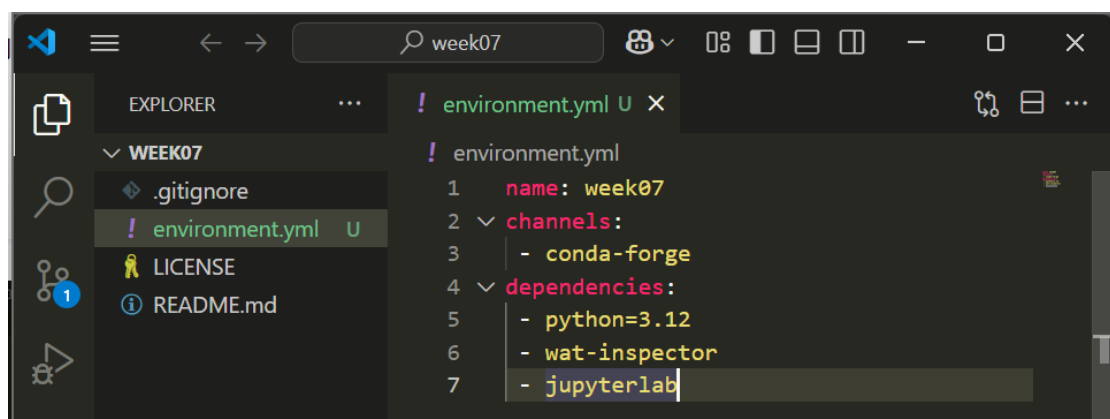


## 第七周 学习笔记——数据可视化与交互（初级）

一、安装 Fork 第 07 周打卡仓库至自己的名下，然后将名下的这个仓库 Clone 到本地计算机

```
Lenovo@DESKTOP-C70G2VK MINGW64 /e/研究生上课资料/研一下 上课资料/金融编程与计算
$ git clone git@gitcode.com:twinkledahaha/week07.git
Cloning into 'week07'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), 8.45 KiB | 4.22 MiB/s, done.
(base)
```

二、用 VS Code 打开项目目录，新建 `environment.yml` 文件，指定安装 Python 3.12 和 `jupyterlab`，然后运行 `conda env create` 命令创建 Conda 环境



```
Lenovo@DESKTOP-C70G2VK MINGW64 /e/研究生上课资料/研一下 上课资料
/金融编程与计算/week07 (main)
$ conda env create
D:\Anaconda\Anaconda3\Lib\argparse.py:2006: FutureWarning: `remote_definition` is deprecated and will be removed in 25.9. Use `conda env create --file=URL` instead.
  action(self, namespace, argument_values, option_string)
Retrieving notices: ...working... done
Channels:
- conda-forge
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

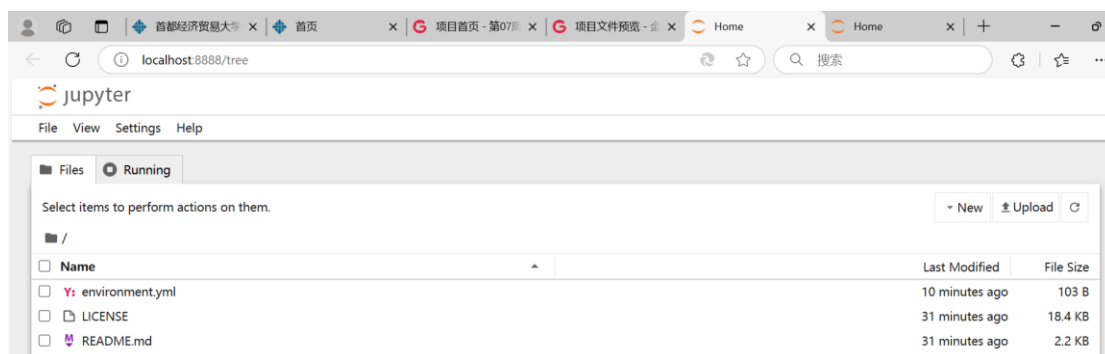
done
#
# To activate this environment, use
#
#     $ conda activate week07
#
# To deactivate an active environment, use
#
#     $ conda deactivate
#
(base)
```

三、在项目目录下，运行 **jupyter lab** 命令，启动 后端 (Backend) 服务，在浏览器里粘贴地址访问 前端 (Frontend) 页面

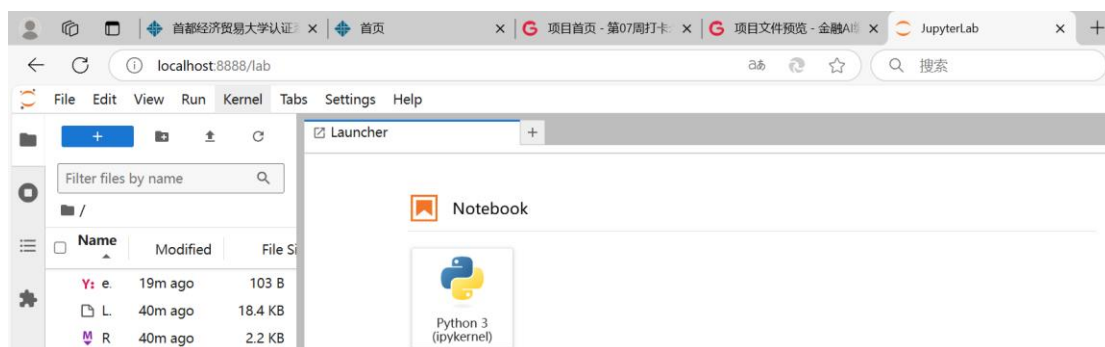
### 3.1 运行 jupyter notebook

```
MINGW64:/e/研究生上课资料 × + v
eek07
[I 2025-04-22 18:34:22.001 ServerApp] Jupyter Server 2.14.1 is running at:
[I 2025-04-22 18:34:22.001 ServerApp] http://localhost:8888/tree?token=19e0eb4b70fb3e740bef4f16b04028997c72257d6b32e13f
[I 2025-04-22 18:34:22.001 ServerApp] http://127.0.0.1:8888/tree?token=19e0eb4b70fb3e740bef4f16b04028997c72257d6b32e13f
[I 2025-04-22 18:34:22.001 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2025-04-22 18:34:22.056 ServerApp]

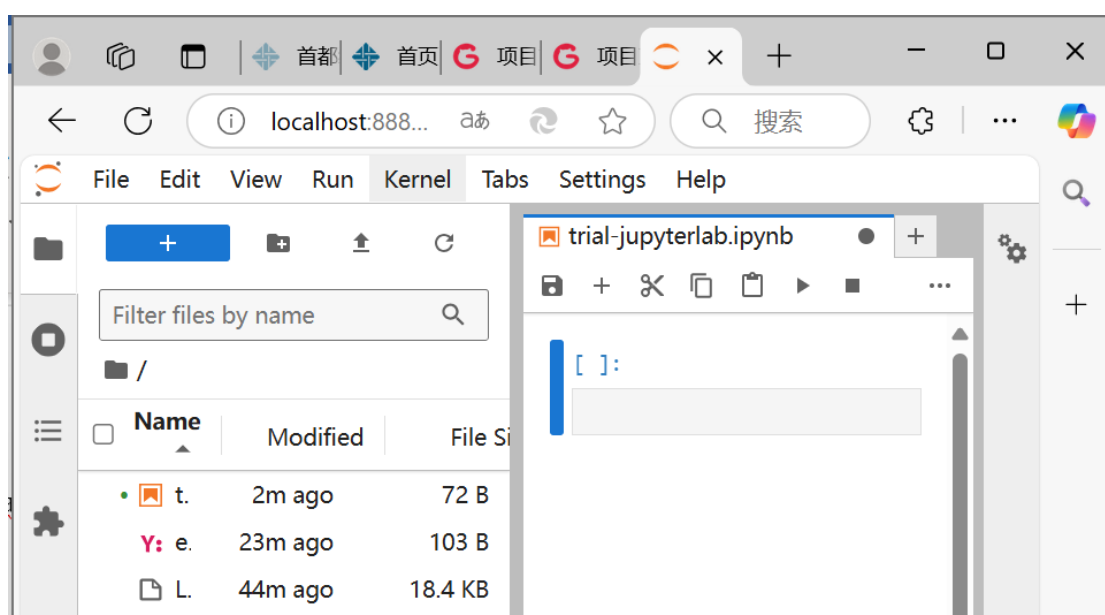
To access the server, open this file in a browser:
file:///C:/Users/Lenovo/AppData/Roaming/jupyter/runtime/jpserver-4808-open.html
Or copy and paste one of these URLs:
http://localhost:8888/tree?token=19e0eb4b70fb3e740bef4f16b04028997c72257d6b32e13f
http://127.0.0.1:8888/tree?token=19e0eb4b70fb3e740bef4f16b04028997c72257d6b32e13f
[I 2025-04-22 18:34:22.263 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyright, python-language-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
0.01s - Debugger warning: It seems that frozen modules are being used, which may
0.00s - make the debugger miss breakpoints. Please pass -Xfrozen_modules=off
0.00s - to python to disable frozen modules.
0.00s - Note: Debugging will proceed. Set PYDEVD_DISABLE_FILE_VALIDATION=1 to disable this validation.
```



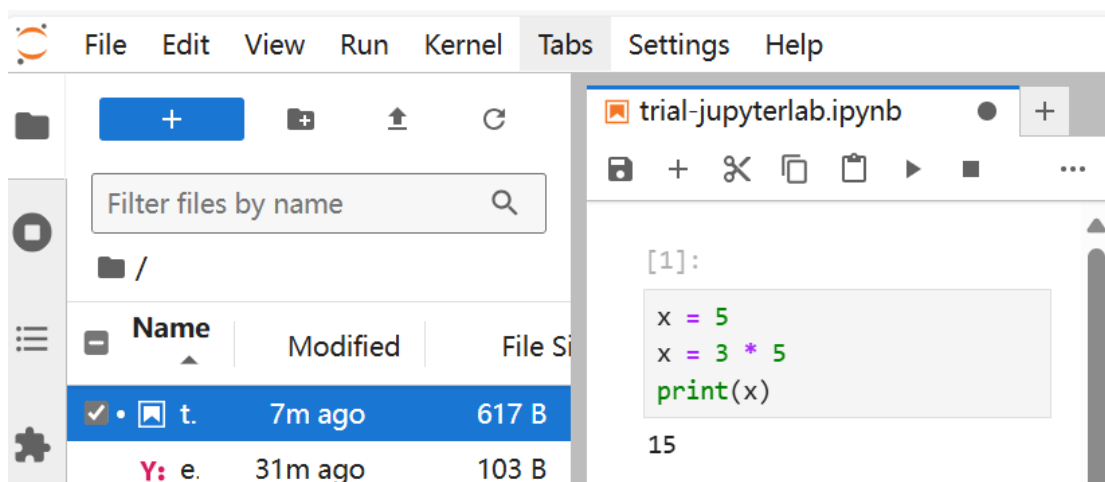
### 3.2 运行 jupyter lab



四、在 JupyterLab 页面里，新建一个 Notebook，改名为 **trial-jupyterlab.ipynb**

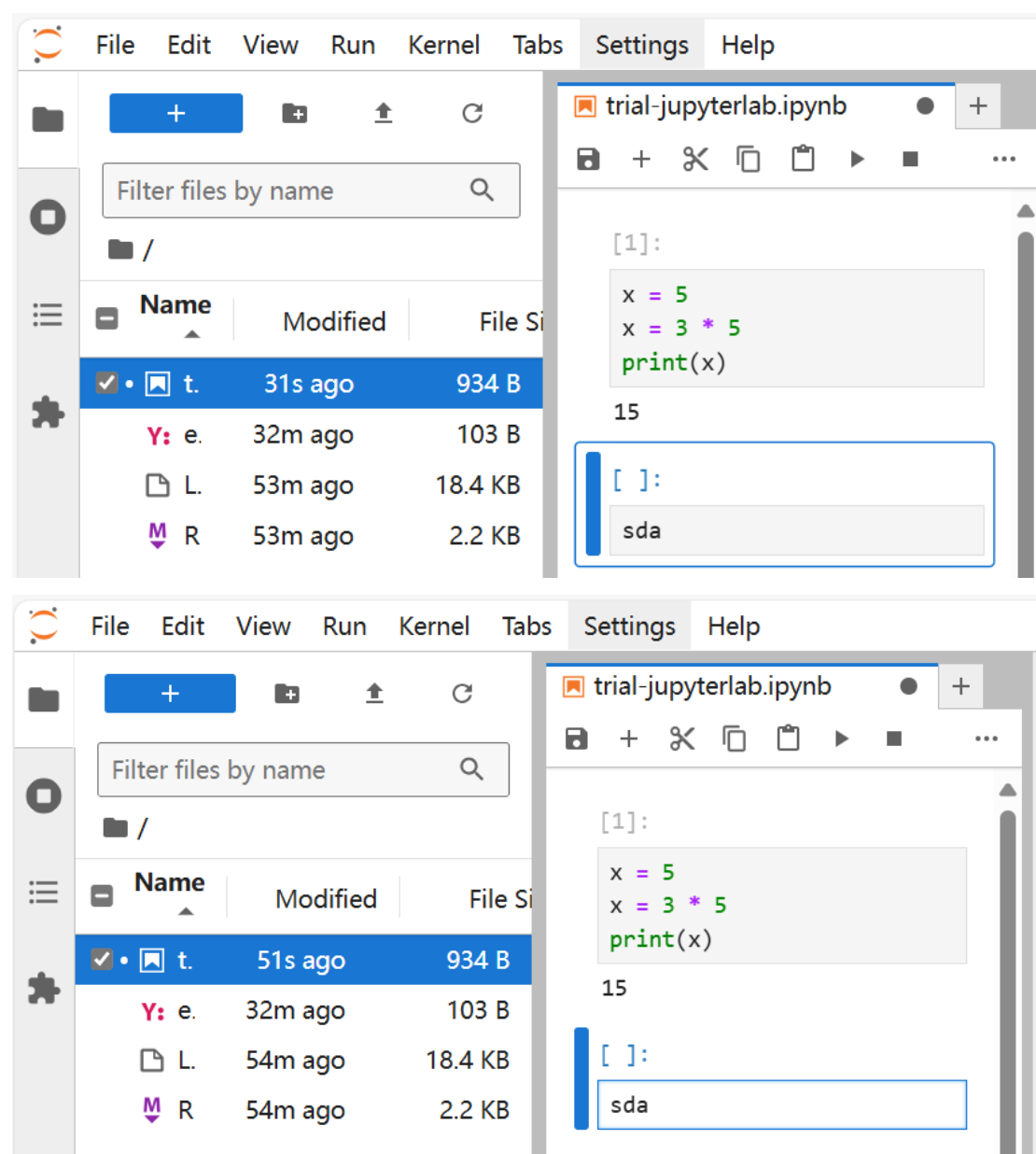


4.1 在单元格 (Cell) 里编写 Python 代码, 按 **Shift+Enter** 运行 Cell 并下移



4.2 在单元格 (Cell) 上按 **ESC** 切换到 命令模式 (command mode), 按

**Enter** 切换到 编写模式 (edit mode)



4.3 在单元格 (Cell) 的命令模式下, 按 **j** 选择下一个, 按 **k** 选择上一个, 按 **a** 在上方添加, 按 **b** 在下方添加, 按 **dd** 删除, 按住 **Shift** 多选, 按 **x** 剪切, 按 **c** 复制, 按 **v** 粘贴, 按 **Shift+M** 合并, 按 **z** 撤销, 按 **Shift+Z** 重做, 按 **Shift+L** 显示/隐藏代码行号

4.4 在单元格 (Cell) 的编写模式下, 按 **Ctrl+Shift+-** 切分单元格

4.5 按按钮显示/隐藏 **Minimap**

4.6 运行单元格 (Cell) 注意序号单调递增

4.7 单元格最后一行如果是 表达式 (expression) 且运行后返回的对象不是

**None**，则计输出 (Out)，否则只计输入 (In)，序号为 **i** 的输出，可以用 **\_i** 变量来引用

```
[2]: 1 a = [2, 4, 9]
      2 a

[2]: [2, 4, 9]

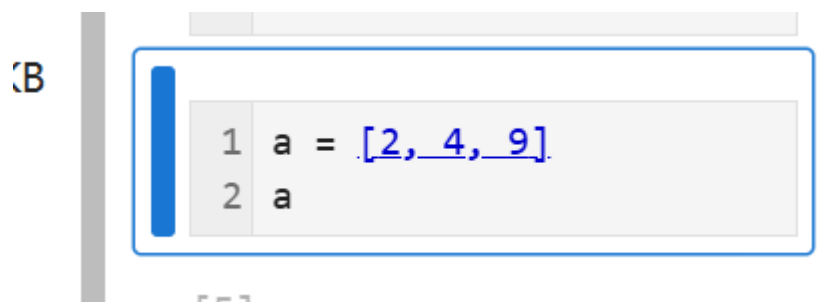
[5]: 1 _2 + [1, 1, 1]
      2 a

[5]: [2, 4, 9, 1, 1, 1]
```

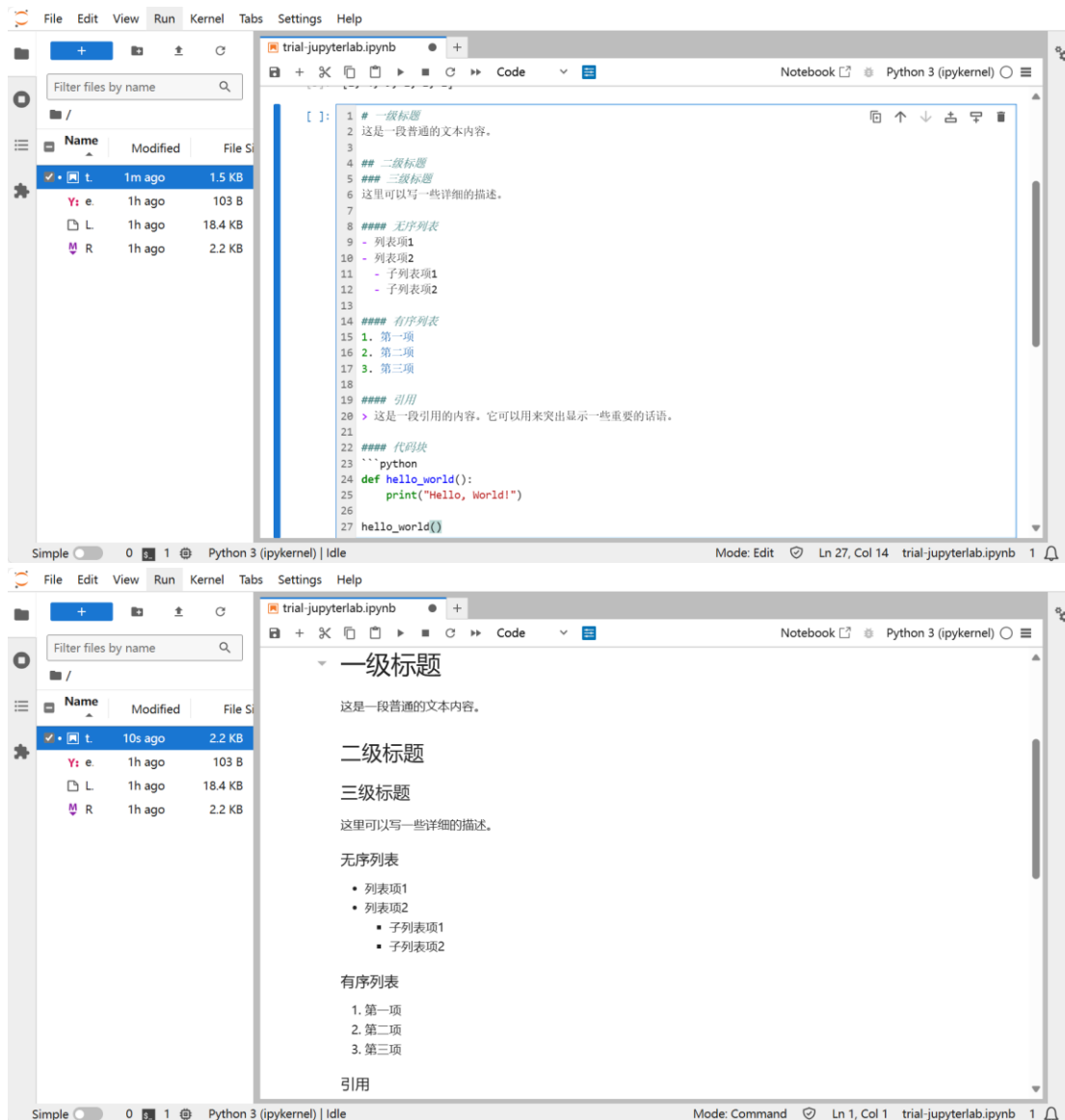
4.8 单元格 (Cell) 序号为 \* 表示代码运行中，尚未返回，按 **ii** 可以打断 (KeyboardInterrupt) (类似于终端的 **Ctrl+C**)

4.9 在单元格 (Cell) 的命令模式下，按 **00** 重启后端 Python 解释器 (被 Jupyter 称为 **Kernel**)，重启后需要从上至下重新运行一遍代码 (**Shift+Enter**)，运行前建议先在菜单里选择 “**Edit / Clear Outputs of All Cells**” 清空全部页面显示的输出

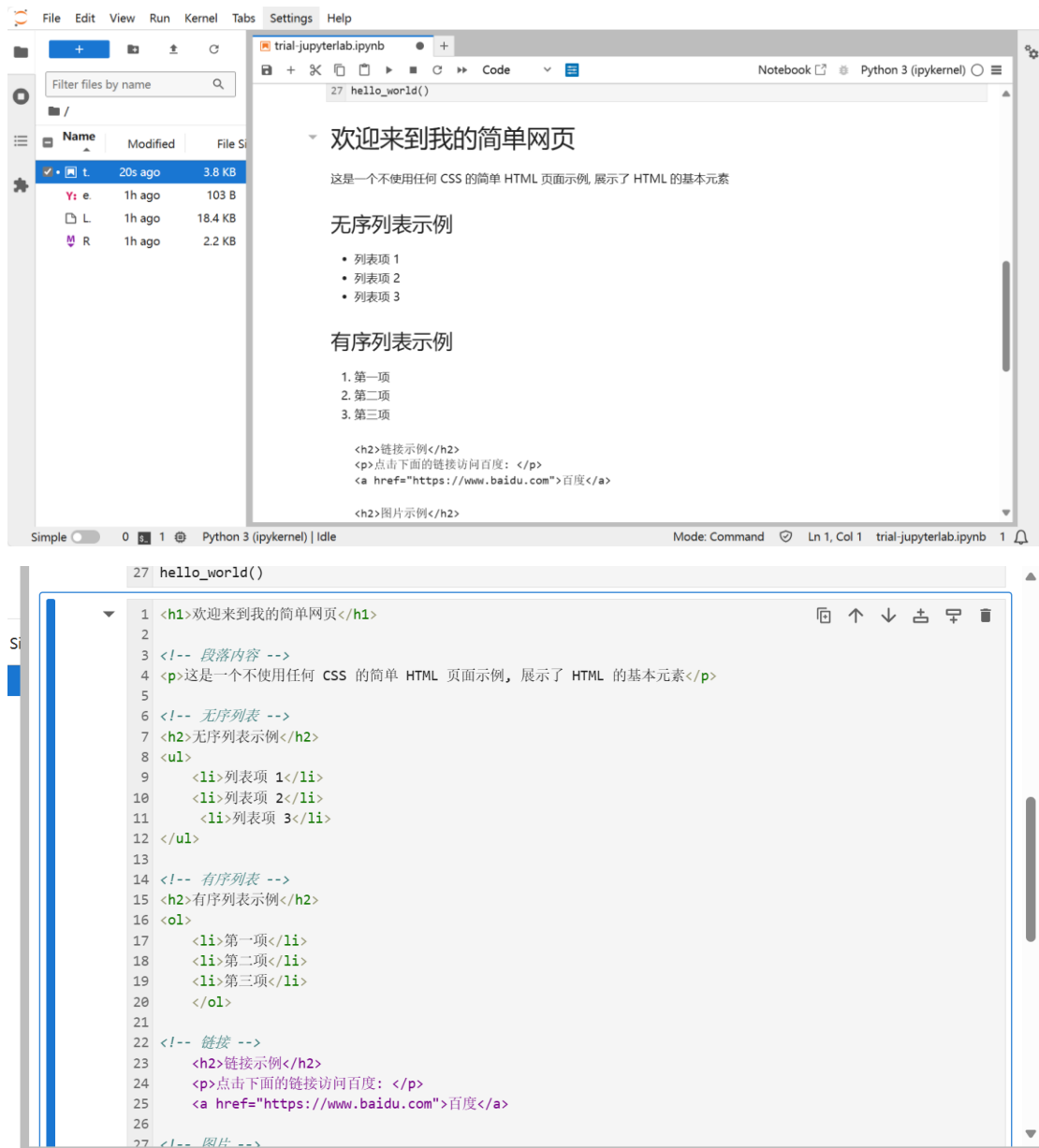
4.10 在单元格 (Cell) 的命令模式下，按 **m** 切换至 **Markdown** 模式，按 **y** 切换至 **Python** 模式



4.11 用豆包 (或 DeepSeek 等任何大模型) 生成一段示例 **Markdown** 代码，复制粘贴进 **Markdown** 单元格，运行以呈现 (Render)



**4.12 用豆包 (或 DeepSeek 等任何大模型) 生成一段示例 HTML 代码，复制粘贴进 Markdown 单元格，运行以呈现 (Render)；注意不支持 CSS**



4.13 用豆包 (或 DeepSeek 等任何大模型) 生成一段示例 LaTeX 数学公式代码, 复制粘贴进 Markdown 单元格, 运行以呈现 (Render); 注意要用  $(行内模式) 或  $(整行模式) 包围$$

$$i\hbar \frac{\partial}{\partial t} \Psi(\mathbf{r}, t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(\mathbf{r}, t) \right] \Psi(\mathbf{r}, t)$$

```

1 
$$i\hbar \frac{\partial}{\partial t} \Psi(\mathbf{r}, t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(\mathbf{r}, t) \right] \Psi(\mathbf{r}, t)$$


```

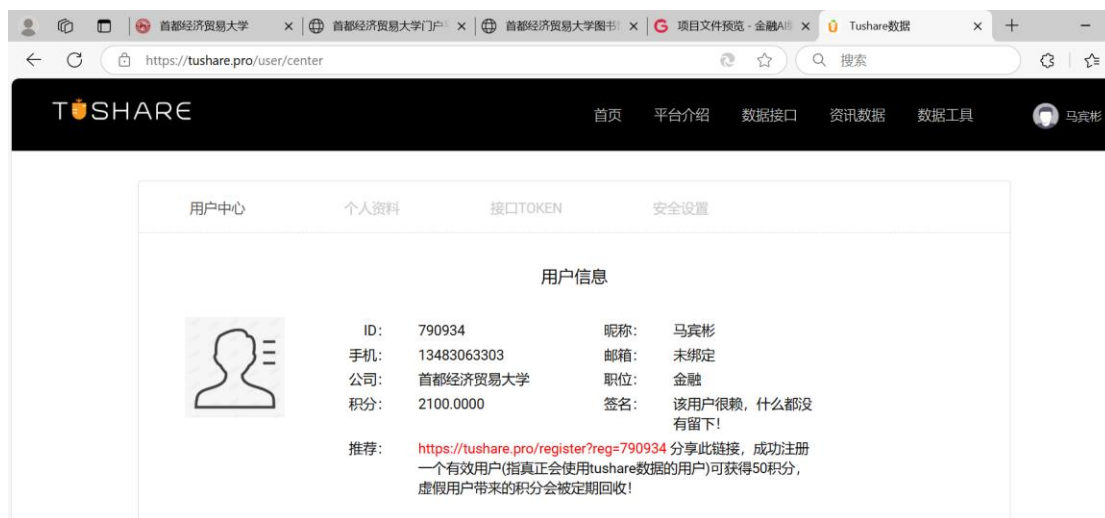
4.14 关闭前端页面, 在后端按 **Ctrl+C** 打断运行中的服务, 回到 **Bash** 提示符



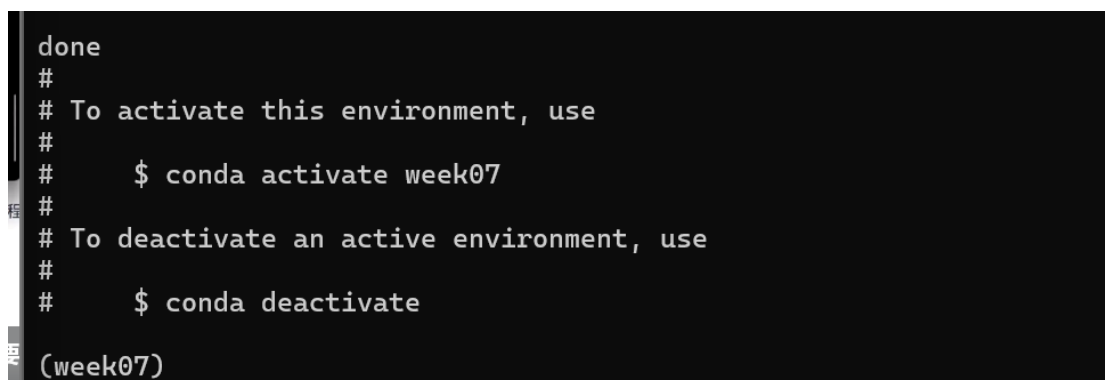
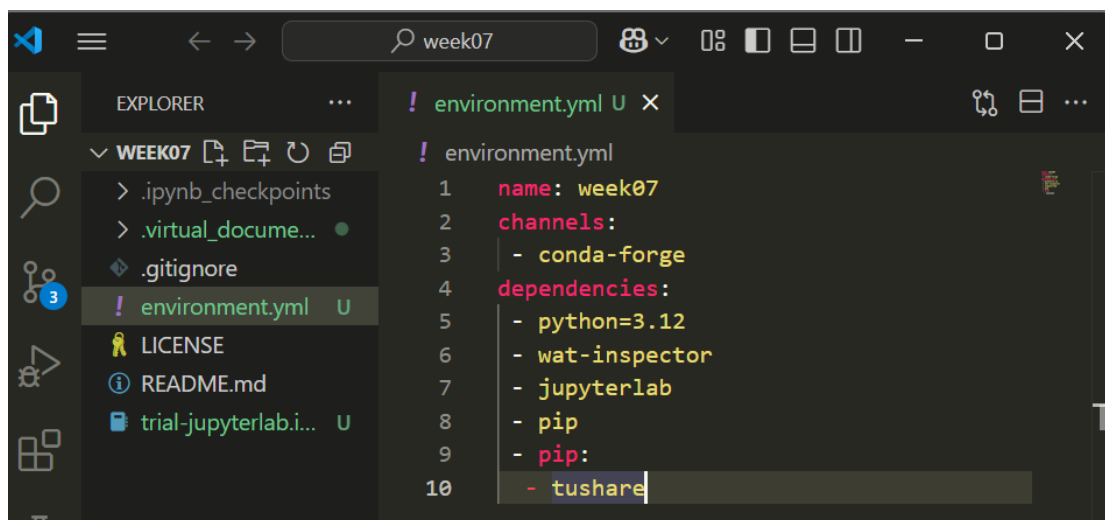
```
MINGW64:/e/研究生 × MINGW64:/e/研究生 × + - □ ×
terlab.ipynb
[I 2025-04-22 19:28:55.129 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:34:57.048 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:36:57.784 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:38:57.866 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:40:58.041 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:44:58.161 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:46:59.040 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:48:59.131 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:50:59.236 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:52:59.331 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:55:00.043 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:59:02.036 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:59:40.137 ServerApp] Saving file at /trial-jupy
terlab.ipynb
[I 2025-04-22 19:59:44.733 ServerApp] Starting buffering for e2d
aee6e-6680-4dbb-a54f-92884b8387c3:b717ffffd-4f01-41ef-9277-294c3f
2b1ac0
[I 2025-04-22 20:00:01.575 ServerApp] Interrupted...
[IPKernelApp] WARNING | Parent appears to have exited, shutting
down.
(base)
Lenovo@DESKTOP-C70G2VK MINGW64 /e/研究生上课资料/研一下 上课资料
```

## 五、通过 tushare 软件包下载保存一些数据

5.1 在 Tushare 网站上 注册 并登陆，完善修改个人资料，浏览阅读 平台介绍 和 数据接口



5.2 修改 `environment.yml` 文件, 添加 `pip: tushare` (注意, `conda-forge` 没有收录 `tushare`, 只能从 `PyPI` 安装, 参考) 依赖项, 运行 `conda env update` 更新 `Conda` 环境



5.3 在终端 (Terminal) 激活 `week07` `Conda` 环境, 运行 `ipython` 命令启动 `IPython` 交互界面 (`IPython` 是 `Jupyter` 项目的一部份, `ipython` 是 `jupyterlab`

的依赖项之一)

```
Lenovo@DESKTOP-C70G2VK MINGW64 /e/研究生上课资料/研一下 上课资料
/金融编程与计算/week07 (main)
$ ipython
Python 3.12.10 | packaged by conda-forge | (main, Apr 10 2025, 2
2:08:16) [MSC v.1943 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 9.1.0 -- An enhanced Interactive Python. Type '?' for he
lp.
Tip: We can't show you all tips on Windows as sometimes Unicode
characters crash the Windows console, please help us debug it.
```

5.4 在 IPython 提示符下，运行下面的 Python 代码设置 Tushare Token

```
import tushare as ts
```

```
ts.set_token("****") # 将 **** 修改成你自己的 Token 字符串
```

其中 \*\*\*\* 要替换成你在 Tushare 平台上的 接口 TOKEN —— 复制粘贴即可。运行 set\_token 函数会把 Token 字符串保存在 ~/tk.csv 文件里，今后每次使用 tushare 软件包请求数据时都会自动读取并发送 Token，不需要反复设置。

```
(week07)
Lenovo@DESKTOP-C70G2VK MINGW64 /e/研究生上课资料/研一下 上课资料
/金融编程与计算/week07 (main)
$ python
Python 3.12.10 | packaged by conda-forge | (main, Apr 10 2025, 2
2:08:16) [MSC v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more inform
ation.
>>> import tushare as ts
>>> ts.set_token(
>>>
>>> quit()
(week07)
```

5.5 按 Ctrl+D 结束前面的 IPython 进程，再重新启动一个新的 IPython 进程，运行下面的 Python 代码向 Tushare 服务器请求 IPO 新股列表 数据，并保存在本地

```
import tushare as ts
```

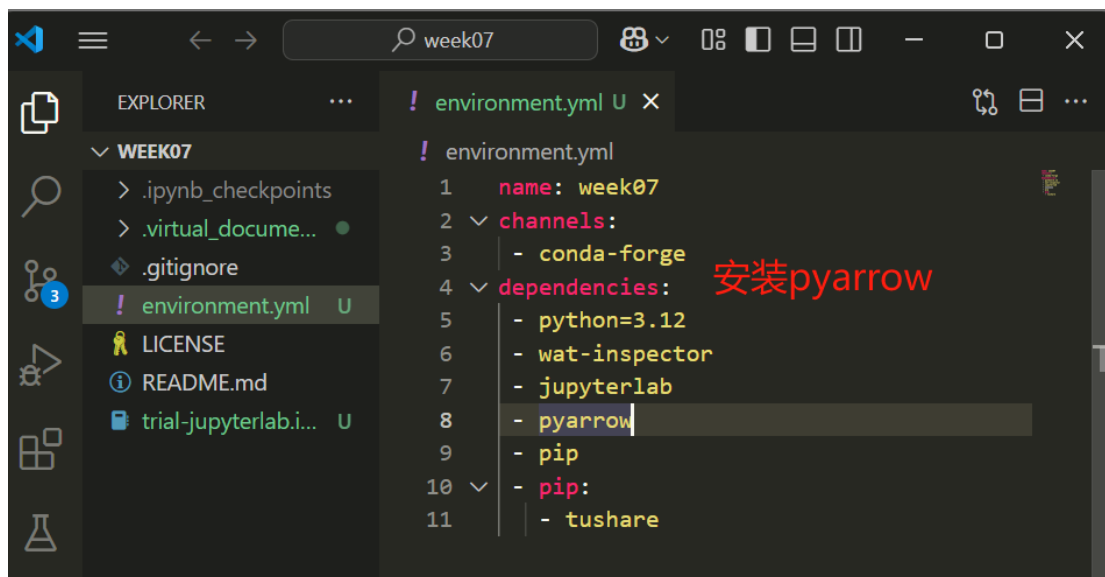
```
pro = ts.pro_api()
```

```
df = pro.new_share()
```

```
df.to_parquet("new_share.parquet")
```

其中请求数据函数返回的对象 df 是 pandas.DataFrame 类型，调用其 to\_parquet 方法能够将内存 (memory) 中的 DataFrame 数据按照 Parquet 格式 (Parquet 是大数据领域的首选格式，已经成为业界标准) 序列化 (serialize) 为字

节串 (bytes) 保存到磁盘 (disk)。



```
environment.yml
1  name: week07
2  channels:
3    - conda-forge
4  dependencies:
5    - python=3.12
6    - wat-inspector
7    - jupyterlab
8    - pyarrow
9    - pip
10   - pip:
11     - tushare
```

安装pyarrow

```
In [1]: import tushare as ts
In [2]: pro = ts.pro_api()
In [3]: df = pro.new_share()
In [4]: df
Out[4]:
```

	ts_code	sub_code	name	...	limit_amount	funds	ballot
0	301636.SZ	301636	泽润新能	...	0.45	0.000	
1	001400.SZ	001400	江顺科技	...	1.50	5.604	
2	301560.SZ	301560	众捷汽车	...	0.70	5.016	
3	603202.SH	732202	天有为	...	1.25	37.400	0
4	301662.SZ	301662	宏工科技	...	0.45	5.320	
...	...	...	...	...	...	...	...
1995	002953.SZ	002953	日丰股份	...	1.70	4.526	
1996	603697.SH	732697	有友食品	...	3.10	6.257	
1997	300772.SZ	300772	运达股份	...	2.80	4.792	
1998	603967.SH	732967	中创物流	...	2.60	10.213	

```
In [6]: pro.stock_basic()
Out[6]:
```

	ts_code	symbol	...	act_name	act_ent_type
0	000001.SZ	000001	...	无实际控制人	
1	000002.SZ	000002	...	深圳市人民政府国有资产监督管理委员会	地方国企
2	000004.SZ	000004	...	李映彤	民营
3	000006.SZ	000006	...	深圳市人民政府国有资产监督管理委员会	地方国企
4	000007.SZ	000007	...	王玩虹	民营
...	...	...	...	...	...
5407	920111.BJ	920111	...	None	None
5408	920116.BJ	920116	...	None	None
5409	920118.BJ	920118	...	None	None
5410	920128.BJ	920128	...	None	None
5411	689009.SH	689009	...	None	None

[5412 rows x 10 columns]

## 5.6 询问豆包 (或 DeepSeek 等任何大模型), 初步了解 Parquet 格式和 CSV 格式的特点和适用领域

特点:

### (1) Parquet 格式

列式存储: 数据按列进行存储, 同一列的数据连续存放, 这种方式有利于提高查询性能, 特别是在只需要查询部分列时, 可以只读取所需列的数据, 减少 I/O 开销。

压缩效率高: 支持多种压缩算法, 如 Snappy、Gzip 等, 能够有效地减少数据存储空间, 降低存储成本。

支持复杂数据类型: 可以存储嵌套结构的数据, 如数组、结构体等, 适用于存储半结构化或结构化程度较高的数据。

文件格式规范: Parquet 是一种基于 Hadoop 生态系统的开源列存储格式, 有统一的文件格式规范, 便于不同系统和工具之间的数据交换和共享。

### (2) CSV 格式

文本格式: 以纯文本形式存储数据, 每行表示一条记录, 字段之间用逗号等分隔符分隔。这种格式简单直观, 易于理解和处理, 任何文本编辑器都可以打开和编辑 CSV 文件。

通用性强：几乎所有的数据库、数据处理工具和编程语言都支持 CSV 格式的导入和导出，具有很强的兼容性和通用性。

不支持复杂数据类型：通常只能存储简单的字符、数字等基本数据类型，对于复杂的嵌套结构数据支持有限。

无压缩或简单压缩：CSV 文件本身一般不进行压缩，如果需要压缩，通常使用外部压缩工具，如 gzip 等。与 Parquet 格式相比，压缩效果可能相对较差。

适用领域：

#### （1）Parquet 格式

大数据分析：在 Hadoop、Spark 等大数据处理框架中广泛应用，由于其列式存储和高效压缩的特点，能够大大提高数据分析的效率，减少查询时间和资源消耗。

数据仓库：适合存储大规模的结构化和半结构化数据，用于构建数据仓库，支持复杂的查询和分析操作。

嵌套数据存储：当数据包含复杂的嵌套结构时，如 JSON 格式的数据，Parquet 能够很好地存储和处理这类数据，方便进行后续的分析 and 处理。

#### （2）CSV 格式

数据交换：常用于不同系统之间的数据交换，由于其通用性强，几乎可以被任何系统识别和处理，因此是一种常用的数据共享格式。

简单数据处理：对于简单的数据分析和处理任务，特别是在数据量较小的情况下，CSV 格式可以方便地使用各种工具进行处理，如 Excel、文本编辑器等。

日志文件：一些系统的日志文件通常以 CSV 格式保存，因为它易于记录和查看，方便后续的日志分析和处理。

**5.7 访问 stock\_basic 接口，并将数据保存为 stock\_basic.parquet 文件 (注意，需要指定 fields 参数获取全部字段)**

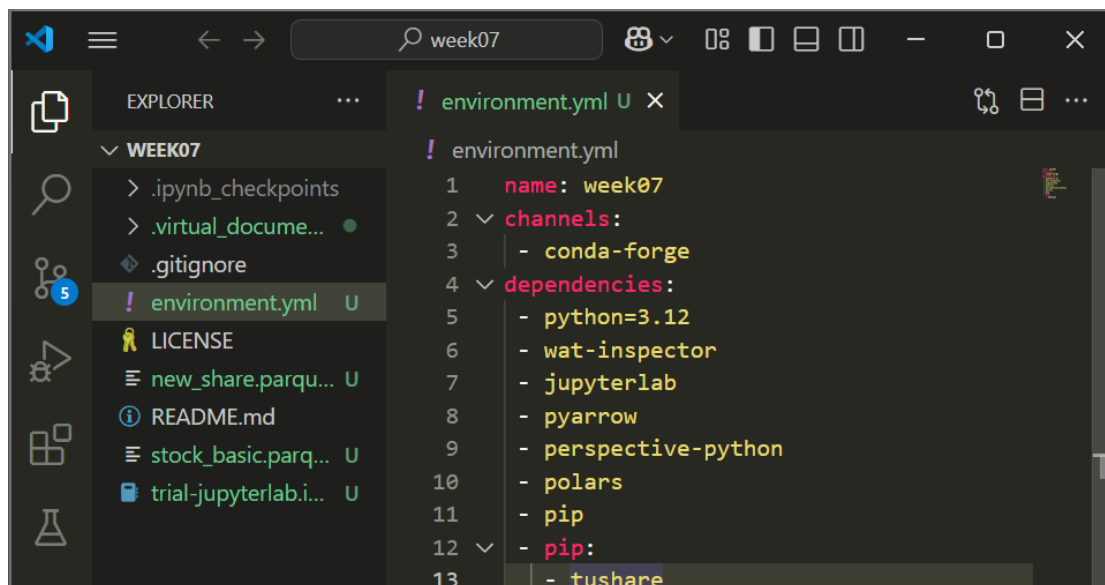
```

Lenovo@DESKTOP-C70G2VK MINGW64 /e/研究生上课资料/研一下 上课资料
/金融编程与计算/week07 (main)
$ ls -lh
total 569K
-rw-r--r-- 1 Lenovo 197609 146 4月 22 21:56 environment.yml
-rw-r--r-- 1 Lenovo 197609 19K 4月 22 18:07 LICENSE
-rw-r--r-- 1 Lenovo 197609 118K 4月 22 22:00 new_share.parquet
-rw-r--r-- 1 Lenovo 197609 2.2K 4月 22 18:07 README.md
-rw-r--r-- 1 Lenovo 197609 419K 4月 22 22:10 stock_basic.parque
t
-rw-r--r-- 1 Lenovo 197609 3.8K 4月 22 19:59 trial-jupyterlab.i
pynb
(week07)
Lenovo@DESKTOP-C70G2VK MINGW64 /e/研究生上课资料/研一下 上课资料
/金融编程与计算/week07 (main)

```

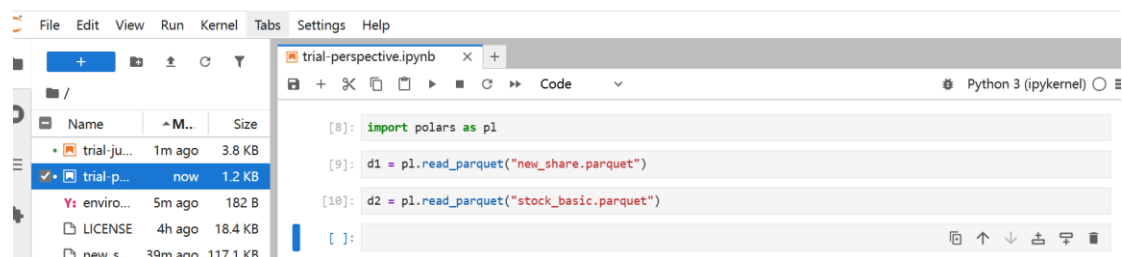
六、通过 perspective-python 软件包查看 polars.DataFrame 数据，实践交互式可视化

6.1 修改 environment.yml 文件，添加 perspective-python 和 polars 依赖项，运行 conda env update 更新 Conda 环境



6.2 启动 JupyterLab，新建一个 Notebook，改名为 trial-perspective.ipynb

6.3 调用 polars.read\_parquet 函数，分别读取磁盘 (disk) 中的 new\_share.parquet 文件和 stock\_basic.parquet 文件，得到内存 (memory) 中的 polars.DataFrame 对象，命名为 d1 和 d2



6.4 进行适当的列变换，尤其是要把实际上是日期类型的列，从 `polars.String()` 类型转换为 `polars.Date()` 类型

```
[15]: d1.with_columns(
      ipo_date=pl.col.ipo_date.str.to_date("%Y%m%d"),
      issue_date=pl.col.issue_date.str.to_date("%Y%m%d"),
    )
[15]: shape: (2,000, 12)
```

ts_code	sub_code	name	ipo_date	issue_date	amount	market_amount	price	pe	limit_amount	funds
str	str	str	date	date	f64	f64	f64	f64	f64	f64
"301636.SZ"	"301636"	"泽润新能"	2025-04-28	null	1597.0	0.0	0.0	0.0	0.45	0.0
"001400.SZ"	"001400"	"江顺科技"	2025-04-15	null	1500.0	1500.0	37.36	15.32	1.5	5.604
"301560.SZ"	"301560"	"众捷汽车"	2025-04-15	null	3040.0	1216.0	16.5	21.3	0.7	5.016
"603202.SH"	"732202"	"天有为"	2025-04-14	null	4000.0	2611.0	93.5	13.5	1.25	37.4
"301663.SZ"	"301663"	"宏工"	2025-04-14	2025-04-14	2000.0	812.0	26.6	7.05	0.45	5.32

```
[16]: d2.with_columns(
      list_date=pl.col.list_date.str.to_date("%Y%m%d"),
    )
[16]: shape: (5,412, 17)
```

6.5 把 `d1` 或 `d2` 作为参数传递给 `perspective.widget.PerspectiveWidget` 类型进行初始化，返回的对象会呈现在 Notebook 的 Output 里



```
[15]: perspective.widget.widget.PerspectiveWidget
```

```
[16]: d1_df = pd.DataFrame(d1)
```

```
[17]: d1_df = d1_df.dropna()
```

```
[18]: PerspectiveWidget(d1_df)
```

untitled

Datagrid

index	0	1	2	3
13	688757.SH	787757	胜利纳米	2025/3/14
14	301535.SZ	301535	浙江华远	2025/3/14
15	301629.SZ	301629	矽电股份	2025/3/11
16	001382.SZ	001382	新亚电缆	2025/3/11
17	603124.SH	732124	江南新材	2025/3/10
18	301501.SZ	301501	恒鑫生活	2025/3/7
19	301479.SZ	301479	弘景光电	2025/3/4
20	603271.SH	732271	永杰新材	2025/2/28
21	301275.SZ	301275	汉朔科技	2025/2/28
22	603409.SH	732409	汇通控股	2025/2/21
23	301173.SZ	301173	毓恬冠佳	2025/2/20

Order By

Where

Columns

- # index
- 0
- 1
- 2
- 3
- 4

```
[19]: d2_df = pd.DataFrame(d2)
```

```
[21]: PerspectiveWidget(d2_df)
```

untitled

Datagrid

index	0	1	2	3	4
0	000001.SZ	000001	平安银行	深圳	银行
1	000002.SZ	000002	万科A	深圳	全国
2	000004.SZ	000004	国华网安	深圳	软件
3	000006.SZ	000006	深振业A	深圳	区域
4	000007.SZ	000007	全新好	深圳	其他
5	000008.SZ	000008	神州高铁	北京	运输
6	000009.SZ	000009	中国宝安	深圳	电子
7	000010.SZ	000010	美丽生态	深圳	建筑
8	000011.SZ	000011	深物业A	深圳	房产
9	000012.SZ	000012	南玻A	深圳	玻璃
10	000014.SZ	000014	沙河股份	深圳	全国
11	000016.SZ	000016	深康佳A	深圳	家用
12	000017.SZ	000017	深中华A	深圳	服饰
13	000019.SZ	000019	深粮控股	深圳	农业
14	000020.SZ	000020	深华发A	深圳	元器

Group By

Split By

Order By

Where

Columns

- # index
- 0
- 1
- 2
- 3
- 4

## 6.6 在 PerspectiveWidget 默认的 Datagrid 视图下，尝试实践：

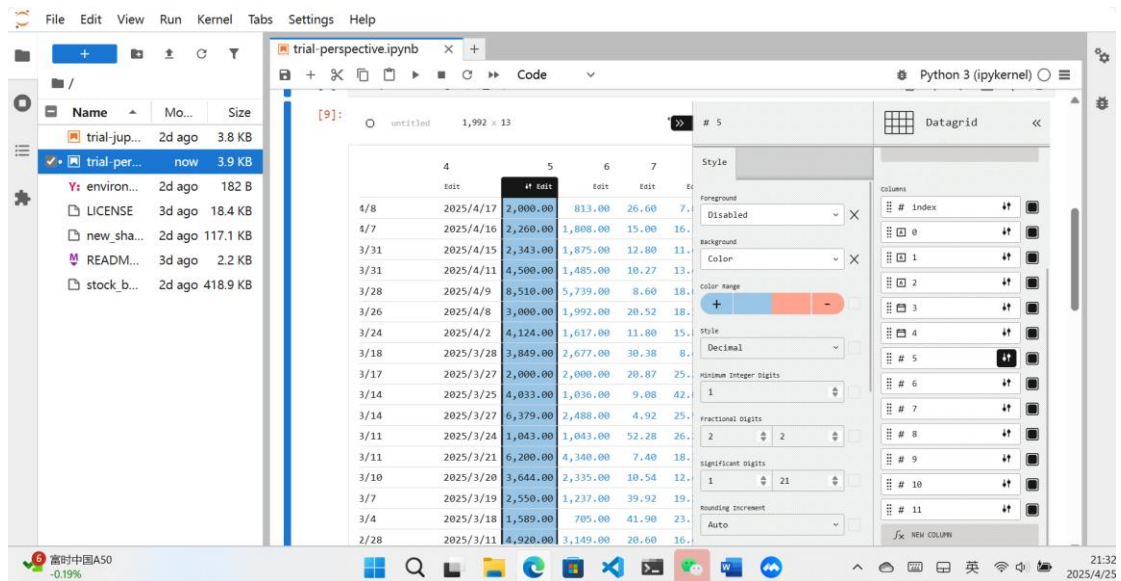
修改各种列数据类型 (文本、数值、日期) 的显示风格 (style)

The image displays two screenshots of the JupyterLab interface, specifically the PerspectiveWidget Datagrid view, demonstrating how to modify the display style of data columns.

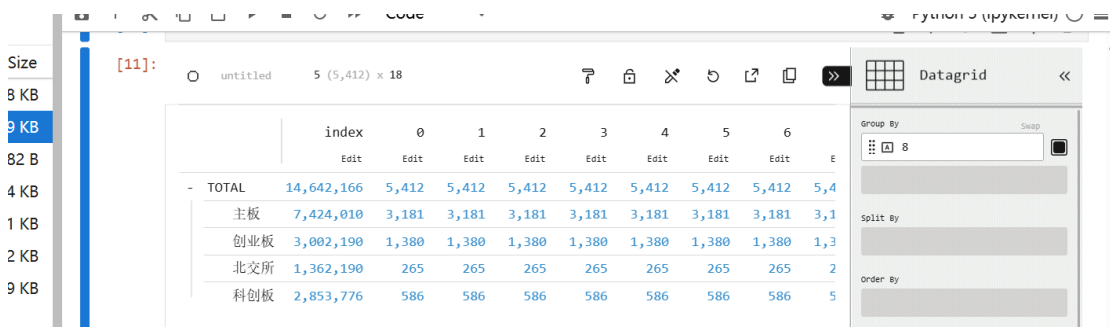
**Top Screenshot:** Shows a table with columns: index, 0, 1, 2, 3, 4. The data includes stock codes, names (e.g., 宏工科技, 伯特催化), and dates (e.g., 2025/4/8, 2025/4/17). The 'Style' panel on the right shows the 'Format' dropdown set to 'bold' and the 'Color' dropdown set to 'series'.

**Bottom Screenshot:** Shows the same table, but the date columns (3 and 4) are now formatted as full dates (e.g., 2025年4月8日星期二). The 'Style' panel on the right shows the 'Color' dropdown set to 'foreground' and the 'Date Style' dropdown set to 'full'.

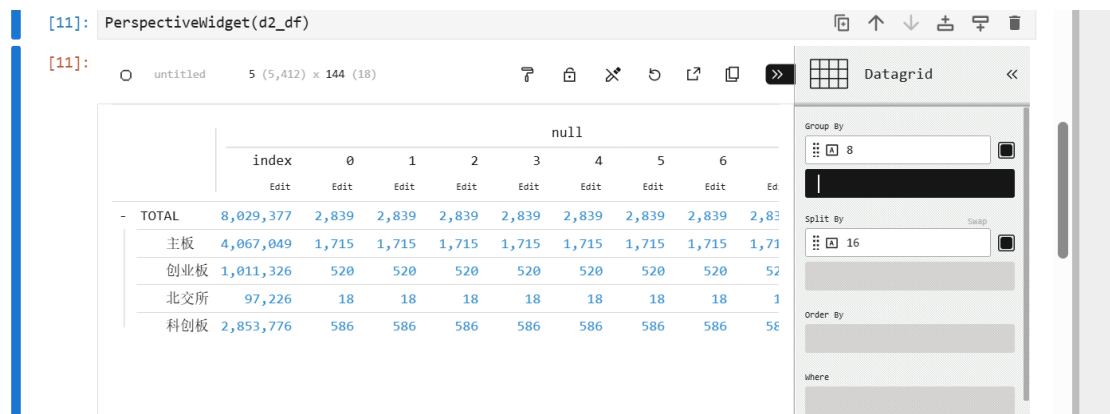
The interface includes a file explorer on the left, a code editor at the top, and a terminal at the bottom. The system tray at the bottom right shows the time as 21:31 on 2025/4/25.



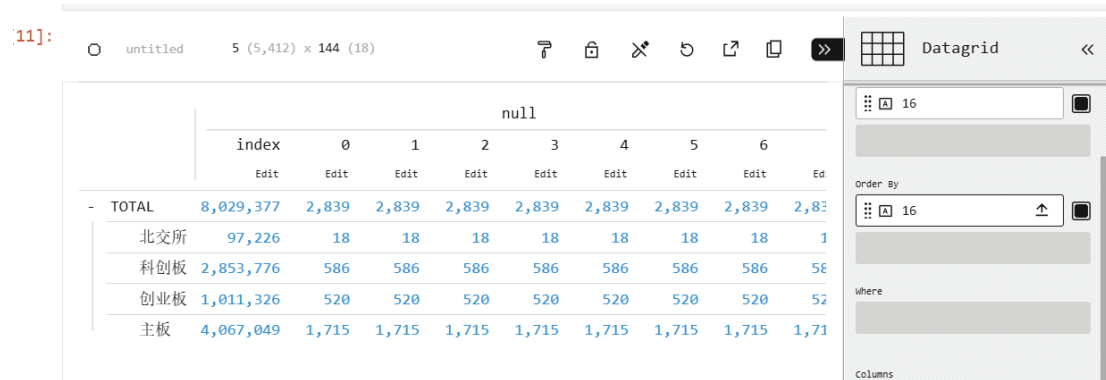
设置 Group By 选项，选择某些列作为分组依据 (纵向排列)，选择其他某些列进行汇总 (注意汇总方式有多种函数选项)



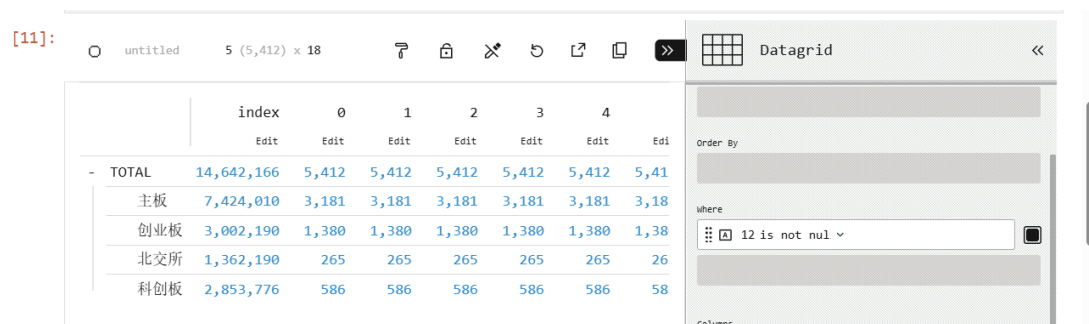
设置 Split By 选项，选择某些列作为拆分依据 (横向排列)



设置 Order By 选项，选择某些列作为排序依据 (注意可以切换 升序/降序)

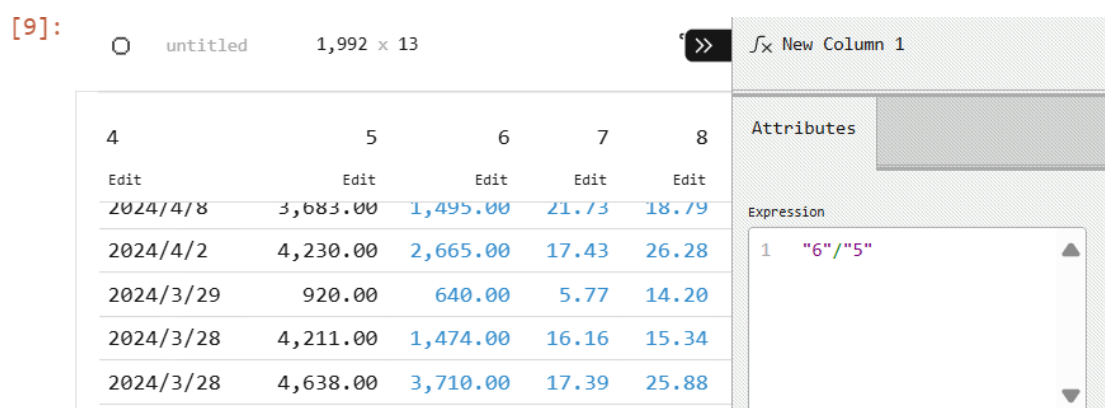


设置 Where 选项，选择某些列，进一步设置条件，进行数据行（观测）方向的过滤



设置 Columns 选项，选择要显示的数据列（变量），及其显示的先后顺序

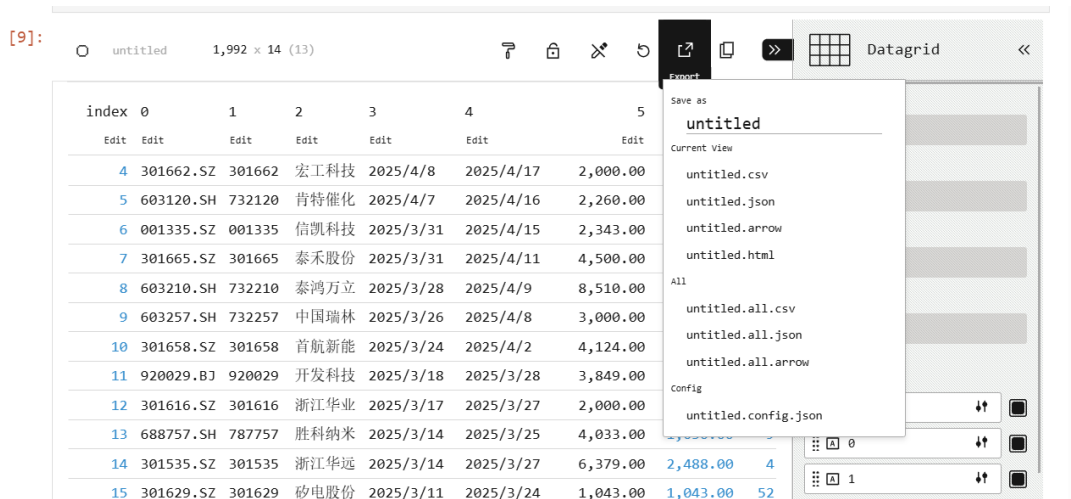
在 All Columns 部分，是能够显示但没有显示的数据列（变量），可以点击 NEW COLUMN 添加衍生计算出的新列，需要用 ExprTK 语言书写表达式代码，变量名用双引号 (") 包围，字符串用单引号 (') 包围



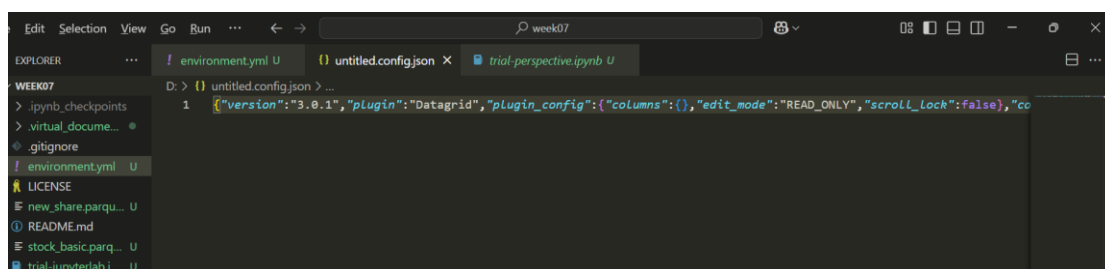
6.7 在 PerspectiveWidget 图形界面依靠鼠标（手动）所做的设置 (configure)，可以导出代码，根据导出的代码，可以修改我们的代码，使得我们运行代码直接就能得到我们所需要的视图（自动化）

在 PerspectiveWidget 的右上方有按钮，可以把图形界面的数据或设置 (configure) 导出 (export) 为文件，或复制 (copy) 到剪贴板





把设置 (config.json) 复制到剪贴板，粘贴进 Notebook Cell，保存成字符串 (str)



```
[12]: '10","11","New Column 1"],["filter":[],"sort":[],"expressions":{"New Column 1":"6"/"5"},"aggregates":{}}'
```

```
[13]: print(config)
```

```
{
  "version": "3.0.1",
  "plugin": "Datagrid",
  "plugin_config": {
    "columns": {},
    "edit_mode": "READ_ONLY",
    "scroll_lock": false,
    "columns_config": {
      "5": {
        "number_fg_mode": "disabled",
        "number_bg_mode": "pulse"
      }
    },
    "settings": true,
    "theme": "Pro Light",
    "title": null,
    "group_by": [],
    "split_by": [],
    "columns": [
      "index",
      "0",
      "1",
      "2",
      "3",
      "4",
      "5",
      "6",
      "7",
      "8",
      "9",
      "10",
      "11",
      "New Column 1"
    ],
    "filter": [],
    "sort": [],
    "expressions": {
      "New Column 1": "6"/"5"
    },
    "aggregates": {}
  }
}
```

也可以把设置 (config.json) 导出为文件，用 `pathlib.Path.read_text` 方法从文件读取字符串 (str)

```
[16]: from pathlib import Path
```

```
[20]: file_path = Path("D:/untitled.config.json")
```

```
[21]: config2 = file_path.read_text()
```

```
[22]: print(config2)
```

```
{
  "version": "3.0.1",
  "plugin": "Datagrid",
  "plugin_config": {
    "columns": {},
    "edit_mode": "READ_ONLY",
    "scroll_lock": false,
    "columns_config": {
      "5": {
        "number_fg_mode": "disabled",
        "number_bg_mode": "pulse"
      }
    },
    "settings": true,
    "theme": "Pro Light",
    "title": null,
    "group_by": [],
    "split_by": [],
    "columns": [
      "index",
      "0",
      "1",
      "2",
      "3",
      "4",
      "5",
      "6",
      "7",
      "8",
      "9",
      "10",
      "11",
      "New Column 1"
    ],
    "filter": [],
    "sort": [],
    "expressions": {
      "New Column 1": "6"/"5"
    },
    "aggregates": {}
  }
}
```

可以用 `json.loads` 函数将无结构的 (unstructured) 字符串 (str) 解析为有结构的 (structured) Python 字典 (dict)，这样就容易在 Notebook 里美化呈现，也

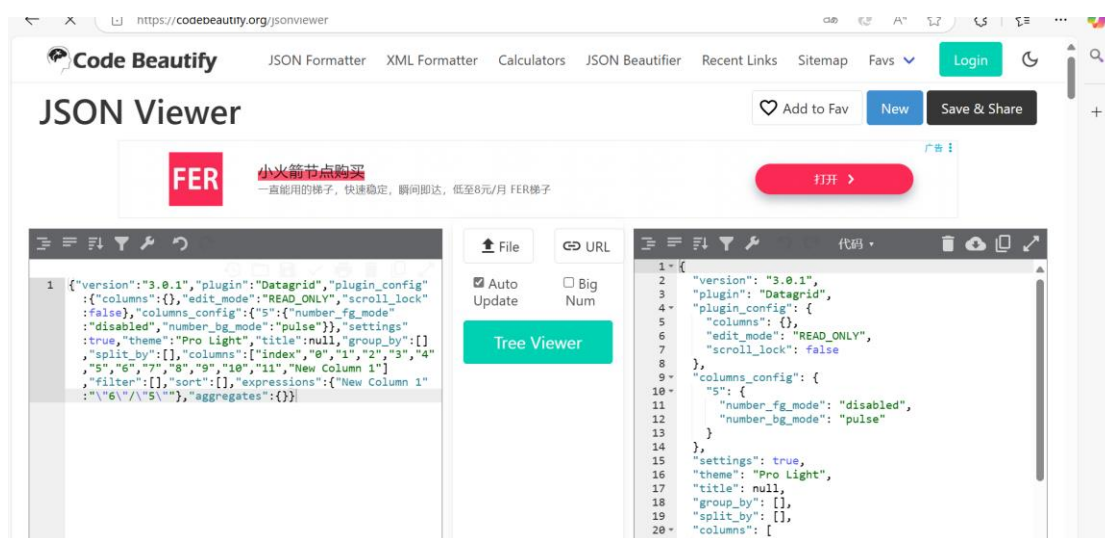
容易进一步通过 Python 代码访问内部的具体设置

```
[28]: import json

[30]: json.loads(config2)

[30]: {'version': '3.0.1',
      'plugin': 'Datagrid',
      'plugin_config': {'columns': {}},
      'edit_mode': 'READ_ONLY',
      'scroll_lock': False},
      'columns_config': {'5': {'number_fg_mode': 'disabled',
                                'number_bg_mode': 'pulse'}},
      'settings': True,
      'theme': 'Pro Light',
      'title': None,
      'group_by': [],
      'split_by': [],
      'columns': ['index',
                  '0',
                  '1',
                  '2',
                  '3',
                  '4',
                  '5',
                  '6',
                  '7',
                  '8',
                  '9',
                  '10',
                  '11',
                  'New Column 1'],
```

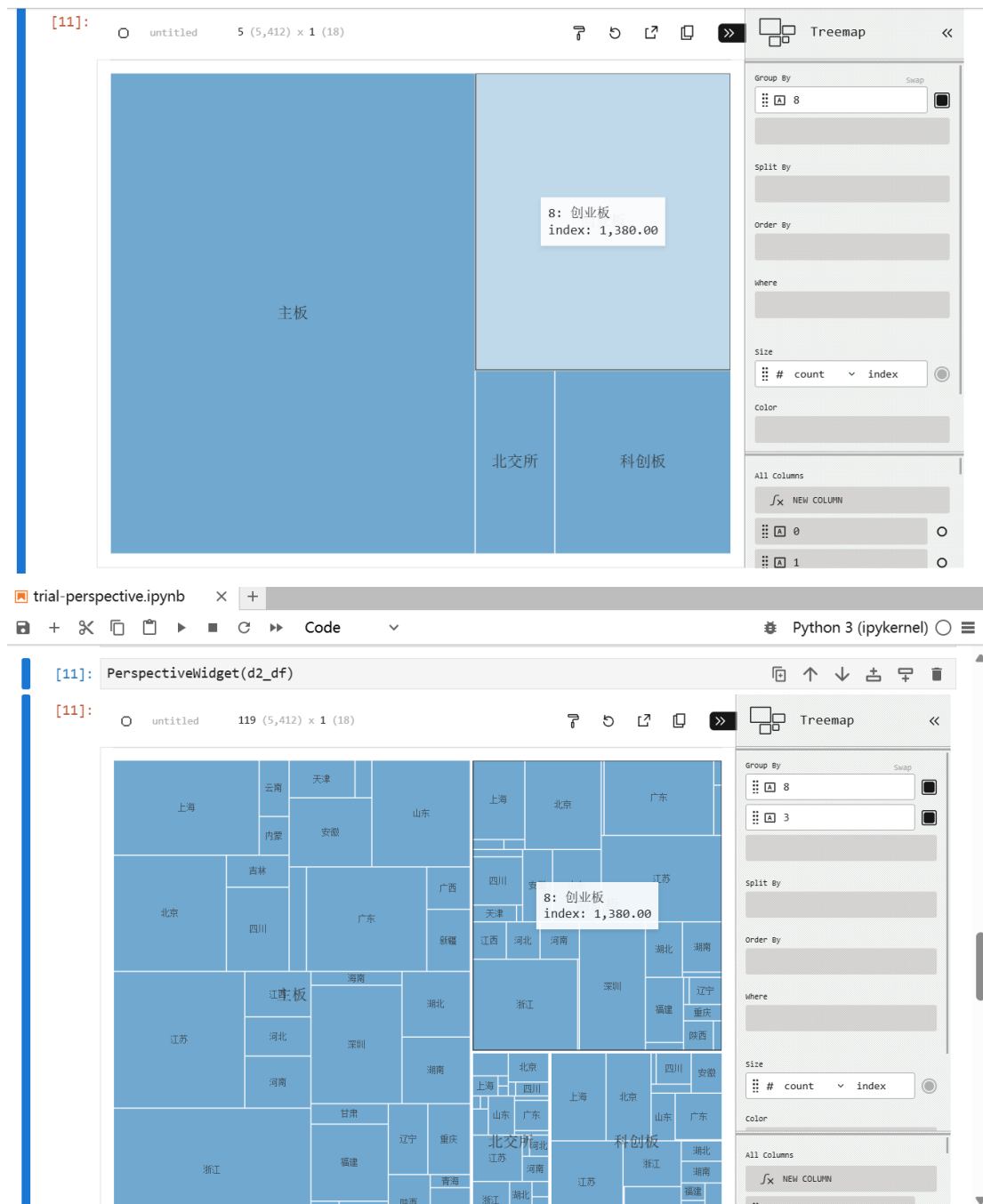
也可以把复制到剪贴板的 JSON 字符串，粘贴进某个在线的 JSON 工具网站 (比如 [链接](#)) 进行美化



根据导出的设置代码，在初始化 (init) PerspectiveWidget 类型时，传入适当的参数进行设置，运行代码，观察是否符合我们的期望

## 6.8 把 PerspectiveWidget 切换为 Treemap 视图，尝试设置各种选项 (configure)，观察数据可视化的实际效果

Treemap (树形结构图) 用不同大小的矩形来体现数据的分类占比构成情况，还可以用矩形的颜色来体现第二个维度的数据 (文本或数值都可以)



点击 [链接 1](#) 或 [链接 2](#) 学习了解更多 Treemap 的概念、适用情形以及实现代码

## Description

Treemaps are an alternative way of visualising the hierarchical structure of a [Tree Diagram](#) while also displaying quantities for each category via area size. Each category is assigned a rectangle area with the subcategory rectangles nested inside.

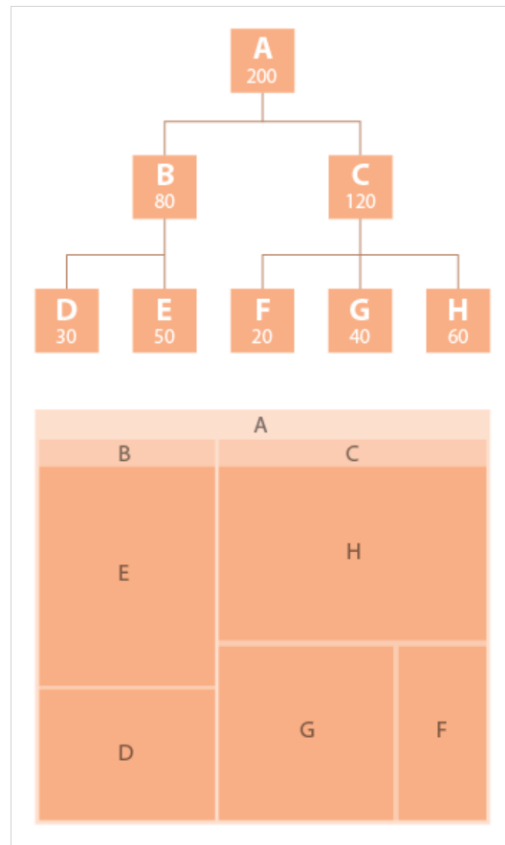
When a quantity is assigned to a category, its area size is in proportion to that quantity and any other quantities within the same parent category in a part-to-whole relationship. Also, the area size of the parent category is the total of its subcategories. If no quantity has been assigned to a subcategory, then its area is divided equally amongst the other subcategories within the parent category.

The way rectangles are divided and ordered into sub-rectangles depends on the tiling algorithm used. Many tiling algorithms have been developed, but the "squarified algorithm", which keeps each rectangle as square-like as possible is the one commonly used.

Ben Shneiderman originally developed Treemaps as a way of visualising a vast file directory on a computer, without taking up too much space on the screen. This makes Treemaps a more compact and space-efficient option for displaying hierarchies, that can give a quick overview of the hierarcal structure. Treemaps are also great at comparing the proportions between categories via their area size.

The downside to Treemaps is that they doesn't show the hierarchal levels as clearly as other charts that visualise

## Anatomy



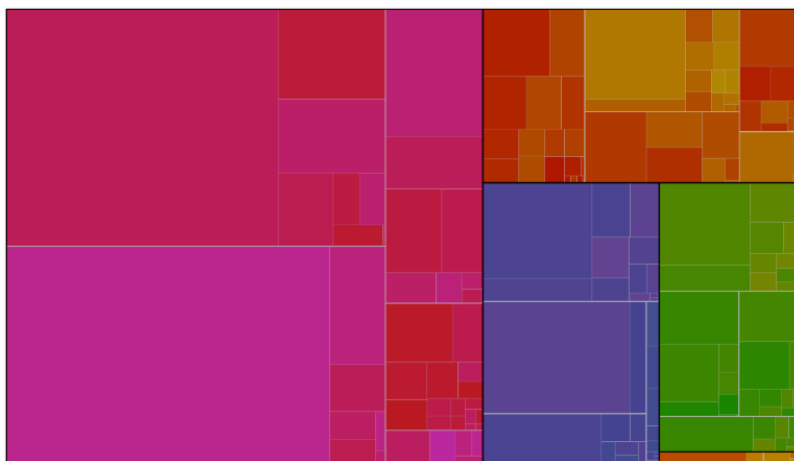


---

A **Treemap** displays **hierarchical** data as a set of nested rectangles. Each group is represented by a rectangle, which area is proportional to its value. Using color schemes and or interactivity, it is possible to represent several dimensions: groups, subgroups etc.

Here is an example describing the **world population** of 250 countries. The world is divided in continent (group), continent are divided in regions (subgroup), and regions are divided in countries. In this tree structure, countries are considered as leaves: they are at the end of the branches.

[SHOW](#)



*Note:* You can read more about this story [here](#), with many more way to visualize this dataset. Data source: wikipedia and formatting by [1](#) and [2](#).

## 6.9 把 **PerspectiveWidget** 切换为 **Y Bar** 视图，尝试设置各种选项 (**configure**)，观察数据可视化的实际效果

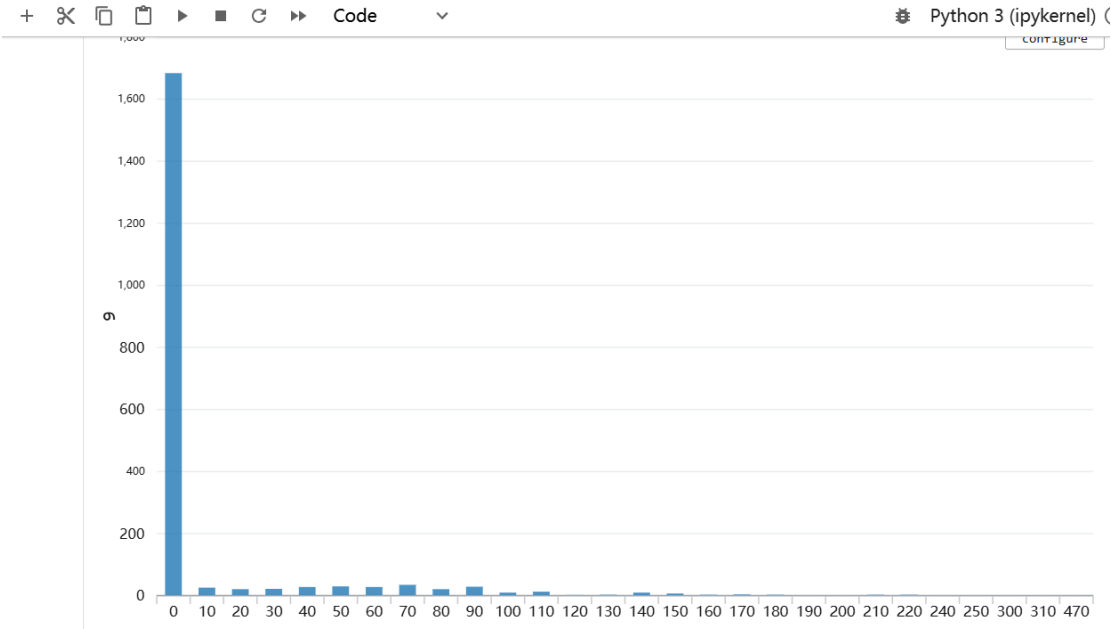
**Y Bar** (条形图/柱状图) 的横轴 (不同的条形) 是第一个维度，用 **Group By** 控制，纵轴 (条形的高度) 是第二个维度，用 **Y Axis** 控制 (支持多变量并列显示)，还可以把每个条形进一步拆分为多个颜色，用 **Split By** 控制



点击 [链接 1](#) 或 [链接 2](#) 学习了解更多 Bar Chart 的概念、适用情形以及实现代码

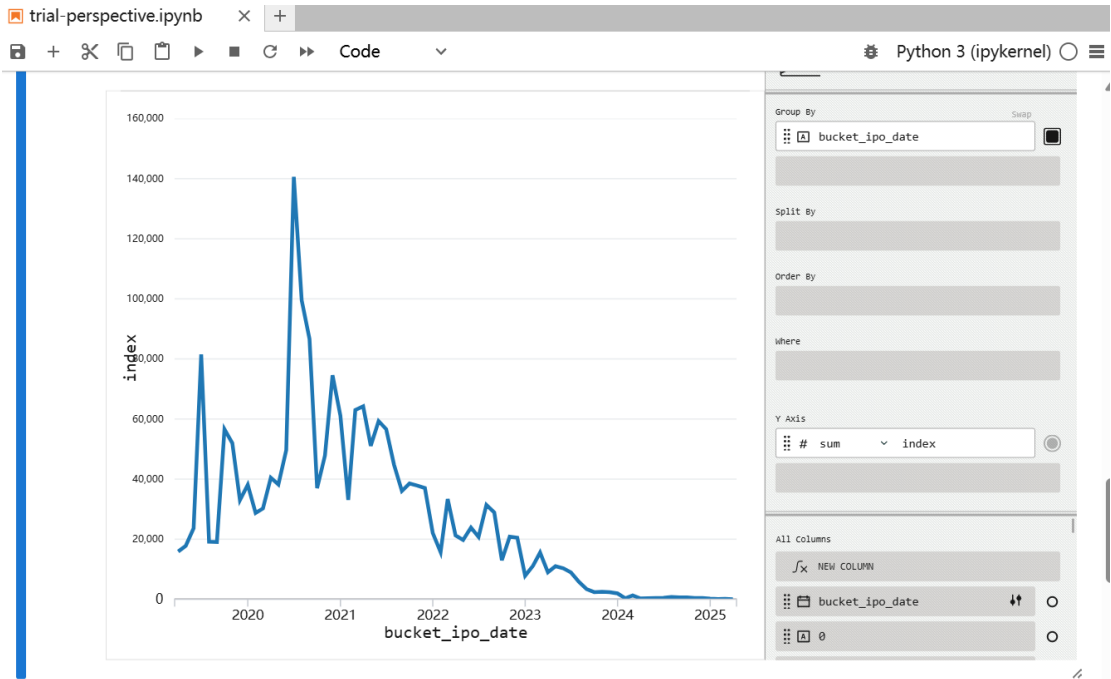
Y Bar 视图还可以用来实现一类很重要的统计制图 —— 直方图 (histogram)。对于数据表中的某一系列连续型数值变量 (比如新股发行的市盈率 `pe`)，我们经常希望观察其 分布 (distribution)。可以用 `bucket` 函数对连续变量进行 “分桶” (比如表达式 `bucket("pe", 10)`)，生成一个新的离散变量 (比如命名为 `bucket_pe`)，然后把离散变量设置为 Y Bar 的横轴 (Group By)，把任意其

他一系列变量用 `count` (计数) 函数汇总, 设置为纵轴 (Y Axis)。这样看到的就是直方图。“分桶” 在有的地方也叫 “分箱” (bin), 其粒度大小需要根据数据适当调节。



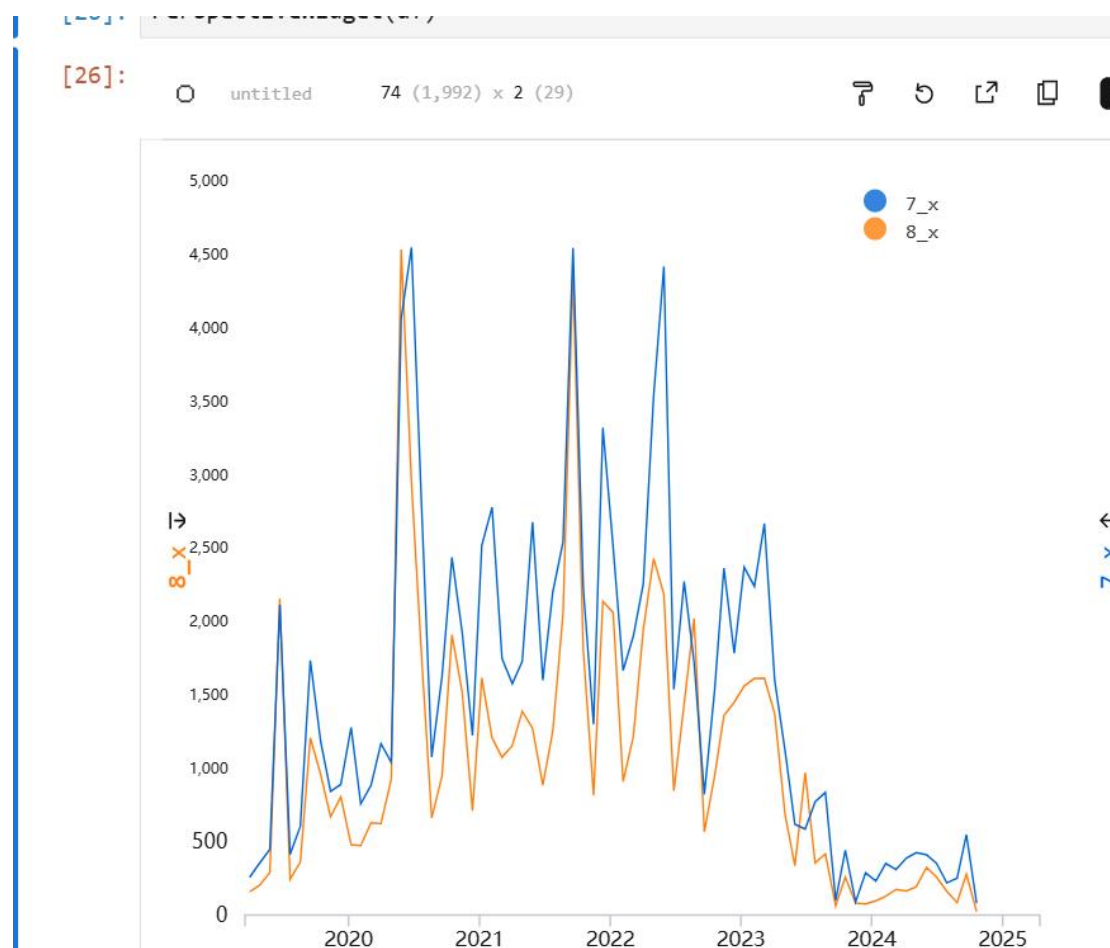
**6.10 把 `PerspectiveWidget` 切换为 `Y Line` 视图, 尝试设置各种选项 (configure), 观察数据可视化的实际效果**

`Y Line` (折线图) 常用来绘制时间序列, 横轴通常是时间, 用 `Group By` 控制, 纵轴 (折线的 `Y` 坐标) 通常是连续型数值变量 (经过汇总), 用 `Y Axis` 控制 (支持多序列同时显示), 还可以进一步拆分为多条序列, 用 `Split By` 控制

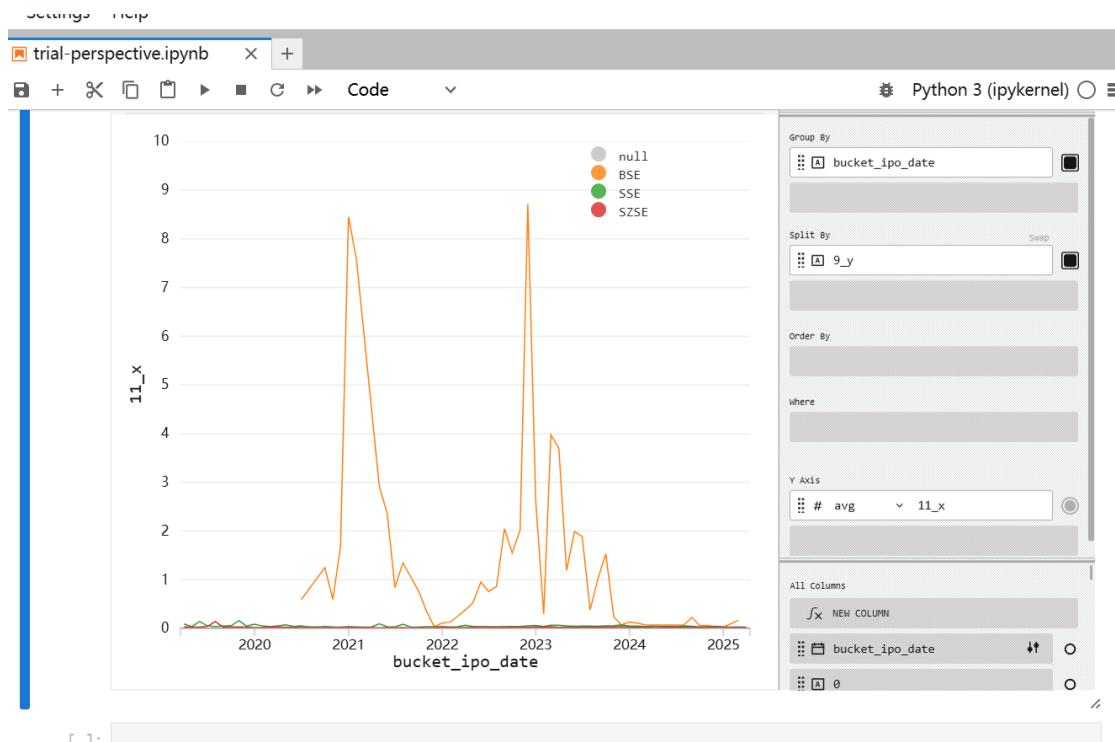


点击 [链接 1](#) 或 [链接 2](#) 学习了解更多 **Line Chart** 的概念、适用情形以及实现代码

使用我们的示例数据，可以尝试观察最近几年 **A 股 IPO** 市场的 **融资额 (funds)** 与 **市盈率 (pe)** 变化情况。为了加深对数据的理解和验证，可以询问豆包 (或 DeepSeek 等任何大模型)，在某个时间段内发生了哪些影响 **A 股 (或 IPO)** 的重大国内外财经事件，由此加强我们对现实背景的理解

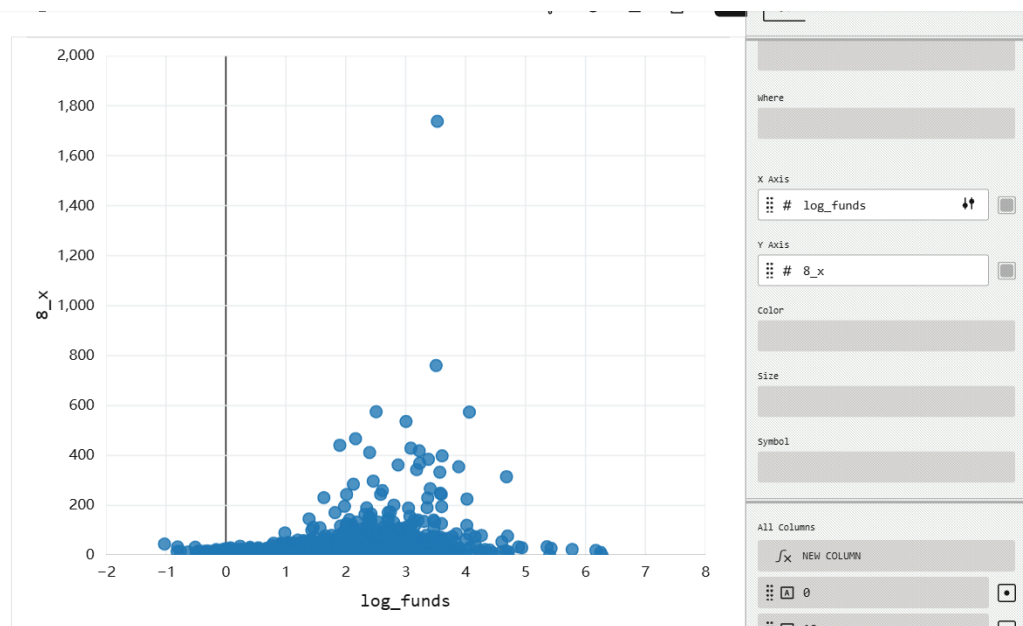


也可以使用示例数据，观察对比最近几年不同交易所 (**exchange**) 或市场 (**market**) 的平均中签率 (**ballot**) 情况



## 6.11 把 PerspectiveWidget 切换为 X/Y Scatter 视图，尝试设置各种选项 (configure)，观察数据可视化的实际效果

X/Y Scatter (散点图) 常用来观察两个数值型连续变量之间的相关关系 (correlation)。数据首先可以进行分组汇总，每一个组对应一个散点，用 Group By 控制。然后把两个连续型数值变量分别设置为 X Axis 和 Y Axis，其汇总数值将作为每个散点的坐标



点击 [链接 1](#) 或 [链接 2](#) 学习了解更多 Scatter Plot 的概念、适用情形以及

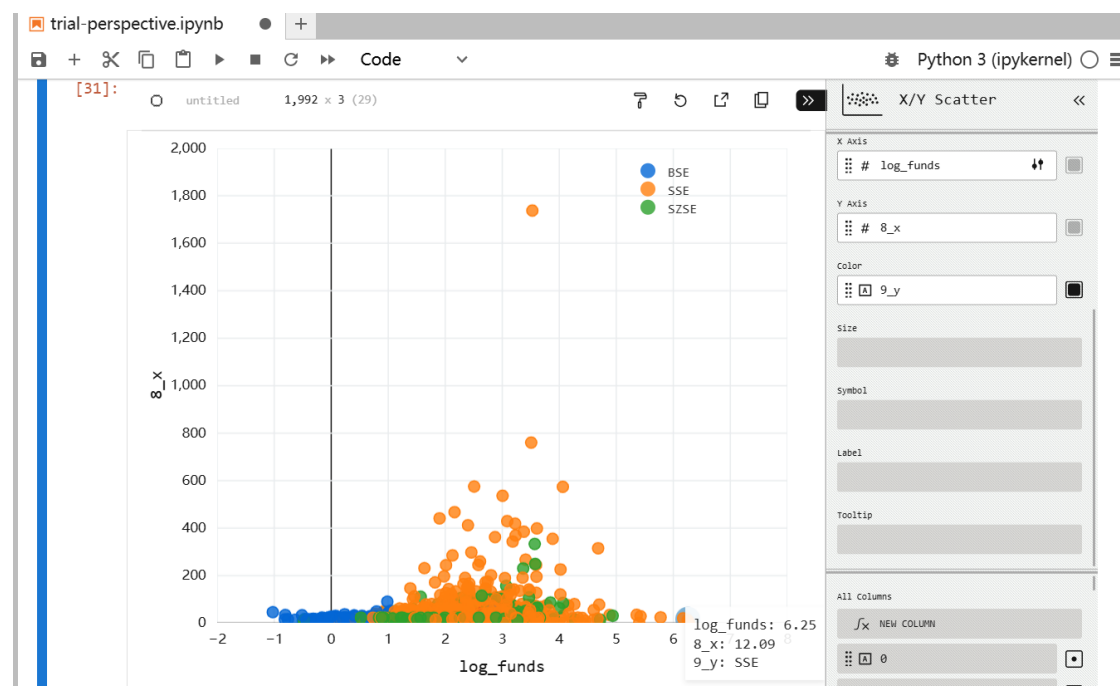
实现代码

散点的分布如果特别不均匀，则意味着变量单位可能有问题，或者需要经过变换 (比如取对数, `log_funds`)

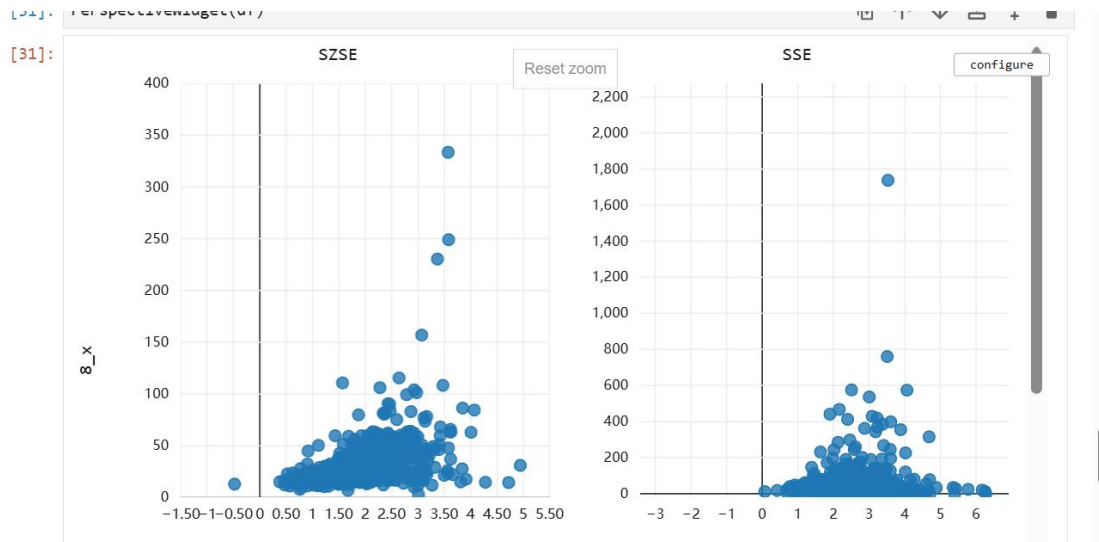
散点的分布如果杂乱无规律，则意味着 `X` 与 `Y` 没有相关性

散点的分布如果看起来能够拟合成一条直线 (即回归线, `regression`), 则意味着 `X` 与 `Y` 具有正的或负的相关性, 意味着可能存在某些规律

散点图上可以进一步体现更多的变量维度, 比如可以把更多变量映射为散点的不同颜色 (Color)、大小 (Size)、符号 (Symbol)、标签 (Label)、提示框 (Tooltip) 等



我们经常还可以把用于分类的类别变量 (类别不宜太多) 设置为 `Split By`, 从而把一个散点图拆分为多个小散点图 (small multiple), 从而更细致地观察是否存在规律



在我们的示例数据中，融资额 (funds)、市盈率 (pe)、中签率 (ballot) 是数值型连续变量，适合用散点图观察他们的规律，散点可以以个股为单位 (不汇总)，也可以按 行业 (industry) 汇总，或者按 上市时间 (ipo\_date) 汇总 (每月分桶)，都可以大胆尝试探索

