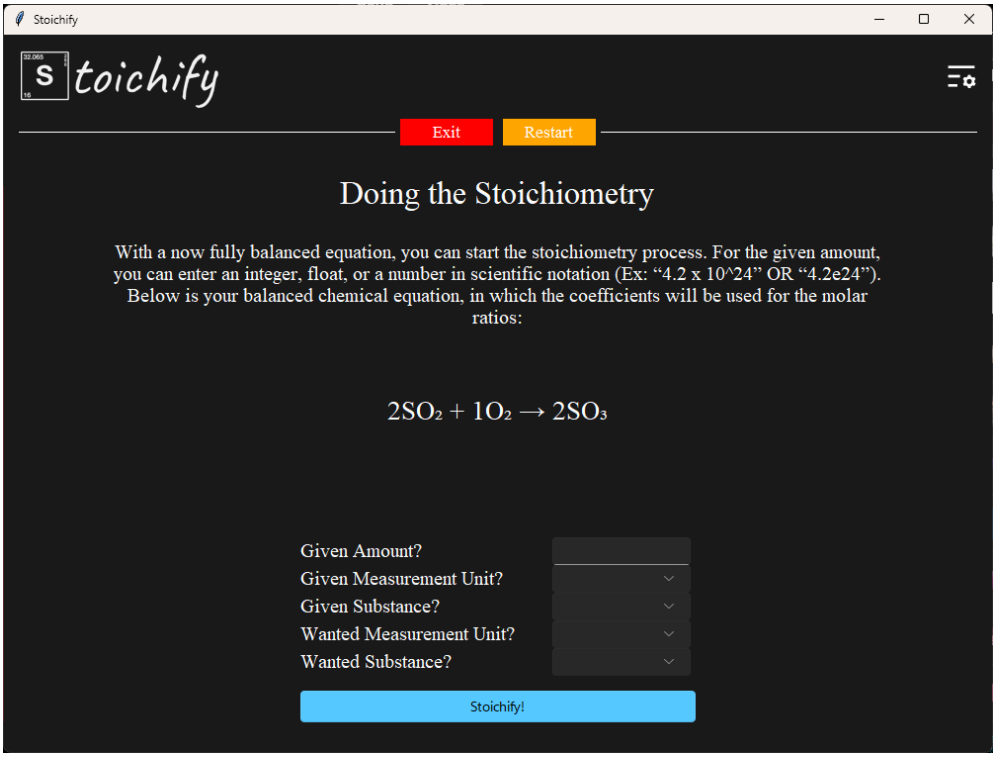
And more. Stoichify's type checking is quite strict, and should[4] catch most errors (if you do encounter an error, and can't fix it, report it to GitHub). After submission of your *Equation* or *Substance*, you'll move on to a unified *Stage 3*.

[3] Stoichify isn't perfect, and might not catch *every* error.

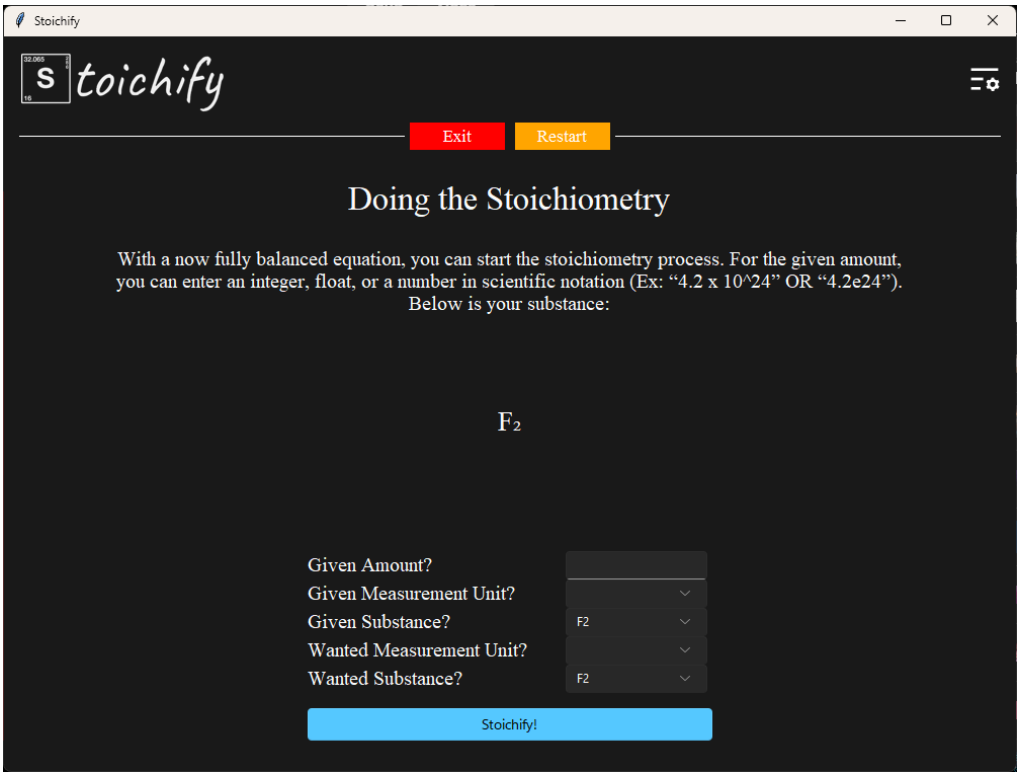
[4] Stoichify isn't perfect, and might not catch *every* error.

Stage 3 (Stoichiometric Calculations)

Now with the *Equation* or *Substance* is saved, Stoichify will automatically format and balance equations (***Figure 7***), or show your formatted substance. If you choose a substance, the fields of the *Given Substance?* and *Wanted Substance?* will already be filled out (***Figure 8***).



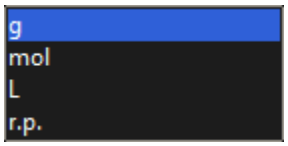
(Figure 7)



(Figure 8)

The first input (*Given Amount?*), an *Entry* requires you to know the given amount, which can either be a float, an integer, or a number in scientific notation (4.2 x 10^24, 4.2 * 10^24, 4.2e24, 4.2e-24, etc.). This entry, upon submission, will count your significant figures for you (even in scientific notation (if parsed correctly)). *Unfortunately*, if you adapt Stoichify into a CLI type application, it requires you to know the given significant figures… As Python and other languages strip zeros, so you’ll have to find a way to keep them via keystrokes (like tkinter) and build a string representation.

For the next inputs, they'll always be dropdowns… Which consist of the *given measurement unit, given substance, wanted measurement unit,* and *wanted substance*. Both measurement unit dropdown consists of (***Figure 9***):



(Figure 9)

With g (grams), mol (moles), L (liters), and r.p. (representative particles, like atoms, molecules, etc.) In terms of *r.p.* it's a general representation of particles.

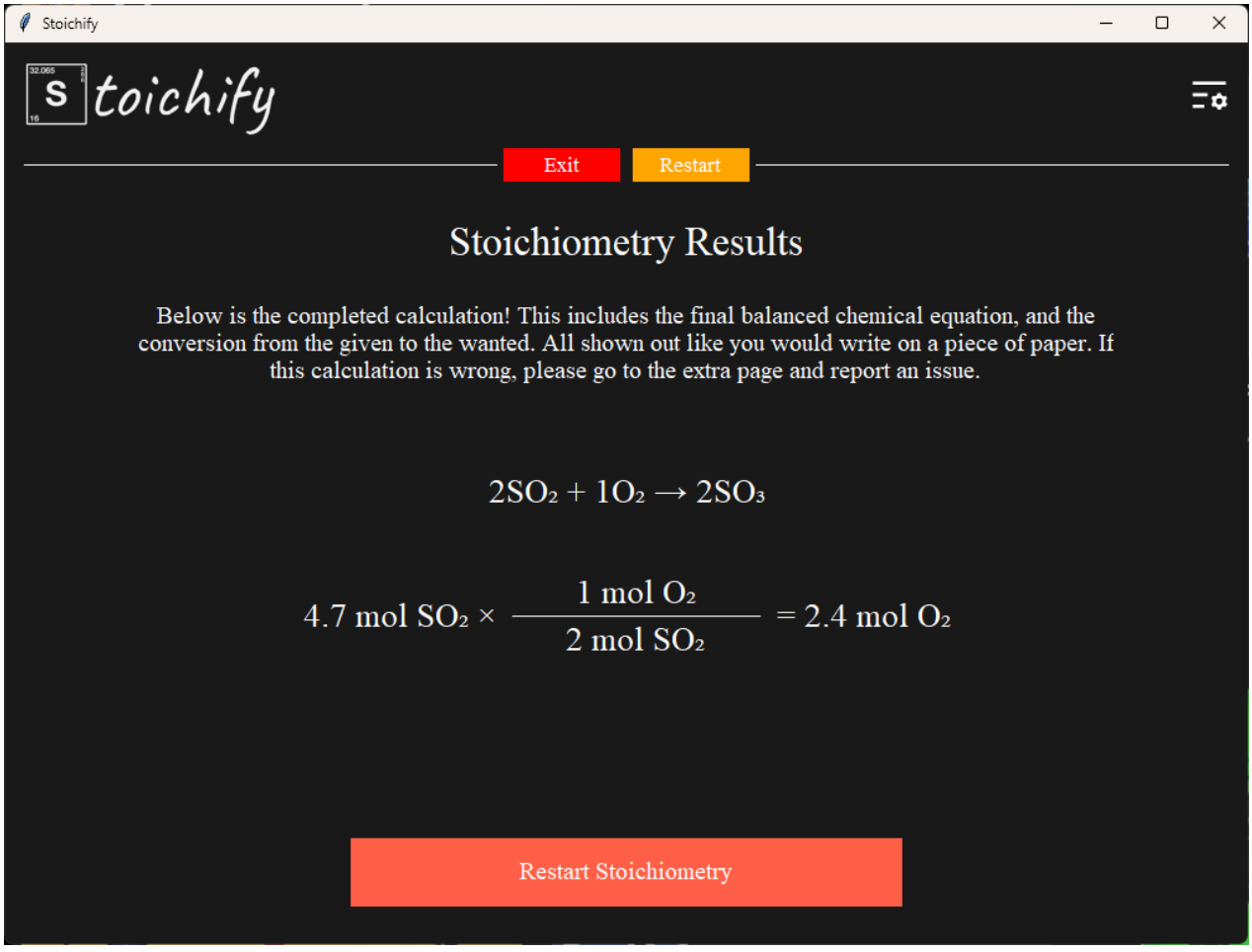
For the *Given Substance?* and *Wanted Substance?* In substance mode, as already mentioned, will already be filled out (as there are no other substances). However, in an equation there are multiple substances, in which will be presented for you to choose from (don't worry, their coefficients are saved) and, as an example, would look like (***Figure 10***):

(Figure 10)

After all values are inputted (with none blank (it'll throw an error)) you can click on the submit button (appropriately named "Stoichify!") you'll move on to *Stage 4*, the final stage with the results!

Stage 4 (Results + Loop Back)

Upon clicking the “Stoichify!” button from *Stage 3*, the GUI (*main.py*) will call the **Stoichify** class (stoichiometry.py) which will water fall calls of other classes to calculate the results. During this process, it’ll build a list of values, that show the program saving its work along the way. Upon completion of this, you’ll be brought to this screen (***Figure 11***):



(Figure 11)

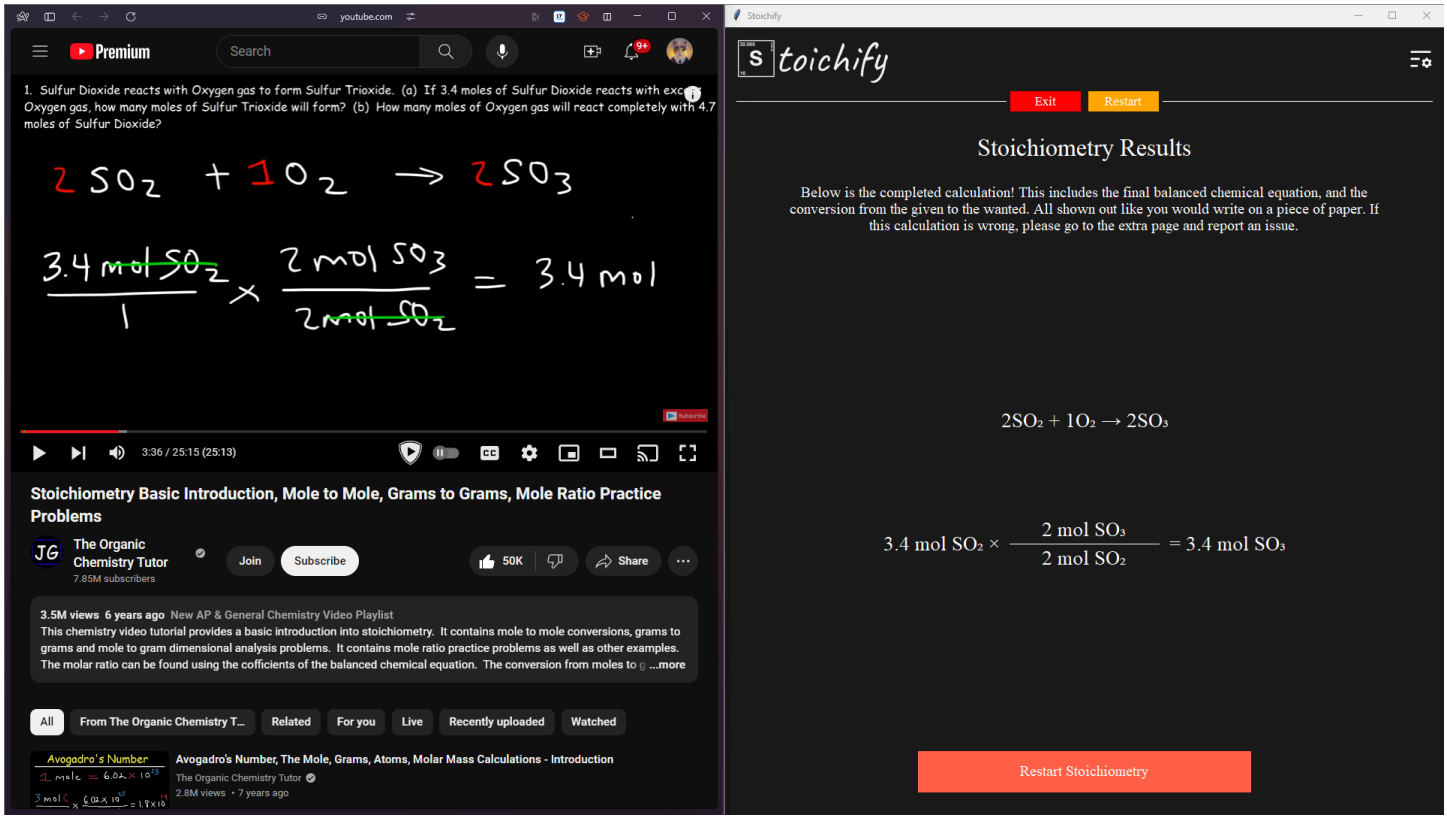
Which will show your final balanced equation / substance and the work shown as a student/teacher would write on a piece of paper! After you're satisfied with the results, you can click the "Restart Stoichiometry" button to go back to the saved balanced equation / substance and do many other stoichiometric calculations. Saving typing time and bringing a quicker experience!

Testing Stoichify

Stoichify was heavily tested during the development and somewhat after development. Comments in source code were some tests and hurdles I had to jump over… However, I made the critical mistake of starting the *testing.py* file, until after development… Regardless, there are **60** tests against Stoichify ensuring type security and consistency with answers across the board. Such test data was gathered from the following sources:

- https://www.youtube.com/watch?v=7Cfq0ilw7ps
- https://www.mcmsnj.net/cms/lib07/NJ01911694/Centricity/Domain/540/Extra%20Practice%20mole%20mass%20rp%20practice%20problems%20answers.pdf
- https://www.youtube.com/watch?v=74-X94OP2XI&t=596s
- https://www.youtube.com/watch?v=XnfATaoubzA
- https://www.youtube.com/watch?v=VJWTnwI1V0g
- … and many more!

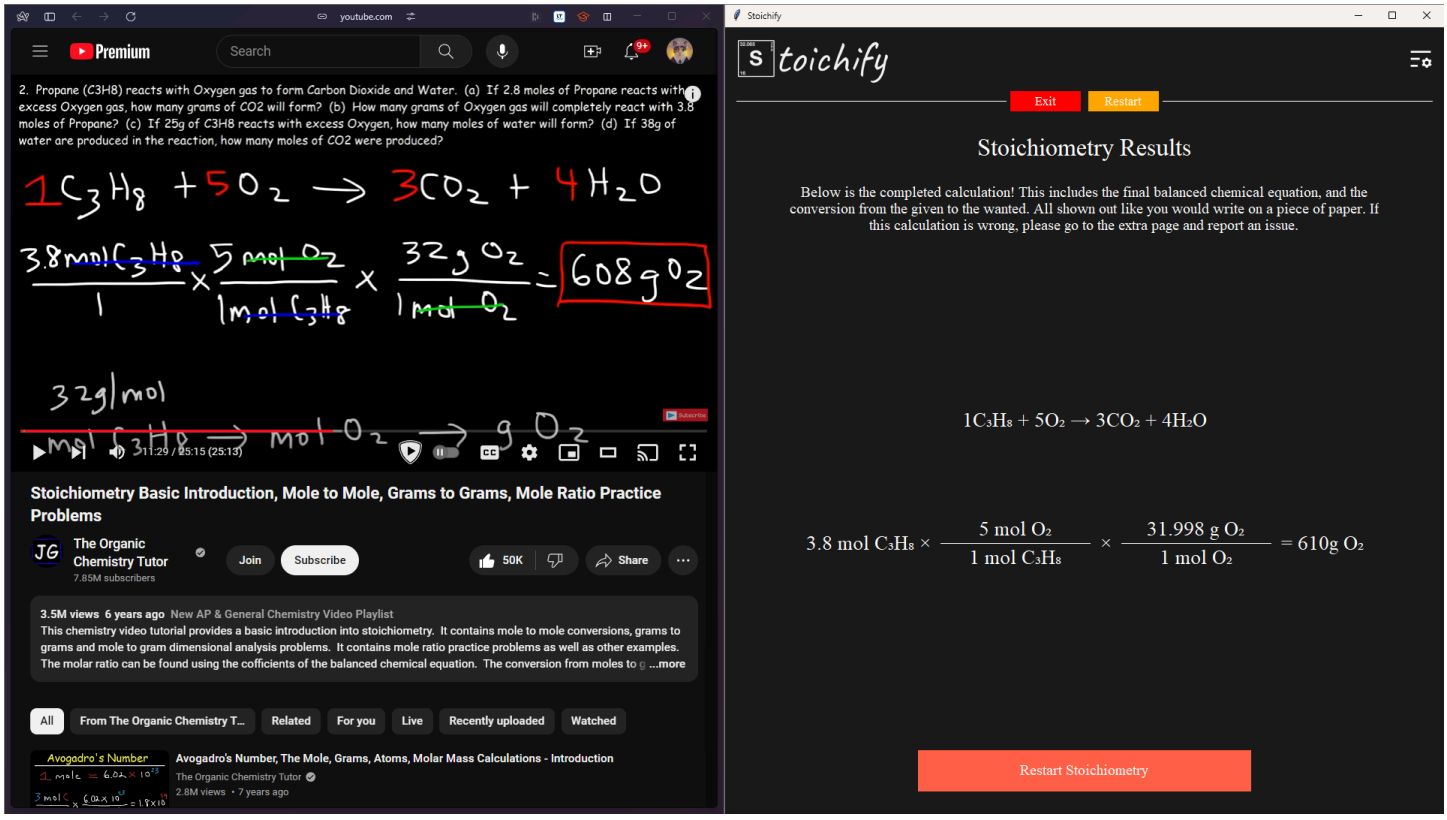
<u>However</u>, it's important to note that some of these sources may not *consider some variables*. Like significant figures (might be an introductory video), have the same molar mass (might be using an out-of-date periodic table), and different rounding. Stoichify uses a newer periodic table and has strict significant figure counting and rounding (rounds up).
Here are some examples of Stoichify in action:



(Figure 12)

SO2 + O2 -> SO3
3.4 mol SO2 -> ? mol SO3 = **3.4**

(taking given * molar ratio of given/wanted coefficients)

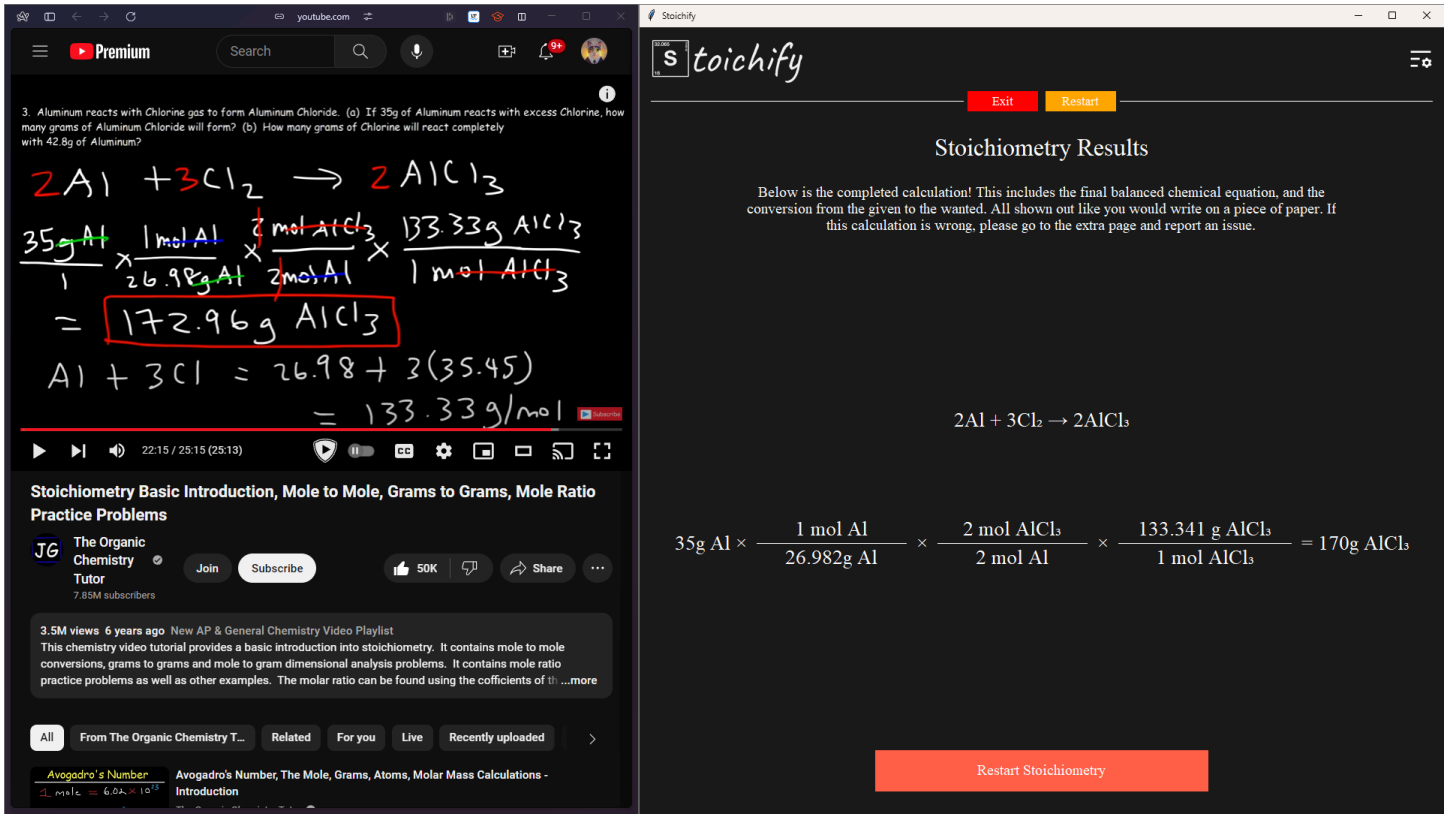


(Figure 13)

C3H8 + O2 -> CO2 + H2O
3.8 mol C3H8 -> ?g O2 = **610**

(Take the given * molar ratio of given/wanted coefficients * a conversion to wanted measurement unit)

(The Organic Chemistry Teacher didn't account for Significant Figures, and the defined user measurement is two significant figures. So, Stoichify rounded up.)

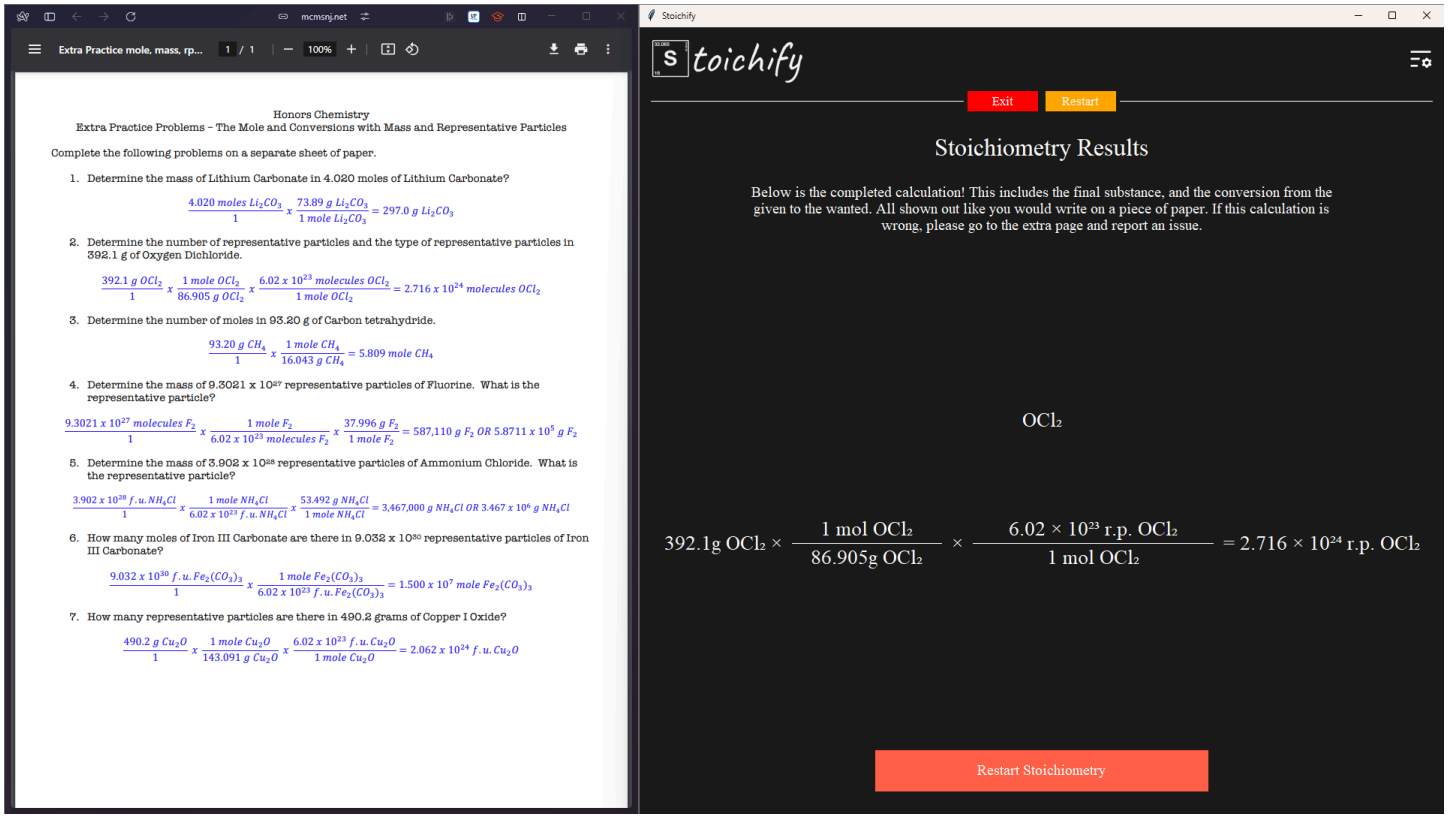


(Figure 14)

Al + Cl2 -> AlCl3
35g Al -> ?g AlCl3 = **170**

(Take the given measurement unit, convert it to moles, * by molar ratio of given/wanted coefficients, and convert back to grams)

(Same thing, Stoichify noticed 2 significant figures, and rounded to it)

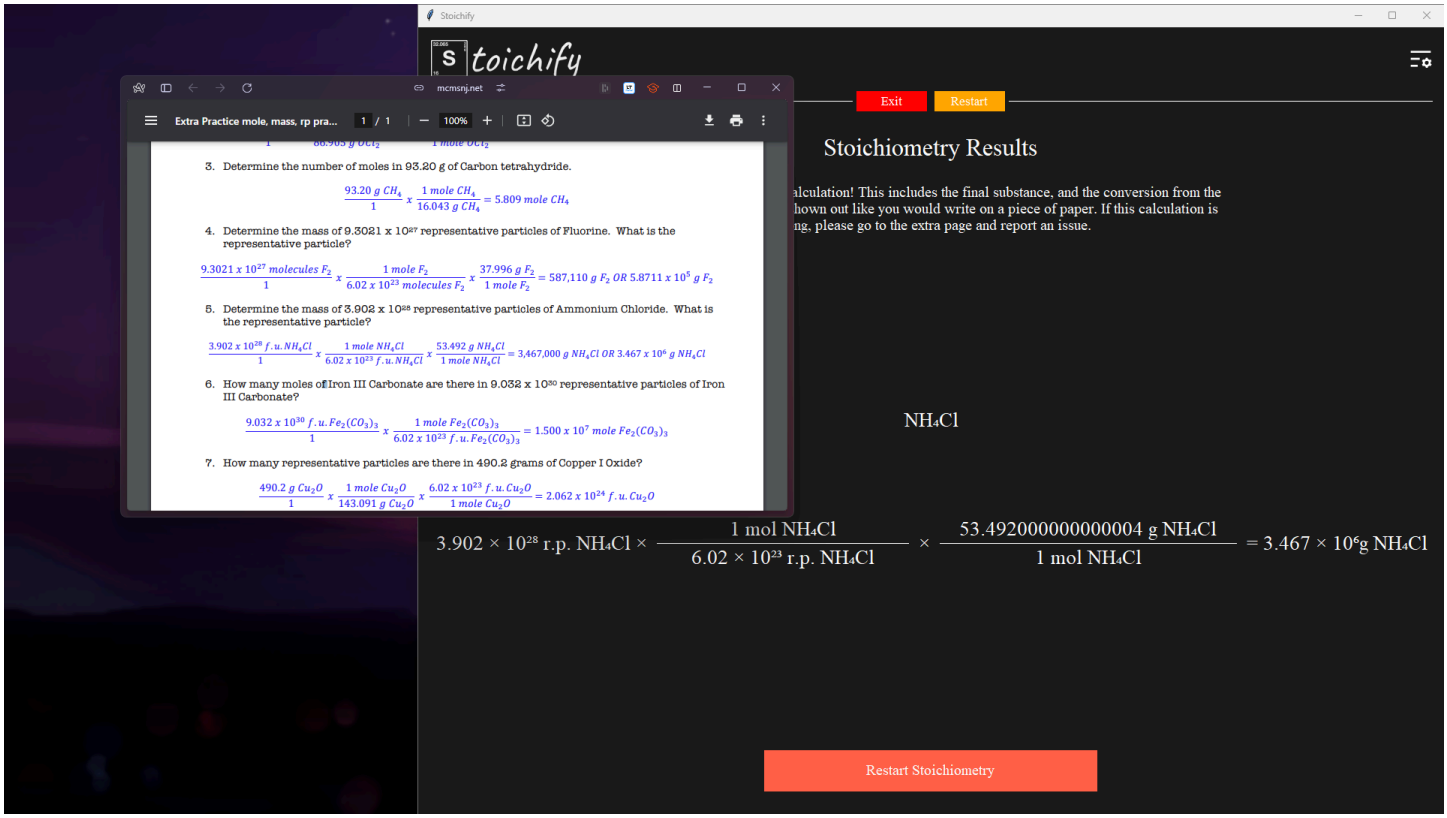


(Figure 15)

OCl2

392.1g OCl2 -> ? r.p. OCl2 = **2.716 x 10^24**

(Take given into moles, then to r.p. (molar ratios don't exist in just substances))



(Figure 16)

NH4Cl

3.902 x 10^28 r.p. NH4Cl -> ?g NH4Cl = **3.467 x 10^6**

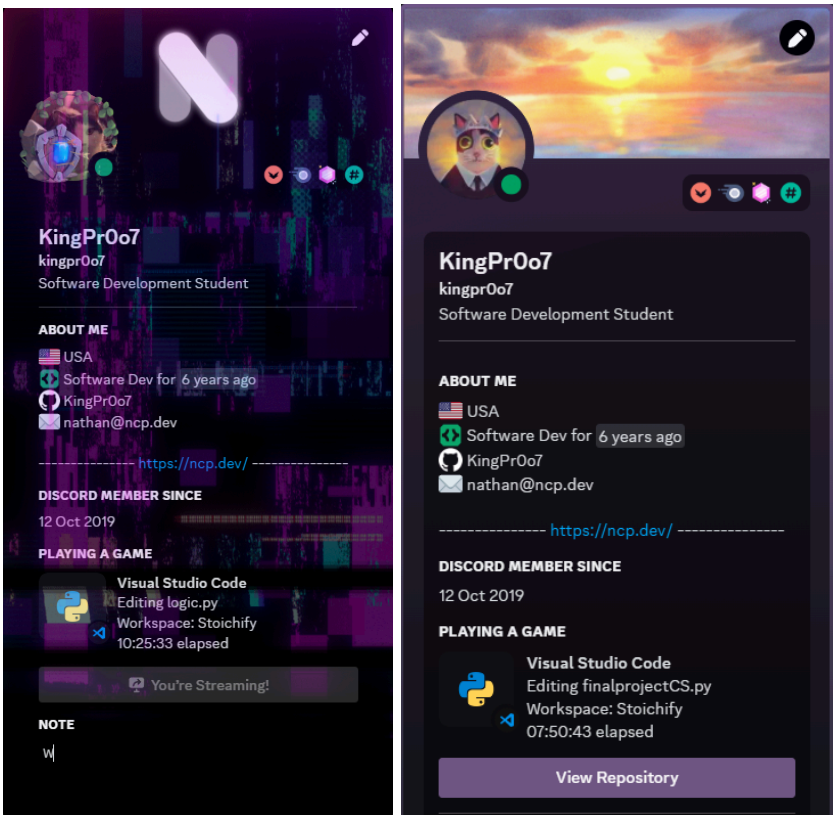
(Go from r.p. to moles, then to grams)

… and many more! Try out Stoichify for yourself to see the results! Please! If you find any errors, want to suggest features, and more, please go to the GitHub: https://github.com/KingPr0o7/Stoichify

The work shown by these people brought up many errors that I had to fix along the way. For example, scientific numbers were a real pain. Because I also had to find the significant figures in them as well. Along with parsing unique ways to format them, and even the float representation of them.

Other things I had to fix were correct balancing, formatting, GUI bugs, and many more. However, the biggest pain, by far, was parsing significant figures. I had gone to many people with my frustrations of Python cutting off my precious significant figures (it haunts me still to this day) until I realized that tkinter saved me, by building a string representation of all characters entered.

All these contributed to a lot of time debugging, asking for help, and hours and hours of Googling my way through issues. But nonetheless, I pushed forward, through the hours (sometimes 10+ hours at a time) (***Figure 16-7***) to create something for the public to enjoy. I want to clarify it, that this project is open to the public, and anyone can report issues to better Stoichify.



(Figures 16 + 17)

Contributors

Even though this project was meant to be a single-person project for Ivy Tech, Stoichify cannot be what it is today without the help from these people below, please give them the credit they deserve. They truly made Stoichify one-of-a-kind.

Dawn Paxson Sowders, Ph.D. — Quality Assurance
Chemlib Creators — Molar Masses Provider
Mohammad-Ali Bandzar — Balancing Algorithm
Evgeny — Significant Figures Rounding (basis)
Benedek Dévényi (rdbende) et al. — Tkinter Theme (sv_ttk)
Jeffrey A. Clark et al. — Pillow (PIL) Image Library
SymPy Development Team — Symbolic Mathematics

Thank you!

This project is in development and **will** be maintained by Nathan Parker (KingPr0o7). I'm a software development student in Indiana, trying to pursue my dreams of making software for the public to enjoy. Follow me on X to get updates about myself.