Machine-actionable data & GUIs

Practical Programming in Chemistry

Prof. Philippe Schwaller

This lecture

- Motivation for this course
- Machine-readable data
- Interactive applications

The Age of Digital Chemistry

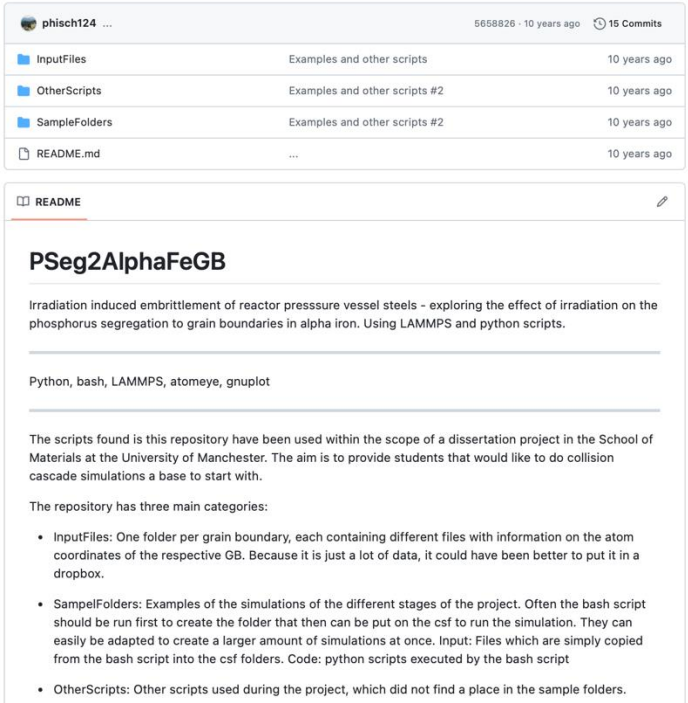
- This course **prepares you for the future**.
- **The earlier** you get in touch with practical programming in chemistry **the better** (you might not realise, yet), but even if you don't program yourself, you will work with people who do.
- Even if you do a **purely experimental degree**, programming will help you do more **structured data analysis**. A handful of experiments are also chemical data.
- Github, code, programming – all scary if you have never done it. But once you get into it (with a proper local setup), it's not that difficult to learn more.
- With what you've seen and done so far, you are more **advanced than 90% of graduated chemistry students**.

The Age of Digital Chemistry

- **“*It is not that machines are going to replace chemists*,”** said Derek Lowe, a longtime drug discovery researcher and the author of In the Pipeline, a widely read blog dedicated to drug discovery. **“*It's that the chemists who use machines will replace those that don’t*.”**
- This practical programming course gives you the **basic tools and knowledge to be able to use machines and computers** efficiently.

At the end of this course, some of you (hopefully all) will be much better at programming than I was in my Bachelor.

Code from my Bachelor (→ it looks terrible)



phisch124 ... 5658826 · 10 years ago 15 Commits

InputFiles	Examples and other scripts	10 years ago
OtherScripts	Examples and other scripts #2	10 years ago
SampleFolders	Examples and other scripts #2	10 years ago
README.md	...	10 years ago

README

PSeg2AlphaFeGB

Irradiation induced embrittlement of reactor presssure vessel steels - exploring the effect of irradiation on the phosphorus segregation to grain boundaries in alpha iron. Using LAMMPS and python scripts.

Python, bash, LAMMPS, atomeye, gnuplot

The scripts found is this repository have been used within the scope of a dissertation project in the School of Materials at the University of Manchester. The aim is to provide students that would like to do collision cascade simulations a base to start with.

The repository has three main categories:

- InputFiles: One folder per grain boundary, each containing different files with information on the atom coordinates of the respective GB. Because it is just a lot of data, it could have been better to put it in a dropbox.
- SampelFolders: Examples of the simulations of the different stages of the project. Often the bash script should be run first to create the folder that then can be put on the csf to run the simulation. They can easily be adapted to create a larger amount of simulations at once. Input: Files which are simply copied from the bash script into the csf folders. Code: python scripts executed by the bash script
- OtherScripts: Other scripts used during the project, which did not find a place in the sample folders.

Irradiation induced embrittlement of reactor presssure vessel steels - exploring the effect of irradiation on the phosphorus segregation to grain boundaries in alpha iron. Using LAMMPS and python scripts.

Readme
Activity
7 stars
3 watching
4 forks
Report repository

Releases

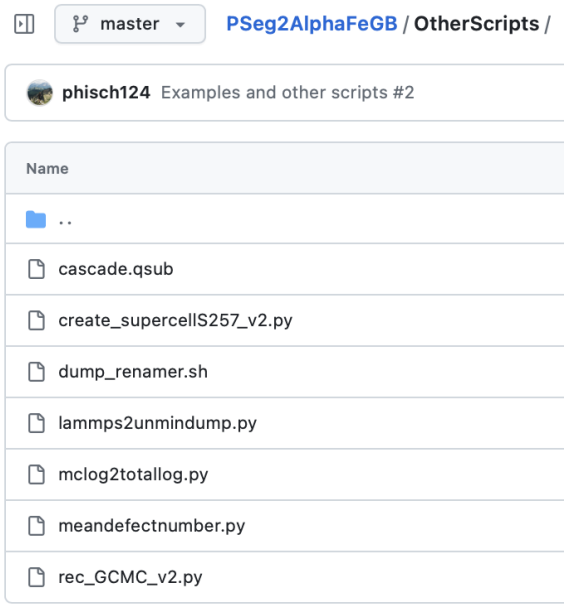
No releases published

Packages

No packages published

Languages

F# 99.0% Other 1.0%

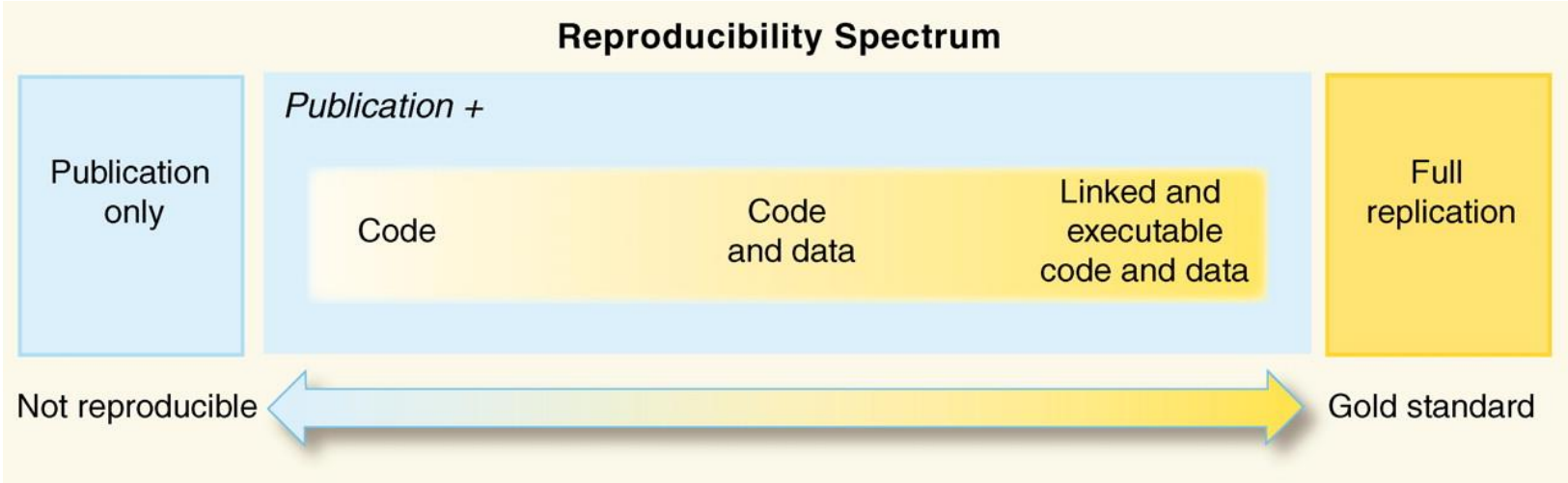


master PSeg2AlphaFeGB / OtherScripts /

phisch124 Examples and other scripts #2

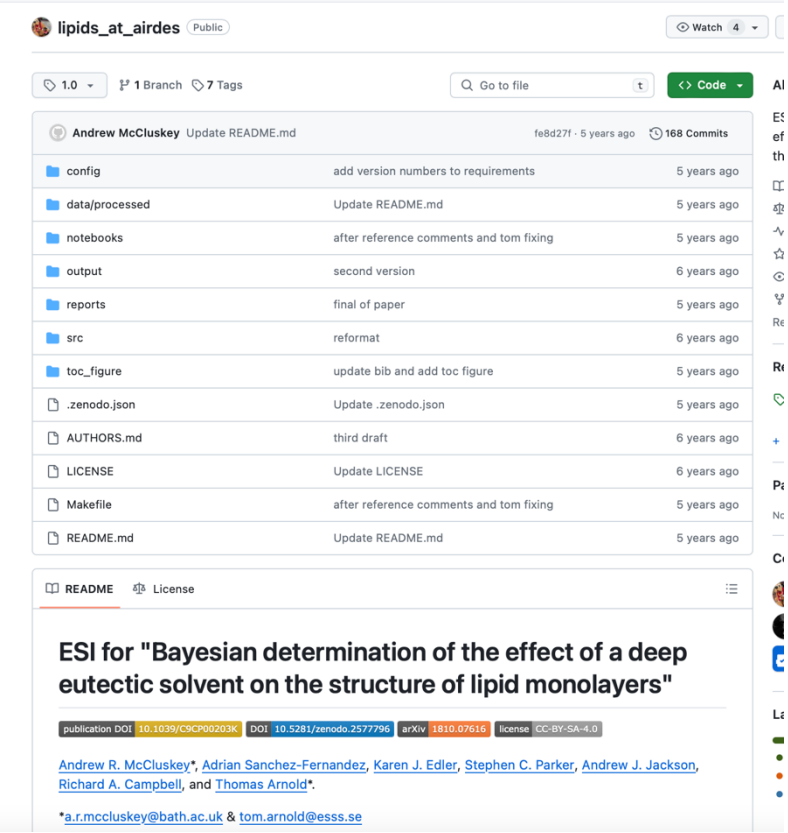
Name
..
cascade.qsub
create_supercellS257_v2.py
dump_renamer.sh
lammps2unmindump.py
mclog2totallog.py
meandefectnumber.py
rec_GCMC_v2.py

Even as experimental chemists – you'll plot graphs and analyze your data → Programming empowers you!



- https://pythoninchemistry.org/reproducible-analysis

Fully reproducible project reports



Reproducible figures and data analysis – that's awesome!!

- Everything you learned in terms of plotting, analyzing chemical data, streamlining data processing will help you throughout your degree.

- If you have to process multiple data files (e.g. excel tables), why not write the code ones and apply it to them, and get much more appealing plots.

https://github.com/arm61/lipids_at_airdes/

The same principles for programming hold true for chemistry.

– the more reusable/reproducible your experiment, the more impactful will be.

Open Science Challenges in Chemistry

- Chemistry data is published in **non-machine-actionable formats** (pdfs and ChemDraw are not fully machine-actionable)
- Not machine-actionable, means you need to **extensive human intervention extract the information**.
- **Knowledge is lost** or tedious to get back
- **No negative data** is published, We learn better by having access to positive and negative examples.
- **Electronic lab notebooks** are super valuable, if you capture all information.

nature chemistry

Explore content | About the journal | Publish with us

nature > nature chemistry > perspectives > article

Perspective | Published: 04 April 2022

Making the collective knowledge of chemistry open and machine actionable

Kevin Maik Jablonka, Luc Patiny & Berend Smit

Nature Chemistry **14**, 365–376 (2022) | Cite this article

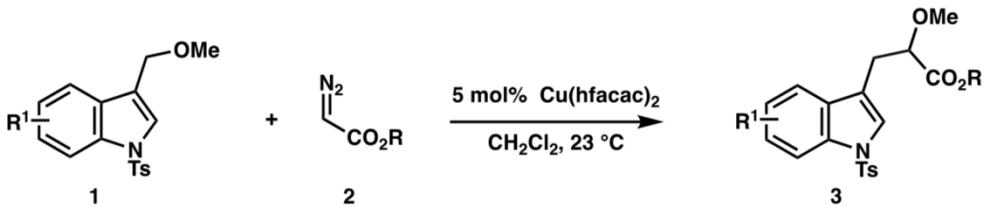
14k Accesses | **13** Citations | **118** Altmetric | Metrics

Kevin Jablonka

Failed project: Automatically extracting information on total syntheses from pdfs.

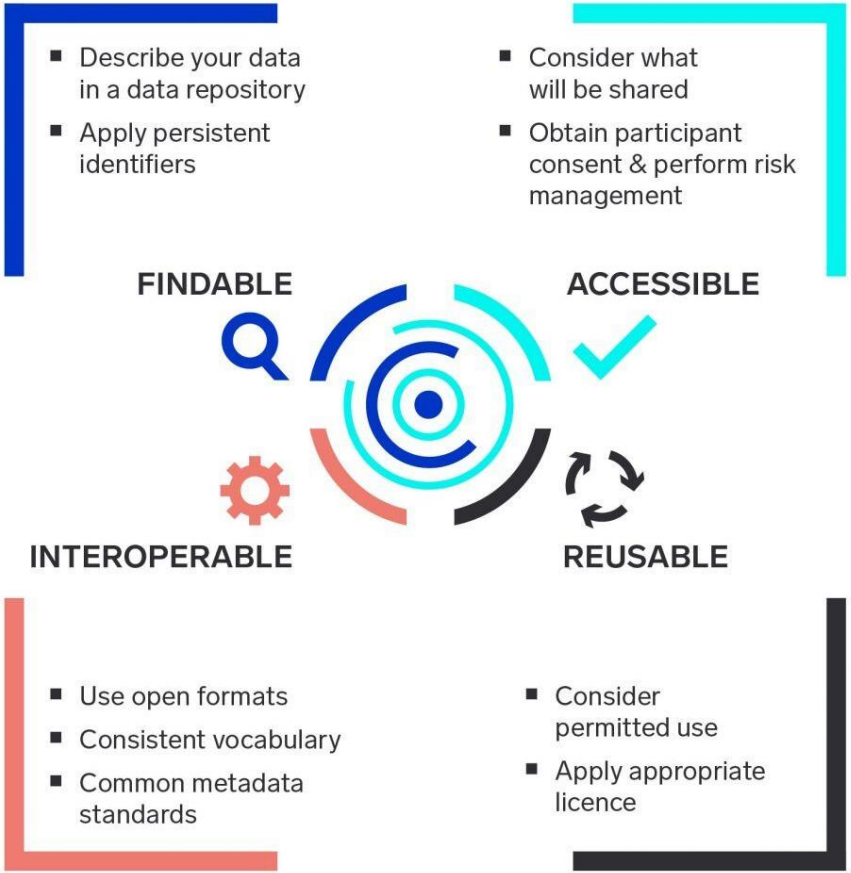
- If the information is incomplete or badly referred from the beginning, the best extraction model cannot recover the information.

4. General Procedure for [1,2]-Rearrangement Reaction



To a flame-dried 8 mL reaction vial charged with indole **1** (0.16 mmol) and $Cu(hfacac)_2$ (5 mol%) under argon, CH_2Cl_2 (1 mL) was added at 23 °C. A solution of diazoester **2** (2.4 equiv) in CH_2Cl_2 (2 mL) was added using a syringe pump at a rate of 2 mL/h to the reaction mixture. After

FAIR data


Describe your data in a data repository
Apply persistent identifiers
Consider what will be shared
Obtain participant consent & perform risk management
FINDABLE
ACCESSIBLE
INTEROPERABLE
REUSABLE
Use open formats
Consistent vocabulary
Common metadata standards
Consider permitted use
Apply appropriate licence

Adding all reaction information and metadata as .csv file

```
,Unnamed: 0,yie_react,SCREEN_ID,NOTEBOOK_ID,REACTION_ID,KeyWord_STD,ReactionGroup,RXN_SMILES,PCAT_CMPD_ID,PRODUCT_STRUCTURE,Product_Yield_PCT_Area_UV,Product_U
V_Analysis_Wavelength_nm,Product_Yield_Mass_Ion_Count,Product_Selectivity,Solvent_1_Name,Reaction_T,Reaction_Time_hrs,Catalyst_1_eq,Catalyst_1_ID.1.,catalyst_1
_ID_1_SMILES,Catalyst_1_ID.2.,catalyst_1_ID_2_SMILES,Catalyst_1_Short_Hand,Catalyst_2_eq,Catalyst_2_ID.1.,catalyst_2_ID_1_SMILES,Catalyst_2_ID.2.,catalyst_2_ID
_2_SMILES,Catalyst_2_Short_Hand,Reactant_1_eq,Reactant_1_ID,Reactant_1_SMILES,Reactant_2_eq,Reactant_2_ID,reactant_2_SMILES,Reactant_3_eq,Reactant_3_ID,reactan
t_3_SMILES,Reagent_1_eq,Reagent_1_ID,Reagent_1_Short_Hand,Reagent_2_eq,Reagent_2_ID,Reagent_2_Short_Hand,ReactionClass,react_1_canon,react_
2_canon,halide,nuc,catalyst,ligand,Year
0,1,97__PF-05314903__MFCD03419272__,SCRN_298,00119131-2673,RKB-03997369,BUCHWALD,Metal Mediated,Br[c:1]1[n:9][c:8]([c:4]2[cH:3][cH:2]1)[cH:7]c[cH:6][c:5]2[
C:10]#[N:11].[CH3:12][C:13]([O:16][C:17]([N:19]1[CH2:24][C@H:22]([NH2:23])[CH2:21][CH2:20]1)=[O:18])([CH3:15])[CH3:14]>>[CH3:12][C:13]([O:16][C:17]([N:19]1[
CH2:24][C@H:22]([NH:23][c:4]2[cH:10][n:11][c:1]([N:9]3[CH2:6][CH2:5]O[CH2:7][CH2:8]3)[cH:2][cH:3]2)[CH2:21][CH2:20]1)=[O:18])([CH3:15])[CH3:14],PF-07093146,CC(
C)(C)OC(=O)N1CC[C@H](C1)Nc2ccc(nc2)N3CCOCC3,97,254.0,158883.302107419,,Dioxane,120.0,,0.125,,,MFCD22572666,c1(c(c(ccc10C)OC)P(C1CCCCC1)C1CCCCC1)c1c(cc(cc1C(C)C
)C(C)C)C(C)C.c1(ccccc1c1ccccc1N)[Pd+].[O-]S(C)(=O)=O,,0.125,MFCD22572666,c1(c(c(ccc10C)OC)P(C1CCCCC1)C1CCCCC1)c1c(cc(cc1C(C)C)C(C)C)C(C)C.c1(
ccccc1c1ccccc1N)[Pd+].[O-]S(C)(=O)=O,,,3G OMs BrettPhos,1.0,PF-05314903,c1cc(c2c(c1)nc(cc2)Br)C#N,1.0,MFCD03419272,N1(C[C@@H](CC1)N)C(OC(C)(C)
C)=O,,,,,MFCD00064808,KOtPn,,,,Pd coupling,N#Cc1cccc2nc(Br)ccc12,CC(C)(C)OC(=O)N1CC[C@@H](N)C1,N#Cc1cccc2nc(Br)ccc12,CC(C)(C)OC(=O)N1CC[C@@H](N)C1,c1(c(c(
ccc10C)OC)P(C1CCCCC1)C1CCCCC1)c1c(cc(cc1C(C)C)C(C)C)C(C)C.c1(ccccc1c1ccccc1N)[Pd+].[O-]S(C)(=O)=O,,2018
1,2,96__PF-05314903__MFCD03419272__,SCRN_298,00119131-2673,RKB-03997370,BUCHWALD,Metal Mediated,Br[c:1]1[n:9][c:8]([c:4]2[cH:3][cH:2]1)[cH:7]c[cH:6][c:5]2[
C:10]#[N:11].[CH3:12][C:13]([O:16][C:17]([N:19]1[CH2:24][C@H:22]([NH2:23])[CH2:21][CH2:20]1)=[O:18])([CH3:15])[CH3:14]>>[CH3:12][C:13]([O:16][C:17]([N:19]1[
CH2:24][C@H:22]([NH:23][c:4]2[cH:10][n:11][c:1]([N:9]3[CH2:6][CH2:5]O[CH2:7][CH2:8]3)[cH:2][cH:3]2)[CH2:21][CH2:20]1)=[O:18])([CH3:15])[CH3:14],PF-07093146,CC(
C)(C)OC(=O)N1CC[C@H](C1)Nc2ccc(nc2)N3CCOCC3,96,254.0,143607.498285105,,Dioxane,120.0,,0.125,,,MFCD22572666,c1(c(c(ccc10C)OC)P(C1CCCCC1)C1CCCCC1)c1c(cc(cc1C(C)C
)C(C)C)C(C)C.c1(ccccc1c1ccccc1N)[Pd+].[O-]S(C)(=O)=O,,0.125,MFCD22572666,c1(c(c(ccc10C)OC)P(C1CCCCC1)C1CCCCC1)c1c(cc(cc1C(C)C)C(C)C)C(C)C.c1(
ccccc1c1ccccc1N)[Pd+].[O-]S(C)(=O)=O,,,3G OMs BrettPhos,1.0,PF-05314903,c1cc(c2c(c1)nc(cc2)Br)C#N,1.0,MFCD03419272,N1(C[C@@H](CC1)N)C(OC(C)(C)
C)=O,,,,,MFCD00064808,KOtPn,,,,Pd coupling,N#Cc1cccc2nc(Br)ccc12,CC(C)(C)OC(=O)N1CC[C@@H](N)C1,N#Cc1cccc2nc(Br)ccc12,CC(C)(C)OC(=O)N1CC[C@@H](N)C1,c1(c(c(
ccc10C)OC)P(C1CCCCC1)C1CCCCC1)c1c(cc(cc1C(C)C)C(C)C)C(C)C.c1(ccccc1c1ccccc1N)[Pd+].[O-]S(C)(=O)=O,,2018
2,3,96__PF-05314903__MFCD03419272__,SCRN_298,00119131-2673,RKB-03997371,BUCHWALD,Metal Mediated,Br[c:1]1[n:9][c:8]([c:4]2[cH:3][cH:2]1)[cH:7]c[cH:6][c:5]2[
```

Can easily be created, if you are using an **electronic laboratory notebook** and saving all the information.

Article | Open access | Published: 02 January 2024

Probing the chemical 'reactome' with high-throughput experimentation data

Emma King-Smith, Simon Berritt, Louise Bernier, Xinjun Hou, Jacquelyn L. Klug-McLeod, Jason Mustakis, Neal W. Sach, Joseph W. Tucker, Qingyi Yang, Roger M. Howard & Alpha A. Lee

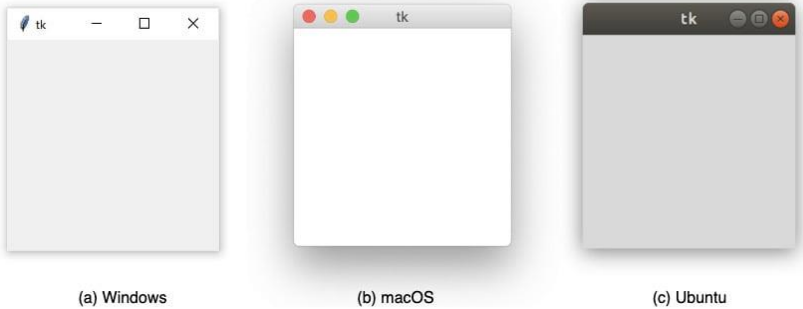
Nature Chemistry **16**, 633–643 (2024) | Cite this article

Change of topic.

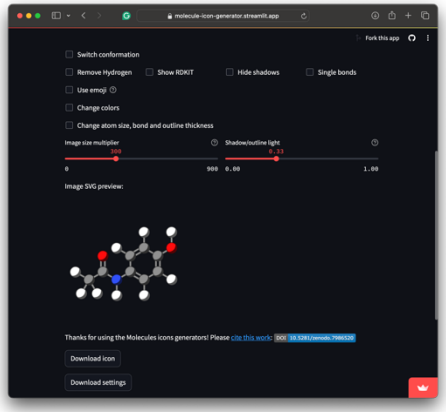
Making your Python code into an app.
→ Creating a graphical user interface (GUI)

Desktop vs Web applications

- **Desktop applications** are software programs that are installed on a specific computer or workstation and run on the operating system of that machine.
- **Web applications** are hosted on local/remote servers and accessed (over the internet) using a web browser.
- VS Code (desktop) / Jupyter lab (browser-based)



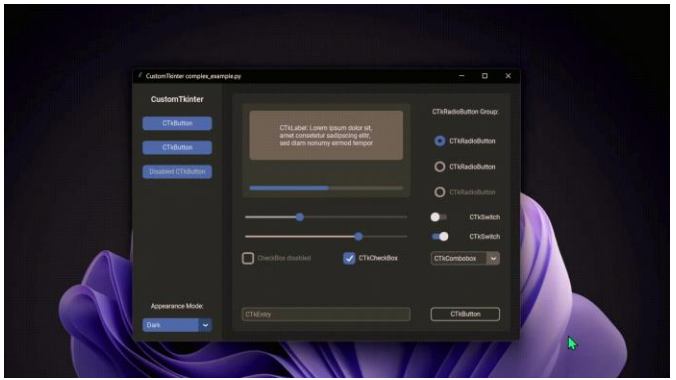
(a) Windows (b) macOS (c) Ubuntu



https://realpython.com/python-gui-tkinter/

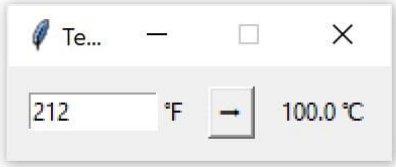
Desktop GUIs with Python

- **Tkinter:** This is the standard GUI toolkit for Python. It is simple to use for small applications and comes pre-installed with Python, making it a great choice for beginners.
- Other libraries
  - **PyQt or PySide**: These are set of bindings for the Qt application framework. They are more powerful than Tkinter and suitable for complex applications but have a steeper learning curve.
  - **wxPython:** This is another option for creating cross-platform desktop applications, known for having native look and feel.



https://github.com/TomSchimansky/CustomTkinter/blob/master/examples/complex_example.py

Example of Tkinter



- Temperature converter app

```
import tkinter as tk
```

```
# Set up the window
window = tk.Tk()
window.title("Temperature Converter")
window.resizable(width=False, height=False)
```

```
# Create the Fahrenheit entry frame with an Entry
# widget and label in it
frm_entry = tk.Frame(master=window)
ent_temperature = tk.Entry(master=frm_entry, width=10)
lbl_temp = tk.Label(master=frm_entry, text="\N{DEGREE FAHRENHEIT}")
```

```
# Layout the temperature Entry and Label in frm_entry
# using the .grid() geometry manager
ent_temperature.grid(row=0, column=0, sticky="e")
lbl_temp.grid(row=0, column=1, sticky="w")
```

```
# Create the conversion Button and result display Label
btn_convert = tk.Button(
    master=window,
    text="\N{RIGHTWARDS BLACK ARROW}",
    command=fahrenheit_to_celsius
)
lbl_result = tk.Label(master=window, text="\N{DEGREE CELSIUS}")
```

```
def fahrenheit_to_celsius():
    """Convert the value for Fahrenheit to Celsius and insert the
    result into lbl_result.
    """
    fahrenheit = ent_temperature.get()
    celsius = (5 / 9) * (float(fahrenheit) - 32)
    lbl_result["text"] = f"{round(celsius, 2)} \N{DEGREE CELSIUS}"
```

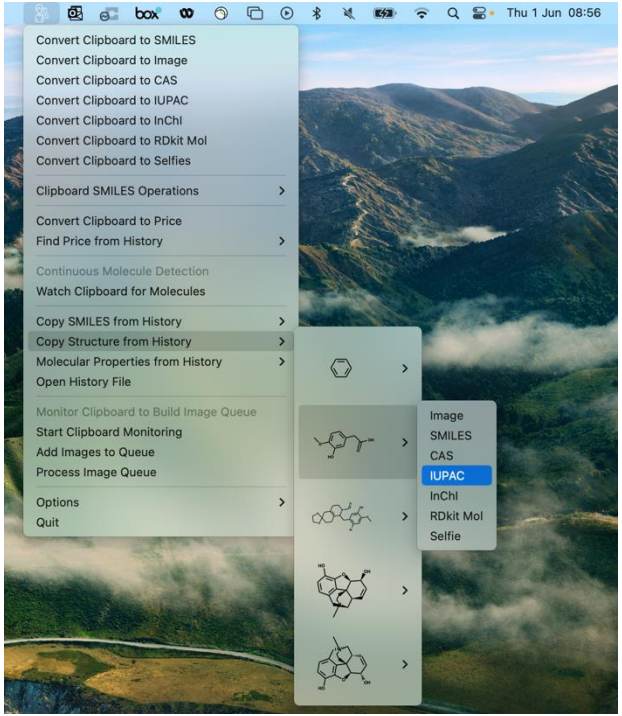
```
# Set up the layout using the .grid() geometry manager
frm_entry.grid(row=0, column=0, padx=10)
btn_convert.grid(row=0, column=1, pady=10)
lbl_result.grid(row=0, column=2, padx=10)
```

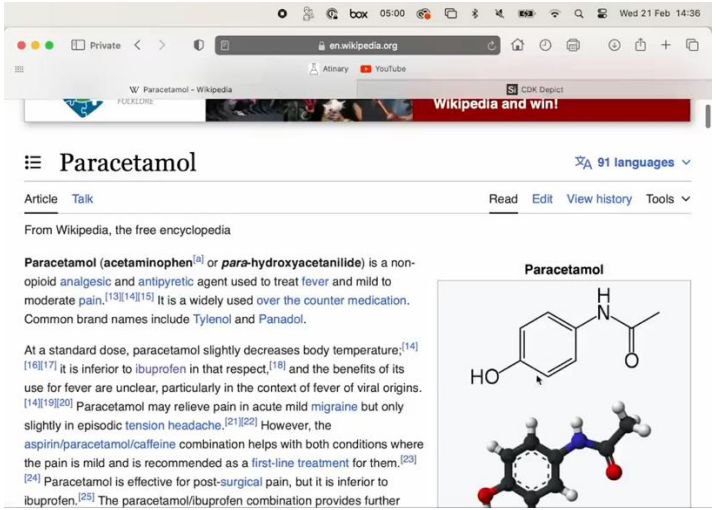
```
# Run the application
window.mainloop()
```

- https://realpython.com/python-gui-tkinter/#building-a-temperature-converter-example-app

CMD + V for chemistry: Image to chemical structure conversion directly done in the clipboard

(for the moment Mac only, windows version in the making)





Based on rumps: https://rumps.readthedocs.io/en/latest/
https://github.com/O-Schilter/Clipboard-to-SMILES-Converter
(I'm not claiming this is a well designed repository)

Web-based GUIs in Python

- What is more common to demonstrate simple Python projects are web-based apps.
- Advantage: Platform Independence. They can be accessed from any device or operating system that supports a compatible web browser.

	Simplicity	Maturity	Flexibility	Primary Use
Gradio	A	C	B	ML Model Demos
Streamlit	A	C	B	Dashboards
Dash	B	B	B	Dashboards
Flask	C	A	A	Web Interfaces

https://towardsdatascience.com/gradio-vs-streamlit-vs-dash-vs-flask-d3defb1209a2

More details

- **Gradio** is specifically built with machine learning models in mind. So if you want to create a web UI specifically for a machine learning model that you built, Gradio's **simple syntax and setup** is the way to go.
- **Streamlit** is useful if you want to get a **dashboard up and running quickly**, and have the flexibility to add lots of components and controls. As well, Streamlit allows you to build a web UI or a dashboard much faster than Dash or Flask. → what we will use **in the exercises**
- **Dash** if you want to be a production-ready dashboard for a larger company, since it's mainly tailored for enterprise companies.
- **Flask** if you have knowledge of Python/HTML/CSS programming and you want to build your own solution completely from scratch.

https://towardsdatascience.com/gradio-vs-streamlit-vs-dash-vs-flask-d3defb1209a2

Example from last year's LLM in Chemistry/Materials Hackathon

- Mainly coded by a chemistry undergrad in 24h

Magdalena Lederbauer

GlossaGen

Your Manuscript's Glossary, Instantly Created

https://github.com/mlederbauer/glossagen/blob/feat/4-gui/src/glossagen/app.py

Made with Gradio.

Streamlit

A faster way to build and share data apps

Turn your data scripts into shareable web apps in minutes.
All in pure Python. No front-end experience required.

https://streamlit.io

Streamlit – probably the fastest way of getting a browser-based UI

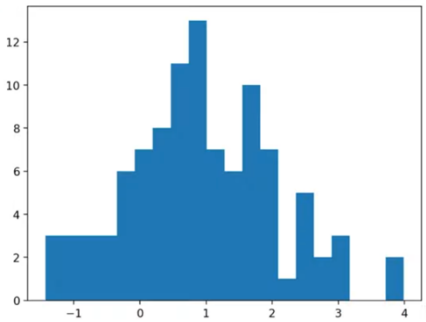
- Create an app.py file with
- And run `streamlit run app.py`

app.py

```
import streamlit as st
```

- Use **st.write**(any variable) to show in text, dataframes, images, ... in the app → it's like a **print()**, just for streamlit webapps

Hello world



	a	b	c
0	0.2348	-1.4700	1.1083
1	0.4511	-0.6847	0.2444
2	-0.2451	-0.5670	-0.6300
3	0.6000	-1.1485	0.7858
4	-0.5392	1.7035	1.1931
5	-1.0795	-0.7488	-2.0435
6	0.2096	1.1038	-1.1573
7	-0.4214	-0.0336	0.9281
8	1.7883	0.3988	0.1005
9	-0.2234	-0.1532	-0.8803

- st.write("Hello world")
- st.write(matplotlib_figure)
- st.write(df)

https://www.youtube.com/watch?v=vIQQR_yq-8I

st.write is simple, but for more control there are more specific functions

```
st.markdown(
    """
# My Streamlit App

**Hello** _there_, `how are you`?

> Quickly build Data Web Apps

- This
- is
- Markdown

---

```python
name = 'Fanilo'
print ("Hello World")
```
```

My Streamlit App

Hello *there*, `how are you` ?

> Quickly build Data Web Apps

- This
- is
- Markdown

```
name = 'Fanilo'
print ("Hello World")
```

```
penguins = sns.load_dataset(
    "penguins"
)
st.dataframe(penguins)
```

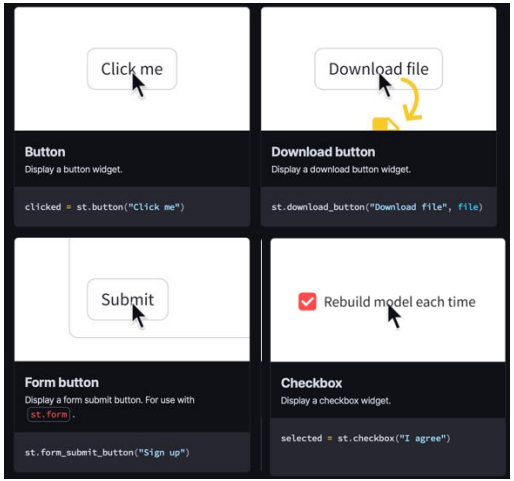
	species	island	bill_length_mm	bill_depth_mm	flip
0	Adelie	Torgersen	39.1000	18.7000	
1	Adelie	Torgersen	39.5000	17.4000	
2	Adelie	Torgersen	40.3000	18.0000	
3	Adelie	Torgersen	<NA>	<NA>	
4	Adelie	Torgersen	36.7000	19.3000	
5	Adelie	Torgersen	39.3000	20.6000	
6	Adelie	Torgersen	38.9000	17.8000	
7	Adelie	Torgersen	39.2000	19.6000	
8	Adelie	Torgersen	34.1000	18.1000	
9					

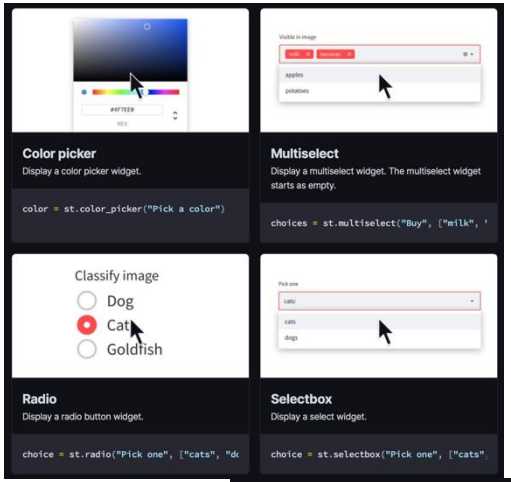
```
results = download_youtube_api()
st.json(results)
```

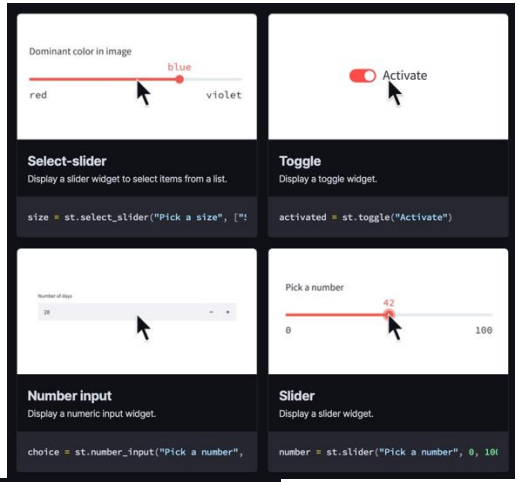

```
[
  0 : {
    "publishedAt" : "2020-01-17T18:19:14Z"
    "channelId" :
    "UC3LD42rjj-Owtxsa6PwGU5Q"
    "title" : "1/4: What is Streamlit"
    "description" :
    "Part 1 of a short video series of
    Adrien's presentation at PyData in
    December. In this episode, he
    discusses the problem he saw in ..."
    "thumbnails" : {
      "default" : {
        "url" :
        "https://i.ytimg.com
        /vi/R2nr1uZ8ffc/default.jpg"
```

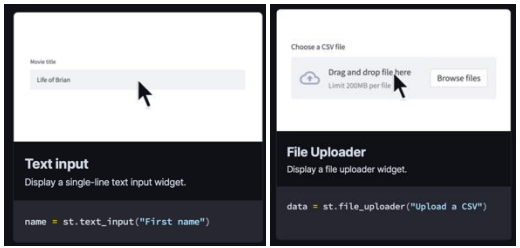

Reasonably good looking JSON viewer.

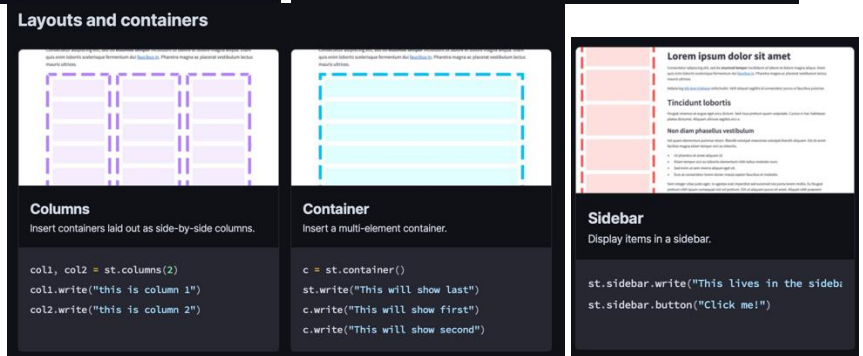
Instead of having to write HTML/Javascript code, you get this out of the box.





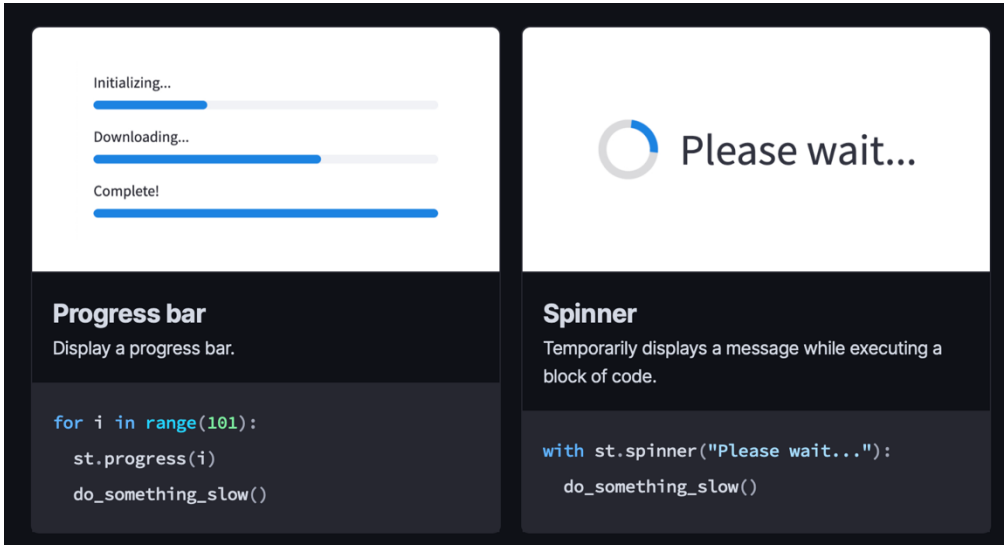


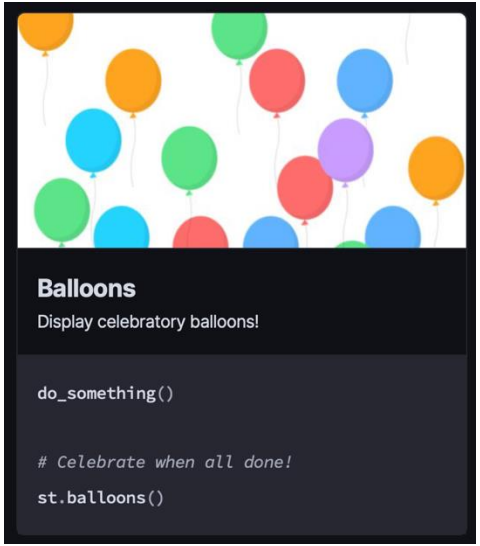




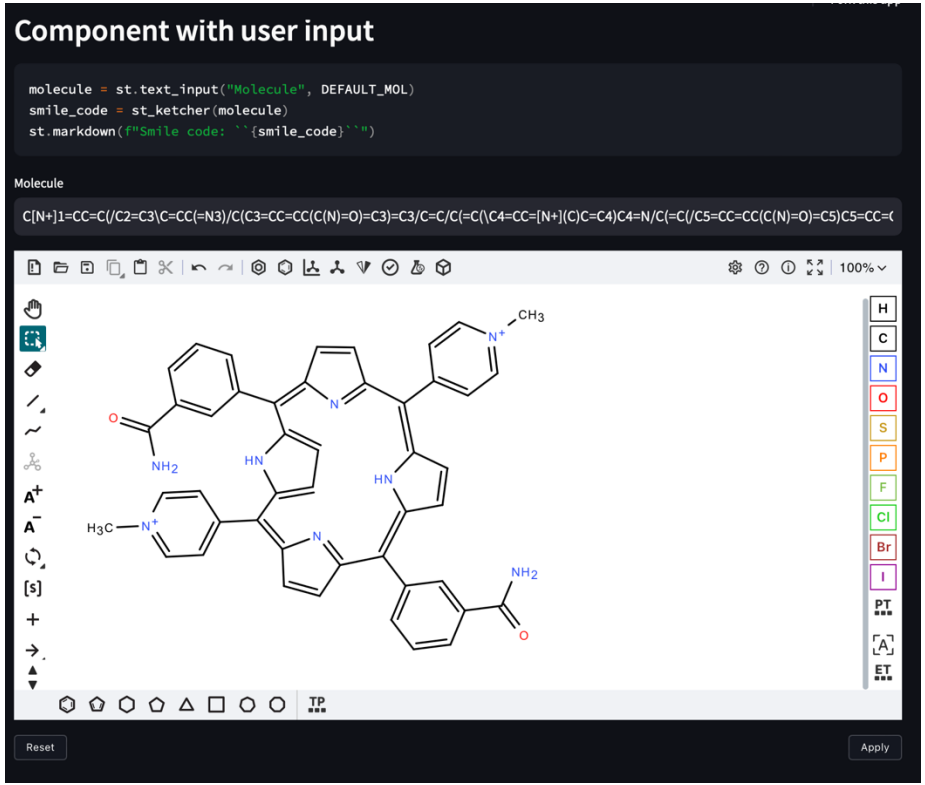
- https://docs.streamlit.io/develop/api-reference

Status elements

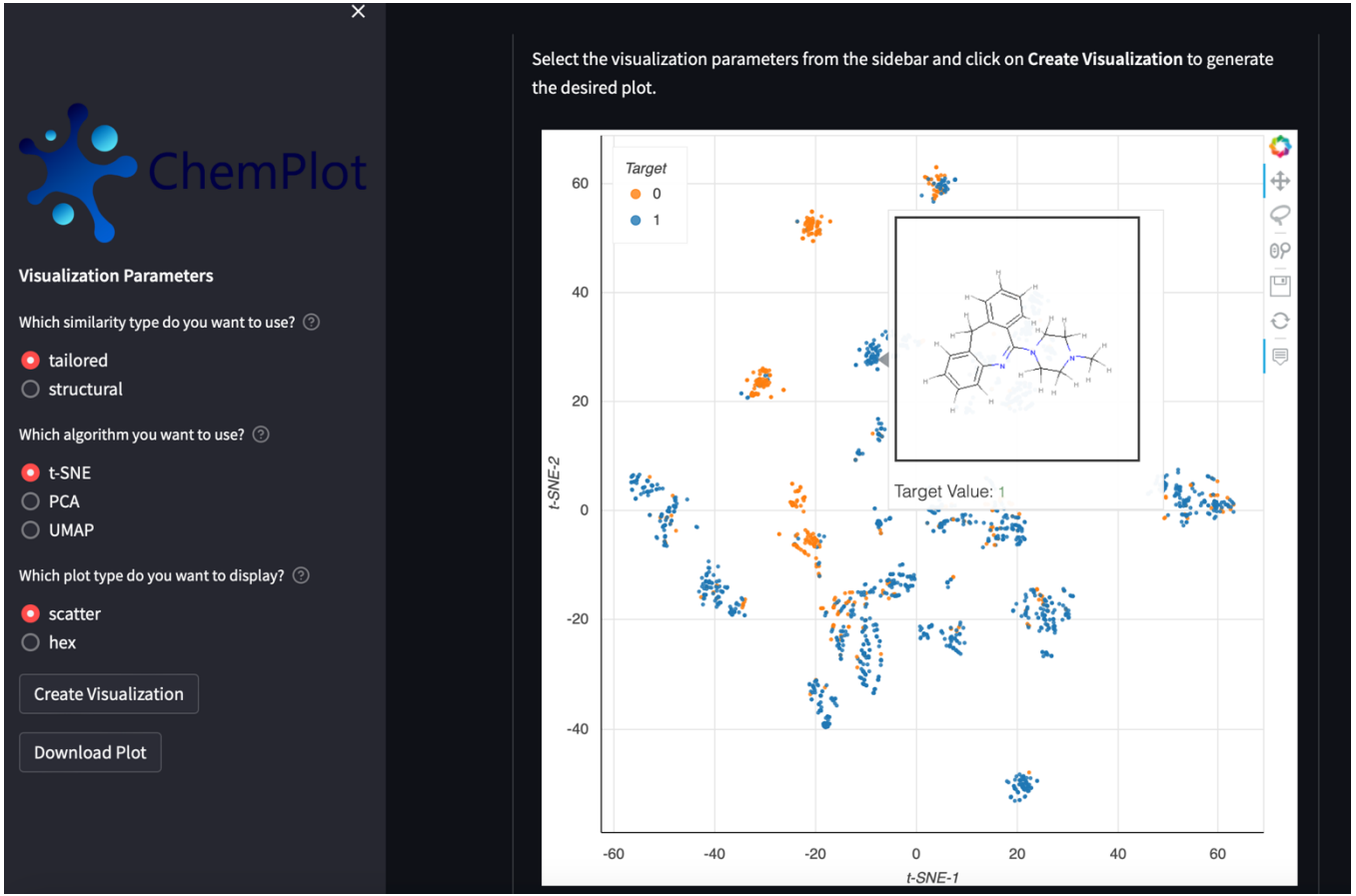




Streamlit-ketcher (molecule drawing frame)



https://github.com/mik-laj/streamlit-ketcher --> pip install streamlit-ketcher



https://chemplot.streamlit.app

https://github.com/mcsorkun/ChemPlot-web

This week's exercises

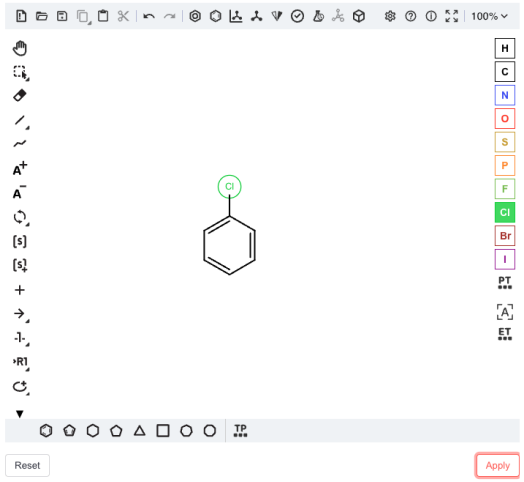
Are you drug like?!

Practical Proramming In Chemistry Week 11

Draw a molecule and see how it stacks up against known drugs

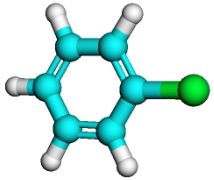
Input Smiles: CCO

Smiles: C1C=CC=CC=1Cl



3D

Generate 3D coordinates for your molecule using rdkit's ETKDGv3 algorithm.



Lipinski Stats

How does your molecule compare to the average for FDA approved drugs?

Molecular weight	LogP
112.01	2.34
↓ -305.21	↓ -0.21

H-Bond Donors	H-Bond Acceptors
0	0
↓ -2.44	↓ -5.7

Passes Lipinski's Rule of Five

Lipinski's rule of five is a rule of thumb to evaluate the druglikeness of a molecule.

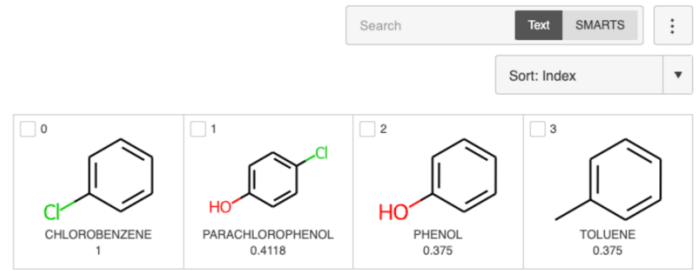
Lipinski's Rules:

1. # H-Bond donors < 5
2. # H-bond acceptors < 10
3. MW < 500 daltons
4. LogP < 5.

As with any rule, there are many exceptions

Most similar drug molecules

Based on Tanimoto similarity of Morgan fingerprints



Recommendation for the project

- To get a starting project structure, you can do it yourself, but if you want some help use https://github.com/schwallergroup/copier-liac-minimal

Usage

Install `copier`:

```
pip install copier
```

Generate a Python package:

```
copier copy gh:schwallergroup/copier-liac-minimal path/to/destination
```

Best in a separate environment (e.g. copier), as you don't want the copier dependencies to interact with your project dependencies.

```
tree .
.
├── LICENSE
├── README.md
├── assets
│   └── banner.png
├── data
│   ├── README.md
│   ├── chirality.cdxml
│   └── single_molecule.cdxml
├── docs
│   └── source
│       ├── _static
│       ├── conf.py
│       └── index.md
├── notebooks
│   ├── README.md
│   └── report.ipynb
├── pyproject.toml
├── src
│   └── cdxml_extractor
│       ├── __init__.py
│       ├── extractor.py
├── tests
│   ├── test_extractor.py
│   └── test_import.py
└── tox.ini
```

License → How the code can be reused (e.g. MIT, default).

Readme → Your landing page, quick intro, how to install, example usage of core functionality

data → If not too large, directly in the repository. Otherwise, exact instructions how to get the data.

docs → You can compile a documentation from the doc strings if you want. Uses sphinx.

notebooks → There can be multiple notebooks, but it should be clear which one is the project report.

pyproject.toml → Define the dependencies, make sure the package is pip installable.

src/package_name → one .py file per module (a single module package is fine, if it makes sense)

tests → one test_*.py file per module, try to test all the functions in the module

tox.ini → automation settings to run the tests

If you create an app.py, either in root folder or create app folder.

Use google/stackoverflow/... (or your favourite language model)

But make sure you understand what you are copying.
→ programming is learned by doing mistakes and iterating