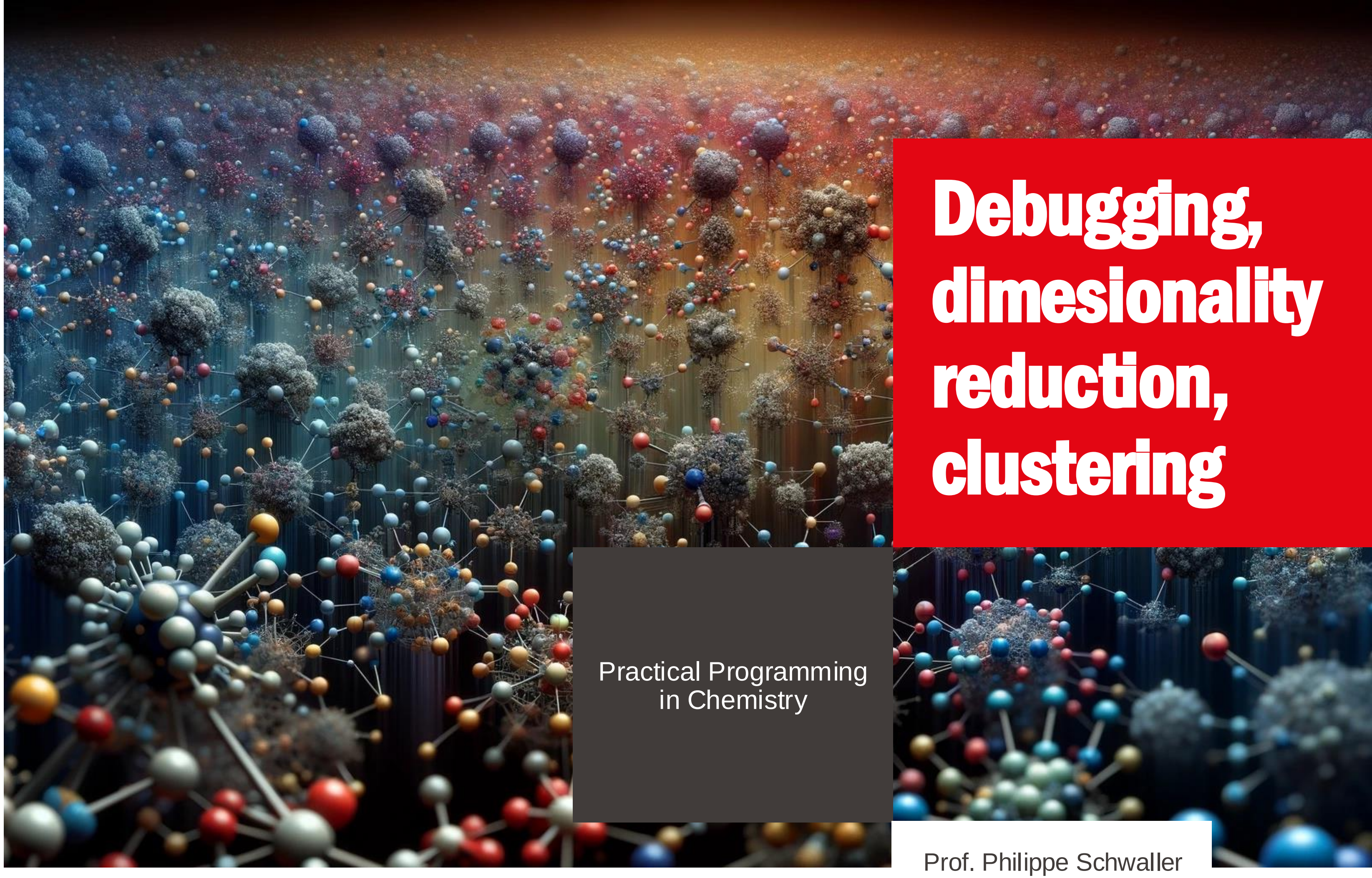
EPFL

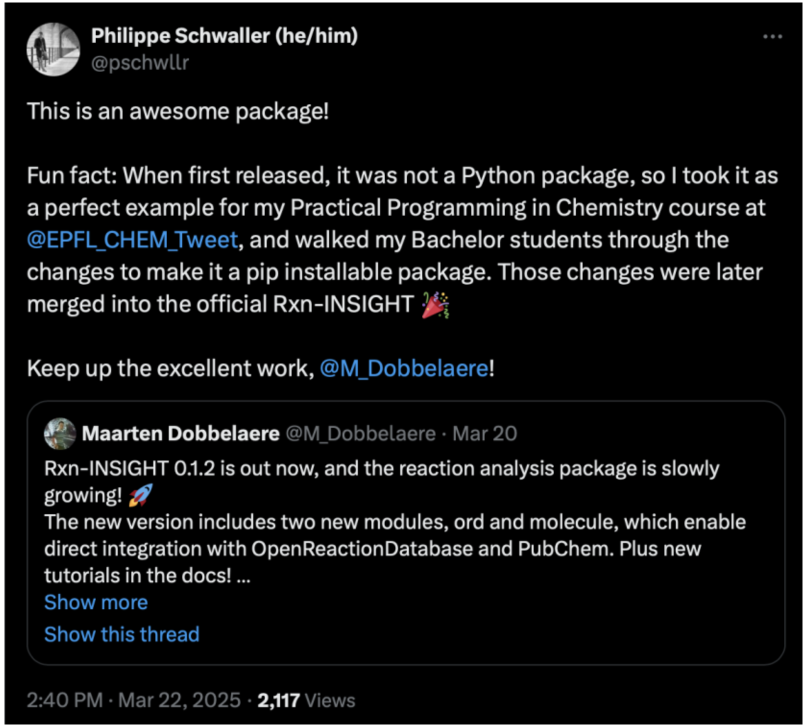


Debugging, dimesionality reduction, clustering

Practical Programming in Chemistry

Prof. Philippe Schwaller

Impact of this course beyond EPFL ☺



Philippe Schwaller (he/him)
@pschwllr

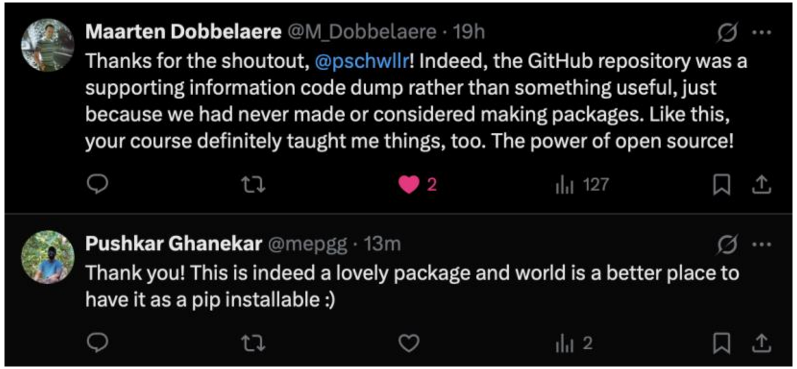
This is an awesome package!

Fun fact: When first released, it was not a Python package, so I took it as a perfect example for my Practical Programming in Chemistry course at @EPFL_CHEM_Tweet, and walked my Bachelor students through the changes to make it a pip installable package. Those changes were later merged into the official Rxn-INSIGHT 🎉

Keep up the excellent work, @M_Dobbelaere!

Maarten Dobbelaere @M_Dobbelaere · Mar 20
Rxn-INSIGHT 0.1.2 is out now, and the reaction analysis package is slowly growing! 🚀
The new version includes two new modules, ord and molecule, which enable direct integration with OpenReactionDatabase and PubChem. Plus new tutorials in the docs! ...
Show more
Show this thread

2:40 PM · Mar 22, 2025 · **2,117** Views



Maarten Dobbelaere @M_Dobbelaere · 19h
Thanks for the shoutout, @pschwllr! Indeed, the GitHub repository was a supporting information code dump rather than something useful, just because we had never made or considered making packages. Like this, your course definitely taught me things, too. The power of open source!

Pushkar Ghanekar @mepgg · 13m
Thank you! This is indeed a lovely package and world is a better place to have it as a pip installable :)

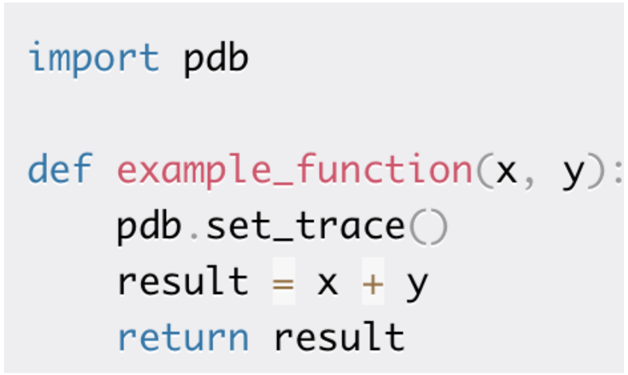
This lecture

- I have an error, what do I do → **debugging**
- Back to more chemistry and cheminformatics-related examples.
  - Visualizing chemical space
  - Clustering

Debugging code

Python Debugger (pdb) or ipdb

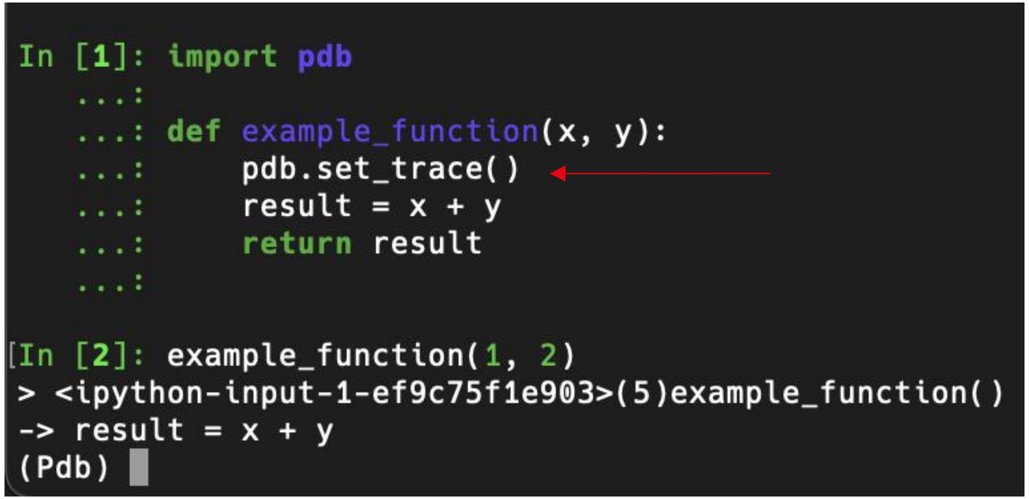
- Python comes with a built-in debugger called PDB (Python Debugger). It allows you to **pause the execution** of your Python code, **inspect variables**, and **step through your code** line by line to **find and fix issues**.



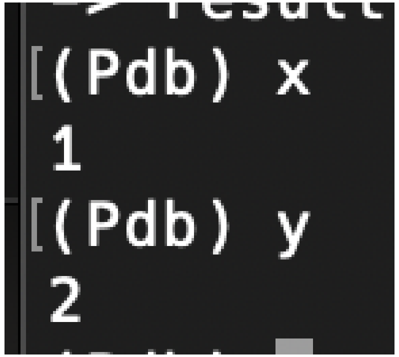
```
import pdb

def example_function(x, y):
    pdb.set_trace()
    result = x + y
    return result
```

https://www.freecodecamp.org/news/python-debugging-handbook/#:~:text=Python%20comes%20with%20a%20built,to%20find%20and%20fix%20issues.

Setting breakpoints



```
In [1]: import pdb
   ...:
   ...: def example_function(x, y):
   ...:     pdb.set_trace()
   ...:     result = x + y
   ...:     return result
   ...:

[In [2]: example_function(1, 2)
> <ipython-input-1-ef9c75f1e903>(5)example_function()
-> result = x + y
(Pdb)
```



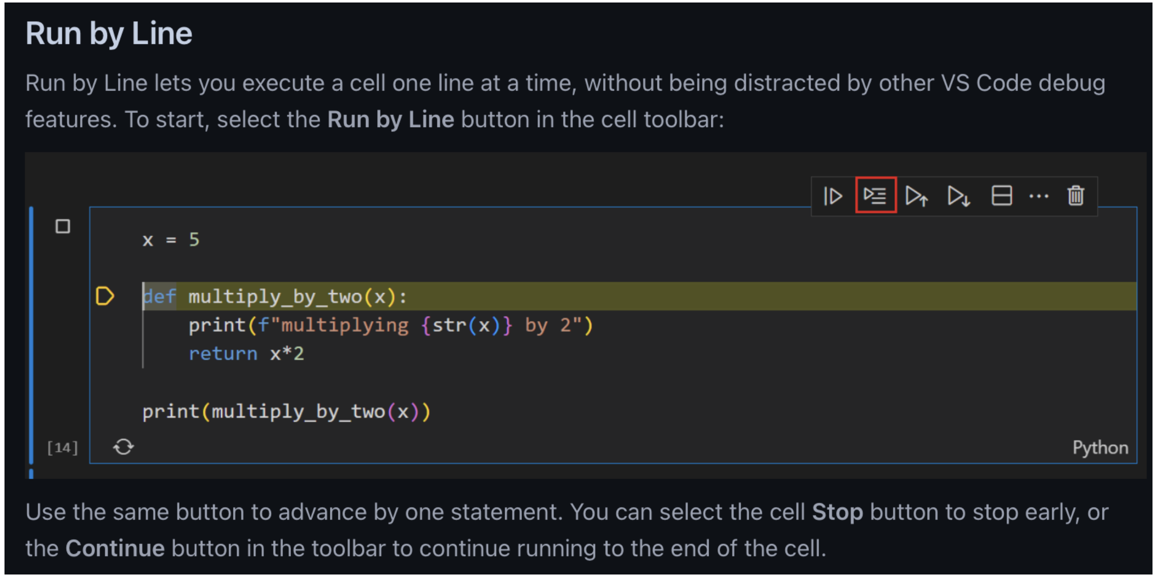
```
[(Pdb) x
1
[(Pdb) y
2
```

Basic commands

- Once you have stopped at a breakpoint, how can you navigate around in your code.
- **n (next):** Continue execution until the next line in the current function is reached. If the current line contains a function call, it will *not step into the called function*.
- **c (continue):** Continue execution until the next breakpoint is encountered.
- **s (step):** Execute the current line of code and stop at the first possible occasion (either in a function that is called or *at the next line in the current function*).
- **q (quit):** Exit the debugger and terminate the program.

Debugging a Jupyter notebook in VSCode

- https://code.visualstudio.com/docs/datascience/jupyter-notebooks#_debug-a-jupyter-notebook



Run by Line

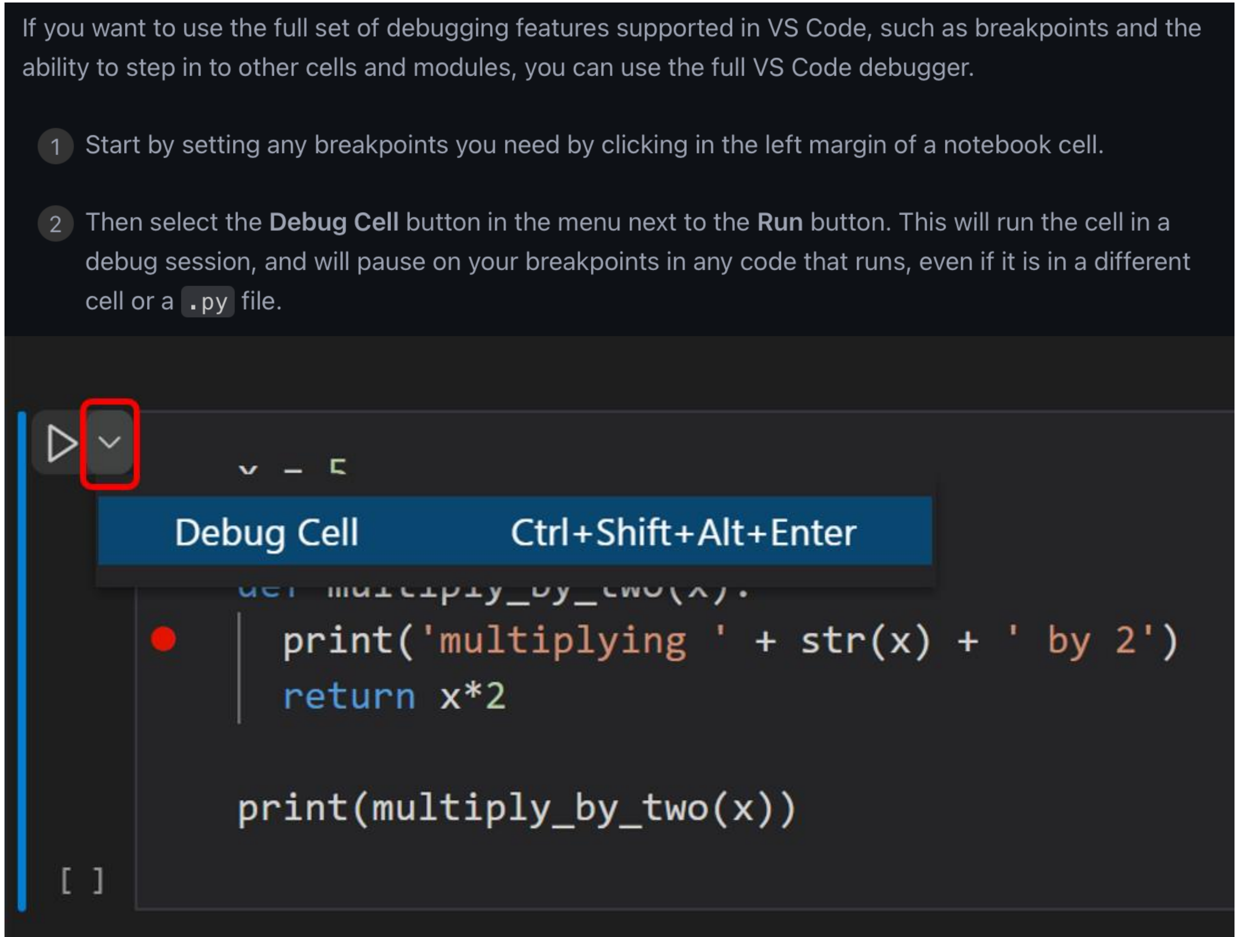
Run by Line lets you execute a cell one line at a time, without being distracted by other VS Code debug features. To start, select the **Run by Line** button in the cell toolbar:

```
x = 5

def multiply_by_two(x):
    print(f"multiplying {str(x)} by 2")
    return x*2

print(multiply_by_two(x))
```

Use the same button to advance by one statement. You can select the cell **Stop** button to stop early, or the **Continue** button in the toolbar to continue running to the end of the cell.

Speaker

Debug cell or .py code



If you want to use the full set of debugging features supported in VS Code, such as breakpoints and the ability to step in to other cells and modules, you can use the full VS Code debugger.

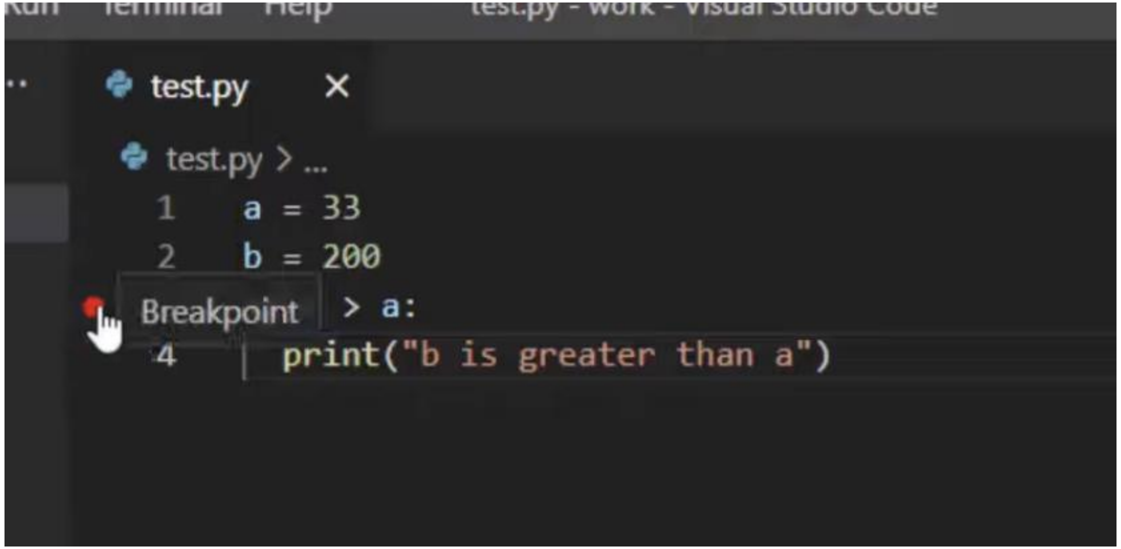
1. Start by setting any breakpoints you need by clicking in the left margin of a notebook cell.
2. Then select the **Debug Cell** button in the menu next to the **Run** button. This will run the cell in a debug session, and will pause on your breakpoints in any code that runs, even if it is in a different cell or a `.py` file.

```
x = 5

def multiply_by_two(x):
    print('multiplying ' + str(x) + ' by 2')
    return x*2

print(multiply_by_two(x))
```

EPFL

Youtube video (11 min) on Debugging Python with VSCode – easy steps to follow (optional)

- https://www.youtube.com/watch?v=b4p-SBjHh28

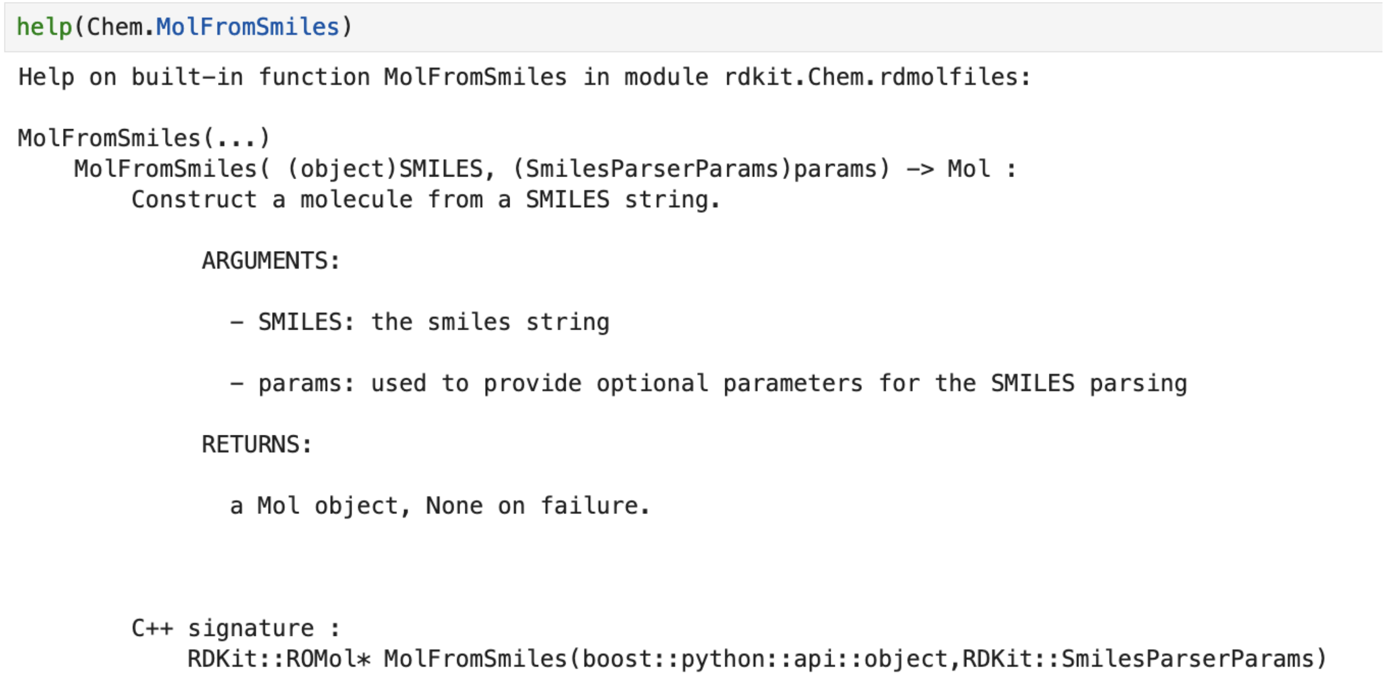


```
a = 33
b = 200
 > a:
    print("b is greater than a")
```

Setting a breakpoint in VSCode is equivalent to writing a line with “pdb.set_trace()”

Accessing the documentation in Jupyter notebooks

Accessing the documentation in Jupyter notebooks

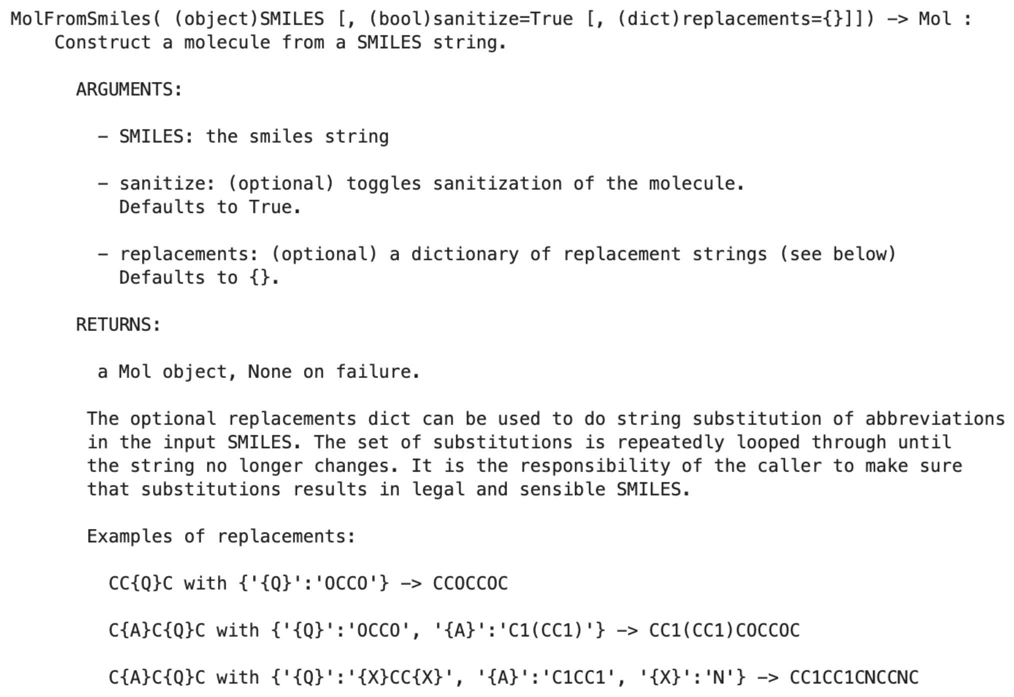


```python
help(Chem.MolFromSmiles)
```

```
Help on built-in function MolFromSmiles in module rdkit.Chem.rdmolfiles:

MolFromSmiles(...)
    MolFromSmiles( (object)SMILES, (SmilesParserParams)params) -> Mol :
        Construct a molecule from a SMILES string.

             ARGUMENTS:

               - SMILES: the smiles string

               - params: used to provide optional parameters for the SMILES parsing

             RETURNS:

               a Mol object, None on failure.



        C++ signature :
            RDKit::ROMol* MolFromSmiles(boost::python::api::object,RDKit::SmilesParserParams)
```

Alternative: Chem.MolFromSmiles(+ ”TAB”
This will also open the documentation.

Accessing the documentation in Jupyter notebooks



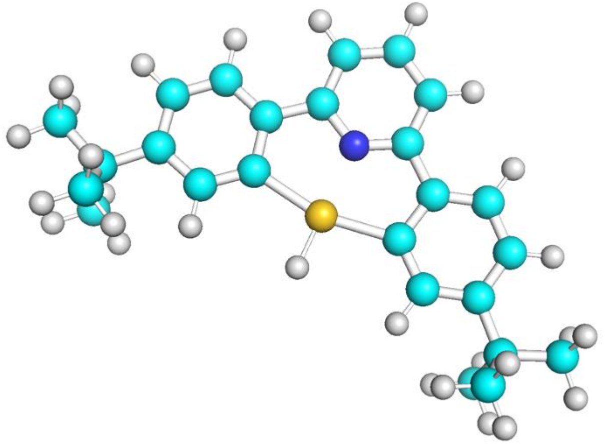
```
MolFromSmiles( (object)SMILES [, (bool)sanitize=True [, (dict)replacements={}]]) -> Mol :
    Construct a molecule from a SMILES string.

      ARGUMENTS:

        - SMILES: the smiles string

        - sanitize: (optional) toggles sanitization of the molecule.
          Defaults to True.

        - replacements: (optional) a dictionary of replacement strings (see below)
          Defaults to {}.

      RETURNS:

        a Mol object, None on failure.

       The optional replacements dict can be used to do string substitution of abbreviations
       in the input SMILES. The set of substitutions is repeatedly looped through until
       the string no longer changes. It is the responsibility of the caller to make sure
       that substitutions results in legal and sensible SMILES.

       Examples of replacements:

         CC{Q}C with {'{Q}':'OCCO'} -> CCOCCOC

         C{A}C{Q}C with {'{Q}':'OCCO', '{A}':'C1(CC1)'} -> CC1(CC1)COCCOC

         C{A}C{Q}C with {'{Q}':'{X}CC{X}', '{A}':'C1CC1', '{X}':'N'} -> CC1CC1CNCCNC
```

I learned something new → you can pass it a replacement dictionary.

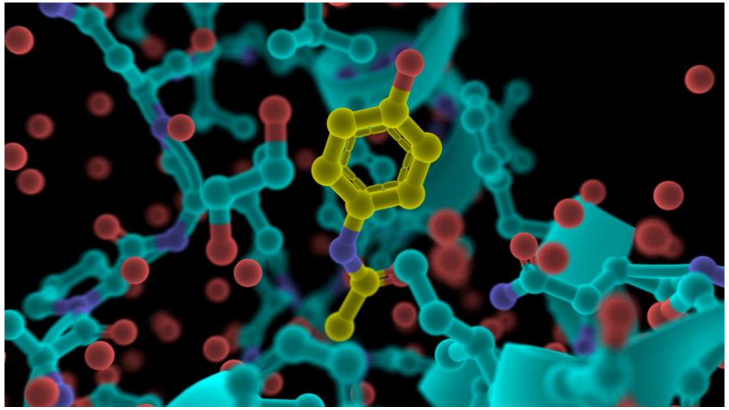
Visualization tools in chemistry

“Hey, this exists, check it out in more details if interested.”

pymol - https://github.com/schrodinger/pymol-open-source

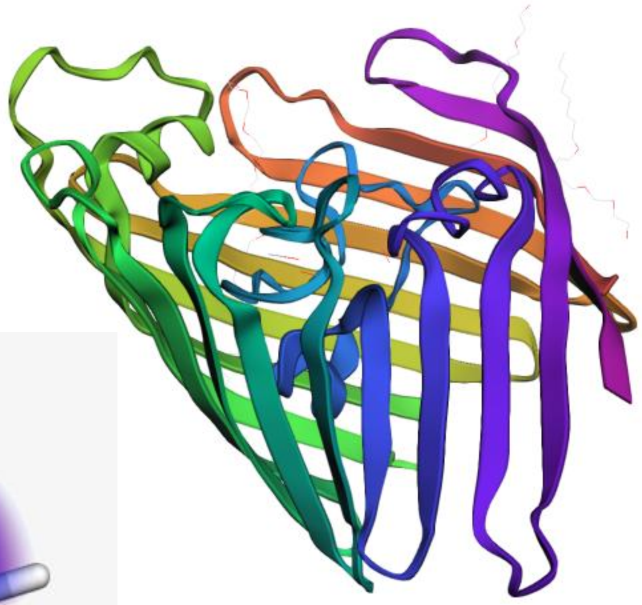


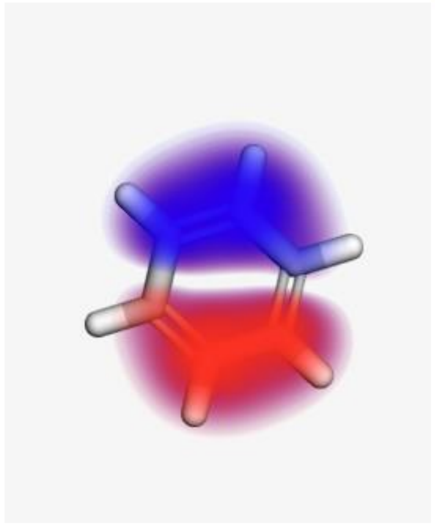
- Paid version maintained by Schrodinger, but open source version available
  - Pip installable
- Software for molecular and biomolecular visualization
- Good for images
- Built on python
  - Intuitive interface
  - Easy scripting
  - But… defaults aethetics not great
- Style guides:
  - https://gist.github.com/bobbypaton/1cdc4784f3fc8374467bae5eb410edef
  - https://www.blopig.com/blog/2024/12/making-pretty-pictures-in-pymol-v2/



3Dmol - https://3dmol.org/doc/index.html

- JavaScript library for molecular visualization
- Render molecular visualizations in webapps

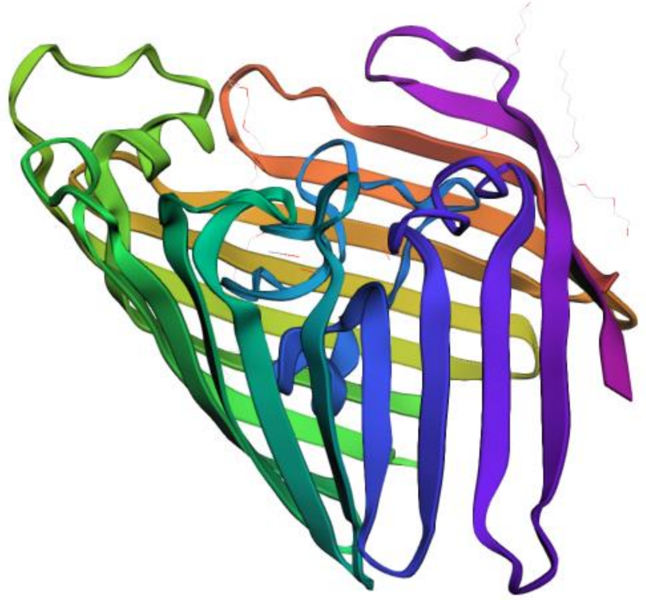




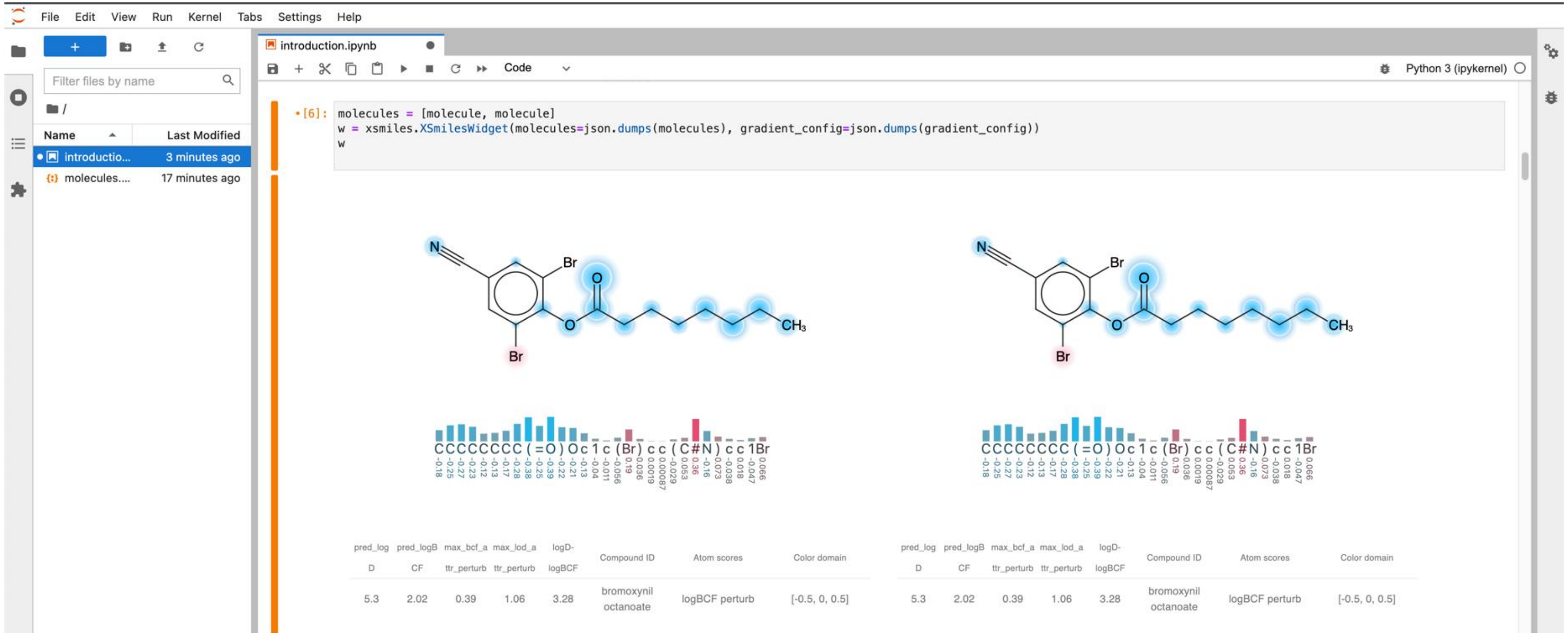
py3Dmol - https://3dmol.org/doc/index.html

- Embed 3Dmol in Jupyter notebook
  - And streamlit (see tomorrow)
  - We've seen a brief overview in the exercises
  - pip install py3Dmol

- Tutorial: https://colab.research.google.com/d/1T2zR59TXyWRcNxRgOAiqVPJW83NV_?usp=sharing



Xsmiles – displaying atom-wise properties



```
[6]: molecules = [molecule, molecule]
     w = xsmiles.XSmilesWidget(molecules=json.dumps(molecules), gradient_config=json.dumps(gradient_config))
     w
```

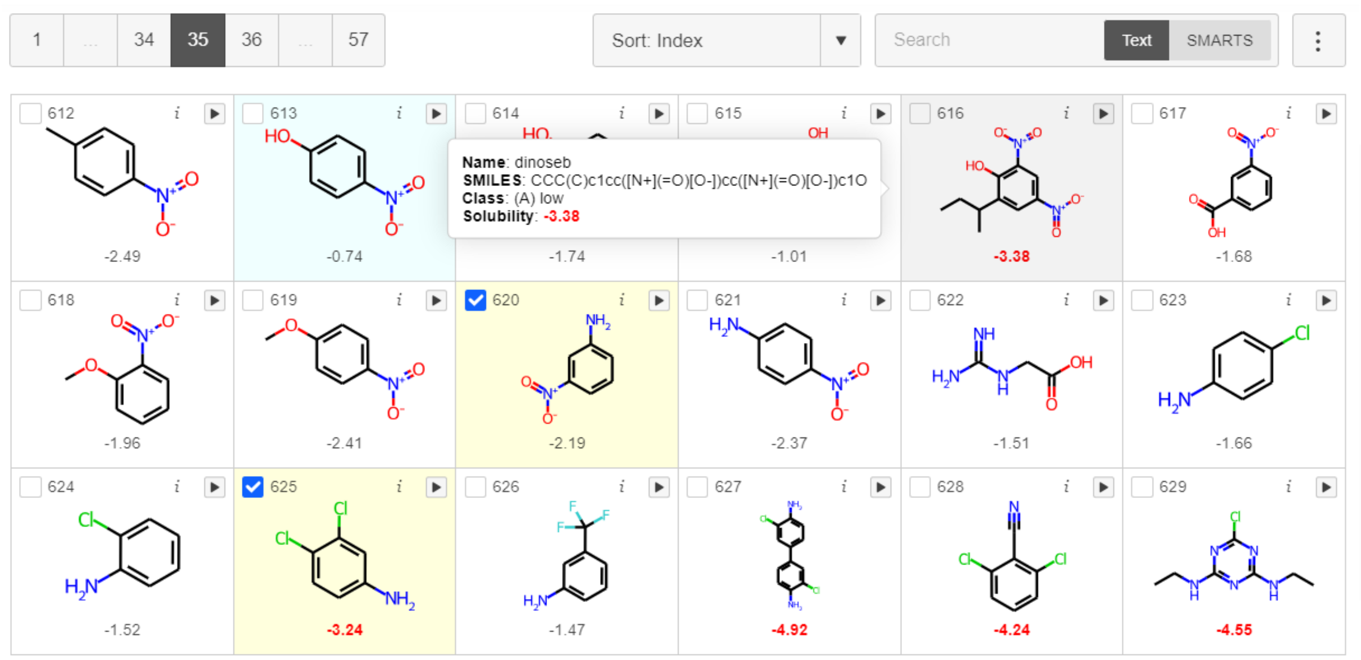
pred_logD	pred_logBCF	max_bcf_attr_perturb	max_lod_attr_perturb	logD-logBCF	Compound ID	Atom scores	Color domain
5.3	2.02	0.39	1.06	3.28	bromoxynil octanoate	logBCF perturb	[-0.5, 0, 0.5]

pred_logD	pred_logBCF	max_bcf_attr_perturb	max_lod_attr_perturb	logD-logBCF	Compound ID	Atom scores	Color domain
5.3	2.02	0.39	1.06	3.28	bromoxynil octanoate	logBCF perturb	[-0.5, 0, 0.5]

Example: https://github.com/Bayer-Group/xsmiles-jupyterlab/tree/main
https://bayer-group.github.io/xsmiles/dist/web/

Mol2grid

- https://mols2grid.readthedocs.io/en/latest/contents.html



Name: dinoseb
SMILES: CCC(C)c1cc([N+](=O)[O-])cc([N+](=O)[O-])c1O
Class: (A) low
Solubility: -3.38

- an SDFile

```
import mols2grid

mols2grid.display("path/to/molecules.sdf")
```

- a `pandas.DataFrame`:

```
mols2grid.display(df, smiles_col="Smiles")
```

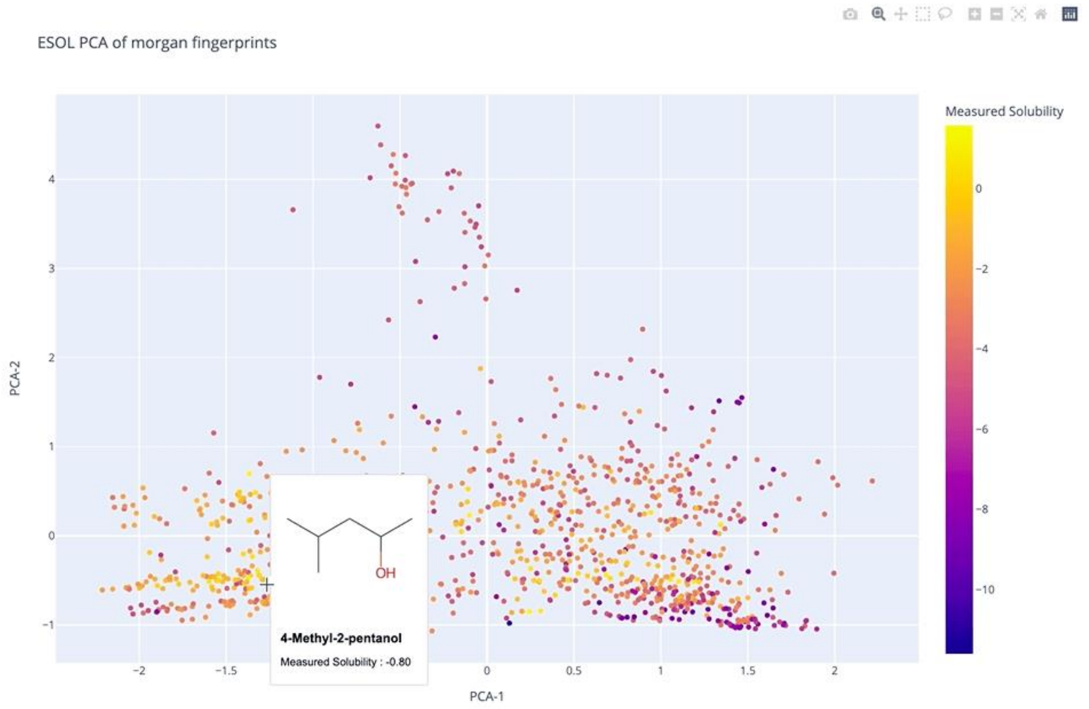
- a list of RDKit molecules:

```
mols2grid.display(mols)
```

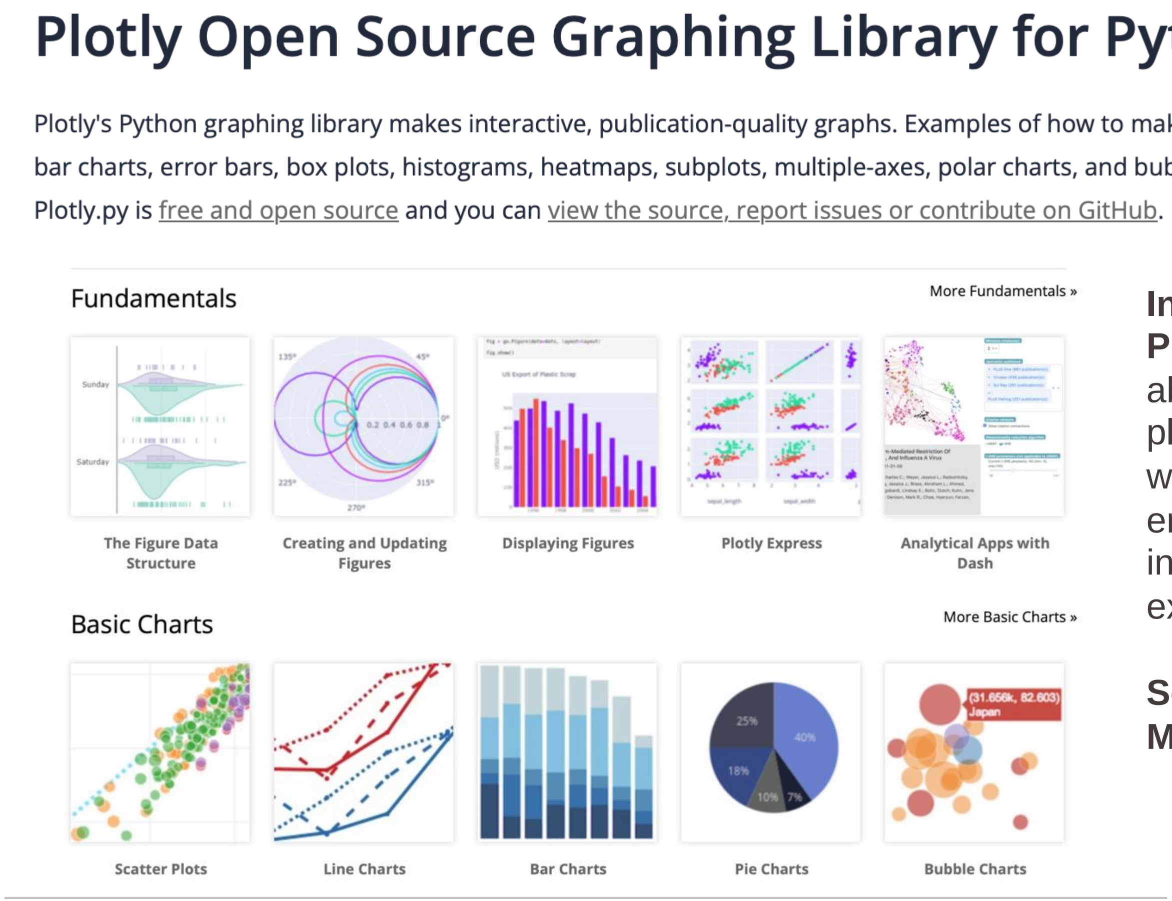
https://colab.research.google.com/github/rdkit/UGM_2021/blob/main/Notebooks/Bouysset_mols2grid.ipynb

molplotly

- https://github.com/wjm41/molplotly (plotly add on for molecules)



molplotly is based on plotly



Plotly Open Source Graphing Library for Python

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

Plotly.py is free and open source and you can view the source, report issues or contribute on GitHub.

Fundamentals

More Fundamentals »

The Figure Data Structure

Creating and Updating Figures

Displaying Figures

Plotly Express

Analytical Apps with Dash

Basic Charts

More Basic Charts »

Scatter Plots

Line Charts

Bar Charts

Pie Charts

Bubble Charts

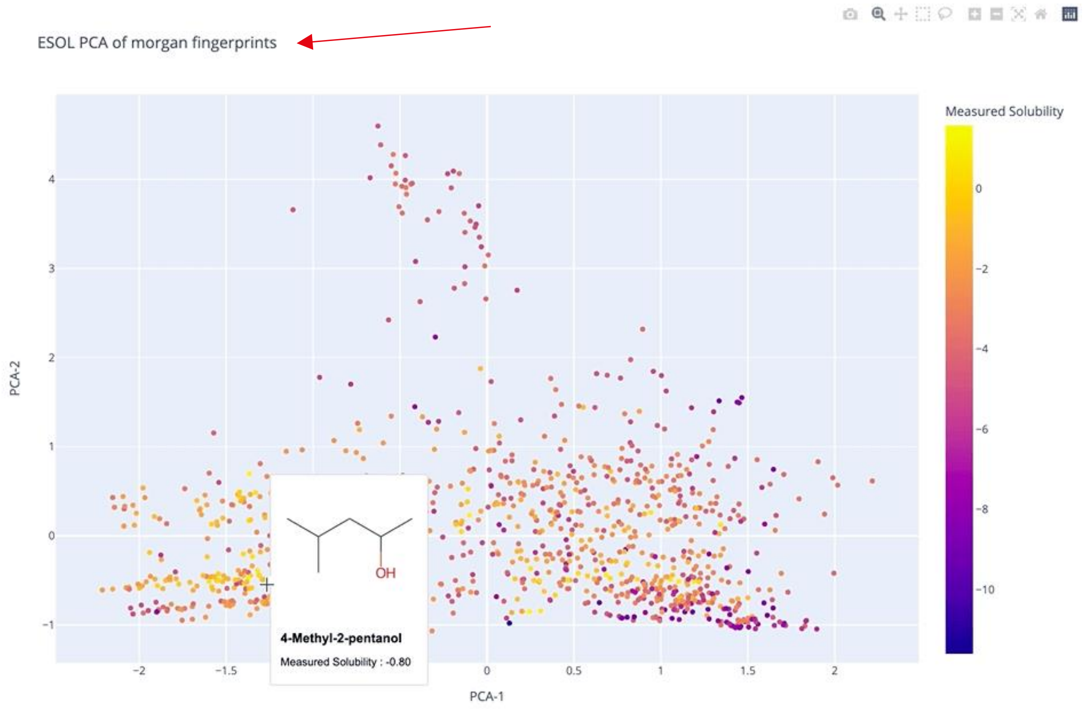
Interactive Visualizations: **Plotly** is best known for its ability to create interactive plots. This feature is handy for web applications or any environment where user interaction can enhance data exploration.

Seaborn -> aesthetics (static)
Matplotlib -> customizability

EPFL

molplotly

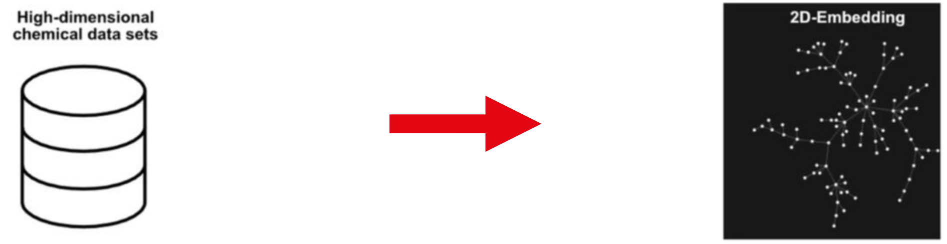
- https://github.com/wjm41/molplotly



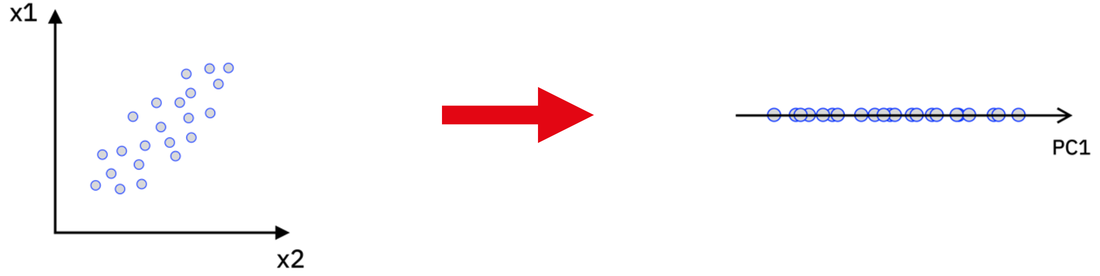
Dimensionality reduction

- high-dimensional space into a low-dimensional space
- How to display molecules described with a 1024-dimensional fingerprint (e.g., morgan fingerprint) in 2D?

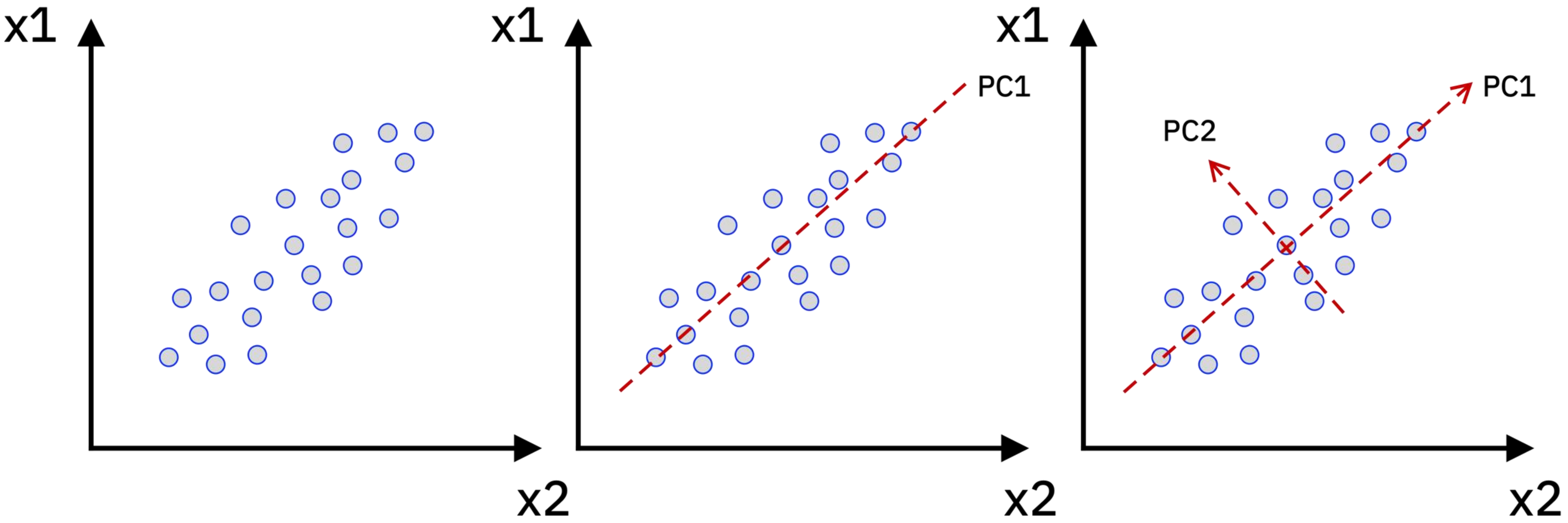
Reality:



Examples
(for simplicity)

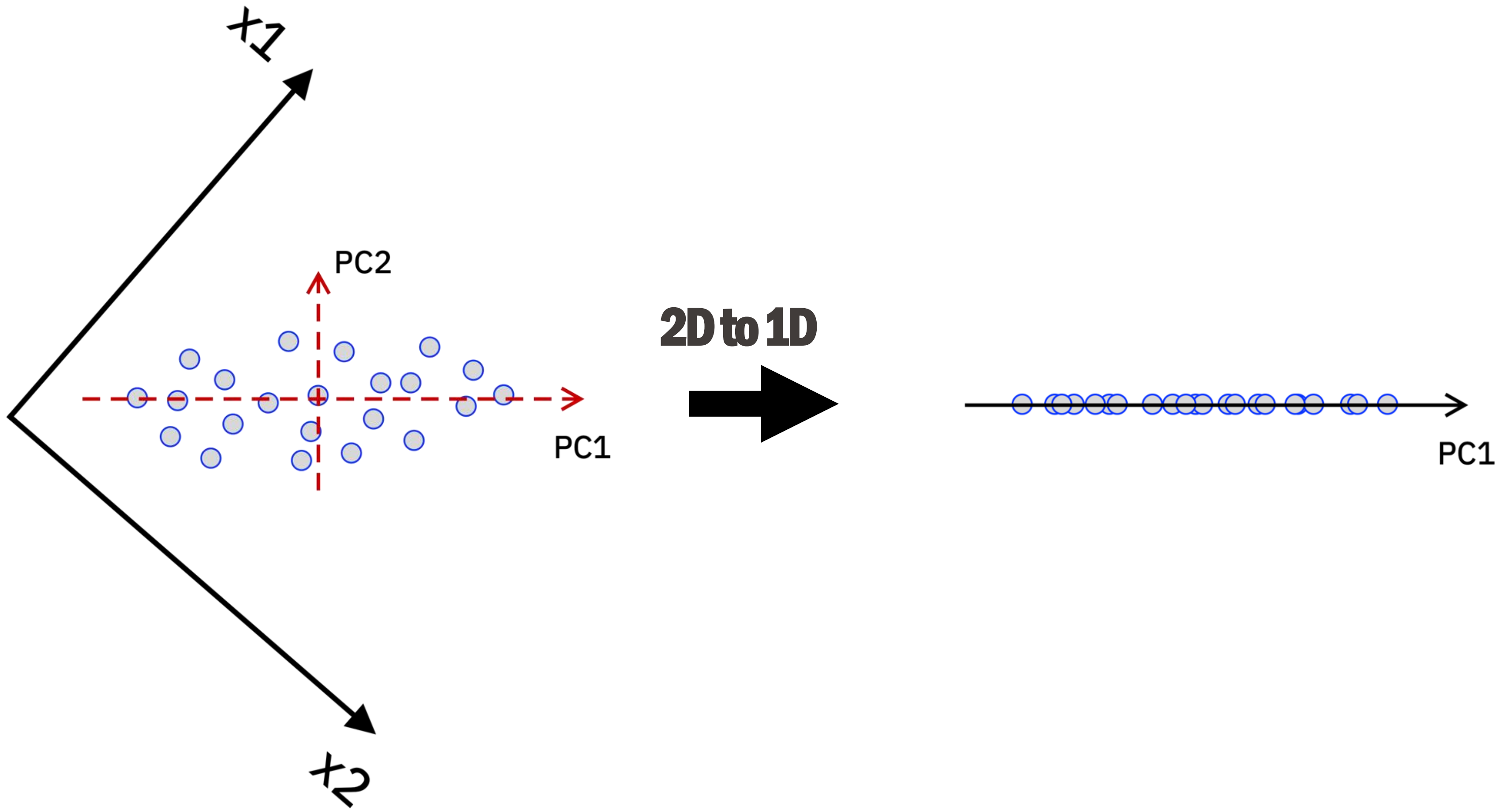


Principal Component Analysis (PCA)



- PCA finds directions of maximum variance
- principal components are orthogonal

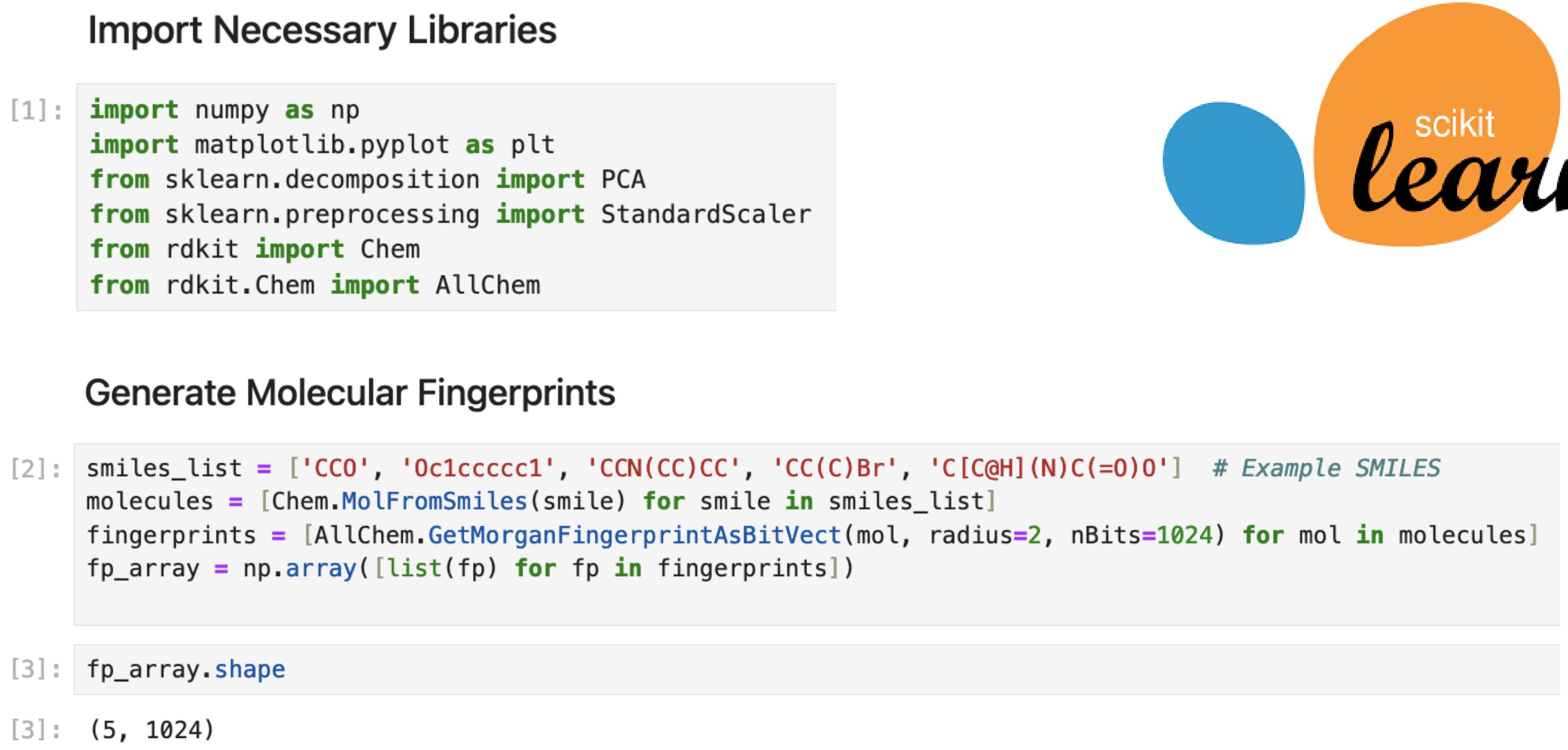
Principal Component Analysis (PCA)



EPFL

Code example

New library: scikit-learn



Import Necessary Libraries

```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.decomposition import PCA
     from sklearn.preprocessing import StandardScaler
     from rdkit import Chem
     from rdkit.Chem import AllChem
```

Generate Molecular Fingerprints

```
[2]: smiles_list = ['CCO', 'Oc1ccccc1', 'CCN(CC)CC', 'CC(C)Br', 'C[C@H](N)C(=O)O']  # Example SMILES
     molecules = [Chem.MolFromSmiles(smile) for smile in smiles_list]
     fingerprints = [AllChem.GetMorganFingerprintAsBitVect(mol, radius=2, nBits=1024) for mol in molecules]
     fp_array = np.array([list(fp) for fp in fingerprints])

[3]: fp_array.shape

[3]: (5, 1024)
```

Apply PCA and reduce the dimensionality to 2D

```
[11]: pca = PCA(n_components=2)  # Using 2 components for 2D visualization
      transformed_values = pca.fit_transform(data_scaled)
```

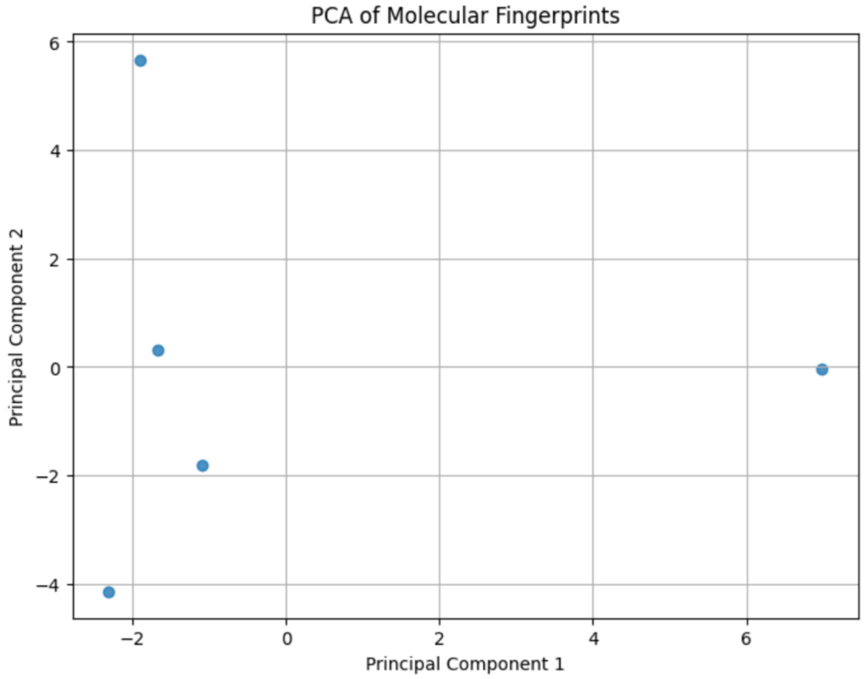
```
[12]: transformed_values
```

```
[12]: array([[-1.09952156, -1.80960721],
             [ 6.98993992, -0.04067157],
             [-2.31701547, -4.14022696],
             [-1.67530037,  0.324116  ],
             [-1.89810252,  5.66638974]])
```

You will do a better plot than that in the exercises ☺

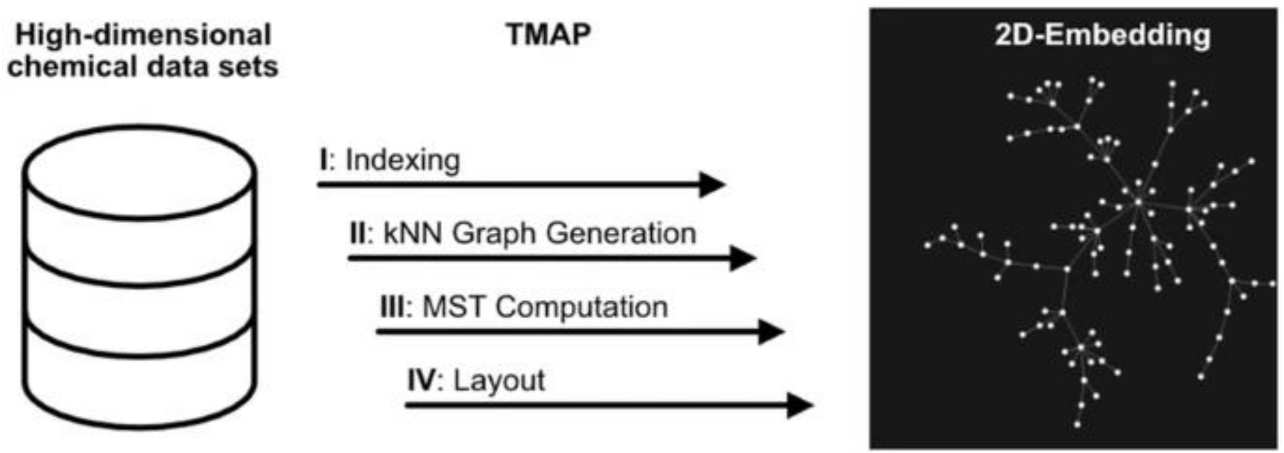
Visualization

```
[13]: plt.figure(figsize=(8, 6))
      plt.scatter(principalComponents[:, 0], principalComponents[:, 1], alpha=0.8)
      plt.xlabel('Principal Component 1')
      plt.ylabel('Principal Component 2')
      plt.title('PCA of Molecular Fingerprints')
      plt.grid(True)
      plt.show()
```



Other dimensionality reductions algorithms

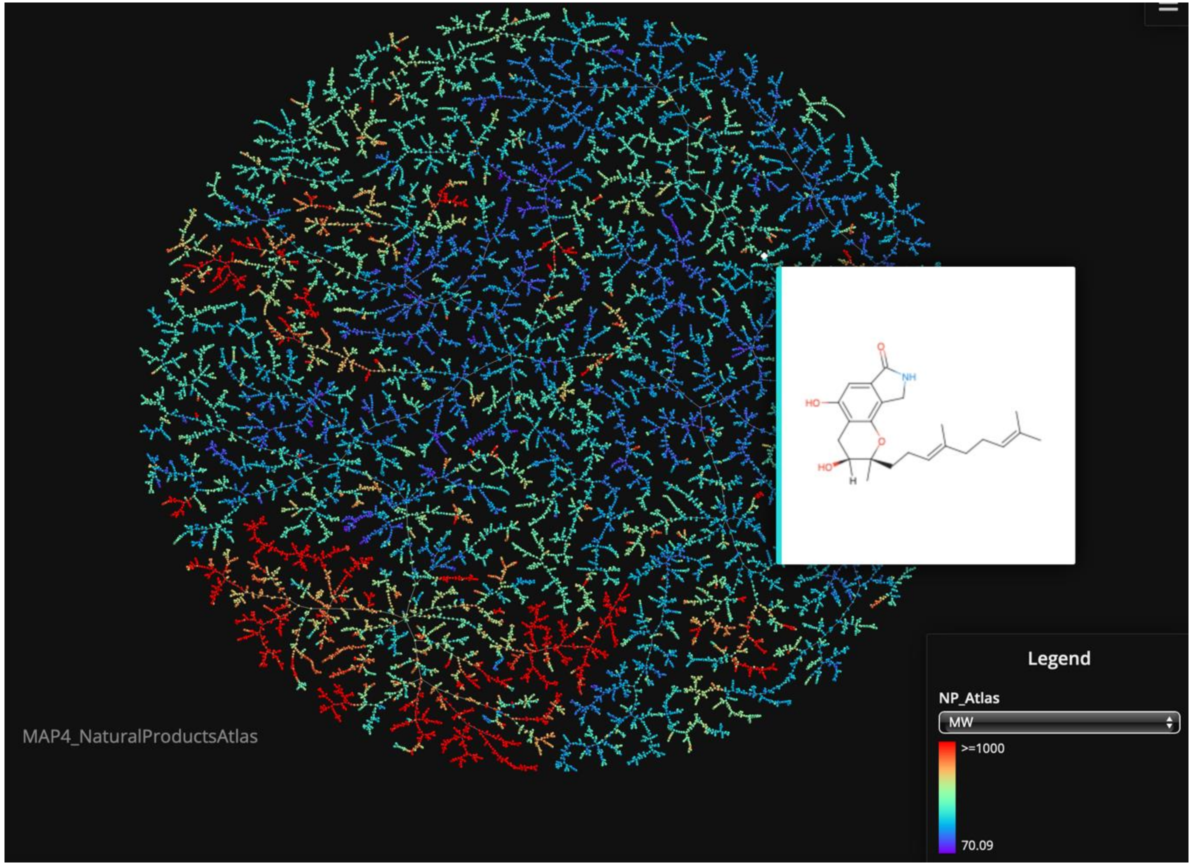
- PCA linearly projects data onto axes of maximum variance
- - **t-SNE (t-Distributed Stochastic Neighbor Embedding)** and **UMAP (Uniform Manifold Approximation and Projection)** are **advanced dimensionality reduction techni**ques that focus on *preserving local neighborhood* structures in the data, making them particularly effective for complex datasets with nonlinear relationships.
- **TMAP (tree map)** is another approach, that produces tree-like 2D maps.



Developed by
Daniel Probst
(at Reymond group)

https://tmap.gdb.tools

Example TMAP

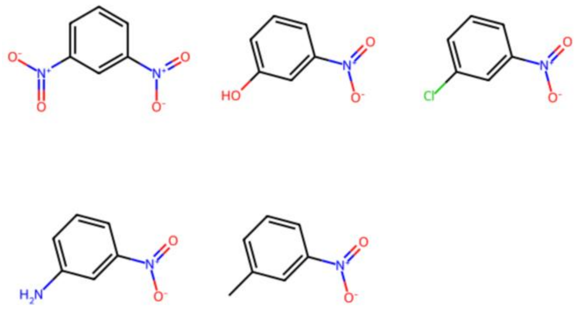


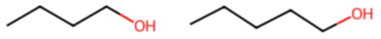
- https://github.com/reymond-group/MAP4-Chemical-Space-of-NPAtlas
  https://tm.gdb.tools/map4/npatlas_map_tmap/

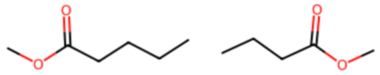
How do we find similar molecules in a dataset and assign group labels?
→ clustering

Clustering – what is it?

- Grouping unlabelled examples
- Typically you have a *large collection of molecules*, and you would like to divide them into *small groups of similar molecules* (clusters)
- This can help you *analyse outcomes* of high-throughput screening or virtual screening, but also *pick diverse molecules* (1 from each cluster)



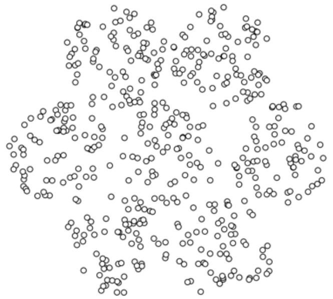


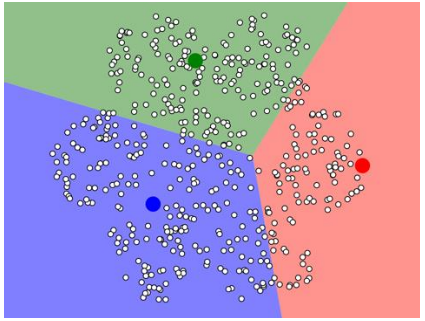


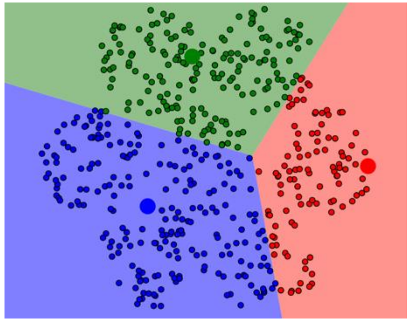
K-Means clustering (centroid-based)

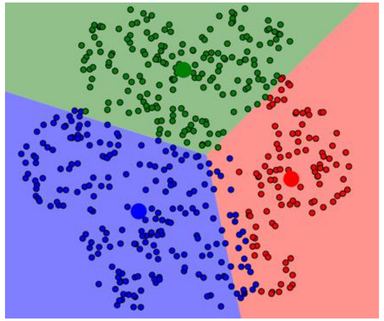
https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

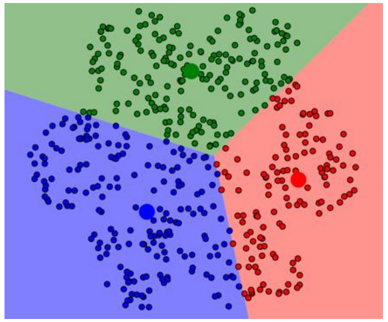
- Centroid == center of the cluster

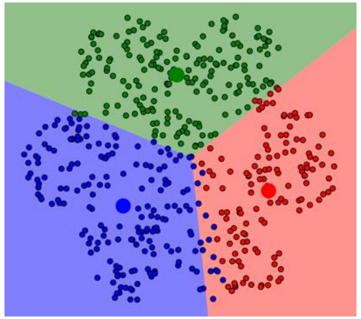

Initial data

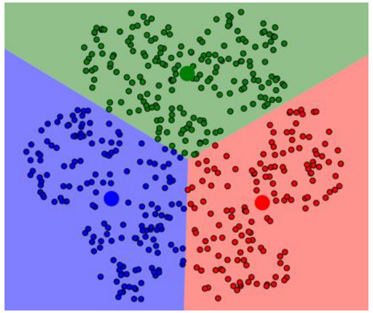

K randomly chosen centroids (C)


Assign points to Cs


Assign new Cs

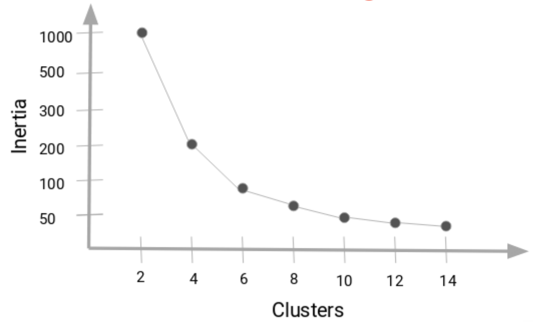

Reassign points to closest C

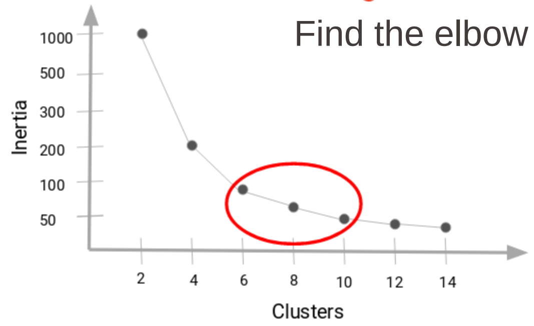

Reassign Cs


Iterate until centroids do not change anymore

How do you define K (the number of clusters)?

- A good model has low inertia and low K.

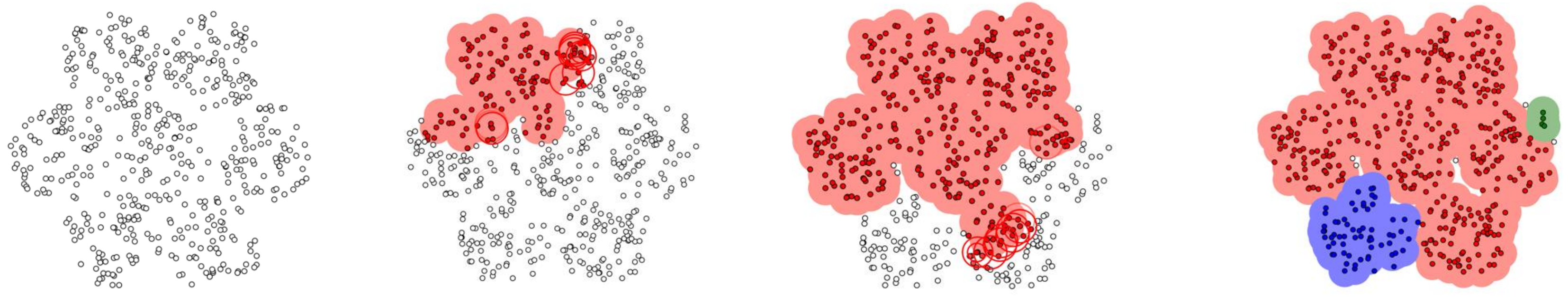




Find the elbow in the inertia plot

Inertia measures how well a dataset was clustered by K-Means.
It is calculated by measuring the **distance between each data point and its centroid**, squaring this distance, and summing these squares across one cluster.

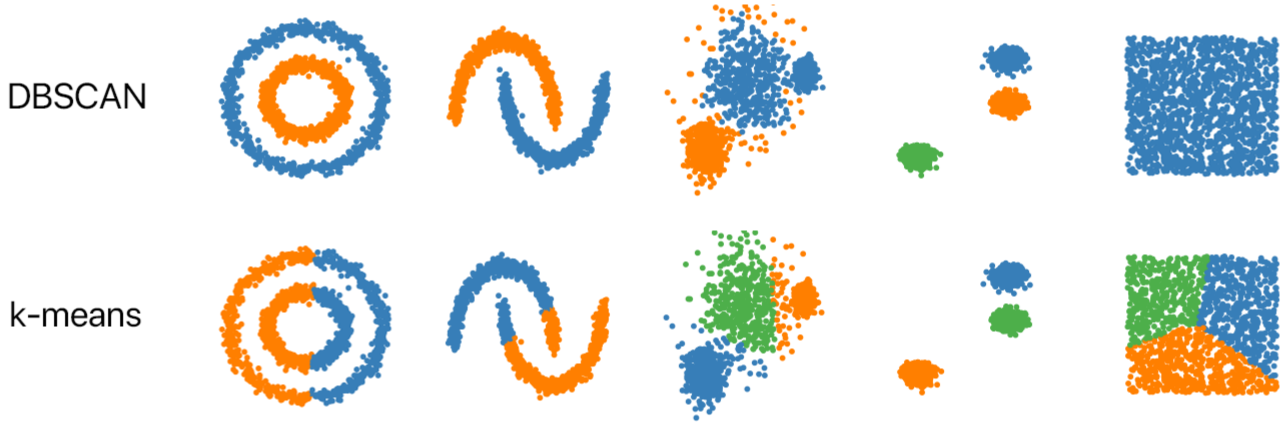
Density-based spatial clustering of applications with noise (DBSCAN)



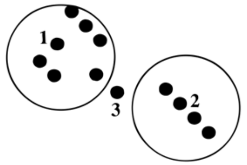
- Point belonging to cluster is near lots of points in that cluster
- Start by picking random point, min points in distance => add to cluster
- When no more point can be added, pick the next arbitrary point

Comparison between k-Means and DBSCAN

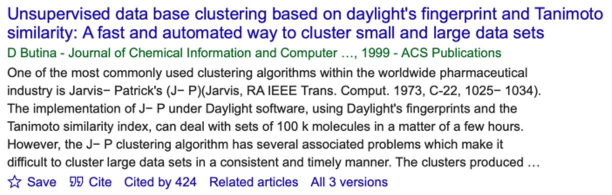
- Advantage: You don’t have to specify the number of clusters in advance in DBSCAN



Butina Clustering method (centroid-based, with Tanimoto similarity exclusion)



- Generation of Fingerprints.
- Identifying Potential Cluster Centroids (rank them by number of neighbors)
- Assign all molecules in the neighborhood to the centroid (within a given Tanimoto similarity)
- Repeat until no molecule with neighbors are left.


Unsupervised data base clustering based on daylight's fingerprint and Tanimoto similarity: A fast and automated way to cluster small and large data sets

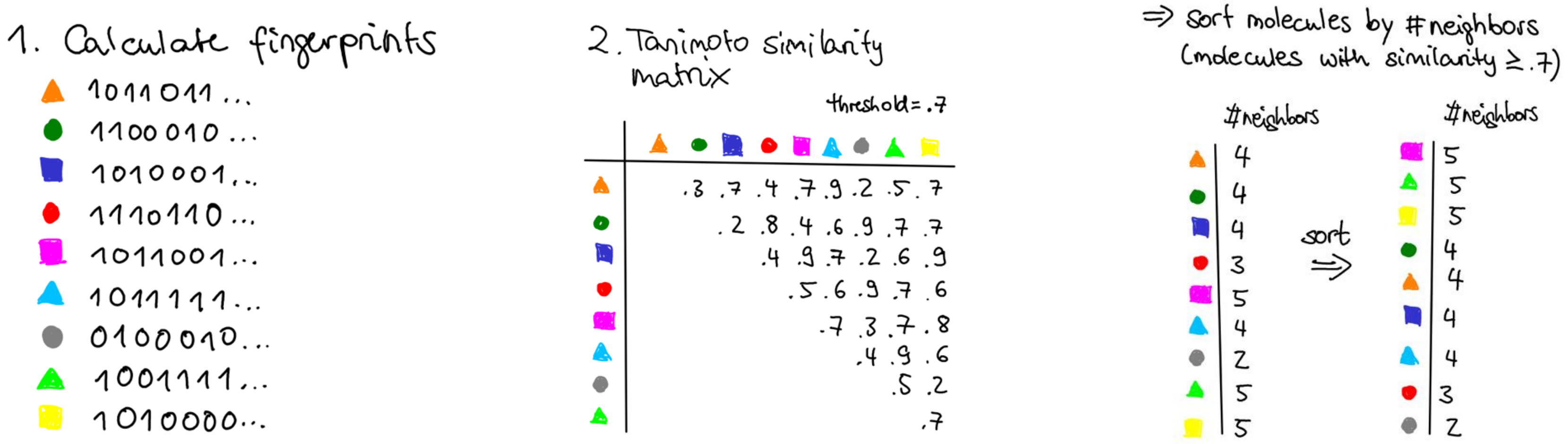
D Butina - Journal of Chemical Information and Computer …, 1999 - ACS Publications

One of the most commonly used clustering algorithms within the worldwide pharmaceutical industry is Jarvis– Patrick's (J– P)(Jarvis, RA IEEE Trans. Comput. 1973, C-22, 1025– 1034). The implementation of J– P under Daylight software, using Daylight's fingerprints and the Tanimoto similarity index, can deal with sets of 100 k molecules in a matter of a few hours. However, the J– P clustering algorithm has several associated problems which make it difficult to cluster large data sets in a consistent and timely manner. The clusters produced …

☆ Save 99 Cite Cited by 424 Related articles All 3 versions

- https://projects.volkamerlab.org/teachopencadd/talktorials/T005_compound_clustering.html

From TeachOpenCADD (Prof. Andrea Volkamer)



1. Calculate fingerprints

▲	1011011 ...
●	1100010 ...
■	1010001...
●	1110110 ...
■	1011001...
▲	1011111...
●	0100010...
▲	1001111...
■	1010000...

2. Tanimoto similarity matrix

threshold = .7

	▲	●	■	●	■	▲	●	▲	■
▲		.3	.7	.4	.7	.9	.2	.5	.7
●			.2	.8	.4	.6	.9	.7	.7
■				.4	.9	.7	.2	.6	.9
●					.5	.6	.9	.7	.6
■						.7	.3	.7	.8
▲							.4	.9	.6
●								.5	.2
▲									.7

⇒ sort molecules by #neighbors (molecules with similarity ≥ .7)

	#neighbors
▲	4
●	4
■	4
●	3
■	5
▲	4
●	2
▲	5
■	5

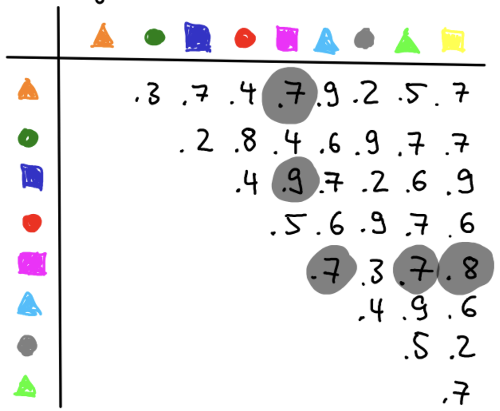
sort ⇒

	#neighbors
■	5
▲	5
■	5
●	4
▲	4
■	4
▲	4
●	3
●	2

- https://projects.volkamerlab.org/teachopencadd/talktorials/T005_compound_clustering.html

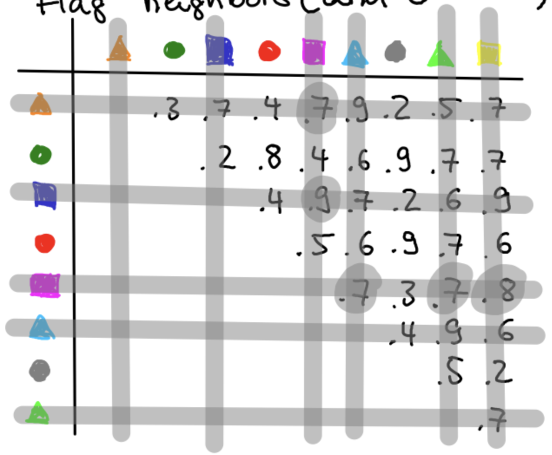
1. Iteration: molecule 🟪

Neighbors:



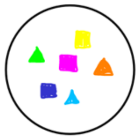
	🔺	🟢	🟦	🔴	🟪	🔹	⚫	▲	🟨
🔺		.3	.7	.4	.7	.9	.2	.5	.7
🟢			.2	.8	.4	.6	.9	.7	.7
🟦				.4	.9	.7	.2	.6	.9
🔴					.5	.6	.9	.7	.6
🟪						.7	.3	.7	.8
🔹							.4	.9	.6
⚫								.5	.2
▲									.7

Flag neighbors (and centroid):



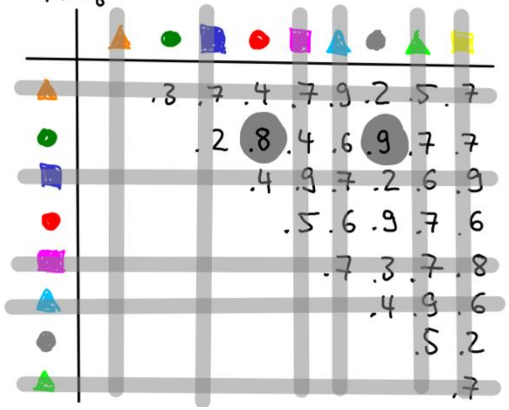
	🔺	🟢	🟦	🔴	🟪	🔹	⚫	▲	🟨
🔺		.3	.7	.4	.7	.9	.2	.5	.7
🟢			.2	.8	.4	.6	.9	.7	.7
🟦				.4	.9	.7	.2	.6	.9
🔴					.5	.6	.9	.7	.6
🟪						.7	.3	.7	.8
🔹							.4	.9	.6
⚫								.5	.2
▲									.7

Build new cluster:



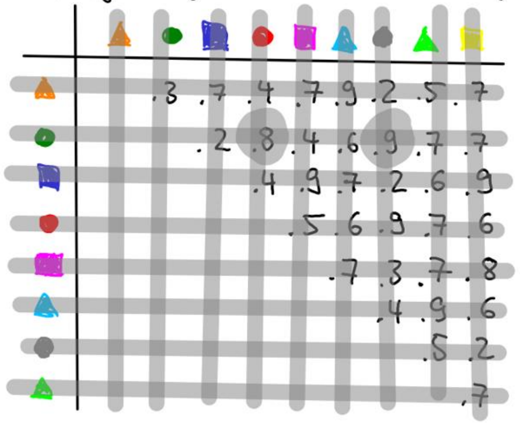
2. Iteration: molecule 🟢

Neighbors:



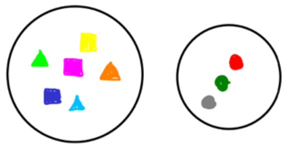
	🔺	🟢	🟦	🔴	🟪	🔹	⚫	▲	🟨
🔺		.3	.7	.4	.7	.9	.2	.5	.7
🟢			.2	.8	.4	.6	.9	.7	.7
🟦				.4	.9	.7	.2	.6	.9
🔴					.5	.6	.9	.7	.6
🟪						.7	.3	.7	.8
🔹							.4	.9	.6
⚫								.5	.2
▲									.7

Flag neighbors (and centroid):



	🔺	🟢	🟦	🔴	🟪	🔹	⚫	▲	🟨
🔺		.3	.7	.4	.7	.9	.2	.5	.7
🟢			.2	.8	.4	.6	.9	.7	.7
🟦				.4	.9	.7	.2	.6	.9
🔴					.5	.6	.9	.7	.6
🟪						.7	.3	.7	.8
🔹							.4	.9	.6
⚫								.5	.2
▲									.7

Build new cluster:



🟪	5
▲	5
🟨	5
🟢	4
🔺	4

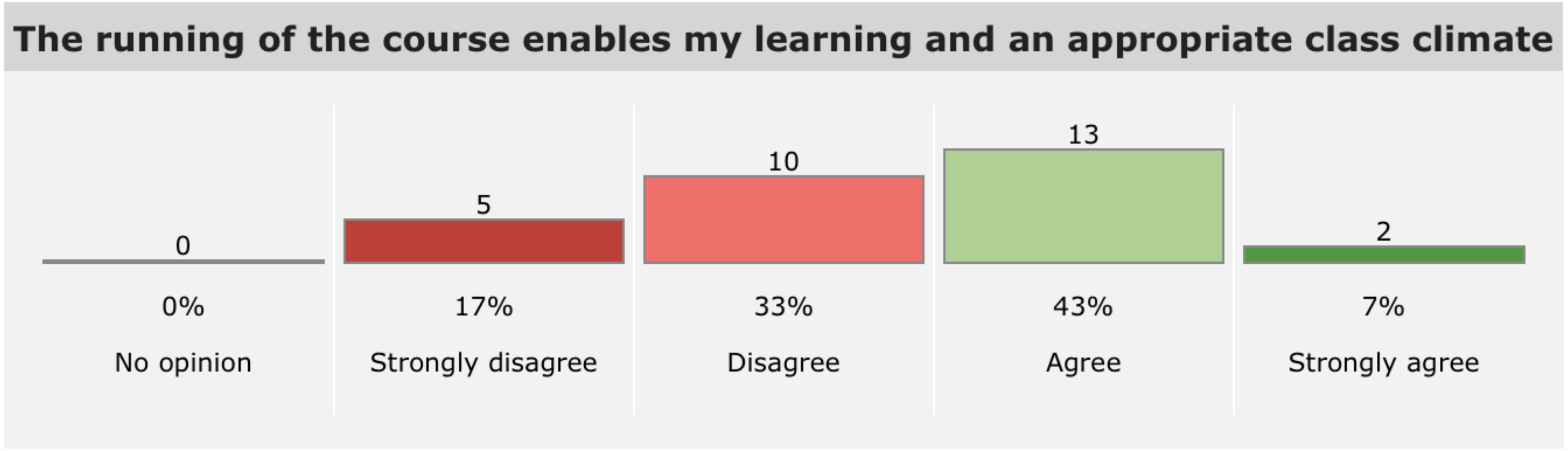
Termination, All molecules assigned or Singletons.

We have 26 groups. Thanks for signing up!

Final presentation slot:
~10% voted for not changing the slot, and finding a large enough room during the day is challenging (according to the Section).

Hence, we will keep the original slot.

Feedback on the course



At least some of you highlighted my enthusiasm for the course, and the helpfulness of the assistants.

Frequent points

- **Condensed Schedule:** The 7-week condensed format (instead of 14 weeks) has created significant pressure with 6 hours of programming per week
- **Exercise vs. Lecture Relationship:**
  - Some students find exercises **well-structured but disconnected** from lectures
  - Others note **redundancy** between lecture content and exercise materials
  - Many feel exercises contain **too much reading and not enough active coding**
- **Project Concerns:** Some anxiety about support during the upcoming project phase (→ sent an email to class delegate, we will organise a slot for office hours during the rest of the semester)

Improvements to be implemented

- **Exercise Format:** Shorter exercises with more hands-on coding and less explanatory text, inclusion of pointers to more advanced exercises (next year). Would it make sense to “collapse” the additional explanations?
- **Documentation:** Reference materials for commands and functions (last year: https://schwallergroup.github.io/practical-programming-in-chemistry/ → but students thought the information was too widespread across platforms | moodle, github, webpage)
- **Project Support:** Office hours in the remaining weeks. What about Thursday 1-2 pm?
- Any other spontaneous suggestions?

Today's exercises are about PCA, molplotly, and clustering.

Tomorrow, will be on making an app with Streamlit.

Next week (last lecture), no exercises, but time for the project.