



# Jupyter for Teaching

Jim Pivarski

University of Chicago – Data Science Institute

September 11, 2025



I almost called this talk “Literate Programming for Teaching,”  
because that’s the paradigm.

I almost called this talk “Literate Programming for Teaching,” because that’s the paradigm.

Jupyter is just the technology.

# Behold, the "Tioga" code-editing environment from 1982!



/ulde/bier/tiogafiles/gargoyl13/GGCirclesimpl.mesa
X11Viewers/Cmd Open New

```

Clear Reset Get GetImp1 PrevFile Menu Save Time Split Places Levels @ A
Find Word Def Position Normalize PrevPlace Rselect StyleKind
FirstLevelOnly MoreLevels FewerLevels AllLevels

CircleMeetsCircle: PUBLIC PROC [circle1, circle2: Circle] RETURNS [points: ARRAY [1..2] OF Point,
    hitCount: [0..2], tangent: BOOL ± FALSE] = {
    o1ToO2, o1ToOHat: Vector;
    epsilon: REAL ± GGUUtility.epsilonInPoints;
    magO1ToO2, outerTangent, innerTangent: REAL;
    IF RatherClose(circle1.origin, circle2.origin) THEN { -- concentric circles
        hitCount ± 0;
        points[1] ± points[2] ± [0.0, 0.0]; -- avoids compiler warnings
        RETURN;
    }
    o1ToO2 ± Vectors2d.Sub(circle2.origin, circle1.origin);
    magO1ToO2 ± Vectors2d.Magnitude[o1ToO2];
    outerTangent ± circle1.radius + circle2.radius;
    innerTangent ± Abs(circle1.radius - circle2.radius);

    SELECT magO1ToO2 FROM
    > outerTangent+epsilon => hitCount ± 0; -- circles far apart
    IN [outerTangent-epsilon .. outerTangent+epsilon] => { -- circles just touch as shown
        hitCount ± 1;
        tangent ± TRUE;
        o1ToOHat ± Vectors2d.Scale[o1ToO2, 1.0/magO1ToO2];
        points[1] ± Vectors2d.Add[circle1.origin, Vectors2d.Scale[o1ToO2Hat, circle1.radius]];
    }
    IN (innerTangent-epsilon .. outerTangent+epsilon) => {
        The two circles overlap. We expect two roots. In the picture below, point C is one of the circle
        intersection points. Segment AB is the segment O1O2 in the picture above. We wish to find the
        altitude h of the triangle. b = r1 and a = r2. From Heron's formula for the area of a triangle, area
        K = sqrt(s(s-a)(s-b)(s-c)), where s = (a+b+c)/2. We also know that the area K = 0.5ch. So h =
        2K/c. This gives us the y coordinate of the intersection points. The x coordinate is the point
        where the altitude hits the base, which we get from the Pythagorean Theorem.
    }
}

```

```

STOP! Find Split
Created Viewer: /R/TiogaDoc.tioga
% findr preview
preview.* =>
/Cedar13.0/Commands/PreView command!!
March 25, 1992 12:09:00 pm PST
/Cedar13.0/Top/PreView.dft6
May 5, 1997 3:08:09 pm PDT
/Cedar13.0/PreView/PreView.icons!!
September 6, 1985 4:59:14 pm PDT
/Cedar13.0/PreView/PreView.tup1
January 23, 1989 11:04:18 am PST
% preview
/usr/ccs/bin/ld -G -o /tmp/ImagerPixelArrayAISimpl-4206F692.so
/project/cedar13.0/versions/ais/sun05/imagerpixelarrayaisimpl.o~9~ => 0
/usr/ccs/bin/ld -G -o /tmp/AISIOimpl-41E958F0.so
/project/cedar13.0/versions/ais/sun05/aisioimpl.o~9~ => 0
/usr/ccs/bin/ld -G -o /tmp/AISStubimpl-41E958F0.so
/project/cedar13.0/versions/ais/sun05/aisstubimpl.o~6~ => 0
/usr/ccs/bin/ld -G -o /tmp/PPreViewimpl-42070138.so
/project/cedar13.0/versions/preview/sun05/ppreviewimpl.o~10~ => 0
/usr/ccs/bin/ld -G -o /tmp/PPreViewTool-42070138.so
/project/cedar13.0/versions/preview/sun05/ppreviewtool.o~11~ => 0
Unable to parse command line
% preview TGMLSiggraph9SSubmitted.ip
%

/ulde/bier/tiogafiles/ % [Split]
STOP! Find Split
March 25, 1992 12:09:00 pm PST
/Cedar13.0/Top/PreView.dft6
May 5, 1997 3:08:09 pm PDT
/Cedar13.0/PreView/PreView.icons!!
September 6, 1985 4:59:14 pm PDT
/Cedar13.0/PreView/PreView.tup1
January 23, 1989 11:04:18 am PST
% preview
/usr/ccs/bin/ld -G -o /tmp/ImagerPixelArrayAISimpl-4206F692.so
/project/cedar13.0/versions/ais/sun05/imagerpixelarrayaisimpl.o~9~ => 0
/usr/ccs/bin/ld -G -o /tmp/AISIOimpl-41E958F0.so
/project/cedar13.0/versions/ais/sun05/aisioimpl.o~9~ => 0
/usr/ccs/bin/ld -G -o /tmp/AISStubimpl-41E958F0.so
/project/cedar13.0/versions/ais/sun05/aisstubimpl.o~6~ => 0
/usr/ccs/bin/ld -G -o /tmp/PPreViewimpl-42070138.so
/project/cedar13.0/versions/preview/sun05/ppreviewimpl.o~10~ => 0
/usr/ccs/bin/ld -G -o /tmp/PPreViewTool-42070138.so
/project/cedar13.0/versions/preview/sun05/ppreviewtool.o~11~ => 0
Unable to parse command line
% preview TGMLSiggraph9SSubmitted.ip
%

```

/ulde/ bier /
tiogafiles/ hello 13/mello xordimpl mesa
Top! Siggraph 9S Submitted tioga
bier / tiogafiles/ hello 13/mello dft
/R/ gargoyl dft
/R/ tiogaDoc .tioga

/ulde/ bier / %
for gargoyl
EditTool
System Dript

# Or "MathCad" for personal computers in 1986



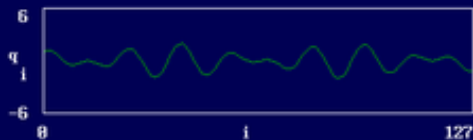
1FFTFILT.MCD1

0 0 auto

FILTERING A NOISY SIGNAL WITH FFT

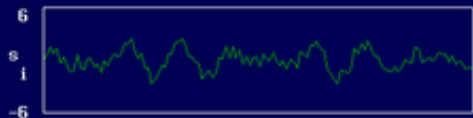
Define the signal:  $i := 0 \dots 127$

$$q_i := \sin\left[\frac{i}{128} \cdot 14 \cdot \pi\right] + \cos\left[\frac{i}{128} \cdot 19 \cdot \pi\right]$$

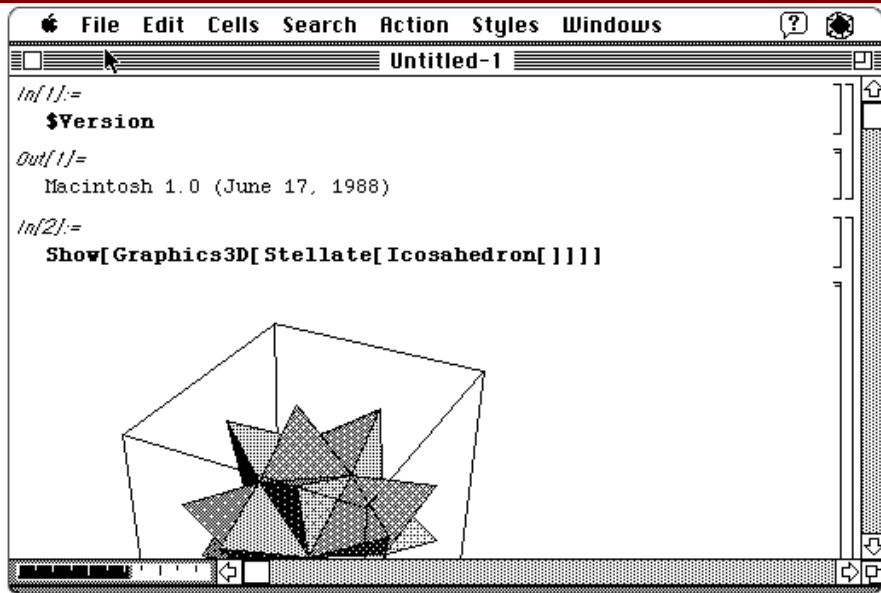


Add some noise:

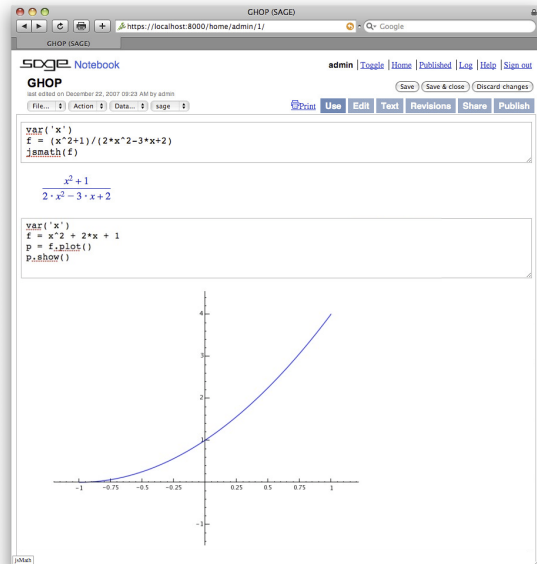
$$s_i := q_i + \text{rnd}(2) - 1$$



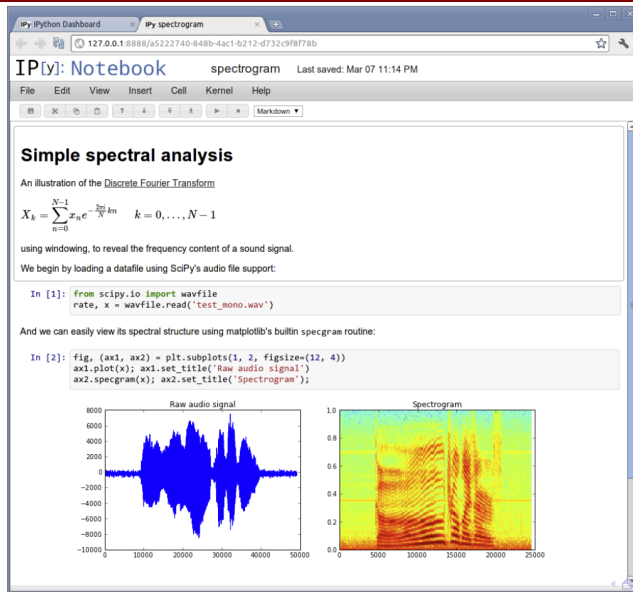
“Mathematica” in 1987 was the first of these to be mainstream



# Jupyter wasn't even the first to use web browsers and Python

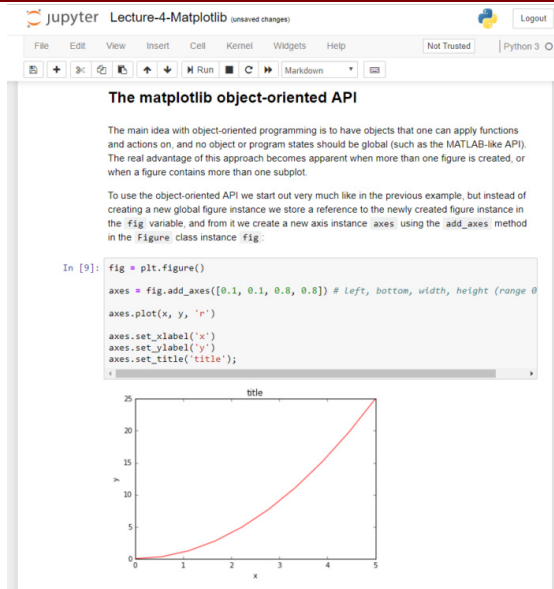


This is “SAGE” in 2007.





But changed its name in 2015 to say it's for **Julia**, **Python**, and **R**



(But now Julia programmers more often use Pluto.jl and R programmers often use RStudio.)

# Rewritten as a complete, Tioga-like environment in 2018: JupyterLab



Files

Running

Commands

Cell Tools

Tabs

File Edit View Run Kernel Tabs Settings Help

+

📁

🔄

> notebooks

| Name                | Last Modified |
|---------------------|---------------|
| Data.ipynb          | an hour ago   |
| Fasta.ipynb         | a day ago     |
| Julia.ipynb         | a day ago     |
| <b>Lorenz.ipynb</b> | seconds ago   |
| R.ipynb             | a day ago     |
| iris.csv            | a day ago     |
| lightning.json      | 9 days ago    |
| lorenz.py           | 3 minutes ago |

Lorenz.ipynb x

Terminal 1 x

Console 1 x

Data.ipynb x

README.md x

Python 3

Code

In this Notebook we explore the Lorenz system of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

```
In [4]: from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

Output View x

lorenz.py x

sigma 10.00

beta 2.67

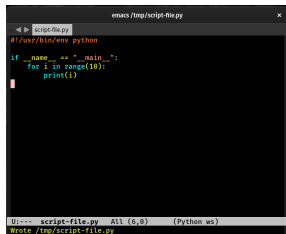
rho 28.00

```
9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
19
20     def lorenz_deriv(x,y,z, t0, sigma=sigma, beta=beta, rho=rho):
21         """Compute the time-derivative of a Lorenz system."""
22         x, y, z = x,y,z
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
24
25     # Choose random starting points, uniformly distributed from -15 to 15
26     np.random.seed(1)
27     x0 = -15 + 30 * np.random.random((N, 3))
28
```

# Notebooks are the middle of three fundamental types of code editors



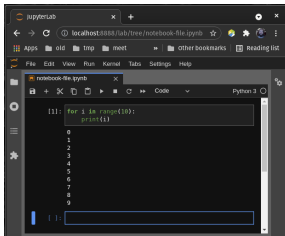
## Batch



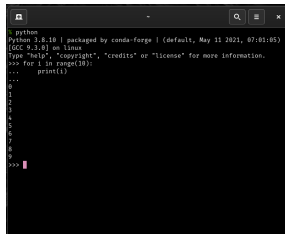
```
#!/usr/bin/env python

if __name__ == "__main__":
    for i in range(10):
        print(i)
```

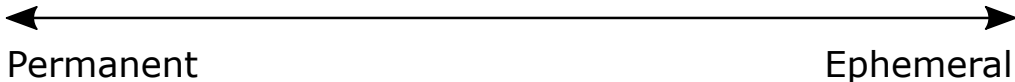
## Notebook



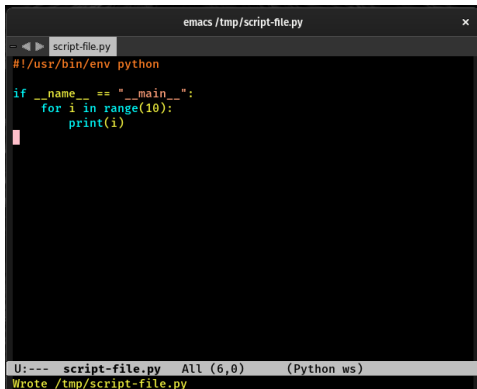
## Terminal



```
python
Python 3.8.10 | packaged by conda-forge | (default, May 11 2021, 07:01:05)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>>
```



- ▶ This is the oldest, from the punch-card, time-sharing era.



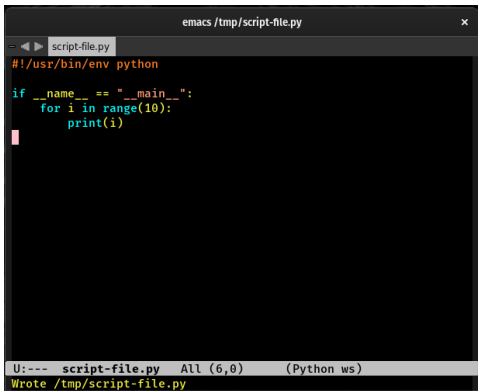
The screenshot shows an Emacs editor window titled 'emacs /tmp/script-file.py'. The editor is displaying a Python script named 'script-file.py'. The script content is as follows:

```
#!/usr/bin/env python

if __name__ == "__main__":
    for i in range(10):
        print(i)
```

The status bar at the bottom of the window shows 'U:--- script-file.py All (6,0) (Python ws)' and 'Wrote /tmp/script-file.py'.

- ▶ This is the oldest, from the punch-card, time-sharing era.
- ▶ Best for developing software libraries, with an emphasis on architecture and the large scale.

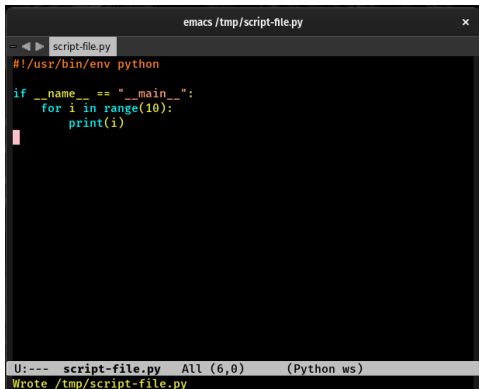


The screenshot shows an Emacs editor window titled 'emacs /tmp/script-file.py'. The editor is displaying a Python script named 'script-file.py'. The script content is as follows:

```
#!/usr/bin/env python

if __name__ == "__main__":
    for i in range(10):
        print(i)
```

The status bar at the bottom of the window shows 'U:--- script-file.py All (6,0) (Python ws)' and a message 'Wrote /tmp/script-file.py'.



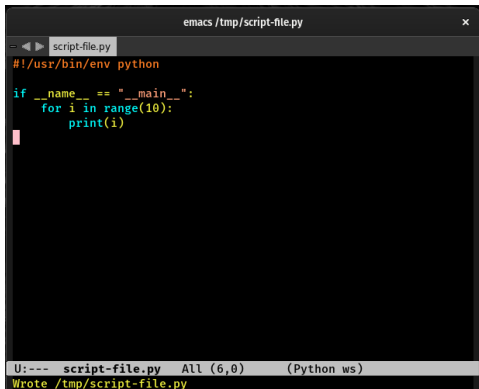
The screenshot shows an Emacs editor window titled 'emacs /tmp/script-file.py'. The editor is displaying a Python script named 'script-file.py'. The script content is as follows:

```
#!/usr/bin/env python

if __name__ == "__main__":
    for i in range(10):
        print(i)
```

The status bar at the bottom of the window shows 'U:--- script-file.py All (6,0) (Python ws)' and a message 'Wrote /tmp/script-file.py'.

- ▶ This is the oldest, from the punch-card, time-sharing era.
- ▶ Best for developing software libraries, with an emphasis on architecture and the large scale.
- ▶ Testing means running the whole program from its beginning (maybe even compiling it if you're using one of *those* languages).



The screenshot shows an Emacs editor window titled 'emacs /tmp/script-file.py'. The editor is displaying a Python script named 'script-file.py'. The script content is as follows:

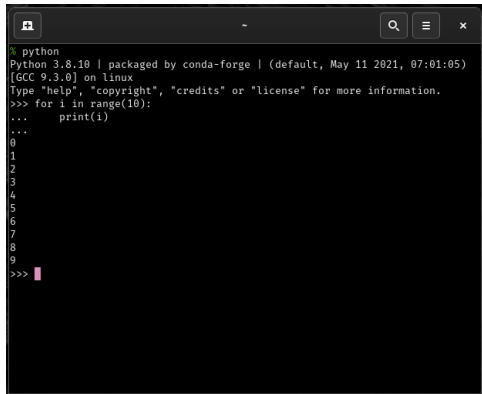
```
#!/usr/bin/env python

if __name__ == "__main__":
    for i in range(10):
        print(i)
```

The status bar at the bottom of the window shows 'U:--- script-file.py All (6,0) (Python ws)' and a message 'Wrote /tmp/script-file.py'.

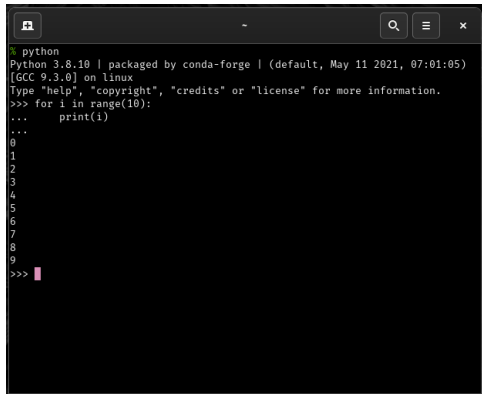
- ▶ This is the oldest, from the punch-card, time-sharing era.
- ▶ Best for developing software libraries, with an emphasis on architecture and the large scale.
- ▶ Testing means running the whole program from its beginning (maybe even compiling it if you're using one of *those* languages).
- ▶ Conceit: you're a sculptor, carving a beautiful edifice that will stand the test of time.

- ▶ Also has a long history in interactive languages like LISP, SPEAKEASY, and BASIC, as well as filesystem shells like UNIX and VAX.



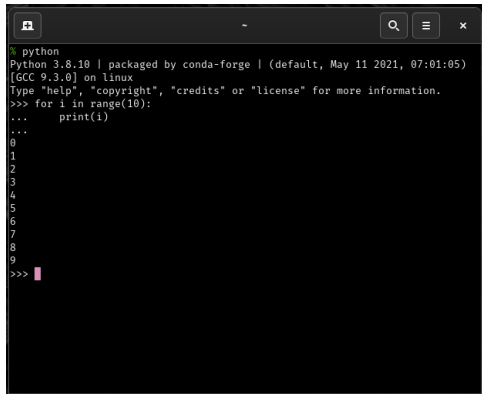
```
% python
Python 3.8.10 | packaged by conda-forge | (default, May 11 2021, 07:01:05)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>>
```





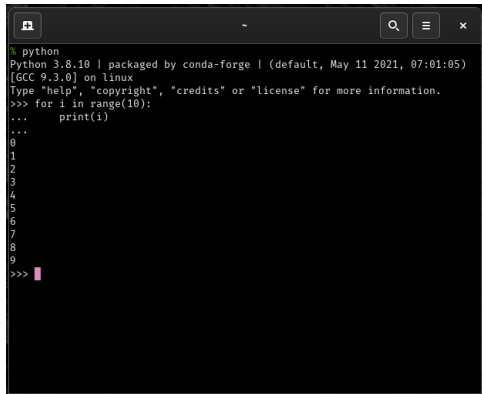
```
% python
Python 3.8.10 | packaged by conda-forge | (default, May 11 2021, 07:01:05)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>> █
```

- ▶ Also has a long history in interactive languages like LISP, SPEAKEASY, and BASIC, as well as filesystem shells like UNIX and VAX.
- ▶ Best for getting quick answers like, “What are the first few elements of this list?” or making simple changes like, “Convert all the images in this directory to JPEG.”



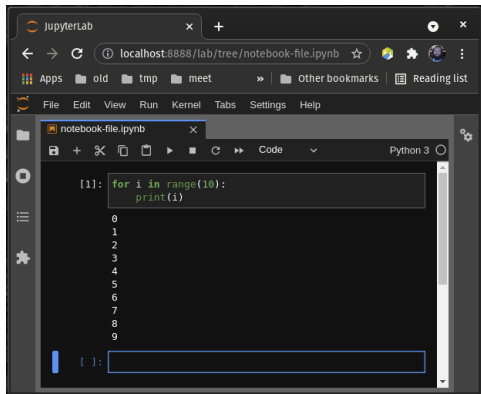
```
% python
Python 3.8.10 | packaged by conda-forge | (default, May 11 2021, 07:01:05)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>> █
```

- ▶ Also has a long history in interactive languages like LISP, SPEAKEASY, and BASIC, as well as filesystem shells like UNIX and VAX.
- ▶ Best for getting quick answers like, “What are the first few elements of this list?” or making simple changes like, “Convert all the images in this directory to JPEG.”
- ▶ Testing is continuous and immediate.

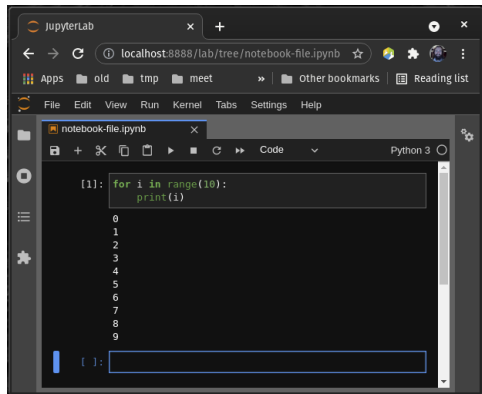


```
% python
Python 3.8.10 | packaged by conda-forge | (default, May 11 2021, 07:01:05)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>> █
```

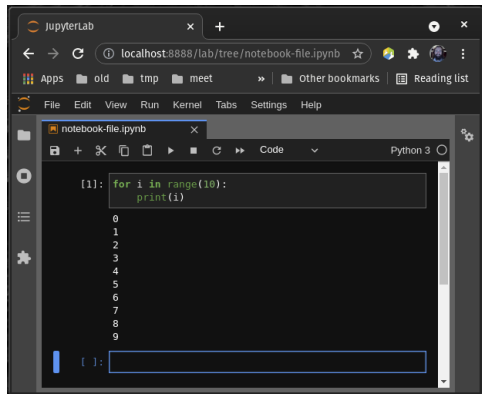
- ▶ Also has a long history in interactive languages like LISP, SPEAKEASY, and BASIC, as well as filesystem shells like UNIX and VAX.
- ▶ Best for getting quick answers like, “What are the first few elements of this list?” or making simple changes like, “Convert all the images in this directory to JPEG.”
- ▶ Testing is continuous and immediate.
- ▶ Conceit: you’re a hack3r, mashing out commands at a breakneck pace.



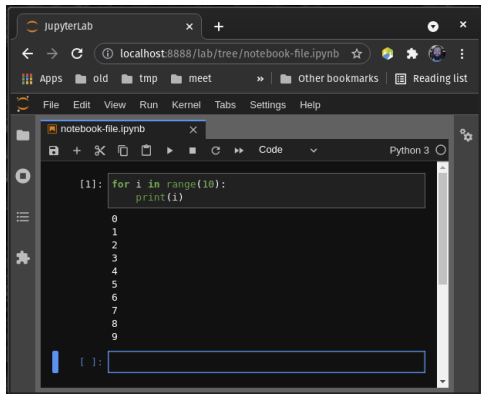
- ▶ Perceived as new, but they've had an important place in mathematical and data analysis software for almost as long as the spreadsheet.



- ▶ Perceived as new, but they've had an important place in mathematical and data analysis software for almost as long as the spreadsheet.
- ▶ Best for quick analyses that will likely be repeated: iterative investigation. Or for presenting code with detailed explanations, like a book, or teaching and running code in real-time.



- ▶ Perceived as new, but they've had an important place in mathematical and data analysis software for almost as long as the spreadsheet.
- ▶ Best for quick analyses that will likely be repeated: iterative investigation. Or for presenting code with detailed explanations, like a book, or teaching and running code in real-time.
- ▶ Testing is continuous and immediate.



- ▶ Perceived as new, but they've had an important place in mathematical and data analysis software for almost as long as the spreadsheet.
- ▶ Best for quick analyses that will likely be repeated: iterative investigation. Or for presenting code with detailed explanations, like a book, or teaching and running code in real-time.
- ▶ Testing is continuous and immediate.
- ▶ Conceit: Literate Programming!



I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: “Literate Programming.”

— Donald Knuth, 1984





I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: “Literate Programming.”

— Donald Knuth, 1984

One deliberately writes a paper, not just comments, along with code.

— Doug McIlroy, 1986



I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: “Literate Programming.”

— Donald Knuth, 1984

One deliberately writes a paper, not just comments, along with code.

— Doug McIlroy, 1986

Instead of writing code containing documentation, the literate programmer writes documentation containing code.

— Ross Williams, 1987



[https://github.com/minrk/ligo-binder/  
blob/master/index.ipynb](https://github.com/minrk/ligo-binder/blob/master/index.ipynb)



Is it a good book?

Is it a good lecture presentation?



Is it a good book?

**Maybe, for an advanced student who's willing to put in effort.**

Is it a good lecture presentation?



Is it a good book?

**Maybe, for an advanced student who's willing to put in effort.**

Is it a good lecture presentation? **No!**



Is it a good book?

**Maybe, for an advanced student who's willing to put in effort.**

Is it a good lecture presentation? **No!**

1. Font is too small; enlarging it would push too much off the screen.



Is it a good book?

**Maybe, for an advanced student who's willing to put in effort.**

Is it a good lecture presentation? **No!**

1. Font is too small; enlarging it would push too much off the screen.
2. As a vertical document, it's prone to scrolling too fast.





Is it a good book?

**Maybe, for an advanced student who's willing to put in effort.**

Is it a good lecture presentation? **No!**

1. Font is too small; enlarging it would push too much off the screen.
2. As a vertical document, it's prone to scrolling too fast.
3. Too much code in the cells, hard to unpack.



Is it a good book?

**Maybe, for an advanced student who's willing to put in effort.**

Is it a good lecture presentation? **No!**

1. Font is too small; enlarging it would push too much off the screen.
2. As a vertical document, it's prone to scrolling too fast.
3. Too much code in the cells, hard to unpack.
4. Students can't start until they install the software and data files.

# "How to give a good Jupyter talk talk" on YouTube



The problems

1. "It's too small!"
2. "You're scrolling too fast!"
3. "Too much code in the cells!"
4. "It doesn't work for me!"

13:05 / 1:38:26 • Key moments ▶

## PyHEP Topical Meeting (2 June 2021) - How to give a good Jupyter talk talk



HEP Software Foundation  
1.64K subscribers

Subscribe



7



Share



Save



### Key moments



Enabling a Slideshow Mode Using Jupyter Notebooks

33:42



To Make the Page Scroll Down and Align with the Cell

35:03



Classic Notebook

36:45



Speaker Notes

40:40



Css Hacking

44:54



Too Much Code in Cells

48:19



Incorporating Graphics

56:25



What Are some Good Examples of a Well-Made Jupyter Presentation

1:02:45

## All of the solutions I'll be presenting address these four problems

1. "It's too small!"
2. "You're scrolling too fast!"
3. "Too much code in the cells!"
4. "It doesn't work for me!"



1. "It's too small!"
  - ▶ browser magnification
  - ▶ configuring the theme settings
2. "You're scrolling too fast!"
  - ▶ slide presentations: `jupyterlab-deck`
  - ▶ hiding hints and solutions
3. "Too much code in the cells!"
  - ▶ self-restraint?
  - ▶ mini-libraries
4. "It doesn't work for me!"
  - ▶ Docker
  - ▶ Binder
  - ▶ Codespaces
  - ▶ JupyterLite

more interactivity



The students watch pre-evaluated slides.

The students watch you evaluate the cells.

The students press “shift-enter” along with you.

You stop now and then to ask, “what if I do *this* instead?”

You include formal exercises in the talk (short or long).

Next stop: the Dockerfile