

科研经验分享

为交互式计算而生——Jupyter项目

赵相杰

2023年2月27日

In 2021, Nature named Jupyter as one of ten computing projects that transformed science

TEN COMPUTER CODES THAT TRANSFORMED SCIENCE

From Fortran to preprint archives, these advances in programming and platforms sent biology, climate science and physics to new heights. **By Jeffrey M. Perkel**

In 2019, the Event Horizon Telescope team gave the world the first glimpse of what a black hole actually looks like. But the image of a glowing, ring-shaped object that the group unveiled wasn't a conventional photograph. It was computed – a mathematical transformation of data captured by radio telescopes in the United States, Mexico, Chile, Spain and the South Pole. The team released the programming code it used to accomplish that feat alongside the articles that documented its findings, so the scientific community could see – and build on – what it had done.

It's an increasingly common pattern. From astronomy to zoology, behind every great scientific finding of the modern age, there is a computer. Michael Levitt, a computational biologist at Stanford University in California who won a share of the 2013 Nobel Prize in Chemistry for his work on computational strategies for modelling chemical structure, notes that today's laptops have about

10,000 times the memory and clock speed that his lab-built computer had in 1967, when he began his prize-winning work. "We really do have quite phenomenal amounts of computing at our hands today," he says. "Trouble is, it still requires thinking."

Enter the scientist-coder. A powerful computer is useless without software capable of tackling research questions – and researchers who know how to write it and use it. "Research is now fundamentally connected to software," says Neil Chue Hong, director of the Software Sustainability Institute, headquartered in Edinburgh, UK, an organization dedicated to improving the development and use of software in science. "It permeates every aspect of the conduct of research."

Scientific discoveries rightly get top billing in the media. But *Nature* this week looks behind the scenes, at the key pieces of code that have transformed research over the past few decades.

Although no list like this can be definitive,

we polled dozens of researchers over the past year to develop a diverse line-up of ten software tools that have had a big impact on the world of science.

Language pioneer: the Fortran compiler (1957)

The first modern computers weren't user-friendly. Programming was literally done by hand, by connecting banks of circuits with wires. Subsequent machine and assembly languages allowed users to program computers in code, but both still required an intimate knowledge of the computer's architecture, putting the languages out of reach of many scientists.

That changed in the 1950s with the development of symbolic languages – in particular the 'Formula translation' language Fortran, developed by John Backus and his team at IBM in San Jose, California. Using Fortran, users could program computers using human-readable instructions, such as $x = 3 + 5$. A compiler then turned such directions into fast, efficient machine code.

It still wasn't easy: in the early days, programmers used punch cards to input code, and a complex simulation might require tens of thousands of them. Still, says Syukuro Manabe, a climatologist at Princeton University in New Jersey, Fortran made programming accessible to researchers who weren't computer scientists. "For the first time, we were able to program [the computer] by ourselves," Manabe says. He and his colleagues used the language to develop one of the first successful climate models.

Now in its eighth decade, Fortran is still widely used in climate modelling, fluid dynamics, computational chemistry – any discipline that involves complex linear algebra and requires powerful computers to crunch numbers quickly. The resulting code is fast, and there are still plenty of programmers who know how to write it. Vintage Fortran code bases are still alive and kicking in labs and on supercomputers worldwide. "Old-time programmers knew what they were doing," says Frank Giraldo, an applied mathematician and climate modeller at the Naval Postgraduate School in Monterey, California. "They were very mindful of memory, because they had so little of it."

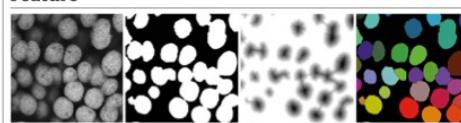
Signal processor:

fast Fourier transform (1965)

When radioastronomers scan the sky, they capture a cacophony of complex signals changing with time. To understand the nature of those radio waves, they need to see what those signals look like as a function of frequency. A mathematical process called a Fourier transform allows researchers to do that. The problem is that it's inefficient, requiring N^2 calculations for a data set of size N .

In 1965, US mathematicians James Cooley and John Tukey worked out a way to accelerate the process. Using recursion, a divide and

Feature



The ImageJ tool can analyse microscope images and automatically identify cell nuclei, as here. ©NIH/National Institutes of Health

neural structure of the brain. For decades, says Geoffrey Hinton, a computer scientist at the University of Toronto, Canada, AI researchers dismissed the latter approach as "nonsense". In 2012, Hinton's graduate students Alex Krizhevsky and Ilya Sutskever proved otherwise.

The venue was ImageNet, an annual competition that challenges researchers to train an AI on a database of one million images of everyday objects, then test the resulting algorithm on a separate image set. At the time, the best algorithms misclassified about one-quarter of them, Hinton says. Krizhevsky and Sutskever's AlexNet, a 'deep-learning' algorithm based on neural networks, reduced that error rate to 16% (ref. 10). "We basically halved the error rate, or almost halved it," notes Hinton.

Hinton says the team's success in 2012 reflected the combination of a big enough training data set, great programming and the newly emergent power of graphical processing units – the processors that were originally designed to accelerate computer video performance. "Suddenly we could run [the algorithm] 30 times faster," he says, "or learn on 30 times as much data."

The real algorithmic breakthrough, Hinton says, actually occurred three years earlier, when his lab created a neural network that could recognize speech more accurately than could conventional AIs that had been refined over decades. "It was only slightly better," Hinton says. "But already that was the writing on the wall."

Those victories heralded the rise of deep learning in the lab, the clinic and more. They're why mobile phones are able to understand spoken queries and why image-analysis tools can readily pick out cells in photomicrographs. And they are why AlexNet took its place among the many tools that have fundamentally transformed science, and with them, the world.

Jeffrey M. Perkel is technology editor at *Nature*.

Take a survey at go.nature.com/1O-computer-codes to weigh in on our code selections.

1. The Event Horizon Telescope Collaboration et al. *Astrophys. J. Lett.* **878**, L1 (2019).

2. Braig, K., Adams, P.D. & Brügge, A.T. *Nature Struct. Biol.* **2**, 1083–1084 (1995).

3. Streater, R.J. *J. Inst. Maths. Appl.* **43**, 623–660 (2000).

4. Nambiar, S. *Comput. Struct.* **19**, 197–199 (1985).

5. Manabe, S. & Bryan, F. *J. Atmos. Sci.* **26**, 786–790 (1969).

6. Lawson, C.L., Hanson, R.J., Kincaid, D.R. & Krogh, F.T. *ACM Trans. Math. Software* **5**, 303–323 (1979).

7. Griesemer, P. <http://arxiv.org/bb/H108.2700> (2011).

8. Nature **63**, 145–146 (2018).

9. Krizhevsky, A., Sutskever, I. & Hinton, G.E. in *Proc. 25th Int. Conf. Neural Information Processing Systems* (eds Pereira, F., Burges, C.J.C., Bottou, L. & Weinberger, K.G.) 1097–1105 (Curran Associates, 2012).

TOP CHOICES FOR SCIENCE CODE

Readers voted on which of the ten software codes in this article had the biggest impact on their work. They could choose up to three. Here are the results.

Fortran compiler 1,872 survey responses

Fast Fourier transform 1,577

IPython Notebook / Jupyter 1,282

arXiv 1,152

BLAS 875

BLAST 812

Biological databases 801

NIH Image / ImageJ / Fiji 505

AlexNet 328

General circulation model of the climate 239

©nature

数据科学岗位任职要求

高级机器视觉工程师  

任职要求：

- 1、硕士及以上学历，具有处理深度学习或工业项目经验者优先；
- 2、掌握以下其中一种编程能力：
 - 1) 掌握 C++ 或 C#，有实际编码经验，熟练使用 VS 或 QT 等编译器；
 - 2) 掌握 python，有实际编码经验，熟练使用 jupyter notebook, PyCharm 等编译器；
- 3、具备数字图像处理及模式识别相关的基础理论和算法知识优先；
- 4、使用任一种视觉软件库 (Halcon/OpenCV)，深度框架（如 PyTorch）进行视觉系统开发，；
- 5、具备较强的自学能力和独立解决问题的能力，善于团队协作与沟通。
- 6、能接受出差工作安排。

数据分析师  

职位描述

hive python 数据分析 运营

任职要求：

- 1、统招本科及以上学历可查学历；
- 2、3年及以上数据分析相关经验，有运营经验优先；
- 3、熟练使用 Python 及 jupyter, sklearn 等工具及包，根据需求进行数据分析。
- 4、熟练编写 HSQL，熟悉 hive、hadoop 等基本操作。
- 5、熟练编写 shell 和 Python 脚本。
- 6、会写 hive+Python 即可，会写 map reduce 任务加分项

数据分析师  

``.

任职要求：

学历及专业：

学历：本科及以上。

专业：计算机、软件工程、通信、数学等相关专业。

专业技能：

1. 熟练掌握 Linux 下的编程，掌握 Python 语言，熟练使用 Jupyter 等工具；
2. 熟练掌握数据分析、探索、BI 等技术；
3. 熟悉数据库操作，SQL 语言，熟悉大数据技术，包括 Hive 等；
4. 深入了解机器学习、人工智能相关算法，且有一定广度者优先；
5. 熟悉人工智能领域当前热点和前沿技术，对至少一个应用领域有系统的认识；
6. 熟悉业界通用的人工智能和机器学习方案和应用流程，具有根据需求制定算法方案并完成工程开发的能力；
7. 有系统学习和应用机器学习 / 人工智能技术经验者优先。



Free software, open standards, and web services for
interactive computing across all programming languages

scripts (.py files)

软件工程

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 #
4 # Created on Sun Jul 21 2019
5 #
6 # @author: Xiangjie Zhao
7 # @email: xjzhao@genetics.ac.cn
8 #
9 # A pipeline for bulk RNA-seq analysis
10 #
11 #
12 # module compatibility
13 #
14 from __future__ import division, print_function, absolute_import, unicode_literals
15 from os.path import join as join_path
16 from datetime import datetime
17 #
18 import os, subprocess
19 import argparse
20 #
21 try:
22     from future_builtins import ascii, filter, hex, map, oct, zip
23 except:
24     pass
25 if os.sys.version_info.major > 2:
26     xrange = range
27 #
28 # utils
29 #
30 # general
31 #
32 class OmicsException(Exception):
33     """
34     custom exception
35     """
36     pass
37     pass
38 
```

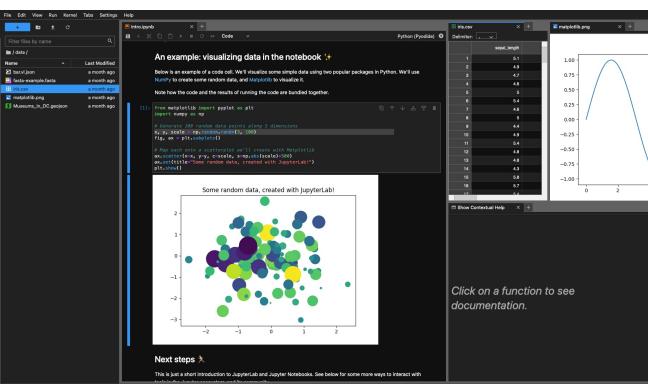
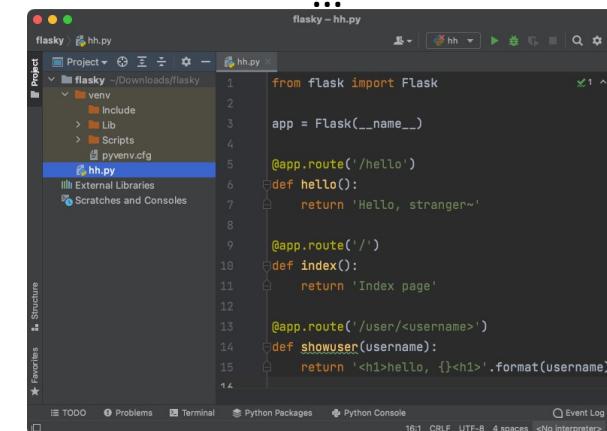
Python

数据科学

```
[In 1]: import numpy as np
[In 2]: x = np.random.randn(1, 4)
[In 3]: x
Out[3]: array([-1.25920396,  0.31454629,  1.38824689, -1.30699719])
[In 4]: x = np.random.randn(1, 4)
[In 5]: x
Out[5]: array([-0.62501251, -0.32113616,  0.4738682 , -1.05463289])
```

IPython

Pycharm
Visual Studio Code



Jupyter

Python

数据科学
软件工程

scripts (.py files)

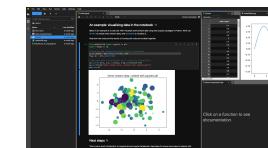
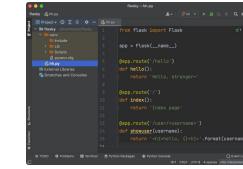
```
Users > xiangjie > Documents > 备忘 > omics-pipeline-master > omxpipeline > bulkRNA-pipeline.py
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Created on Sun Jul 21 2019
5
6  @author: Xiangjie Zhao
7  @email: xjzhao@genetics.ac.cn
8
9  A pipeline for bulk RNA-seq analysis
10 """
11
12 # -----
13 # module, compatibility
14
15 from __future__ import (division, print_function, absolute_import, unicode_literals)
16 from os.path import join as join_path
17 from datetime import datetime
18
19 import os, subprocess
20 import argparse
21
22 try:
23     from future_builtins import ascii, filter, hex, map, oct, zip
24 except:
25     pass
26
27 if os.sys.version_info.major > 2:
28     xrange = range
29
30 # -----
31 # utils
32
33 # -----
34 # general
35
36 class OmicsException(Exception):
37     """custom exception
38     """
39     pass
40
```

```
[1]: import numpy as np
[2]: x = np.random.rand(1, 4)
[3]: x
[4]: array([0.2553936, 0.7655428, 1.0014689, -0.00097191])
[5]: x = np.random.rand(1, 4)
[6]: x
[7]: array([0.1301251, -0.3121369, -0.4739442, -1.0546295])
```

IPython

Pycharm
Visual Studio Code

...



Jupyter

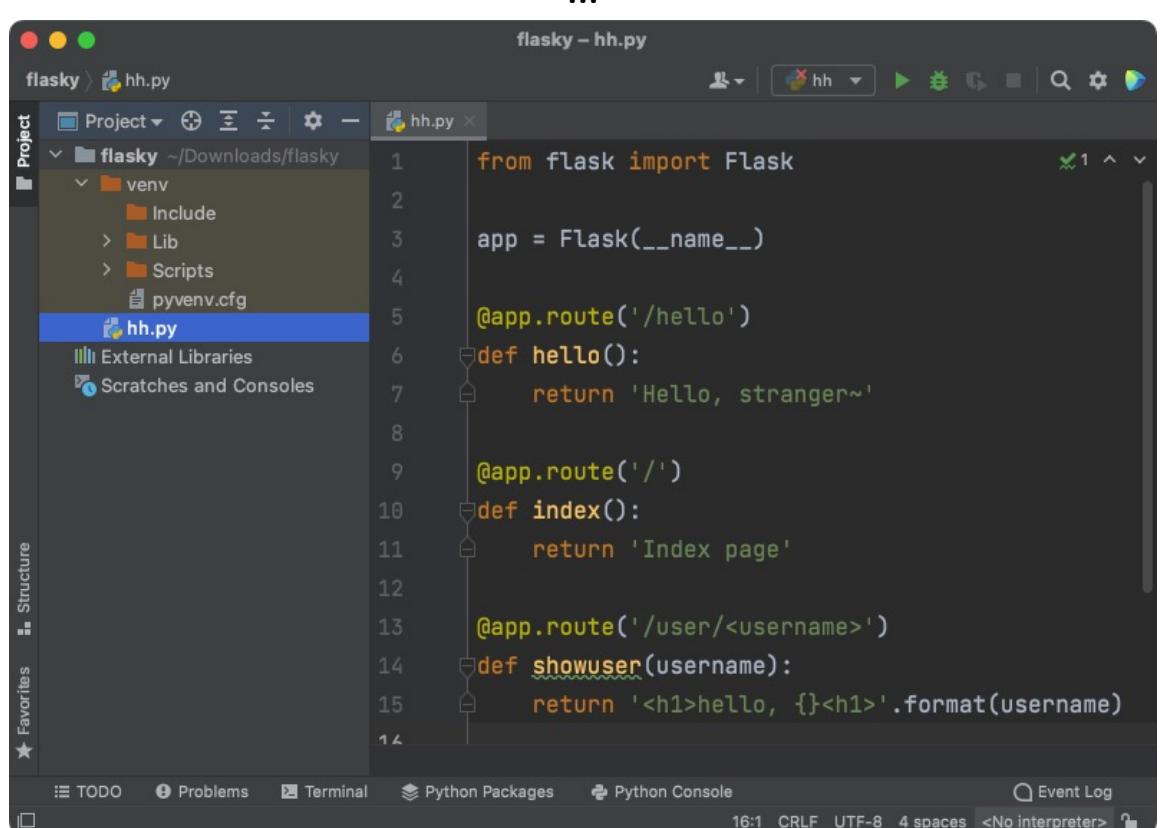
scripts (.py files)

Pycharm
Visual Studio Code

软件工程

Python

数据科学



The screenshot shows the PyCharm IDE interface. The project navigation bar at the top indicates the current project is 'flasky' and the file being edited is 'hh.py'. The left sidebar displays the project structure, including a virtual environment 'venv' with its subfolders 'Include', 'Lib', 'Scripts', and 'pyenv.cfg', and the file 'hh.py' which is currently selected. The main code editor area contains the following Python code:

```
from flask import Flask
app = Flask(__name__)
@app.route('/hello')
def hello():
    return 'Hello, stranger~'
@app.route('/')
def index():
    return 'Index page'
@app.route('/user/<username>')
def showuser(username):
    return '<h1>hello, {}</h1>'.format(username)
```

At the bottom of the interface, there are several tabs: 'TODO', 'Problems', 'Terminal', 'Python Packages', 'Python Console', and 'Event Log'. The status bar at the very bottom shows the file encoding as 'UTF-8', the line count as '16:1', and the character count as 'CRLF 4 spaces <No interpreter> ?'.

```
[1]: x = np.random.randn(4)
[2]: x + np.random.randn(4)
[3]:
[4]: array([1.12900000, 0.21856000, -1.38816000, -1.30690711])
[5]:
[6]: array([-0.83591211, -0.32119611, -0.47088411, -1.45663081])
```

IPython

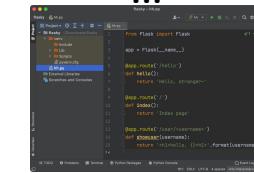


Jupyter

scripts (.py files)

Pycharm
Visual Studio Code

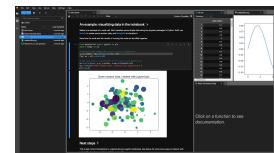
软件工程



Python

数据科学

```
[In [1]: import numpy as np
[In [2]: x = np.random.randn(1, 4)
[In [3]: x
Out[3]: array([-1.25920396,  0.31454629,  1.38824689, -1.30699719])
[In [4]: x = np.random.randn(1, 4)
[In [5]: x
Out[5]: array([-0.62501251, -0.32113616,  0.4738682 , -1.05463289])]
```



Ipython (2001年)

Jupyter

Python

软件工程

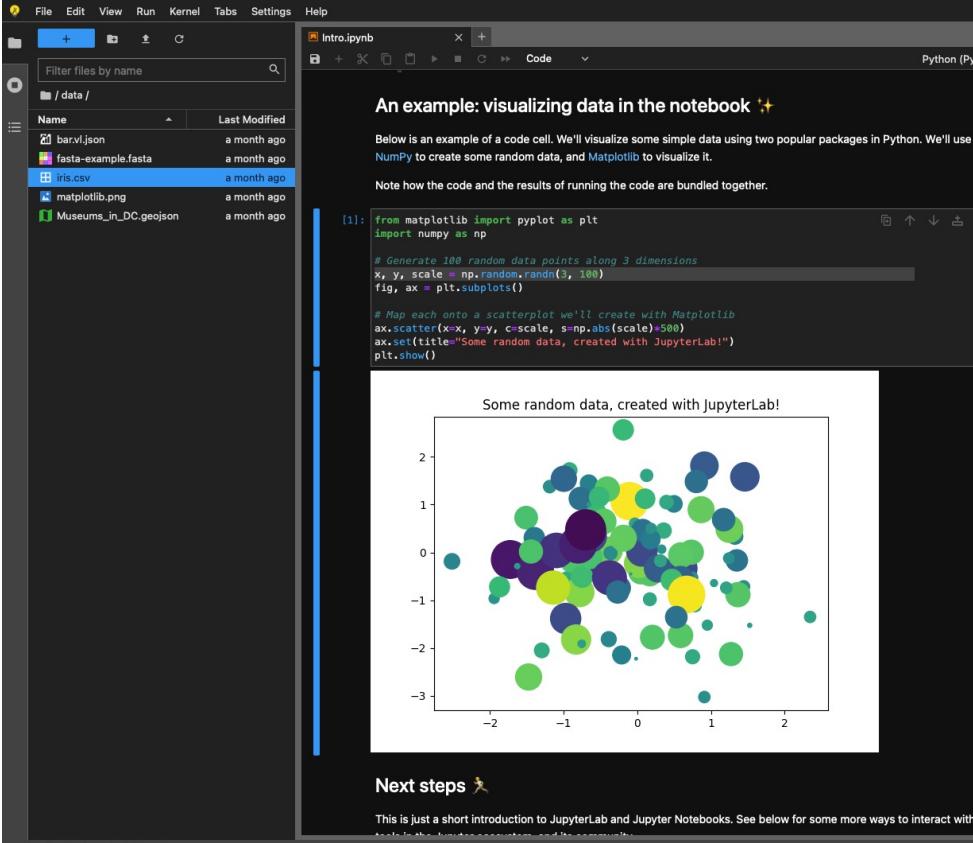
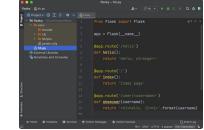
数据科学

scripts (.py files)



IPython

Pycharm
Visual Studio Code



An example: visualizing data in the notebook 

Below is an example of a code cell. We'll visualize some simple data using two popular packages in Python. We'll use NumPy to create some random data, and Matplotlib to visualize it.

Note how the code and the results of running the code are bundled together.

```
[1]: from matplotlib import pyplot as plt
import numpy as np

# Generate 100 random data points along 3 dimensions
x, y, scale = np.random.randn(3, 100)
fig, ax = plt.subplots()

# Map each onto a scatterplot we'll create with Matplotlib
ax.scatter(x, y, c=scale, s=np.abs(scale)*500)
ax.set_title("Some random data, created with JupyterLab!")
plt.show()
```

Some random data, created with JupyterLab!

sepal_length

	sepal_length
1	5.1
2	4.9
3	4.7
4	4.6
5	5
6	5.4
7	4.6
8	5
9	4.4
10	4.9
11	5.4
12	4.8
13	4.8
14	4.3
15	5.8
16	5.7
17	5.4

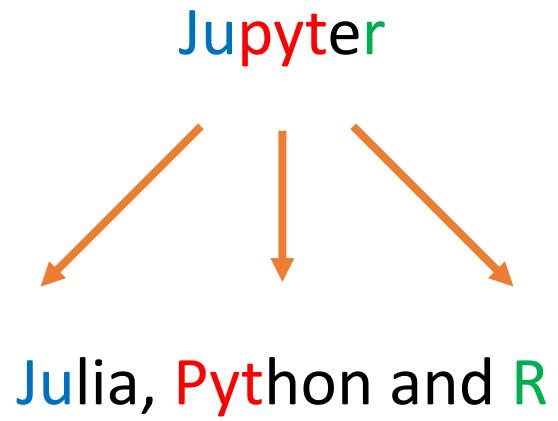
Click on a function to see documentation.

Next steps 

This is just a short introduction to JupyterLab and Jupyter Notebooks. See below for some more ways to interact with them.

Jupyter (2014年)

Jupyter名字的由来



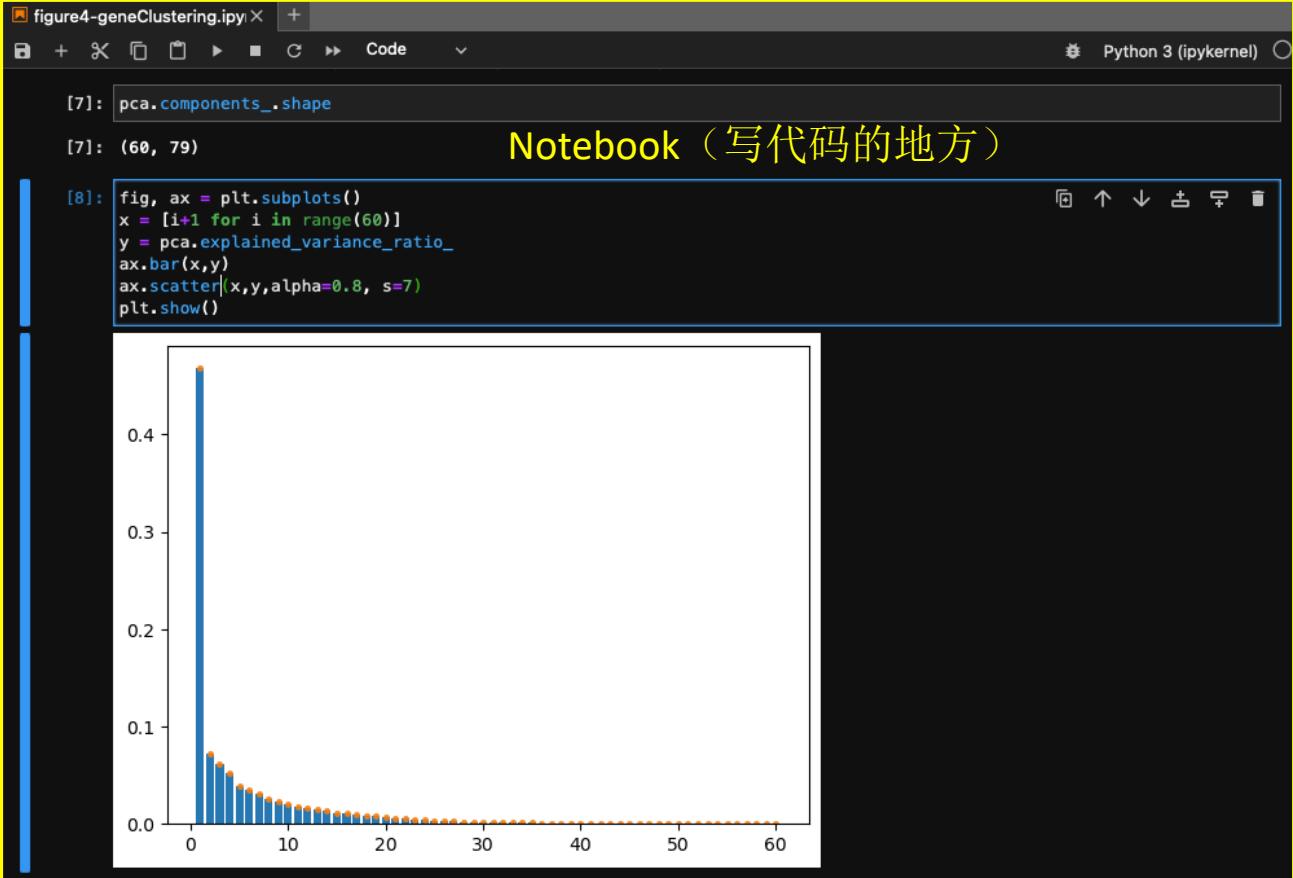
Jupiter (木星)



JupyterLab界面介绍

目录/文件 (command+b)

```
+ Filter files by name
  / ... / HPA-single_cell / figure4 /
Name Last Modified
gene_clust... 8 minutes ago
gene_clust... 4 days ago
umap_clust... 8 minutes ago
```



Show Contextual Help x +

Docstring:
A scatter plot of `*y*` vs. `*x*` with varying marker size and/or color.

Parameters

`x, y : float or array-like, shape (n,)`
The data positions.

`s : float or array-like, shape (n,), optional`
The marker size in points**2.
Default is ``rcParams['lines.markersize'] ** 2``.

`c : array-like or list of colors or color, optional`
The marker colors. Possible values:
- A scalar or sequence of n numbers to be mapped to colors using `*cmap*` and `*norm*`.

函数说明文档 (command+i)

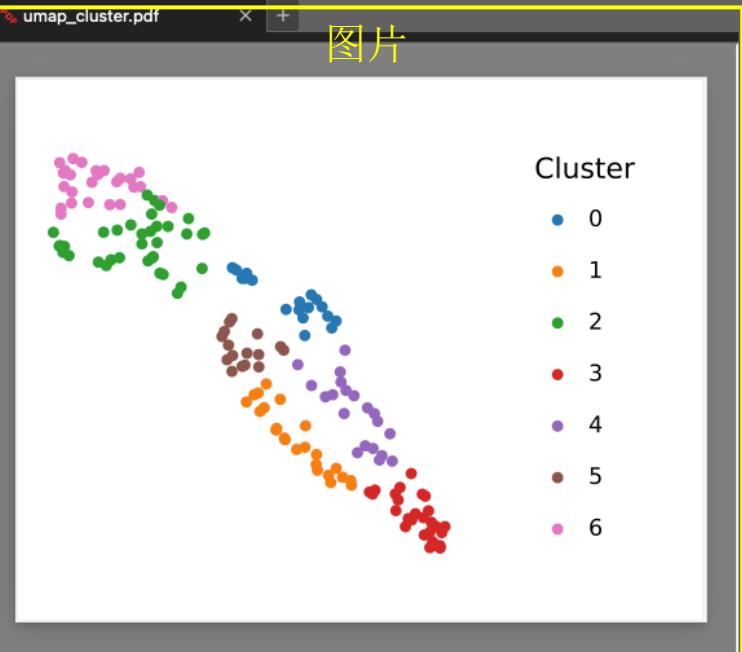
gene_clustering.tsv x + 表格

Delimiter: tab

	Gene	UMR
1	ABHD2	9.60
2	ADCYAP1R1	13.29
3	ADIPO1	9.75
4	ADIPO2	9.80
5	ADRA1A	12.58
6	ADRA1B	13.31
7	ADRA1D	15.32
8	ADRA2A	13.33
9	ADRA2B	14.20
10	ADRA2C	12.81
11	ADRB1	12.21
12	ADRB2	10.85
13	ADRB3	16.14
14	AGTR1	12.16
15	AGTR2	16.01
16	ALDH1A1	9.48
17	ALDH1A3	10.46

Terminal 1 终端

```
42.99 12/1 2844% user,33.84% 66 97 3
5sys,7:0210 3.29, 2.63, 2.58
530* core15.87% user, 3.59% sys, 0
.53% idler 8903 80:14.50 1
72982* com.apple.We 88116 00438.44 2
93.20 lcom.apple.We 88116 00438.44 2
13040*top 1 6.6 00:33.98 2
08.99 2/1 Oiod 52601 09637.27 9
18315 Wind93lserver 5.5 80836.27 3
20.11 22/1 6ft Po 48.1 10701.40 6
72982* com.apple.We 25.9 01:2
394* WindowServer 100.2 80: 5:47
1* launchd 8.2 01:49.30 3
73040* top 6.8 00:12.73 2
/1 0com.apple.We 4.9 01:50.58 1
430* coreaudiod 4.3 09:27.52 2
72782* com.apple.We 4.2 00:38.35 6
0* kernel_task 3.8 36:05.51 5
394* WindowServer 2.6 80:27.06 2
389* AirPlayXPCSe 2.1 00:02.49 1
2487* Microsoft Po 2.0 10:09.51 3
385* bluetoothhd 1.8 05:16.12 1
6 77* NeteaseMusic 1.5 00:39.39 2
1956* com.apple.We 1.4 00:34.76 2
620* donotdisturb 1.0 11:01.89 6
72720* VDecoderXPC 1.0 00:26.83 3
674* Notification 0.8 06:18.07 5
```





File Edit View Run Kernel Tabs Settings Help

显示和选择当前Notebook

的编程语言环境



Untitled1.ipynb

● +



+



☒



□



▶



▶

Code



Python 3 (ipykernel)



对cell进行添加、复制、粘贴、运行等操作（左），中断
和重启整个Notebook（中），显示当前cell的类型（右）

运行状态

[]:

[]:

⟳ ⌂ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋

这叫一个cell，或者说代码块，运行的结果会直接展示在下方

Simple



0

\$

_

2



Python 3 (ipykernel) | Idle

Mode: Command



Ln 1, Col 1

Untitled1.ipynb

用Jupyter的*Hello World*介绍Notebook的两种操作模式

新建一个Notebook

A screenshot of a Jupyter Notebook interface. The top bar shows a cell identifier [:]. Below it is a toolbar with icons for back, forward, search, and other operations. The status bar at the bottom indicates 'Python 3 (ipykernel) | Idle'. In the center, the text 'Mode: Command' is displayed in a box with a yellow border. The main area is empty, showing a cursor in the first cell.

默认是命令模式

A screenshot of a Jupyter Notebook interface. The top bar shows a cell identifier [:]. Below it is a toolbar with icons for back, forward, search, and other operations. The status bar at the bottom indicates 'Python 3 (ipykernel) | Idle'. In the center, the text 'Mode: Edit' is displayed in a box with a yellow border. The main area is empty, showing a cursor in the first cell.

按下ENTER进入编辑模式

A screenshot of a Jupyter Notebook interface. The top bar shows a cell identifier [1] : . Below it is a toolbar with icons for back, forward, search, and other operations. The status bar at the bottom indicates 'Python 3 (ipykernel) | Idle'. In the center, the code 'print("Hello World!")' is written in green, and the output 'Hello World!' is displayed below it. The text 'Mode: Command' is shown in a box with a yellow border. The main area is empty, showing a cursor in the second cell.

编辑模式下输入代码并运行

A screenshot of a Jupyter Notebook interface. The top bar shows a cell identifier [1] : . Below it is a toolbar with icons for back, forward, search, and other operations. The status bar at the bottom indicates 'Python 3 (ipykernel) | Idle'. In the center, the code 'print("Hello World!")' is written in green, and the output 'Hello World!' is displayed below it. The text 'Mode: Command' is shown in a box with a yellow border. The main area is empty, showing a cursor in the second cell.

命令模式下（按下ESC进入）的输入视为各种命令，比如这里依次键入c和v的效果是复制并粘贴了一个cell

Notebook的两种操作模式

编辑模式（按下ENTER进入）：写代码、在一个cell内部进行编辑。

命令模式（按下ESC进入）：进入后无法写代码，而是对整个cell进行操作，比如复制、删除、改变cell类型等。

三种cell类型

This screenshot shows a Jupyter Notebook cell in 'Markdown' mode. The cell contains the following content:

```
## I'm a subtitle.  
[ ]: print('Hello World!')  
我只是一段最朴素的文字。
```

The 'Markdown' tab is highlighted with a yellow box.

This screenshot shows a Jupyter Notebook cell in 'Code' mode. The cell contains the following content:

```
## I'm a subtitle.  
[ ]: print('Hello World!')  
我只是一段最朴素的文字。
```

The 'Code' tab is highlighted with a yellow box.

This screenshot shows a Jupyter Notebook cell in 'Raw' mode. The cell contains the following content:

```
## I'm a subtitle.  
[ ]: print('Hello World!')  
我只是一段最朴素的文字。
```

The 'Raw' tab is highlighted with a yellow box.

三种cell类型

<code>## I'm a subtitle.</code>	Markdown（命令模式下输入m）
[]: <code>print('Hello World!')</code>	Code（命令模式下输入y）
我只是一段最朴素的文字。	Raw（命令模式下输入r） 回 ↑ ↓ 古 早 去

Run ↓

I'm a subtitle.	以Markdown的语法显示
[3]: <code>print('Hello World!')</code>	以当前选择的语言（Python）运行并将结果直接显示在下方
我只是一段最朴素的文字。	无变化

File Edit View Run Kernel Tabs Settings Help

UNTITLED.ipynb

Untitled.ipynb +

Python 3 (ipykernel)

Introduction to the JupyterLab

An example: visualizing data in the notebook ✨

Next steps 🏃

Other notebooks in this demo

Other sources of information in Jupyter

Introduction to the JupyterLab

JupyterLab is a next-generation web-based user interface for Project Jupyter. It enables you to work with documents and activities such as Jupyter notebooks, text editors, terminals, and custom components in a flexible, integrated, and extensible manner. It is the interface that you're looking at right now.

For an overview of the JupyterLab interface, see the **JupyterLab Welcome Tour** on this page, by going to `Help > Welcome Tour` and following the prompts.

> **See Also:** For a more in-depth tour of JupyterLab with a full environment that runs in the cloud, see [\[the JupyterLab introduction on Binder\]](https://mybinder.org/v2/gh/jupyterlab/jupyterlab-demo/HEAD?urlpath=lab/tree/demo)(<https://mybinder.org/v2/gh/jupyterlab/jupyterlab-demo/HEAD?urlpath=lab/tree/demo>).

An example: visualizing data in the notebook ✨

Below is an example of a code cell. We'll visualize some simple data using two popular packages in Python. We'll use [\[NumPy\]](#)(<https://numpy.org/>) to create some random data, and [\[Matplotlib\]](#)(<https://matplotlib.org>) to visualize it.

Note how the code and the results of running the code are bundled together.

```
[3]: from matplotlib import pyplot as plt
      import numpy as np

      # Generate 100 random data points along 3 dimensions
      x, y, scale = np.random.randn(3, 100)
      fig, ax = plt.subplots()

      # Map each onto a scatterplot we'll create with Matplotlib
      ax.scatter(x=x, y=y, c=scale, s=np.abs(scale)*500)
      ax.set(title="Some random data, created with JupyterLab!")
      plt.show()

    •••
```

Next steps 🏃

This is just a short introduction to JupyterLab and Jupyter Notebooks. See below for some more ways to interact with tools in the Jupyter ecosystem, and its community.

Other notebooks in this demo

Here are some other notebooks in this demo. Each of the items below corresponds to a file or folder in the **file browser to the left**.

- [\[**`Lorenz.ipynb`**\]\(Lorenz.ipynb\)](#) (Lorenz.ipynb) uses Python to demonstrate interactive visualizations and computations around the [\[Lorenz system\]\(https://en.wikipedia.org/wiki/Lorenz_system\)](#). It shows off basic Python functionality, including more visualizations.

Simple 0 \$ 5 Python 3 (ipykernel) | Idle Mode: Command Ln 4, Col 1 Untitled.ipynb

File Edit View Run Kernel Tabs Settings Help

UNTITLED.ipynb

Untitled.ipynb +

Python 3 (ipykernel)

Introduction to the JupyterLab

An example: visualizing data in the notebook ✨

Next steps 🚶

Other notebooks in this demo

Other sources of information in Jupyter

Introduction to the JupyterLab

JupyterLab is a next-generation web-based user interface for Project Jupyter. It enables you to work with documents and activities such as Jupyter notebooks, text editors, terminals, and custom components in a flexible, integrated, and extensible manner. It is the interface that you're looking at right now.

For an overview of the JupyterLab interface, see the [JupyterLab Welcome Tour](#) on this page, by going to Help → Welcome Tour and following the prompts.

See Also: For a more in-depth tour of JupyterLab with a full environment that runs in the cloud, see the [JupyterLab introduction on Binder](#).

An example: visualizing data in the notebook ✨

Below is an example of a code cell. We'll visualize some simple data using two popular packages in Python. We'll use NumPy to create some random data, and Matplotlib to visualize it.

Note how the code and the results of running the code are bundled together.

```
[1]: from matplotlib import pyplot as plt
import numpy as np

# Generate 100 random data points along 3 dimensions
x, y, scale = np.random.randn(3, 100)
fig, ax = plt.subplots()

# Map each onto a scatterplot we'll create with Matplotlib
ax.scatter(x=x, y=y, c=scale, s=np.abs(scale)*500)
ax.set(title="Some random data, created with JupyterLab!")
plt.show()
```

Some random data, created with JupyterLab!

Simple 0 \$ 5 Python 3 (ipykernel) | Idle

Mode: Command ↵ Ln 1, Col 1 Untitled.ipynb

File Edit View Run Kernel Tabs Settings Help

UNTITLED.ipynb

Untitled.ipynb +

Python 3 (ipykernel)

Introduction to the JupyterLab

An example: visualizing data in the notebook ✨

Next steps 🏃

Other notebooks in this demo

Other sources of information in Jupyter

Code

```
ax.scatter(x, y, c=cstate, s=100, alpha=0.5)
ax.set(title="Some random data, created with JupyterLab!")
plt.show()
```

...

Next steps 🏃

This is just a short introduction to JupyterLab and Jupyter Notebooks. See below for some more ways to interact with tools in the Jupyter ecosystem, and its community.

Other notebooks in this demo

Here are some other notebooks in this demo. Each of the items below corresponds to a file or folder in the **file browser to the left**.

- [`Lorenz.ipynb`](#)(Lorenz.ipynb) uses Python to demonstrate interactive visualizations and computations around the [\[Lorenz system\]](#)(https://en.wikipedia.org/wiki/Lorenz_system). It shows off basic Python functionality, including more visualizations, data structures, and scientific computing libraries.
- [`sqlite.ipynb`](#)(sqlite.ipynb) demonstrates how an in-browser sqlite kernel to run your own SQL commands from the notebook. It uses the [\[jupyterlite/xeus-sqlite-kernel\]](#)(<https://github.com/jupyterlite/xeus-sqlite-kernel>).

Other sources of information in Jupyter

- **More on using JupyterLab:** See [\[the JupyterLab documentation\]](#)(<https://jupyterlab.readthedocs.io/en/stable/>) for more thorough information about how to install and use JupyterLab.
- **More interactive demos:** See [\[try.jupyter.org\]](#)(<https://try.jupyter.org>) for more interactive demos with the Jupyter ecosystem.
- **Learn more about Jupyter:** See [\[the Jupyter community documentation\]](#)(<https://docs.jupyter.org>) to learn more about the project, its community and tools, and how to get involved.
- **Join our discussions:** The [\[Jupyter Community Forum\]](#)(<https://discourse.jupyter.org>) is a place where many in the Jupyter community ask questions, help one another, and discuss issues around interactive computing and our ecosystem.

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 change (σ, β, ρ) with ipywidgets and examine the trajectories.

Simple 0 \$ 5 Python 3 (ipykernel) | Idle

Mode: Command ⌘ Ln 7, Col 1 Untitled.ipynb

File Edit View Run Kernel Tabs Settings Help

UNTITLED.ipynb

Untitled.ipynb Python 3 (ipykernel)

Introduction to the JupyterLab

An example: visualizing data in the notebook ✨

Next steps 🏃

Other notebooks in this demo

Other sources of information in Jupyter

Next steps 🏃

This is just a short introduction to JupyterLab and Jupyter Notebooks. See below for some more ways to interact with tools in the Jupyter ecosystem, and its community.

Other notebooks in this demo

Here are some other notebooks in this demo. Each of the items below corresponds to a file or folder in the **file browser to the left**.

- **Lorenz.ipynb** uses Python to demonstrate interactive visualizations and computations around the [Lorenz system](#). It shows off basic Python functionality, including more visualizations, data structures, and scientific computing libraries.
- **sqlite.ipynb** demonstrates how an in-browser sqlite kernel to run your own SQL commands from the notebook. It uses the [jupyterlite/xeus-sqlite-kernel](#).

Other sources of information in Jupyter

- **More on using JupyterLab:** See the [JupyterLab documentation](#) for more thorough information about how to install and use JupyterLab.
- **More interactive demos:** See [try.jupyter.org](#) for more interactive demos with the Jupyter ecosystem.
- **Learn more about Jupyter:** See the [Jupyter community documentation](#) to learn more about the project, its community and tools, and how to get involved.
- **Join our discussions:** The [Jupyter Community Forum](#) is a place where many in the Jupyter community ask questions, help one another, and discuss issues around interactive computing and our ecosystem.

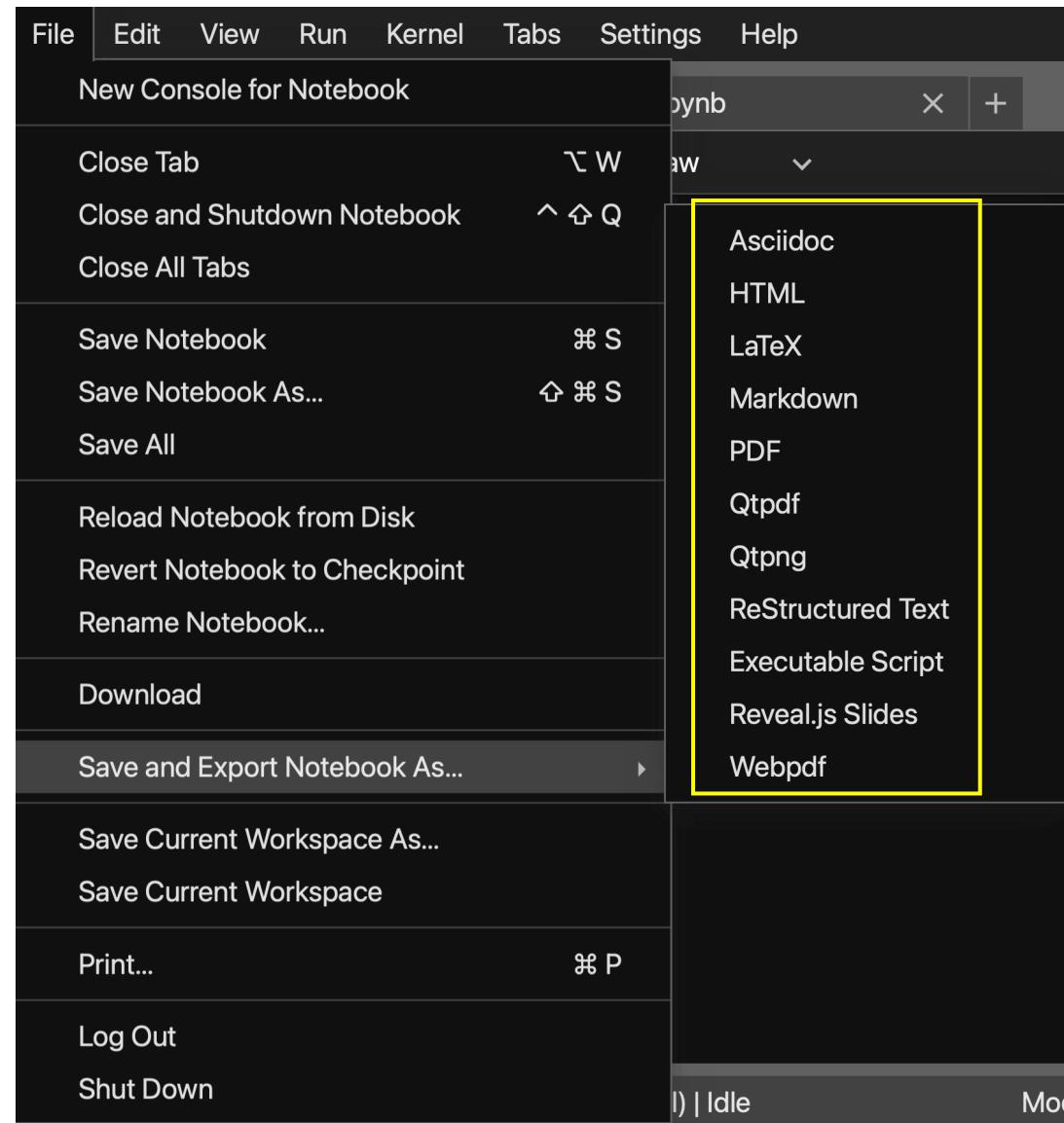
We explore the Lorenz system of differential equations:

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

Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.

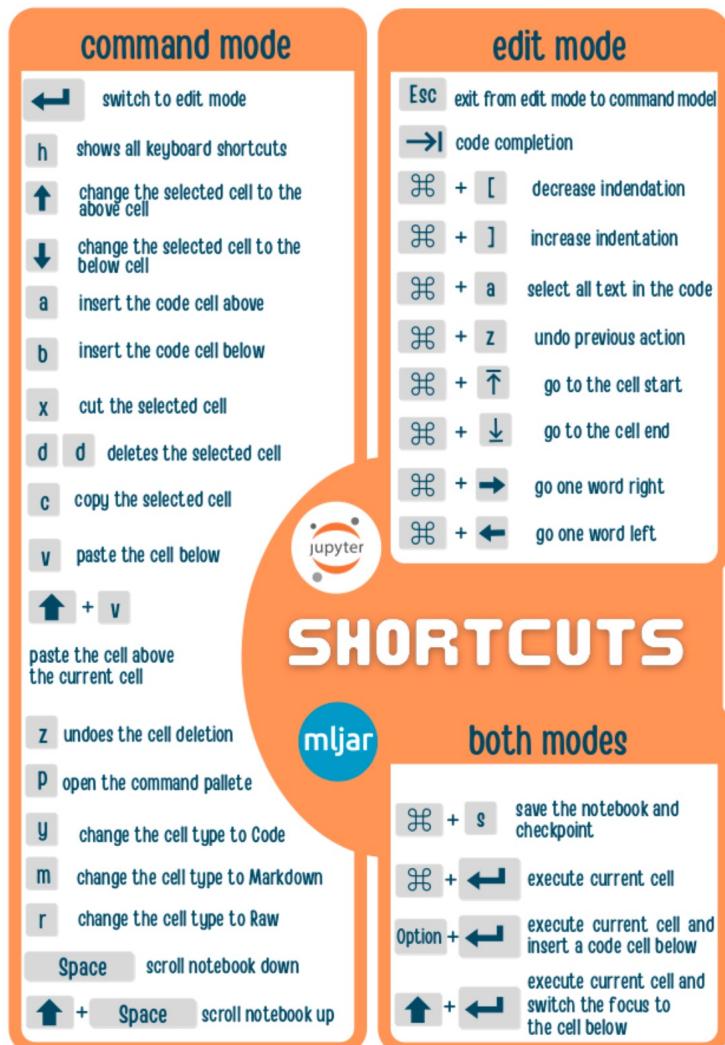
Simple 0 \$ 5 Python 3 (ipykernel) | Idle Mode: Command ⚙️ Ln 1, Col 1 Untitled.ipynb

Jupyter可以导出为多种格式去分享



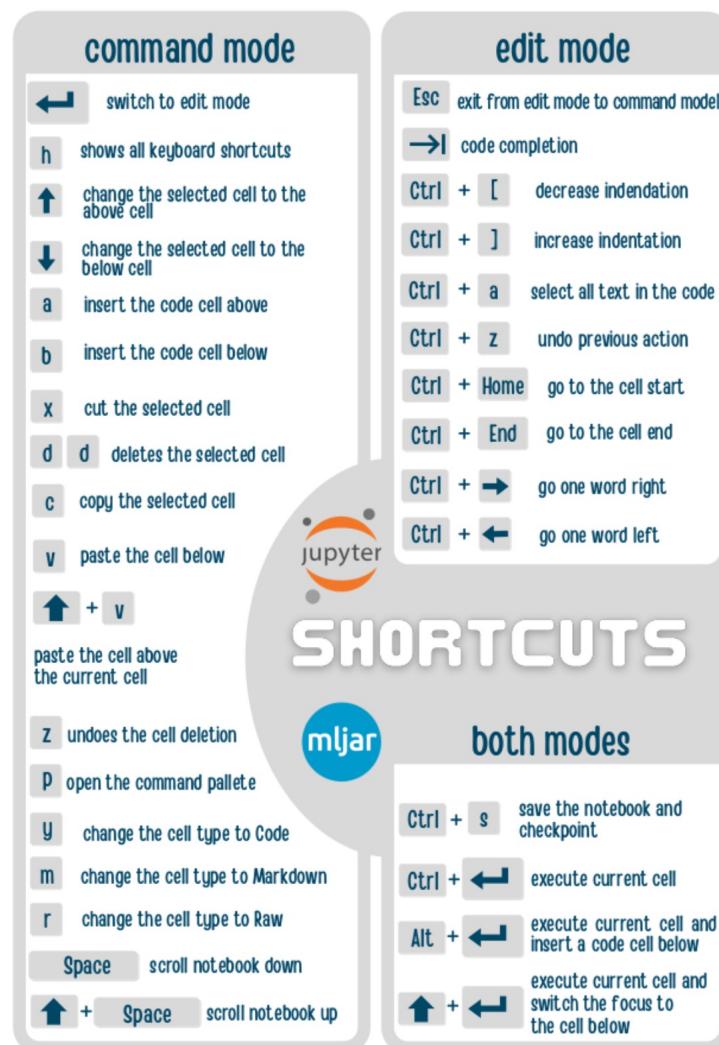
一些常用的快捷键

For Mac users



命令模式下一些常用的快捷键跟vim的类似

For Windows users



Running JupyterLab remotely

The screenshot shows a documentation page for JupyterServer. At the top, there's a navigation bar with links for 'Users', 'Operators' (which is highlighted in blue), 'Developers', 'Contributors', and 'Other'. There's also a search icon and a help icon.

The main content area has a sidebar titled 'Section Navigation' containing links like 'Managing multiple extensions', 'Configuring Extensions', 'Migrating from Notebook Server', 'Running a public Jupyter Server' (which is also highlighted in blue), 'Security in the Jupyter Server', and 'Configuring Logging'.

The main content area features a large title 'Running a public Jupyter Server'. Below it, a note states: 'The Jupyter Server uses a two-process kernel architecture based on ZeroMQ, as well as Tornado for serving HTTP requests.'

There are two callout boxes: one labeled 'Note' stating that by default, the server runs locally at 127.0.0.1:8888 and is accessible only from 'localhost'. It also provides the URL 'http://127.0.0.1:8888'. Another callout box labeled 'Important' cautions against running a multi-user server with a single user, noting potential security risks and the need for a Unix server.

At the bottom, there's a section titled 'Securing a Jupyter server' with a note about protecting the server with a password.

The right sidebar is titled 'On this page' and lists various sub-topics under 'Running a public Jupyter Server', such as 'Prerequisite: A Jupyter server configuration file', 'Automatic Password setup', 'Preparing a hashed password', 'Adding hashed password to your notebook configuration file', 'Using SSL for encrypted communication', 'Running a public notebook server', 'Running the notebook with a customized URL prefix', 'Embedding the notebook in another website', 'Using a gateway server for kernel management', and 'Known issues'. There's also a link to 'Show Source'.

<https://jupyter-server.readthedocs.io/en/stable/operators/public-server.html>

我的JupyterLab工作流的日常

1. 打开VPN



2. 打开浏览器



3. 输入Jupyter Server运行
的地址和密码
(上一张slide中你设置的)

Q https://192.168.10.9:27777

6. 关掉JupyterLab网页，
结束工作



5. 保存Notebook



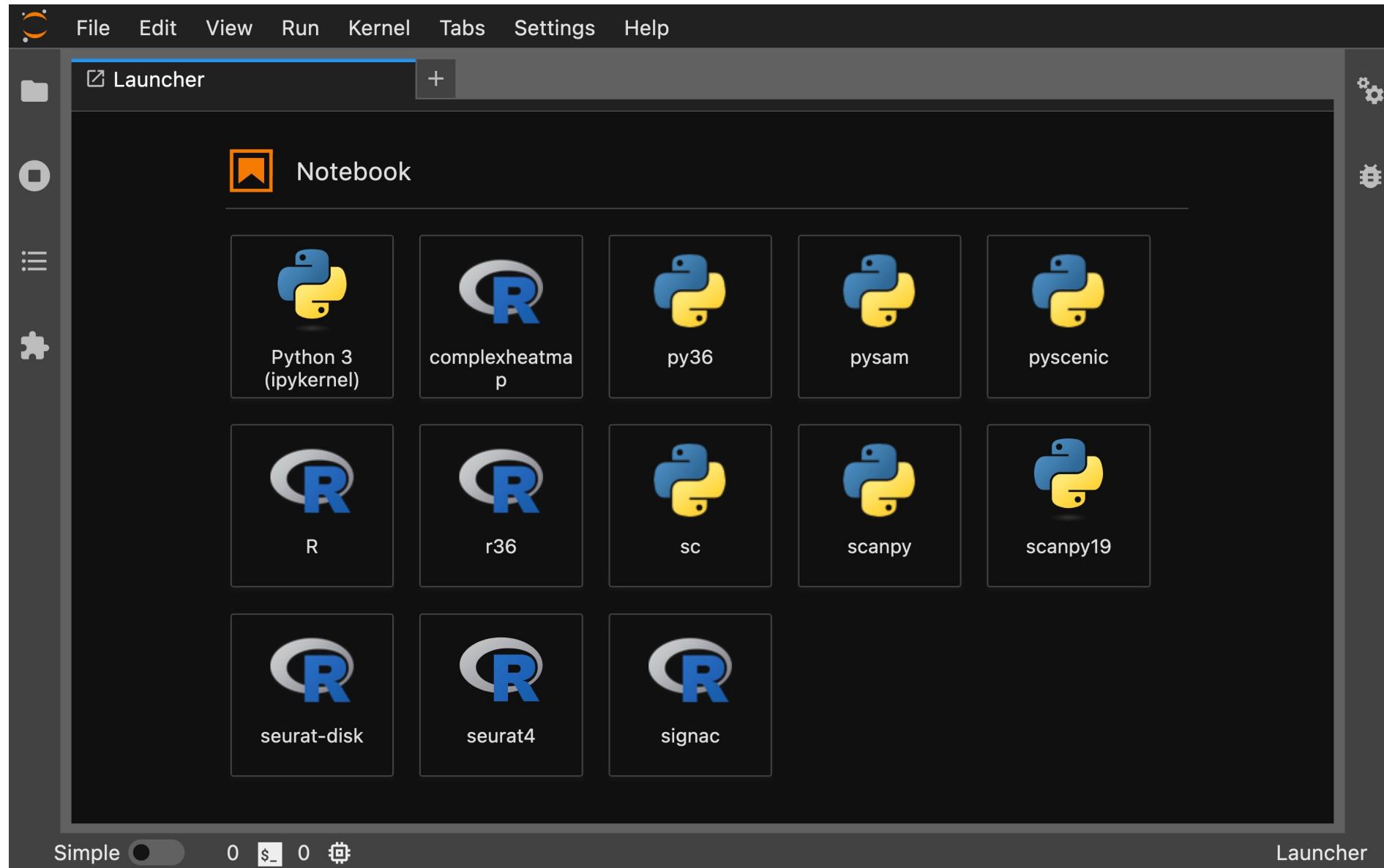
4. 开始/继续工作



JupyterLab的多编程语言支持

1. 安装**，建议给每一种语言单独建立一个conda环境。
2. Google或Bing搜索“i**kernel”，然后点进第一个查看下载和安装方法。
3. **代表编程语言的名字，比如Julia、R、Java。

我用不同的conda环境来解决软件之间的冲突



Thanks for your attention!