

The EPFL logo is displayed in red, bold, sans-serif capital letters.The logo for HES-so Valais Wallis, featuring the text "Hes-so" in black and "VALAIS WALLIS" in a smaller font, with a stylized graphic of two slanted lines.An aerial photograph of the EPFL campus in Lausanne, Switzerland. The image shows modern university buildings, a large lake (Lac Léman) in the background, and distant mountains under a dramatic, cloudy sky. A large red rectangular box is overlaid on the right side of the image, containing the title text.

Tools for (computational) chemists

Dr. Simon Dürr

currently: PostDoc EPFL

from March: Assistant Professor UAS

HES-SO Valais-Wallis

Institut Technologies du vivant / Institute Life Technologies

- Why care?
- Do and Don't data processing
- Simulation data processing
- Research coding basics (Code editor, Linting, Formatting, Testing, Documentation, start a project, coding environments)
- Use AI productively for your research data workflows (Copilot, Cursor)
- Sharing code (Webapps, Marimo, Colab, packaging, Zenodo)

Why care?

Chemists generate a lot of data and we're required to share it in Switzerland

Unveiling a Single-Metal-Mediated Phosphodiester Bond Cleavage Mechanism for Nucleic Acids: A Multiscale Computational Investigation of a Human DNA Repair Enzyme

46
views

34
shares

0
downloads

Unveiling a Single-Metal-Mediated Phosphodiester Bond Cleavage Mechanism for Nucleic Acids: A Multiscale Computational Investigation of a Human DNA Repair Enzyme

Mohamed M. Aboelnga and Stacey D. Wetmore*

Department of Chemistry and Biochemistry, University of Lethbridge, 4401 University Drive West, Lethbridge, Alberta, Canada T1K 3M4

Supporting Information

(1185 pages)

```
O  0 -5.10485 -0.16413 1.95494
H  0 -6.78555 -1.35754 -0.78289
H  0 -5.81577 0.92343 -1.05095
H  0 -6.57024 1.4367 0.47924
```

524

```
H  0 -3.77521 0.19906 -0.88976
H  0 -2.95436 -0.32309 0.56513
C  1 0.00000 0.00000 0.00000
```

[RETURN TO ISSUE](#) | [EDITORIAL](#) | [NEXT >](#)

Guidelines for Reporting Molecular Dynamics Simulations in JCIM Publications

Thereza A. Soares*, Zoe Cournia, Kevin Naidoo, Rommie Amaro, Habibah Wahab, and Kenneth M. Merz Jr.

✓ **Cite this:** *J. Chem. Inf. Model.* 2023, 63, 11, 3227–3229

Publication Date: May 16, 2023 ▾

<https://doi.org/10.1021/acs.jcim.3c00599>

Copyright © Published 2023 by American Chemical Society

[Request reuse permissions](#)

Subscribed

Article Views

10138

Altmetric

66

Citations

2

[LEARN ABOUT THESE METRICS](#)

Input/output files and data availability

ARTICLE SECTIONS

Jump To ▾

JCIM now requires depositing your input files (including topologies, parameters, input files, and initial configurations) and output trajectory files in a public repository providing a DOI and provide the accession link in the manuscript. Some available repositories are Zenodo (<https://zenodo.org/>), Dryad (<https://datadryad.org/>), Nomad (<https://nomad-repository.eu/>), and Figshare (<https://figshare.com/>). If the full trajectories are too large, a selection of clustered structure representing the respective trajectories should be made available in the public repository.

Real problems from my own PhD

- People not sharing their data/models even after inquiring multiple times
- Compiled binary only for 32bit systems and environment not specified

Table 1. Computational tools for metal-binding site and metalloprotein prediction.

Name	Website	Related Metals	Main Algorithm	Reported Performance	Ref.
RDBG	http://www.cerm.unifi.it/home/research/genomebrowsing.html	Zn, Cu, Fe and other metals	Integration of tools for retrieval of protein domains and genome analysis	Accuracy: 89.6%, precision: 85.9%	[32]
Zincfinder	http://zincfinder.dsl.unifi.it	Zn	^a SVM	^b AURPC: 0.590 (local predictor) and 0.633 (gating network)	[33]
Zincpred	http://www.fos.su.se/~nanjiang/zincpred/download/	Zn	SVM- and homology-based algorithm	AURPC: 0.723 (local predictor) and 0.701 (gating network)	[34]
TEMSP	http://metalign.ustc.edu.cn/temsp/	Zn	Structure-based algorithm with a range of geometric criteria	^c AUC: 0.945	[35]
ZincIdentifier	http://protein.cau.edu.cn/zincIdentifier/	Zn	A two-step feature selection method based on random forest algorithm	AUC: 0.955, AURPC: 0.829	[36]
ZincExplorer	http://protein.cau.edu.cn/ZincExplorer/	Zn	A combination of SVM-, cluster- and template-based predictors	AURPC: 0.907	[37]
ZincBinder	http://proteininformatics.org/mkumar/znbinder/	Zn	SVM model trained on PSSM-based input feature	AUC: 0.91	[38]
ZINCLUSTER	http://www.metalactive.in	Zn	SVM-based Ligand Finder and Cluster Finder algorithms	^d MCC: 0.798, F1-score: 0.801	[39]
ZnMachine	http://bioinformatics.fzu.edu.cn/znMachine.html	Zn	A combination of several intensively-trained machine learning models	AUC: 0.933 (SVM) and 0.910 (neural network)	[40]
HemeBIND	http://mleg.cse.sc.edu/hemeBIND/	Fe (heme)	SVM	MCC: 0.504, F1-score: 56.87%	[41]
SCMHBP	http://iclab.life.nctu.edu.tw/SCMHBP/	Fe (heme)	Based on a newly-developed scoring card method for predicting heme-binding proteins	Accuracy: 85.90%	[42]
Isph	http://biodev.extra.ces.fr/isph	Fe (Fe-S)	A penalized linear model based on machine learning approach	Precision: 87.9%, recall: 80.1% (extended model)	[43]
MetalPredator	http://metalweb.cerm.unifi.it/tools/metalpredator/	Fe (Fe-S)	Integration of existing domain-based methodology with a new approach for discovering metal-binding motifs	Precision: 85.2%, recall: 88.6%	[44]
MetSite	N/A	Fe, Zn, Cu, Mn, Ca, Mg	Artificial neural network	Mean accuracy: 94.5%	[45]
FINDSITE-metal	http://cssb.biology.gatech.edu/findsite-metal/	Fe, Zn, Cu, Mn, Ni, Co, Ca, Mg	Integration of structure/evolutionary information and machine learning approach (SVM)	Overall accuracy: 70–90%	[46]
SeqCHED	http://igin.weizmann.ac.il/seqched	Fe, Zn, Cu, Mn, Ni, Co, Ca, Mg	A modification of the CHED algorithm and machine learning filters (decision tree classifier and SVM)	Sensitivity: 84–85%, selectivity: 82–93% (stringent filtration)	[47]
MetalDetector	http://metaldetector.dsl.unifi.it/v2.0/	Transition metals that use cysteine and histidine as ligands	A combination of different machine learning algorithms (SVM-HMM, structured-output SVM)	Precision: 60–79%, recall: 71–88%	[48]
MB	http://bioinfo.cmu.edu.tw/MB/	Ca, Cu, Fe, Mg, Mn, Zn, Cd, Ni, Hg, Co	Fragment transformation method	Overall accuracy: 92.9–95.1%	[49]

review from
2022

Table 1. Computational tools for metal-binding site and metalloprotein prediction.

Name	Website	Related Metals	Main Algorithm	Reported Performance	Ref.
RDGB	http://www.cerm.unifi.it/home/research/genomics/rdgb.htm	Zn, Cu, Fe and other metals	Integration of tools for retrieval of protein domains and genome analysis	Accuracy: 89.6%, precision: 85.9%	[32]
Zincfinder	http://zincfinder.dsi.unifi.it	Zn	^a SVM	^b AURPC: 0.590 (local predictor) and 0.633 (gating network)	[33]
Zincpred	http://www.tos.su.se/~nandje/gatingpred/download/	Zn	SVM- and homology-based algorithm	AURPC: 0.723 (local predictor) and 0.701 (gating network)	[34]
TEMSP	http://metalgn.usc.edu.cn/temsp/	Zn	Structure-based algorithm with a range of geometric criteria	^c AUC: 0.945	[35]
ZincIdentifier	http://protein.cau.edu.cn/zincidentifier/	Zn	A two-step feature selection method based on random forest algorithm	AUC: 0.955, AURPC: 0.629	[36]
ZincExplorer	http://protein.cau.edu.cn/ZincExplorer/	Zn	A combination of SVM-, cluster- and template-based predictors	AURPC: 0.907	[37]
ZincBinder	http://proteininformatics.org/linkum/zincbinder/	Zn	SVM model trained on PSSM-based input feature	AUC: 0.91	[38]
ZINCCLUSTER	http://www.metalactive.in	Zn	SVM-based Ligand Finder and Cluster Finder algorithms	^d MCC: 0.766, F1-score: 0.601	[39]
ZnMachine	http://bioinformatics.fcu.edu.cn/znmachine.html	Zn	A combination of several intensively-trained machine learning models	AUC: 0.933 (SVM) and 0.910 (neural network)	[40]
HemeBIND	http://mleg.cse.sc.edu/hemeBIND/	Fe (heme)	SVM	MCC: 0.504, F1-score: 56.87%	[41]
SCMHBP	http://lelab.life.uctu.edu.tw/SCMHBP/	Fe (heme)	Based on a newly-developed scoring card method for predicting heme-binding proteins	Accuracy: 85.90%	[42]
Isph	http://medev.extra.cba.tmsph	Fe (Fe-S)	A penalized linear model based on machine learning approach	Precision: 87.9%, recall: 80.1% (extended model)	[43]
MetalPredator	http://metabash.cerm.unifi.it/lelab/metalpredator/	Fe (Fe-S)	Integration of existing domain-based methodology with a new approach for discovering metal-binding motifs	Precision: 85.2%, recall: 88.6%	[44]
MetSite	N/A	Fe, Zn, Cu, Mn, Ca, Mg	Artificial neural network	Mean accuracy: 94.5%	[45]
FINDSITE-metal	http://icsh.biology.gatech.edu/findsite-metal/	Fe, Zn, Cu, Mn, Ni, Co, Ca, Mg	Integration of structure/evolutionary information and machine learning approach (SVM)	Overall accuracy: 70–90%	[46]
SeqCHED	http://ligin.wisconsin.edu/seqched	Fe, Zn, Cu, Mn, Ni, Co, Ca, Mg	A modification of the CHED algorithm and machine learning filters (decision tree classifier and SVM)	Sensitivity: 84–85%, selectivity: 82–93% (stringent filtration)	[47]
MetalDetector	http://metaldetector.dsi.unifi.it/v2.0/	Transition metals that use cysteine and histidine as ligands	A combination of different machine learning algorithms (SVM-HMM, structured-output SVM)	Precision: 60–79%, recall: 71–88%	[48]
MIB	this version no longer available http://bioinfo.cmu.edu.tw/MIB/	Ca, Cu, Fe, Mg, Mn, Zn, Cd, Ni, Hg, Co	Fragment transformation method	Overall accuracy: 92.9–95.1%	[49]

Do and Don't data processing

No manual file editing

No manual file copying

Clear track record from raw data to plots/tables (e.g via scripting)

Go from Python to Microsoft Word

https://rowannicholls.github.io/python/data/export_to_word.html

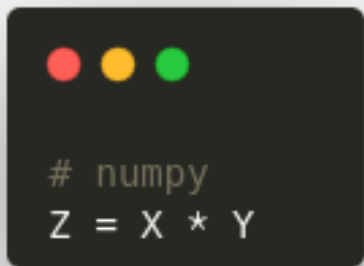


Programming language



Simplicity is key

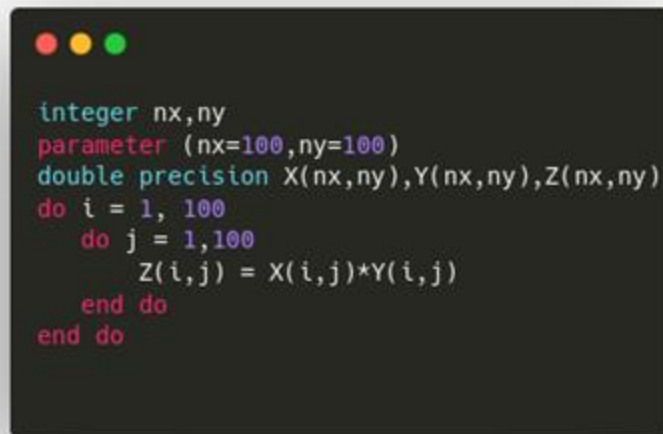
Multiply 2d arrays of size (100,100)



```
# numpy  
Z = X * Y
```



almost
as fast



```
integer nx,ny  
parameter (nx=100,ny=100)  
double precision X(nx,ny),Y(nx,ny),Z(nx,ny)  
do i = 1, 100  
  do j = 1,100  
    Z(i,j) = X(i,j)*Y(i,j)  
  end do  
end do
```



fast

Simulation data processing

Running a lot of computations?

Use workflow tools to keep track:

Computational Chemistry

- AIIDA
- atomate2/jobflow

Machine Learning

- Weights and Biases

Old school

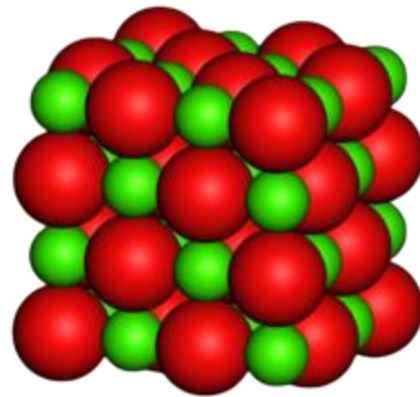
Create structure in Avogadro or GaussView

Run `pw.x < MGO.opt.in > MGO.opt.out`

Extract energies `grep ! MGO.opt.out > energies.dat`

Plot them using `gnuplot` by entering the commands

`> plot 'energies.dat' using 1:2 with lines`



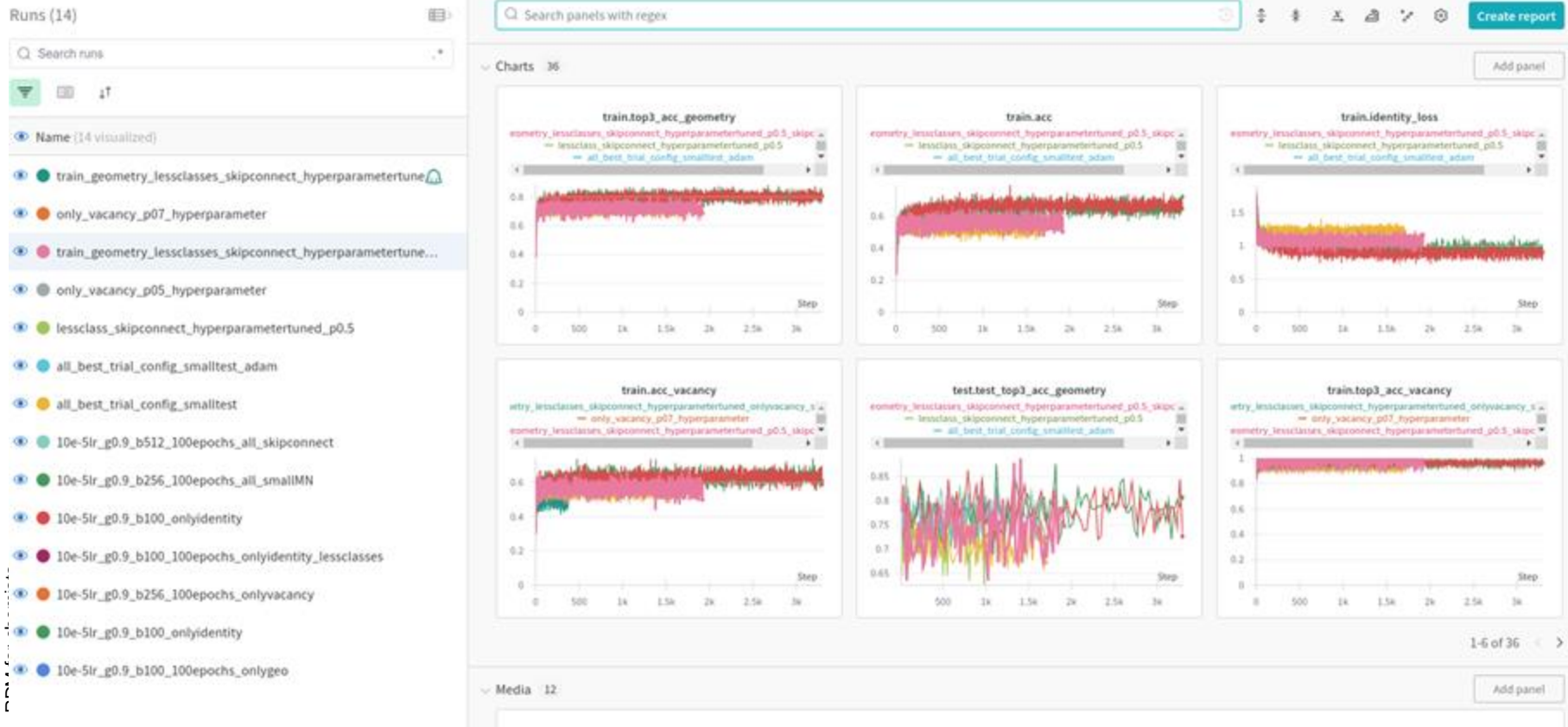
struct_opt_band_calc.py

```
from atomate2.vasp.flows.core import RelaxBandStructureMaker
from jobflow import run_locally
from pymatgen.core import Structure

# construct a rock salt MgO structure
mgo_structure = Structure(
    lattice=[[0, 2.13, 2.13], [2.13, 0, 2.13], [2.13, 2.13, 0]],
    species=["Mg", "O"],
    coords=[[0, 0, 0], [0.5, 0.5, 0.5]],
)

# make a band structure flow to optimise the structure and obtain the band structure
bandstructure_flow = RelaxBandStructureMaker().make(mgo_structure)

# run the flow
run_locally(bandstructure_flow, create_folders=True)
```



Prefer scriptable tools to GUIs (less automatable)

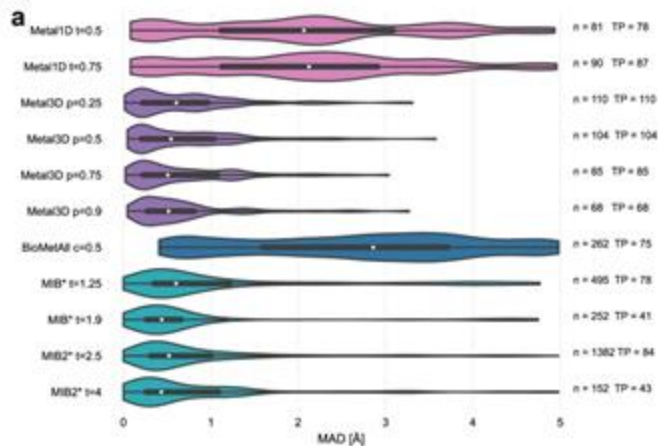
Prefer open software to closed software

Make everything explicit instead of running single commands

Chances that you have to redo something multiple times are high.

Use automated scripts as much as possible to generate plots directly

Fig. 4: Mean absolute deviation of correctly predicted sites and selectivity for other ions.



lcgc-epfl/metal-site-prediction-v0.2.zip

supporting_data.zip

! The previewer is not showing all the files.

supporting_data

Figure2

2CBA_nometal.pdb

2cba_paper.pdb

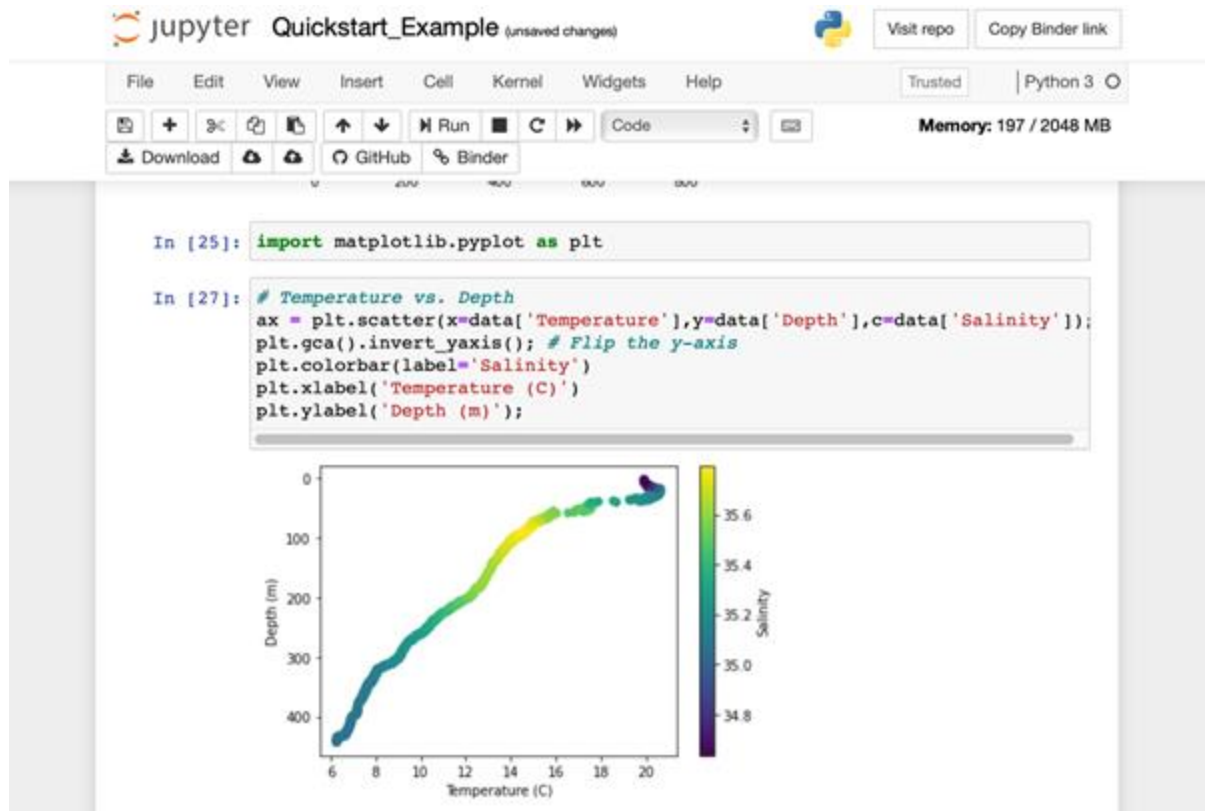
metal_2cba_paper.cube

Figure3A_4A_S4_TableS3_S4

Figure3B

All scripts to
reproduce
figure exactly
is there

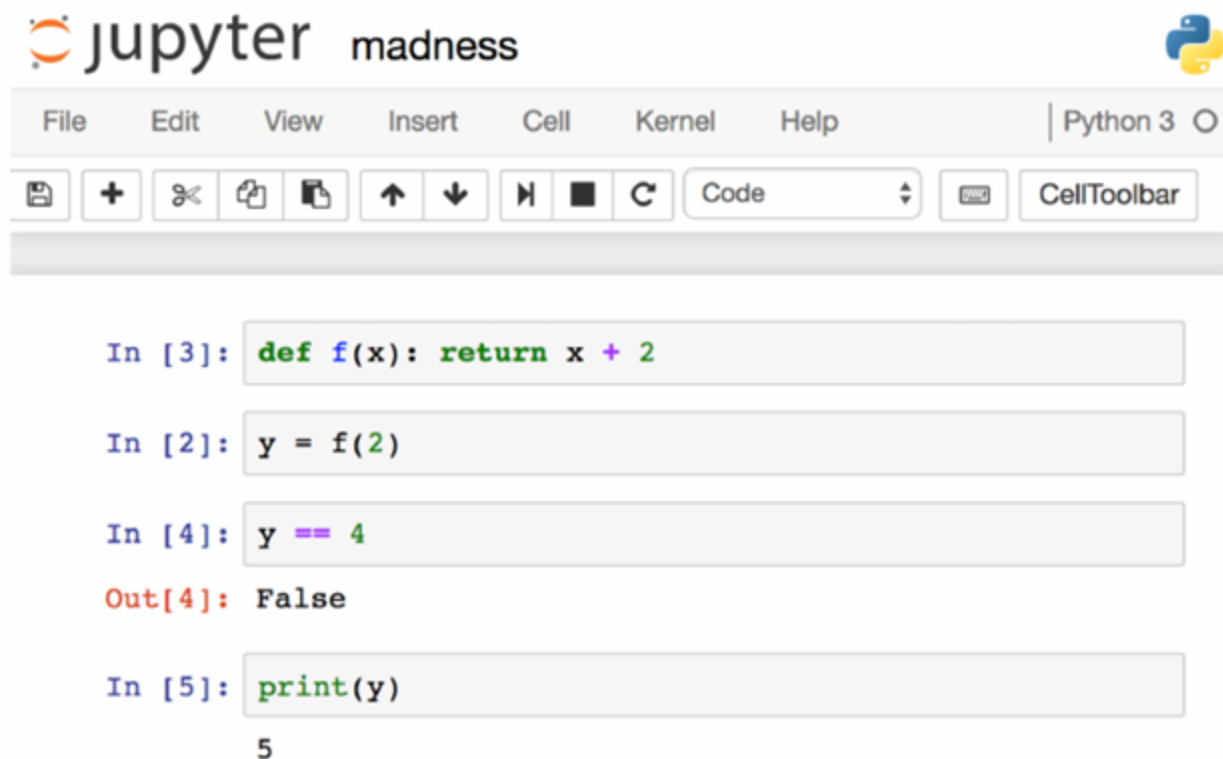
Let's talk about Jupyter Notebooks





UNTITLED25.IPYNB

UNTITLED24.IPYNB



jupyter madness

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

```
In [3]: def f(x): return x + 2
```

```
In [2]: y = f(2)
```

```
In [4]: y == 4
```

Out[4]: False

```
In [5]: print(y)
```

5

if you look at the numbers, it's clear those cells weren't even executed in order!



@joelgrus #jupytercon



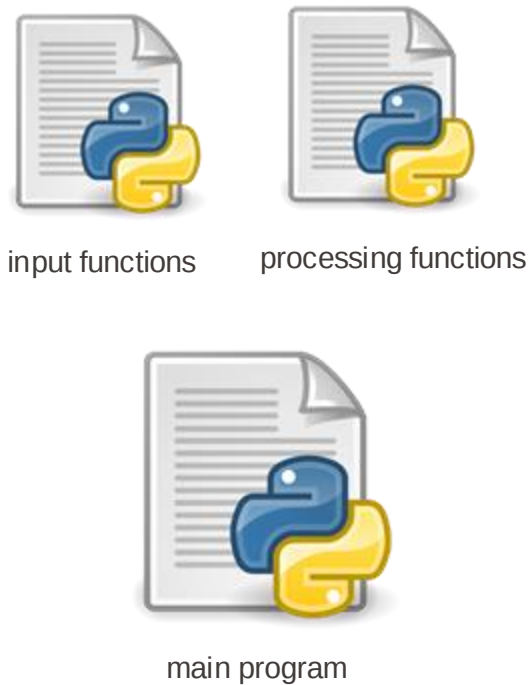
Notebook

vs.



Script

vs.



Code

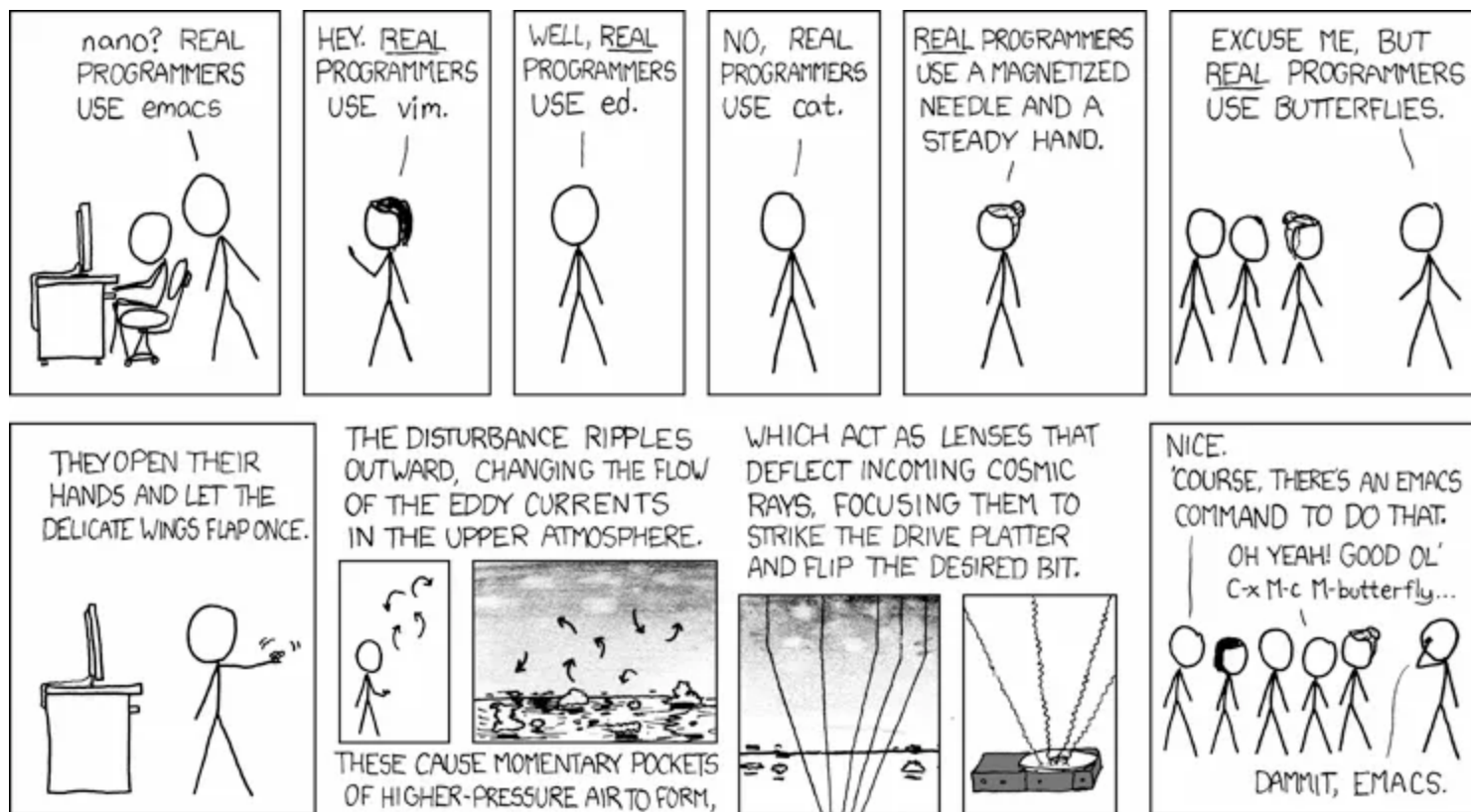
```
1 import marimo as mo
```



```
1 x = mo.ui.slider(1, 9)
2 x
```

$$e^1 = 2.718$$

```
1 mo.md(f"$e^{x.value} = {math.exp(x.value):0.3f}$")
```



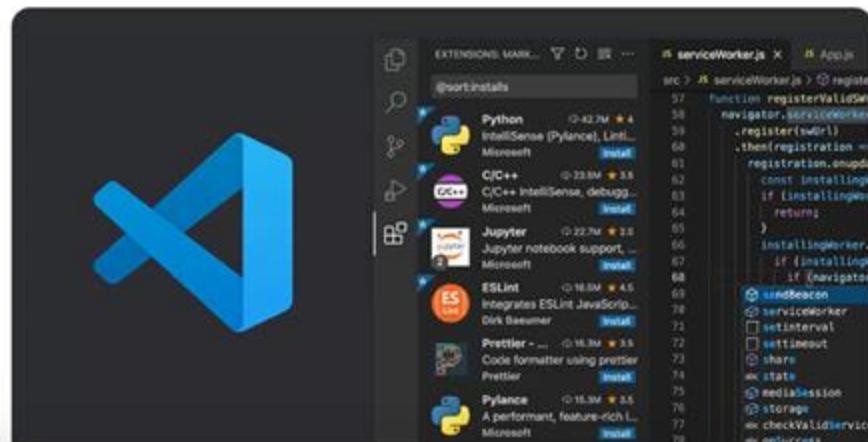
Many choices but if you don't have a strong preference for an editor:

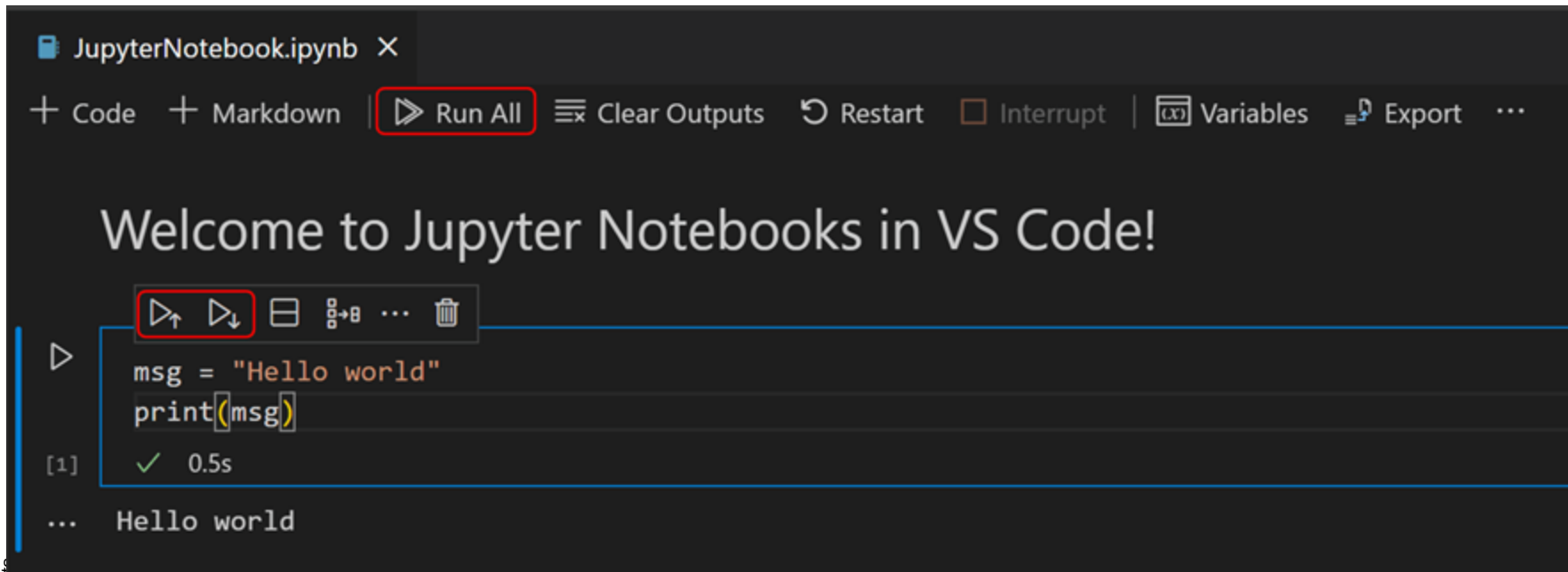
VS Code offers:

- the right abstractions,
- version control,
- inbuilt terminal,
- linting,
- formatting,
- AI completion,
- Jupyter support and
- remote capabilities

Why VS Code remains a developer favorite, year after year

 Anastasiya Uspenski





Syntax Highlighting, Linting, Formatting

Tools for coding

- Linting
- Formatting
- Code environments
- Version control
- Continuous integration
- Documentation

The screenshot shows a code editor with a file named `hello.py`. The code contains the following lines:

```
1 msg = "Hello, Python"
2
3 print msg
4
```

A tooltip is displayed over the `print` statement on line 3, showing the function signature and documentation for `print`:

```
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
Prints the values to a stream, or to sys.stdout by default. Optional keyword arguments:
file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
```

Below the documentation, the tooltip displays a parsing error:

```
Parsing failed: 'Missing parentheses in call to 'print'. Did you mean
print(...)? (<unknown>, line 3)' Pylint(E0001:syntax-error)
```

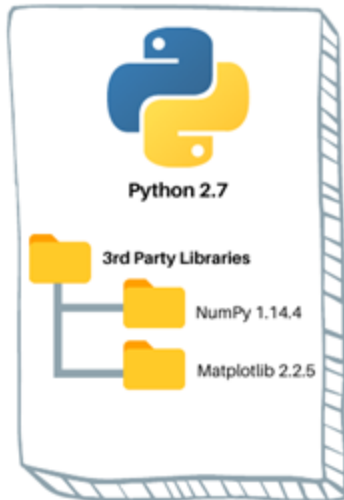
At the bottom of the tooltip, there is a link to [View Problem \(Alt+F8\)](#) and the text `No quick fixes available`.

The bottom of the screenshot shows the **PROBLEMS** panel with two errors listed:

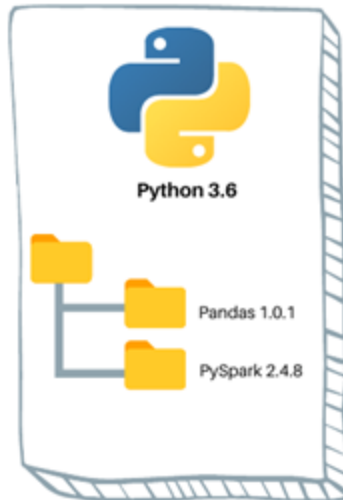
- ✖ Parsing failed: 'Missing parentheses in call to 'print'. Did you mean print(...)? (<unknown>, line 3)' Pylint(E0001:syntax-error) [Ln 3, Col 2]
- ✖ Statements must be separated by newlines or semicolons Pylance. [Ln 3, Col 7]

Do not install everything in one big environment

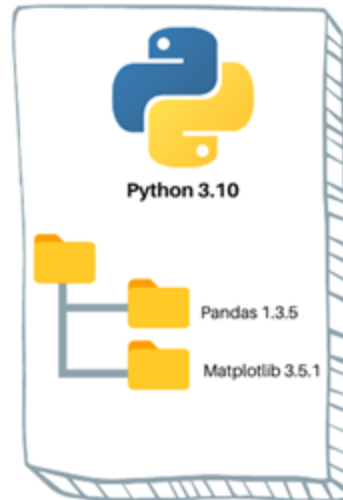
Virtual Environment 1



Virtual Environment 2



Virtual Environment 3



Different options:

- venv/pip/uv
 - smaller/lightweight
 - cannot change python version
 - only python packages from pypi
- conda/mamba
 - change python version
 - heavier/slower
 - install also other scientific software

	conda	venv
Python version (e.g. Python 3.9.7)	easy	hard
Package version (e.g. nltk 3.6.2)	easy	easy



```

1 from seven_dwarfs import Grumpy, Happy, Sleepy, Bashful, Sneezzy,
2 x = { 'a':37,'b':42,
3
4 'c':927}
5
6 x = 123456789.123456789E123456789
7
8 if very_long_variable_name is not None and \
9     very_long_variable_name.field > 0 or \
10     very_long_variable_name.is_debug:
11     z = 'hello '+'world'
12 else:
13     world = 'world'
14     a = 'hello {}'.format(world)
15     f = rf'hello {world}'
16 if (this
17 and that): y = 'hello 'world'#FIXME: https://github.com/psf/black
18 class Foo (    object ):
19     def f (self ):
20         return 37*-2
21     def g(self, x,y=42):
22         return y
23 def f (    a: List[ int ]):
24     return 37-a[42-u : y**3]

```

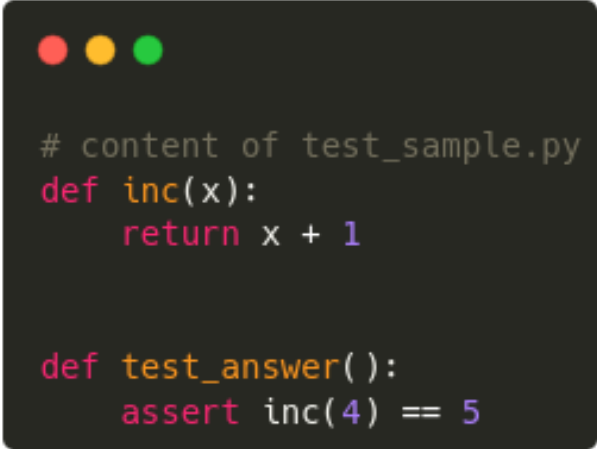
```

1 from seven_dwarfs import Grumpy, Happy, Sleepy, Bashful, Sneezzy, Do
2
3 x = {"a": 37, "b": 42, "c": 927}
4
5 x = 123456789.123456789e123456789
6
7 if (
8     very_long_variable_name is not None
9     and very_long_variable_name.field > 0
10    or very_long_variable_name.is_debug
11 ):
12     z = "hello " + "world"
13 else:
14     world = "world"
15     a = "hello {}".format(world)
16     f = rf"hello {world}"
17 if this and that:
18     y = "hello " "world" # FIXME: https://github.com/psf/black/issue
19
20
21 class Foo(object):
22     def f(self):
23         return 37 * -2
24
25     def g(self, x, y=42):
26         return y
27
28
29 def f(a: List[int]):
30     return 37 - a[42 - u : y**3]
31

```

format everytime on save and ensure
consistent formatting with Python standards

Make sure code does what it is supposed to do

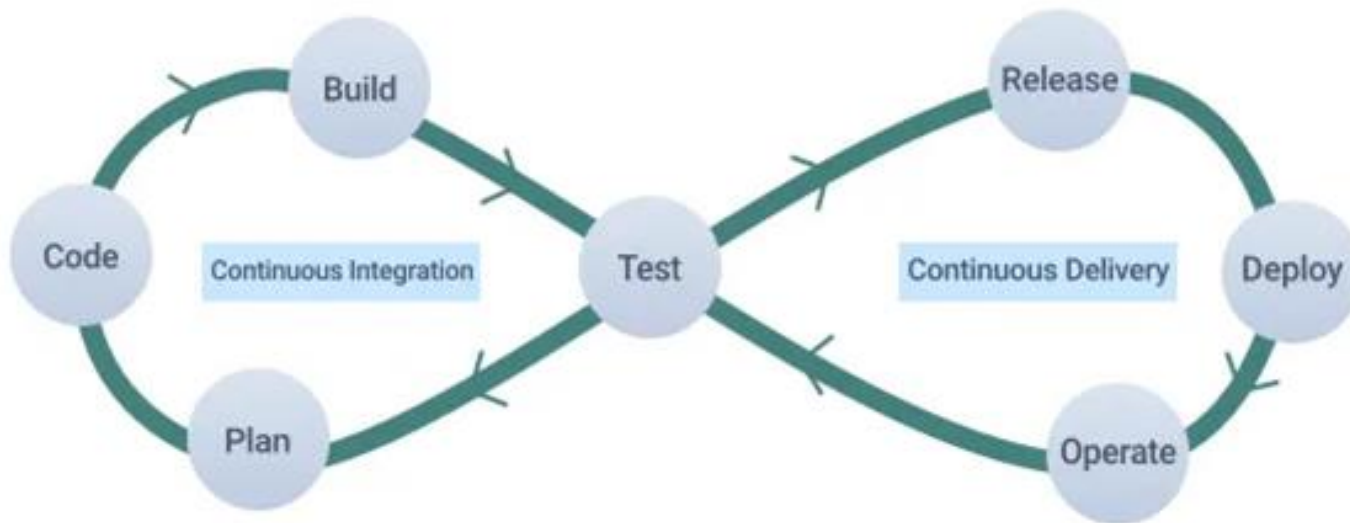


```
# content of test_sample.py
def inc(x):
    return x + 1

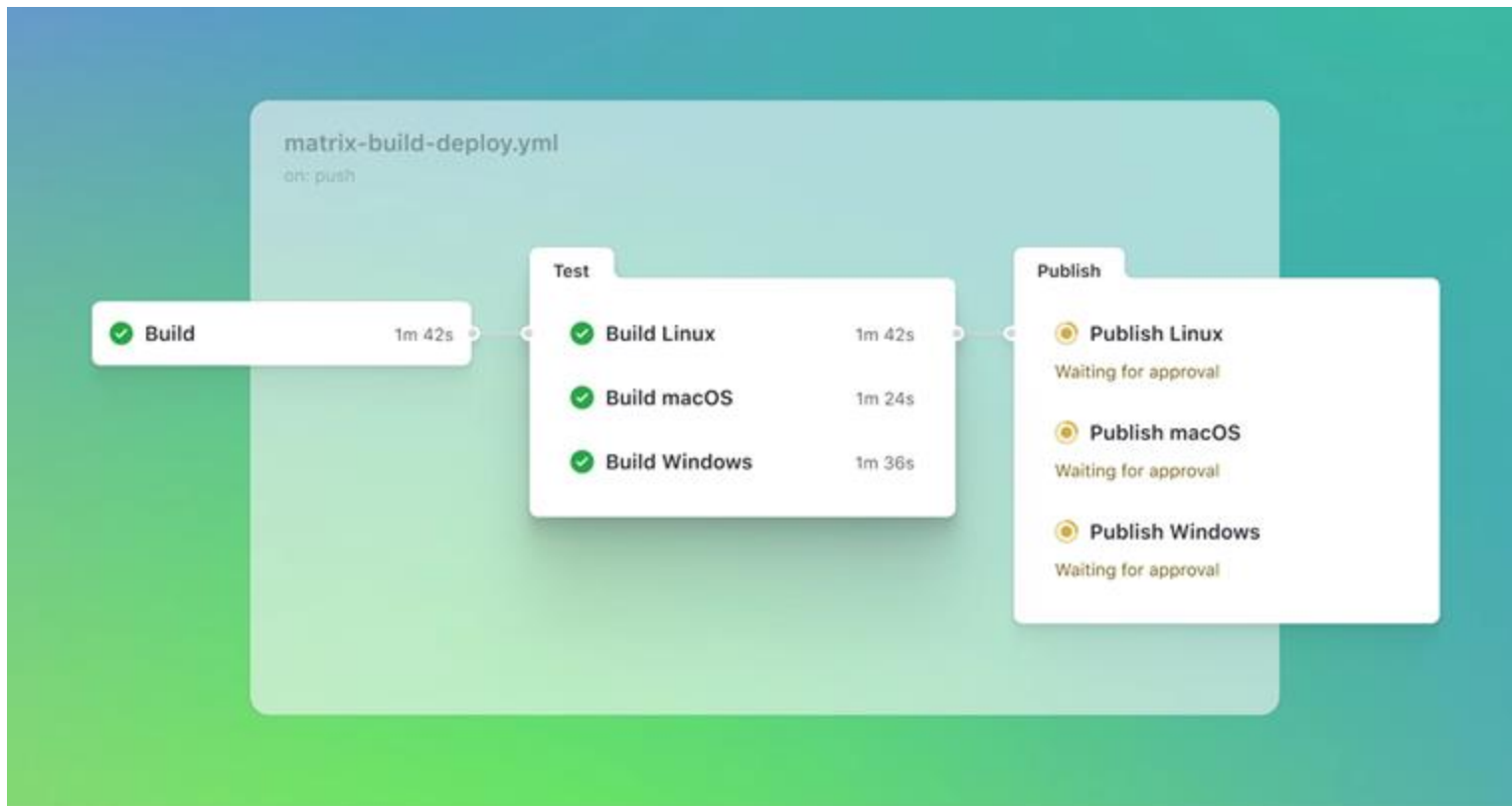
def test_answer():
    assert inc(4) == 5
```



Continuous Integration(CI)



<https://medium.com/digital-transformation-and-platform-engineering/a-quick-guide-to-continuous-integration-and-continuous-delivery-4df594ae281>



CI can be useful for teaching as well.

We automatically extract solutions from notebooks, build PDFs and deploy them to a website for our courses.

Molecular Dynamics and Monte Carlo

Q Search

Molecular Dynamics and Monte Carlo

Exercises

1. Basic Concepts of Molecular Dynamics and Monte Carlo Simulations
2. Statistical Thermodynamics and the Boltzmann Distribution
3. Monte Carlo Simulations and the Photon Gas
4. Introduction to Molecular Dynamics
5. Classical Molecular Dynamics (MD) Simulations
6. Properties from Molecular Dynamics

Molecular Dynamics and Monte Carlo

In this course, we will treat computational methods that are able to describe the properties of thermodynamic ensembles of molecules. These properties are either determined via stochastic sampling methods (Monte Carlo Simulations) or by following the time evolution of a molecular system for a sufficient amount of time (Molecular Dynamics Simulations).

The prediction of molecular properties at finite temperature - no matter whether it be a chemical reaction in solution or in an enzyme - requires more than the tools treated in the course 'Introduction to Electronic Structure Methods'. Finite temperature effects can give rise to substantial differences between an (idealised) isolated system at a hypothetical 0 K and the physical system at $T > 0$ K. Molecules have kinetic energy, and the system's behaviour is not governed by its potential energy alone, but by its *free energy*, with the effects of entropy taken into account. Consider the basic thermodynamical concept of the free energy of a reaction, $\Delta G = \Delta H - T\Delta S$. Although a reaction may be predicted to be endothermic based on quantum chemical calculations at 0 K, the entropic term of the transformation may still favour the reaction to be exergonic. The reaction will thus be spontaneous at finite temperature (cf. the solvation of certain salts in water, where the solute cools down due to an positive reaction enthalpy, but the process is spontaneous due to a considerable rise in entropy).

Both Molecular Dynamics (MD) and Monte Carlo (MC) simulations are techniques that allow to obtain information about the statistical distribution of a system, and thus on its thermodynamic properties at finite temperature, entropic effects included. The link between the microscopic system and the macroscopic thermodynamic observables is established in a branch of physics known as statistical mechanics. During the following exercise sessions, you will apply the techniques that have been treated in the lecture to both theoretical and practical problems; ranging from simple coding over statistical mechanics to actually performing both MD and MC simulations. Although the scope of this course is too small to give you a complete overview of the field, you should be able to gain some insight into the basic methodology and concepts.

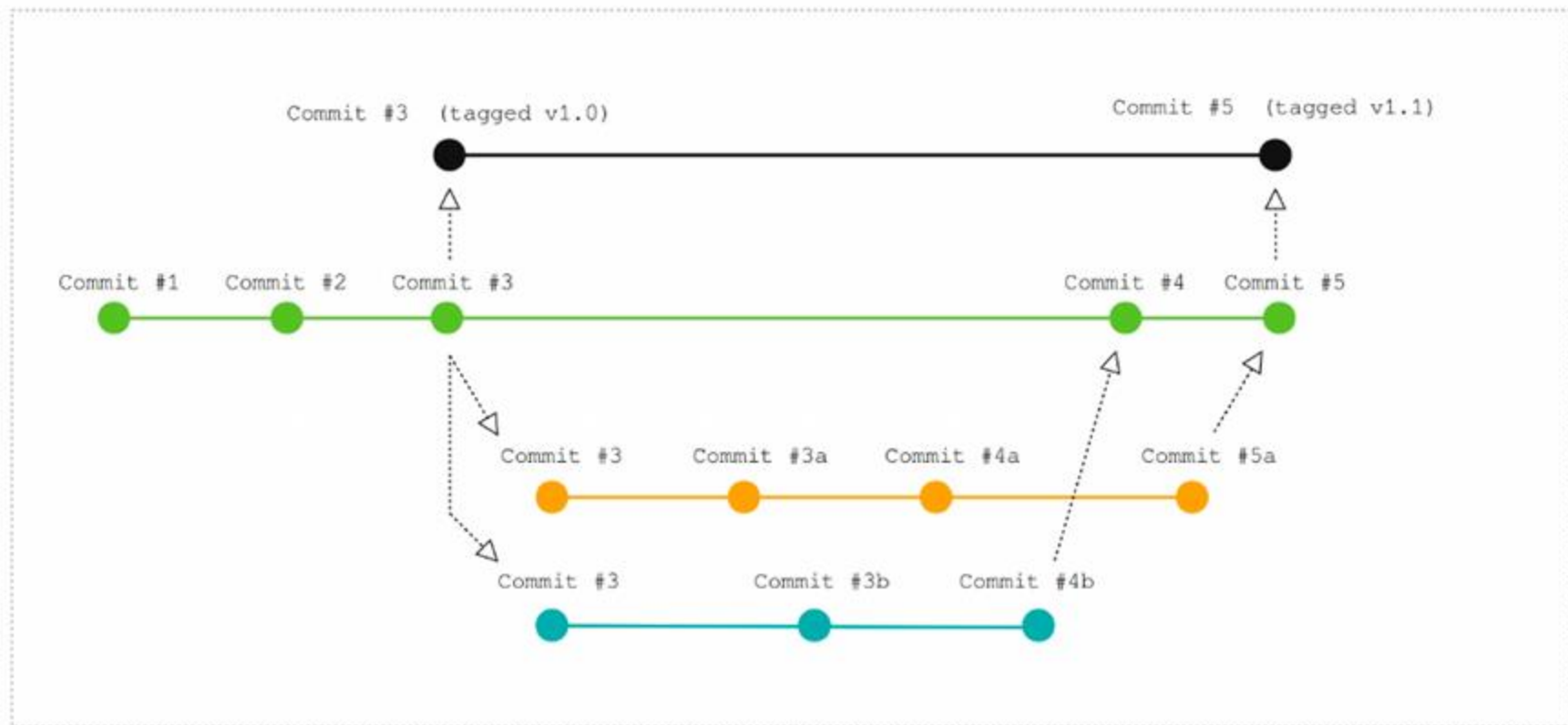
These exercises are based on various textbooks and the Molecular Simulations tutorial provided by the University of Amsterdam.

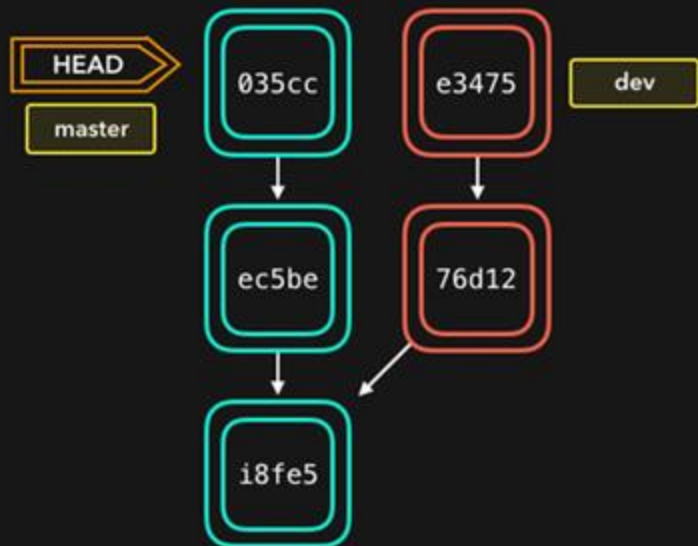
<https://github.com/lcbc-epfl/mdmc-public>

https://github.com/duerrsimon/repo_selective_sync_remove_solutions



— Master Branch (Production)
 — Develop Branch
 — Feature #1 Branch (Developer 1)
 — Feature #2 Branch (Developer 2)



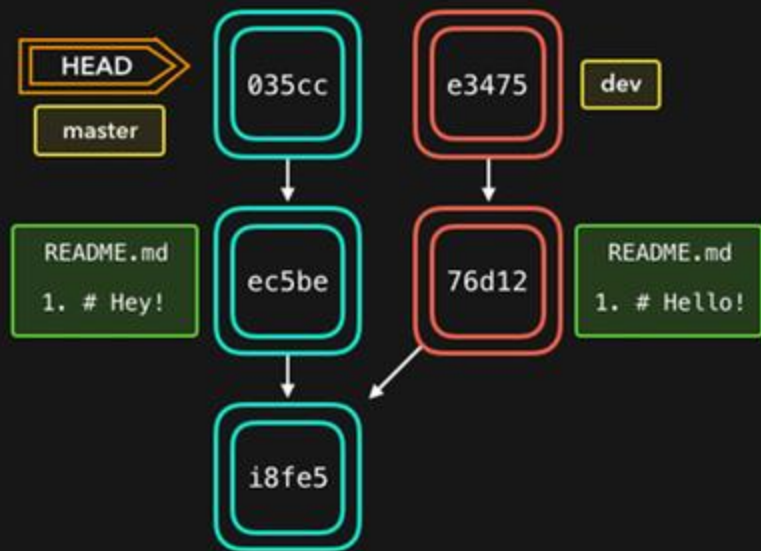


```
bash
master$
```

Git | Merging (no-fast-forward)

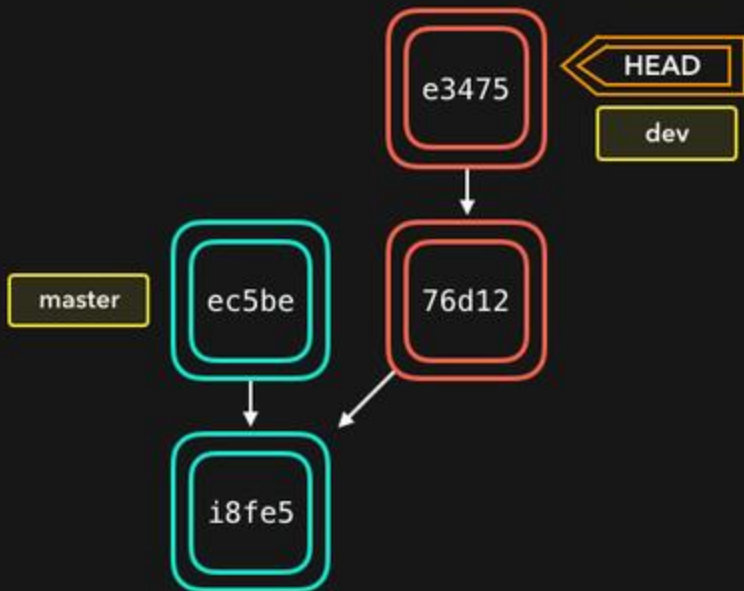
Default behavior when current branch contains commits that the merging branch doesn't have

Creates a new commit which merges two branches together without modifying existing branches



```
bash

master$
```



```
bash
dev$
```

Git | Rebasing

Copies commits on top of another branch without creating a commit, which keeps a linear history

Changes the history as new hashes are created for the copied commits

A note on commit messages

Conventional Commits

A specification for adding human and machine readable meaning to commit messages



```
<type>[optional scope]:  
<description>
```

```
[optional body]  
[optional footer(s)]
```

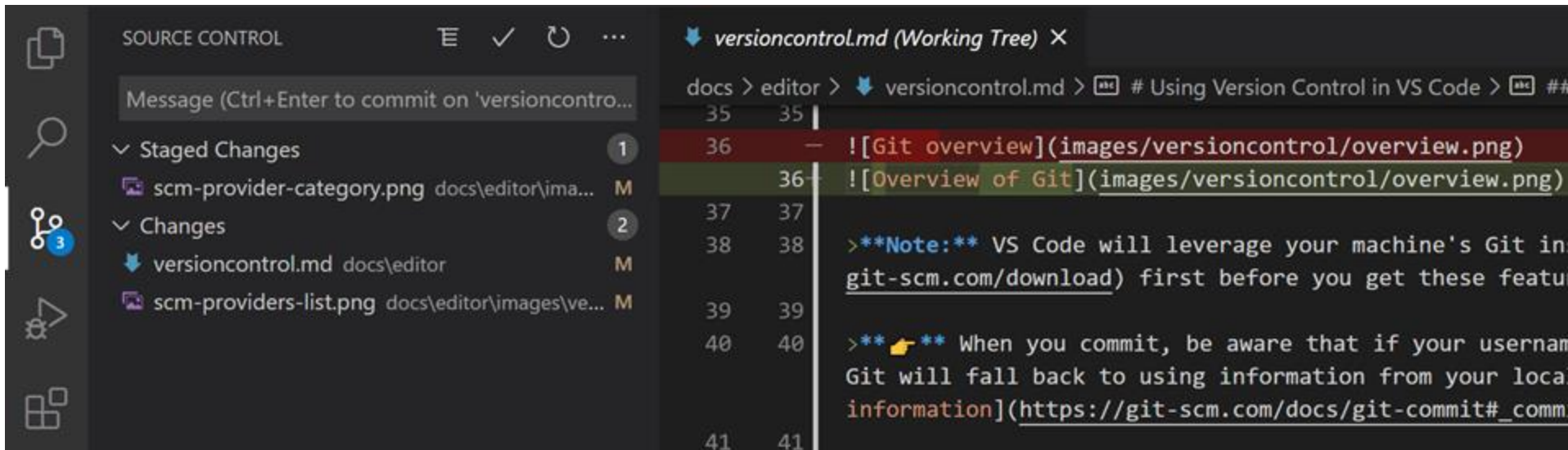
```
fix(api): fix wrong calculation of  
request body checksum
```

```
feat!: remove ticket list endpoint
```

```
docs: update tutorial
```

It pairs well with the principle that each commit should be one complete atomic change that starts and ends with the code base in a working state.

VSCode can also help with git workflows



Git for notebooks

Diffs for notebooks can be large because they mix execution output with code.



```

1 import marimo
2
3 __generated_with = "0.11.7"
4 app = marimo.App(width="medium")
5
6 @app.cell
7 def _():
8     - print("hello world")
8+    print("second version")
9     return
10

```



```

1 {
2   "cells": [
3     {
4       "cell_type": "code",
5       "execution_count": 1,
6       "metadata": {},
7       "outputs": [
8         {
9           "name": "stdout",
10          "output_type": "stream",
11          "text": [
12            - "hello world\n"
12+           "second version\n"
13          ]
14        }
15      ],
16      "source": [
17        - "print(\"hello world\")"
17+       "print(\"second version\")"
18      ]
19    },
20    {
21      "metadata": {
22        "hide_input": false,
23        "kernel_spec": {
24          - "display_name": "Python 3",
24+         "display_name": "base",
25          "language": "python",
26          "name": "python3"
27        },
28        "language_info": {
29          "codemirror_mode": {
30            "name": "ipython",
31            "version": 3
32          },
33          "file_extension": ".py",
34          "mimetype": "text/x-python",
35          "name": "python",
36          "nbconvert_exporter": "python",
37          "pygments_lexer": "ipython3",

```

```
using System;struct a{static int Main()  
{object[]c={"\u0048e\x6c\x6co "+(C\u0068ar)  
(86+1)+"or\x6c\x64"};typeof(Conso\u006ce).GetMethod  
\u0068o\u0064s()[101].Invoke(c,c);return 0;}}
```

worst

script.sh

```
def simple_math(a, b):  
    return a + b  
  
def main():  
    num1 = 10  
    num2 = 5  
    result = simple_math(num1, num2)  
  
    print("Number 1:", num1)  
    print("Number 2:", num2)  
    print("Result:", sum_result)  
  
if __name__ == "__main__":  
    main()
```

bad

good

```
def sum(a, b):
    """
    Perform summation two numbers.

    Parameters
    -----
    a : int or float
        The first number.
    b : int or float
        The second number.

    Returns
    -----
    int or float
        The sum of the two numbers.
    """
    return a + b

def main():
    """
    Entry point of the program.
    """
    num1 = 10
    num2 = 5
    result = sum(num1, num2)

    print("Number 1:", num1)
    print("Number 2:", num2)
    print("Result:", result)

if __name__ == "__main__":
    main()
```

excellent

```
import argparse

def sum(a, b):
    """
    Calculate the sum of two numbers.

    Parameters
    -----
    a : int or float
        The first number.
    b : int or float
        The second number.

    Returns
    -----
    int or float
        The sum of the two numbers.
    """
    return a + b

def main():
    """
    Entry point of the program.
    """
    parser = argparse.ArgumentParser(description="Calculate the sum of two numbers.")
    parser.add_argument("num1", type=int, help="First number")
    parser.add_argument("num2", type=int, help="Second number")
    args = parser.parse_args()
    try:
        result = sum(args.num1, args.num2)
    except Exception as e:
        print("An error occurred:", e)

    print("Number 1:", args.num1)
    print("Number 2:", args.num2)
    print("Result:", result)

if __name__ == "__main__":
    main()
```

```
def foo(arg1, arg2):
    """
    A method's docstring with parameters and return value.

    Use all the cool Sphinx capabilities in this description, e.g. to give
    usage examples ...

    :Example:

    >>> another_class.foo('', AClass())

    :param arg1: first argument
    :type arg1: string
    :param arg2: second argument
    :type arg2: :class:`module.AClass`
    :return: something
    :rtype: string
    :raises: TypeError
    """
    return '' + 1
```



automatically generate documentation
website from code

foo(arg1, arg2)

A method's docstring with parameters and return value.

Use all the cool Sphinx capabilities in this description, e.g. to give usage examples ...

Example:

```
>>> another_class.foo('', AClass())
```

Parameters:

- **arg1** (*string*) – first argument
- **arg2** (*module.AClass*) – second argument

Returns: something

Return type: string

Raises: TypeError

Start a project

```
~/cookiecutter:~/cookiecutter$ cookiecutter gh:choderalab/cookiecutter-cooperbox
project_name [project_name]: Drug Binder Finder
repo_name [drug_binder_finder]:
first_module_name [drug_binder_finder]: drug_search
author_name [your name for your organization/company/youself]: Levi W. Madden
description [a short description of the project, it's best to search for keywords that are drugs or drug-like]
Select open_source_license:
1 - MIT
2 - BSD-3-Clause
3 - LGPLv3
4 - Not Open Source
Choose from 1, 2, 3, 4 [1]:
Select dependency_source:
1 - Prefer conda-forge over the default anaconda channel with pip fallback
2 - Prefer default anaconda channel with pip fallback
3 - Dependencies from pip only (no conda)
Choose from 1, 2, 3 [1]:
```

<https://github.com/MolSSI/cookiecutter-cms>

<https://github.com/choderalab/software-development>

The screenshot displays the GitHub repository page for `choderalab / software-development`. The repository is public and has 239 stars, 79 forks, and 2 pull requests. The main content area shows a list of files and a merge pull request by `jchodera`. The files list includes:

File	Description	Time
<code>chapter_images</code>	Fill in the Structure page	7 years ago
<code>CONTINUOUS_INTEGRATION.md</code>	Other minor edits.	5 years ago
<code>DOCUMENTATION.md</code>	adding back to top	7 years ago
<code>LICENSING_GUIDELINES.md</code>	Edits for readability	5 years ago
<code>PACKAGING_AND_DEPLOYMENT.md</code>	Link PyPI tutorial to PyPA user guide	6 years ago
<code>PHILOSOPHY_AND_SCOPE.md</code>	Fix typo	5 years ago
<code>PYTHON_CODING.md</code>	Update PYTHON_CODING.md	4 years ago
<code>PYTHON_OPTIMIZATION.md</code>	Fixing typos	6 years ago
<code>README.md</code>	Add MoSSI Education Resources link	5 years ago
<code>STRUCTURING_YOUR_PROJECT.md</code>	Add CMS cookiecutter info to structuring your project	5 years ago
<code>UNIT_TESTING.md</code>	UNIT_TESTING.md: mention codecov	4 years ago
<code>VERSION_CONTROL.md</code>	Cleanup edits to Python Coding section.	5 years ago

The README section is titled "Software Development Best Practices for Computational Chemistry".

What do you really need?

	Research scripts/Notebooks	Research code
Unit tests	-	yes
Publish	Zenodo	Zenodo, PyPi, conda-forge, GitHub
Linting/Formatt ing	yes	yes
Documentation	inline/docstrings	Web documentation, tutorials (e.g using Sphinx)
Format	e.g Notebook, scripts	e.g Python modules
Environment	environment.yml	Docker container
CI/CD	-	yes
Versioning	yes, e.g git	yes, e.g git

ev



Example case:

I had a ML dataset of h5 files and corresponding csv files

```
folder
|
├─ file1.h5
├─ file2.h5
└─ envs
    ├─ file1_envs.csv
    └─ file2_envs.csv
```

I wanted to split the dataset into a smaller test set and validation set.

Use AI productively for your research data workflows

Be precise and split your commands



You

I have a folder with .h5 files with an identifier. In this folder there is a subfolder "envs" that contains corresponding csv files in the format identifier_envs.csv.

Take all of the files in the folder and move half of them to a folder test and half of them in vals

GPT-3.5 generates non working python code

**You**

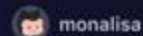
Write a python programm that iterates over a pair of h5 files contained in a folder. Each h5 file has an associated _envs.csv file in the envs folder

generates working example that one can debug

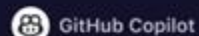
**You**

Now add functionality to move half the h5/csv pairs to a new directory. Sample them pairs to choose randomly

Adds the required functionalities and results in working code



Write unit tests for this function



```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK\n2023-01-03 -2.72 EUR'
        expected_output = [(datetime.datetime(2023, 1, 2), -34.01, 'USD'),
                             (datetime.datetime(2023, 1, 3), 2.59, 'DKK'),
                             (datetime.datetime(2023, 1, 3), -2.72, 'EUR')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string), expected_output)

if __name__ == '__main__':
    unittest.main()
```

The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format.

```
1 import datetime
2
3 def parse_expenses (expenses_string):
4     """Parse the list of expenses and return the list of triples (date, amount, currency)
5     Ignore lines starting with #.
6     Parse the date using datetime.
7     Example expenses_string:
8         2023-01-02 -34.01 USD
9         2023-01-03 2.59 DKK
10        2023-01-03 -2.72 EUR
11    """
12    expenses = []
13
14    for line in expenses_string.splitlines():
15        if line.startswith("#"):
16            continue
17        date, value, currency = line.split(" ")
18        expenses.append((datetime.datetime.strptime (date, "%Y-%m-%d"),
19                        float (value),
20                        currency))
21    return expenses
22
23 expenses_data = '''2023-01-02 -34.01 USD
24                  2023-01-03 2.59 DKK
25                  2023-01-03 -2.72 EUR'''
26
27
28
29
30
31
32
33
34
```



transactions.py

parallelize this because the api is slow

Submit Edit

quick question

gpt-4o ⌂K to toggle

```
def insert_failed_transactions_from_stripe():
    import sqlite3
    import stripe
    from datetime import datetime, timedelta

    # Set up Stripe API key
    stripe.api_key = 'your_stripe_api_key_here'

    # Connect to SQLite database (or create it if it doesn't exist)
    conn = sqlite3.connect('transactions.db')
    cursor = conn.cursor()

    # Create table if it doesn't exist
    cursor.execute('''
CREATE TABLE IF NOT EXISTS transactions (
    id TEXT PRIMARY KEY,
    amount INTEGER,
    currency TEXT,
    status TEXT,
    created TIMESTAMP
)
''')

    # Get today's date range
```

v0 Private Alpha

Discover Our Unique Features
A hero section in dark mode
9 hours ago

Twitter Post
A Tweet UI
13 hours ago

Pricing Plans
A sleek pricing page for a SaaS.
17 hours ago

Cookie Consent
A cookie consent banner
18 hours ago

Financial Invoices
A table of financial invoices
20 minutes ago

Music Player
A simple music player
14 hours ago

FAQ Terms AI Policy Privacy



McKay Wrigley

@mckaywrigley

Subscribe

It's a golden age for builders.

People who leverage AI can now build things that were unimaginable 2yrs ago.

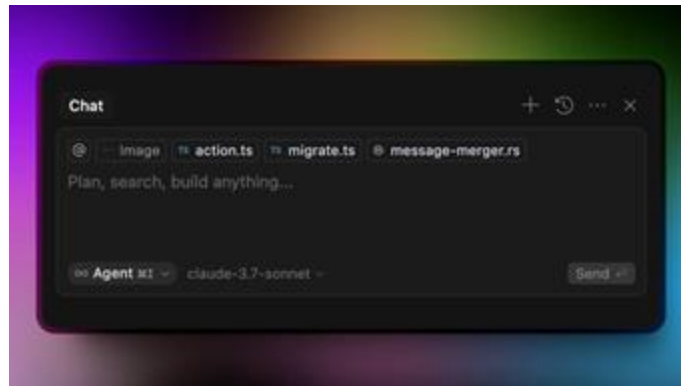
What used to take a team of 5 experts, \$1M+, and a ton of time can now be built by a solo dev using Cursor in a few days.

The age of superbuilders has begun.

1:41 · 03 Sept 24 · 87,5K Views

This is a bit hyperbolic but you certainly can become much more productive using AI.

Agentic programming



Suggestions directly in the terminal



apart
from
chatGPT

HuggingChat v0.7.0

Making the community's best AI chat models available to everyone.

Current Model

mistralai/Mixtral-8x7B-Instruct-v0.1



[↗ Model page](#)

[🌐 Website](#)

Examples

Write an email from bullet list

Code a snake game

Assist in a task

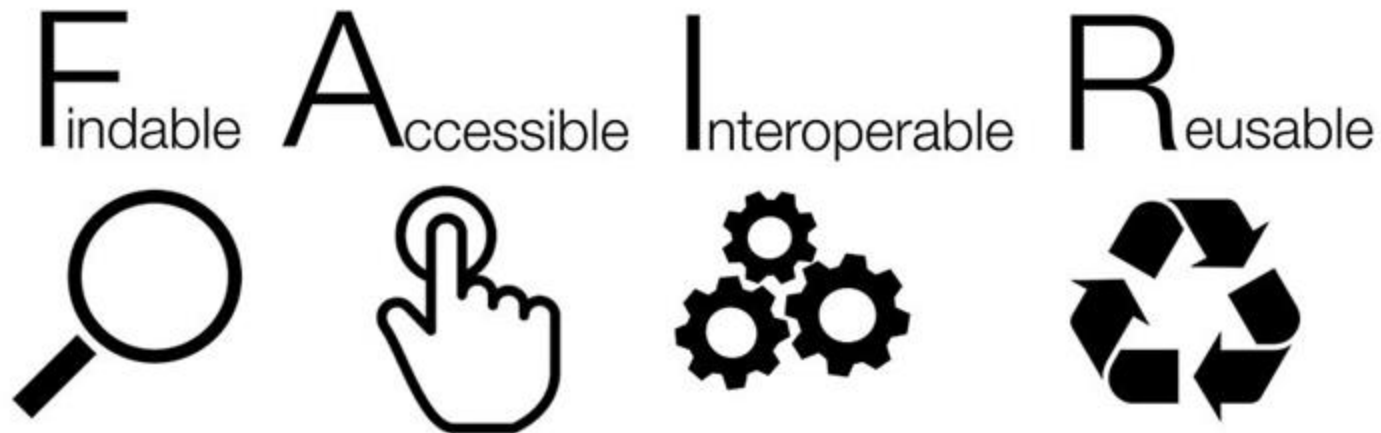
☐ Search web ⓘ

Ask anything



Sharing code

We need our code to be FAIR



How to package code

Just giving someone a code file does not mean the code will run as intended.

We need:

- data
- code
- environment specification

So how can we do this?

	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
							

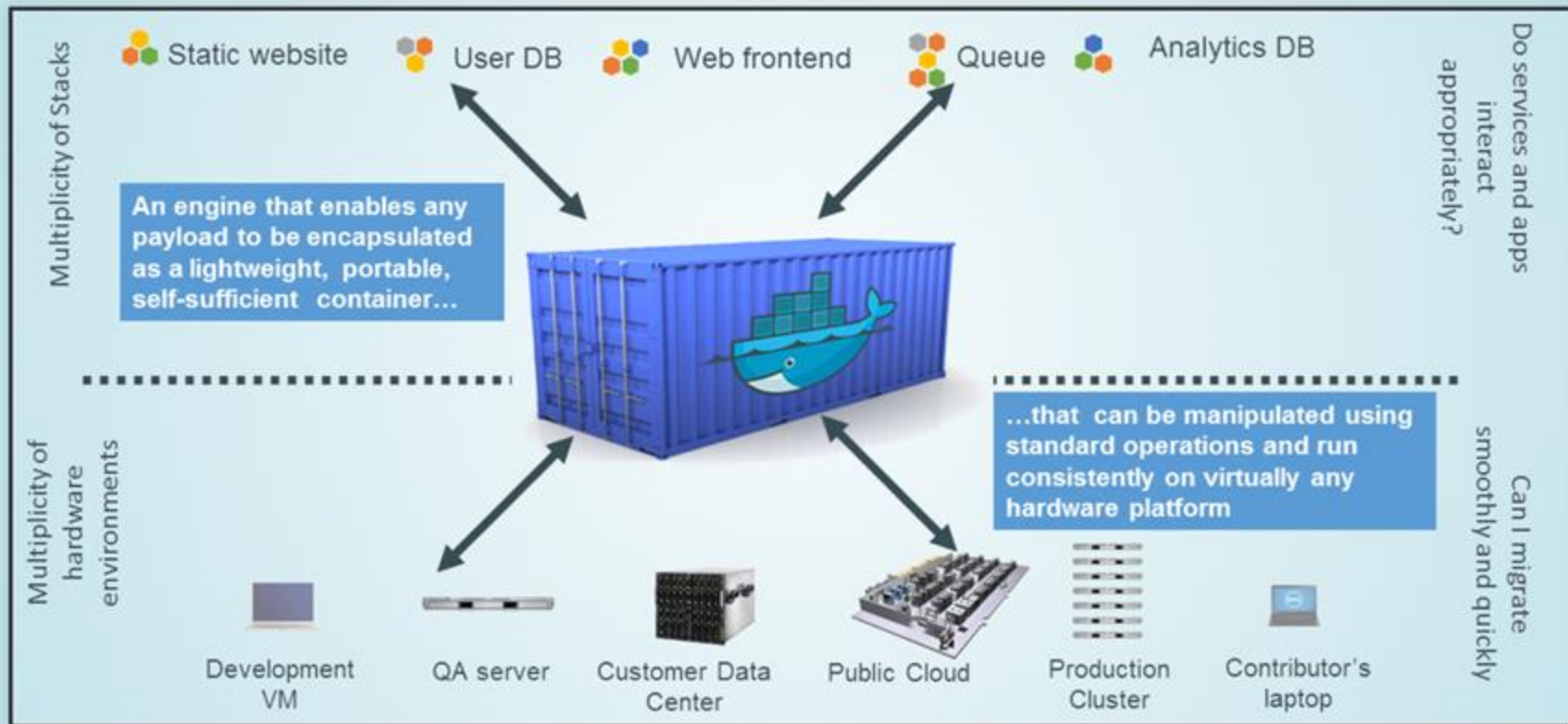
Solution: Intermodal Shipping Container



The Matrix from Hell

	Static website	?	?	?	?	?	?	?
	Web frontend	?	?	?	?	?	?	?
	Background workers	?	?	?	?	?	?	?
	User DB	?	?	?	?	?	?	?
	Analytics DB	?	?	?	?	?	?	?
	Queue	?	?	?	?	?	?	?
		Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
								

Docker is a Container System for Code



Build once... (finally) run *anywhere**

- A clean, safe, hygienic, portable runtime environment for your app.
- No worries about missing dependencies, packages and other pain points during subsequent deployments.

* *Where "anywhere" means an x86 server running a modern Linux kernel (3.2+ generally or 2.6.32+ for RHEL 6.5+, Fedora, & related)*



```
FROM python:3.7-slim-buster
```

```
COPY install_packages.sh .
```

```
RUN ./install_packages.sh
```

```
RUN useradd cheminfo
```

```
WORKDIR /home/cheminfo
```

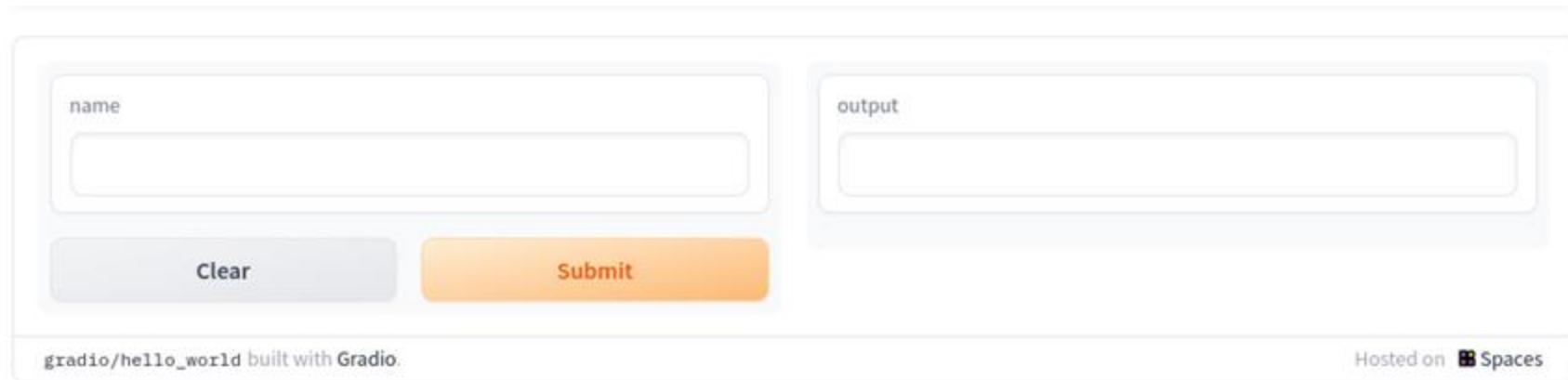
```
COPY requirements.txt .
```

```
COPY dimensionality_reduction ./dimensionality_reduction
```

```
COPY README.md .
```

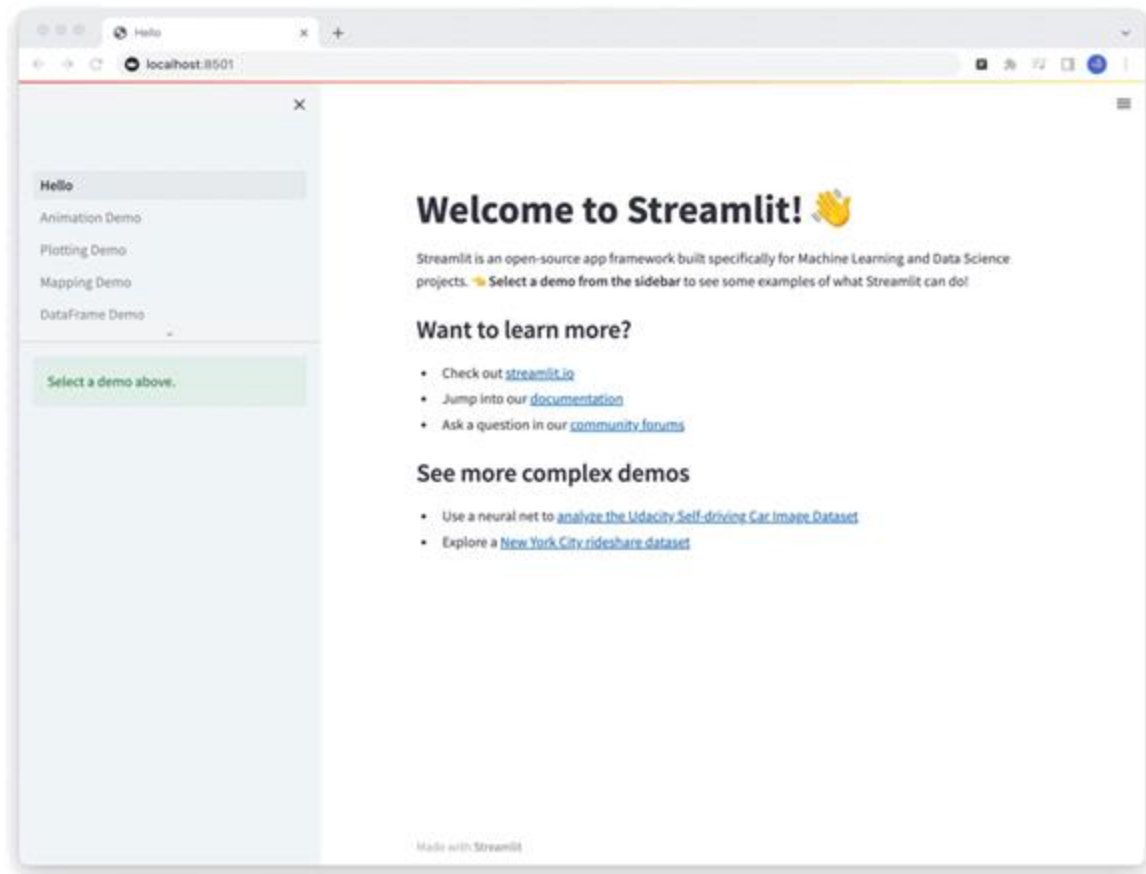
```
RUN pip install --no-cache-dir -r requirements.txt
```

Gradio

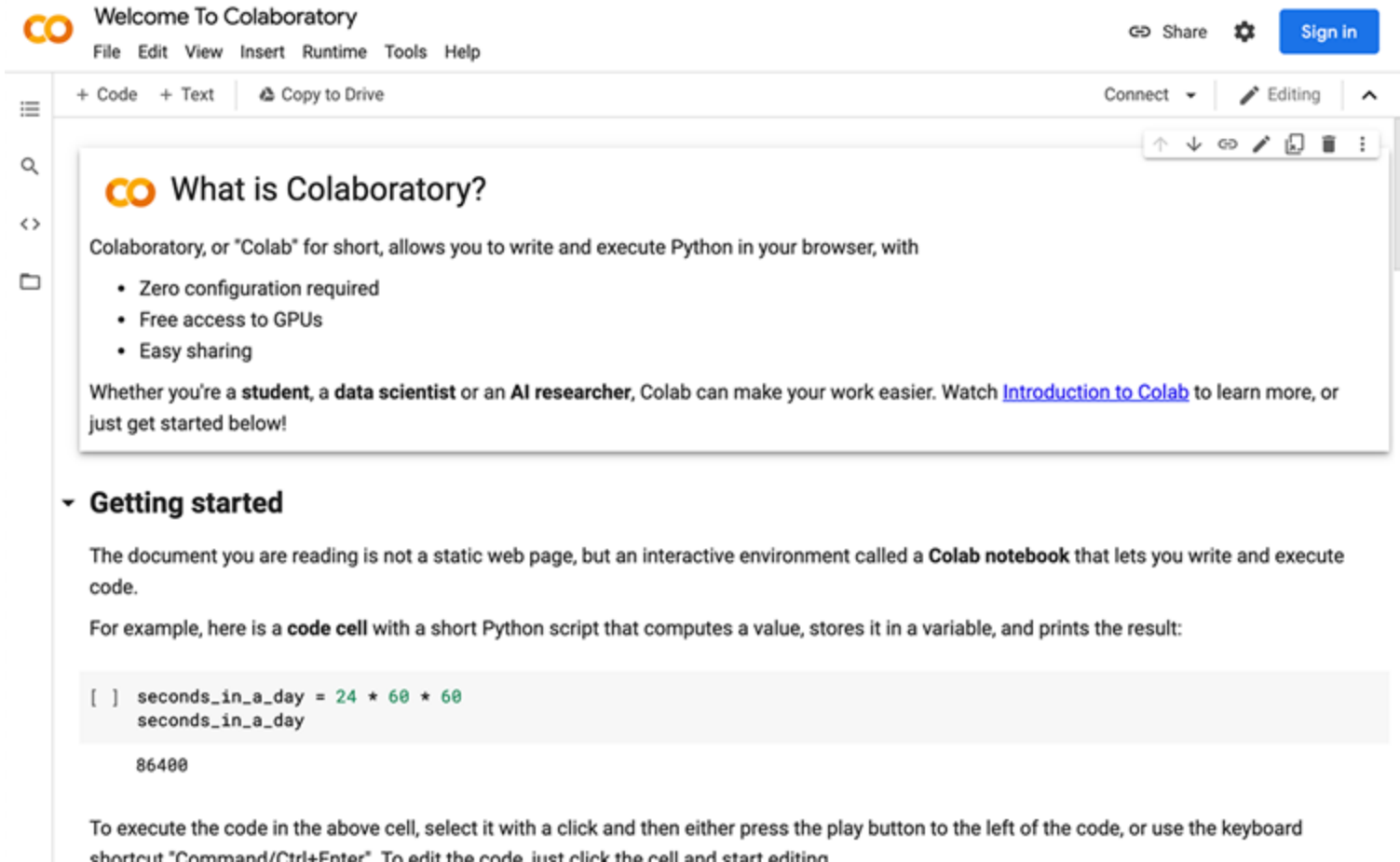


The screenshot shows a web application interface for Gradio. It features a light gray background. On the left, there is a white rounded rectangle containing a text input field labeled "name" and a "Clear" button below it. To the right of this is another white rounded rectangle containing a text input field labeled "output". Below the "name" input field is an orange "Submit" button. At the bottom left of the interface, the text "gradio/hello_world built with Gradio." is displayed. At the bottom right, it says "Hosted on Spaces" with a small icon.

<https://www.gradio.app/guides/quickstart>



<https://docs.streamlit.io/streamlit-community-cloud/get-started/quickstart>



The screenshot shows the Google Colaboratory web interface. At the top, there's a navigation bar with the Colab logo, 'Welcome To Colaboratory', and a menu (File, Edit, View, Insert, Runtime, Tools, Help). On the right, there are links for 'Share', 'Settings', and a 'Sign in' button. Below the navigation bar, there's a toolbar with '+ Code', '+ Text', 'Copy to Drive', 'Connect', 'Editing', and a vertical ellipsis. The main content area is titled 'What is Colaboratory?' and contains a paragraph explaining that Colab allows writing and executing Python in the browser. It lists three bullet points: 'Zero configuration required', 'Free access to GPUs', and 'Easy sharing'. Below this, it says 'Whether you're a student, a data scientist or an AI researcher, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!'. A section titled 'Getting started' follows, explaining that the document is an interactive environment called a 'Colab notebook'. It then provides an example of a 'code cell' containing a Python script that calculates the number of seconds in a day. The code is:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

. The output of the code is '86400'. At the bottom, it explains how to execute the code (click and play button or keyboard shortcut) and how to edit the code (click the cell and start editing).

co Welcome To Colaboratory
File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive Connect Editing

co What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

86400

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

ColabFold v1.5.5: AlphaFold2 using MMseqs2

Easy to use protein structure and complex prediction using [AlphaFold2](#) and [Alphafold2-multimer](#). Sequence alignments/templates are generated through [MMseqs2](#) and [HHsearch](#). For more details, see [bottom](#) of the notebook, checkout the [ColabFold GitHub](#) and read our manuscript. Old versions: [v1.4](#), [v1.5.1](#), [v1.5.2](#), [v1.5.3-patch](#)
[Mirdita M, Schütze K, Moriawaki Y, Heo L, Ovchinnikov S, Steinegger M. ColabFold: Making protein folding accessible to all. Nature Methods, 2022](#)



> Input protein sequence(s), then hit Runtime -> Run all



query_sequence: "PIAQIHILEGRSDEQKETLIREVSEAIRSLDAPLTSVRVIITEMAKGHFGIGGELASK"

- Use : to specify inter-protein chainbreaks for **modeling complexes** (supports homo- and hetro-oligomers). For example **PI...SK:PI...SK** for a homodimer

jobname: "test"

num_relax: 0

- specify how many of the top ranked structures to relax using amber

template_mode: none

- none = no template information is used. pdb100 = detect templates in pdb100 (see [notes](#)). custom - upload and search own templates (PDB or mmCIF format, see [notes](#))

[Show code](#)

> Install dependencies



[Show code](#)

colab frequently updates
the environment

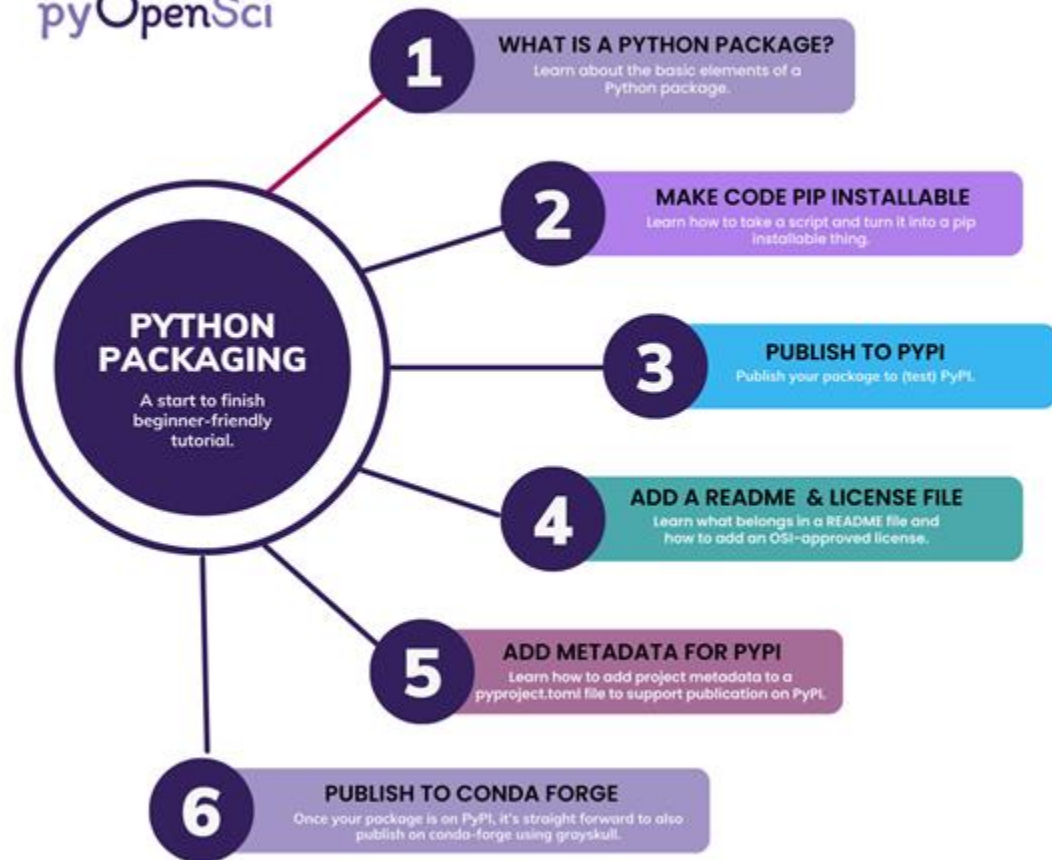
not reliable enough for
sharing research code

You will need to constantly
maintain it

2023-12-18

- Expanded access to AI coding has arrived in Colab across 175 locales for all tiers of Colab users
- Improvements to display of ML-based inline completions (for eligible Pro/Pro+ users)
- Started a series of [notebooks](#) highlighting Gemini API capabilities
- Enable `%/Ctrl+L` to select the full line in an editor
- Fixed [bug](#) where we weren't correctly formatting output from multiple execution results
- Python package upgrades
 - CUDA 11.8 to CUDA 12.2
 - tensorflow 2.14.0 -> 2.15.0
 - tensorboard 2.14.0 -> 2.15.0
 - keras 2.14.0 -> 2.15.0
 - Nvidia drivers 525.105.17 -> 535.104.05
 - tensorflow-gcs-config 2.14.0 -> 2.15.0
 - bigframes 0.13.0 -> 0.17.0
 - geemap 0.28.2 -> 0.29.6
 - pyarrow 9.0.0 -> 10.0.1
 - google-generativeai 0.2.2 -> 0.3.1
 - jax 0.4.20 -> 0.4.23
 - jaxlib 0.4.20 -> 0.4.23
- Python package inclusions
 - kagglehub 0.1.4
 - google-cloud-aiplatform 1.38.1

<https://colab.research.google.com/notebooks/reInotes.ipynb>





= Citable Code

<https://www.youtube.com/watch?v=gp3D4mf6MHQ>

When should you start doing this?

