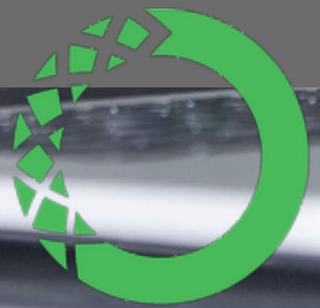
數據分析

日常工作流分享

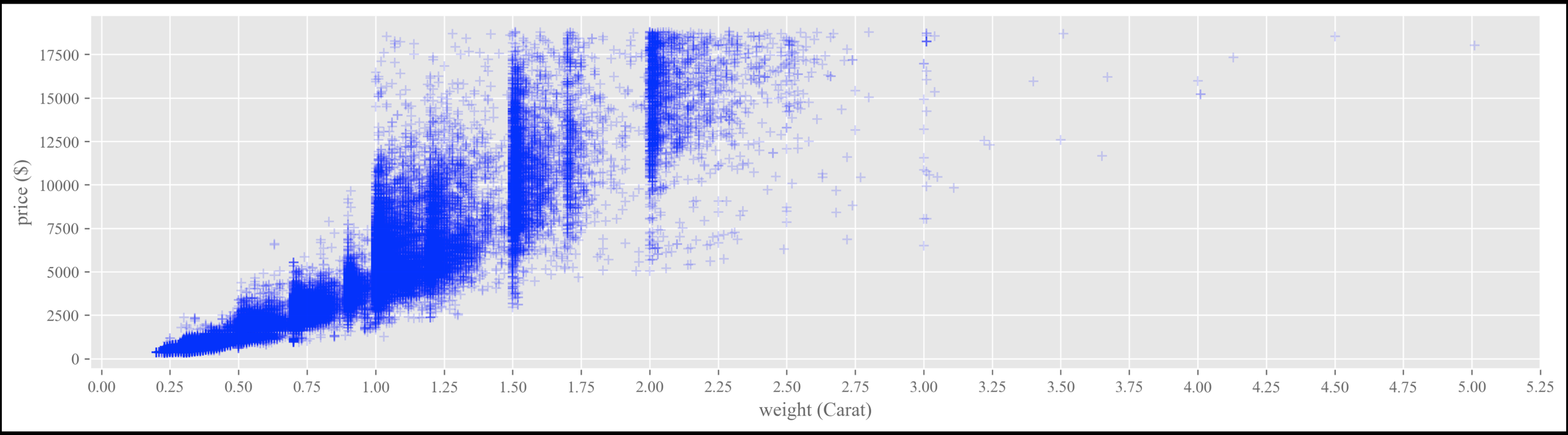


ANACONDA®

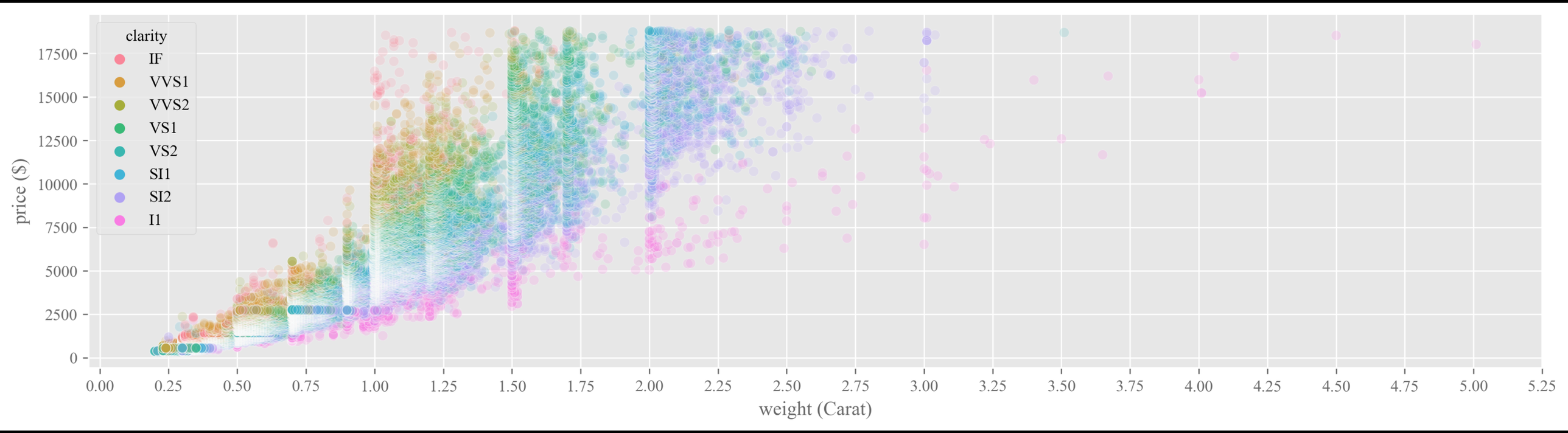
环境配置(environment)

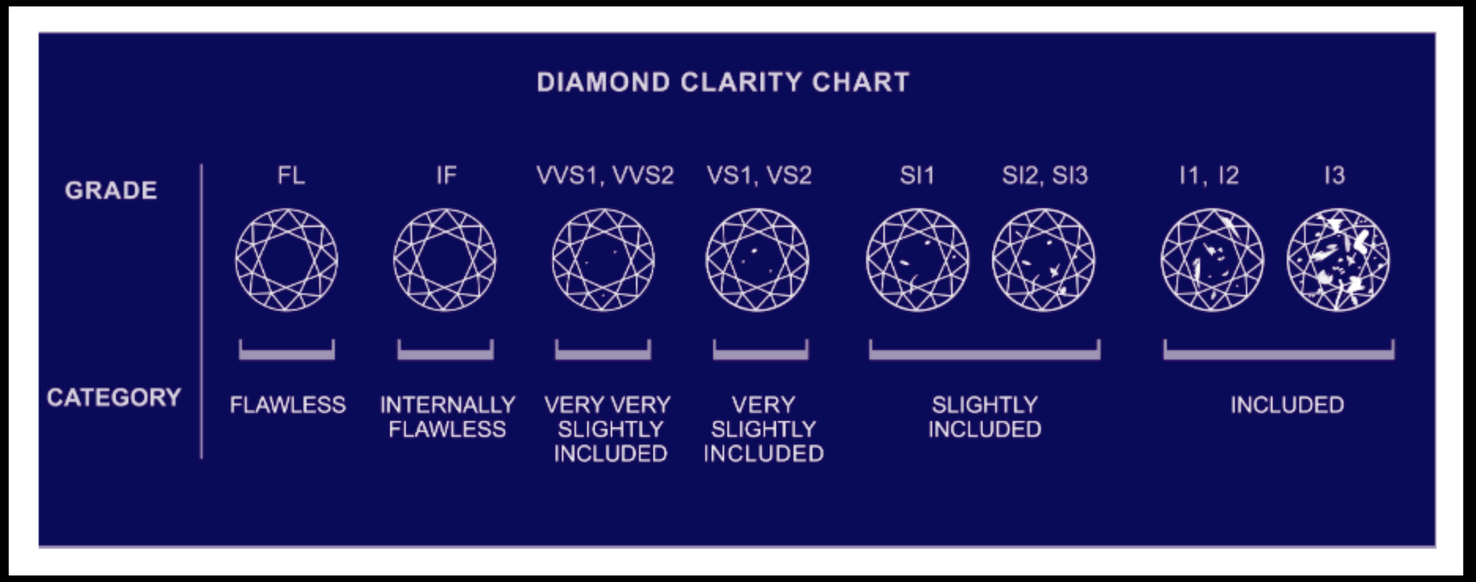
- 使用Anaconda/conda建立多个环境
  - ds: 处理基本的数据处理/可视化
  - r_env: R语言环境
  - dl_torch: pytorch
  - dl_tf: tensorflow
  - 专门的项目配置专门环境：
    - 如：Flask网页；Manim视频；
    - 仿真；强化学习，等等.
  - 策略：
    - 核心包使用conda安装
    - “边缘”包使用conda中的pip安装

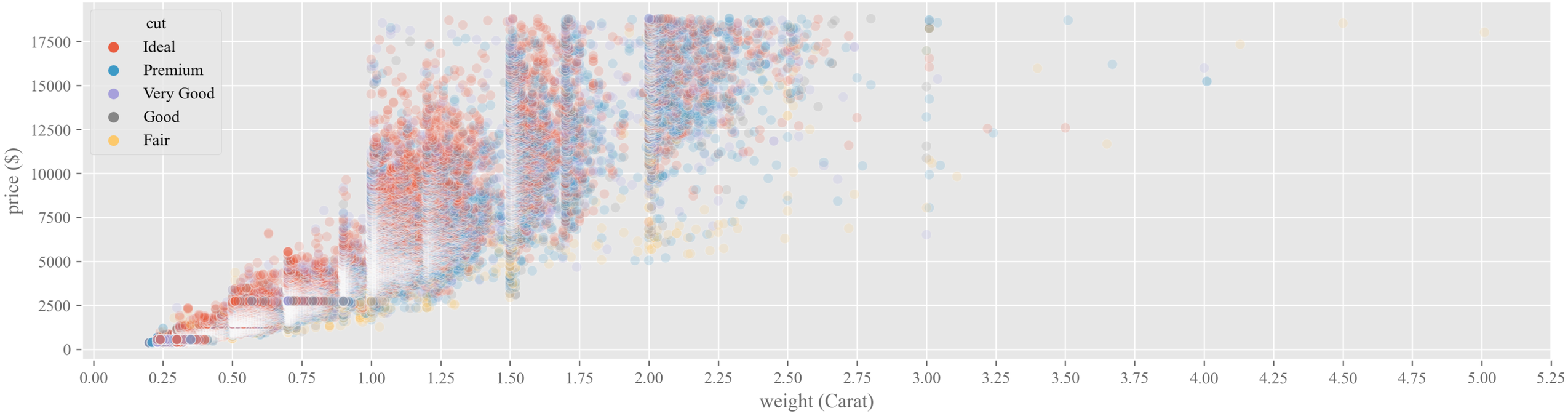
```
(base)                                  ~ % conda env list
# conda environments:
#
                                        
base                     *  /opt/anaconda3
                                        
                                        
                                        
                                        
                                        
                                        
dl                         /opt/anaconda3/envs/dl
ds                         /opt/anaconda3/envs/ds
manim                      /opt/anaconda3/envs/manim
manim2                     /opt/anaconda3/envs/manim2
r_env                      /opt/anaconda3/envs/r_env
spinningup                 /opt/anaconda3/envs/spinningup
```

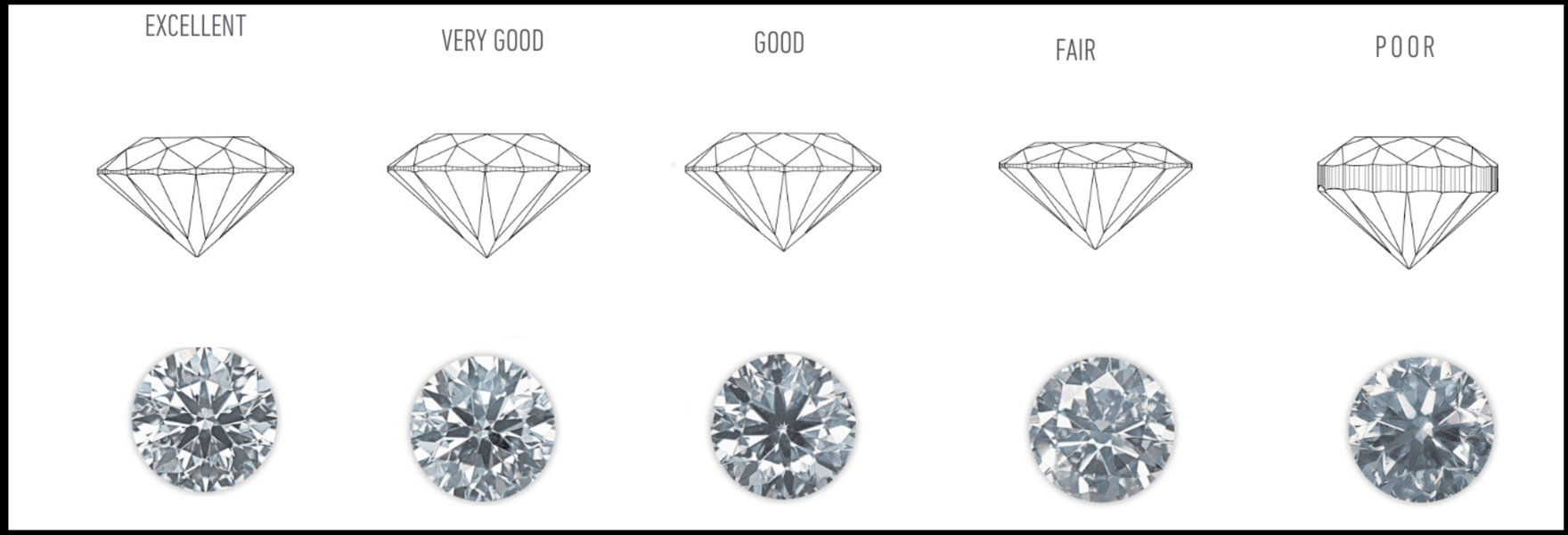

price ($)
17500
15000
12500
10000
7500
5000
2500
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
4.25
4.50
4.75
5.00
5.25
weight (Carat)

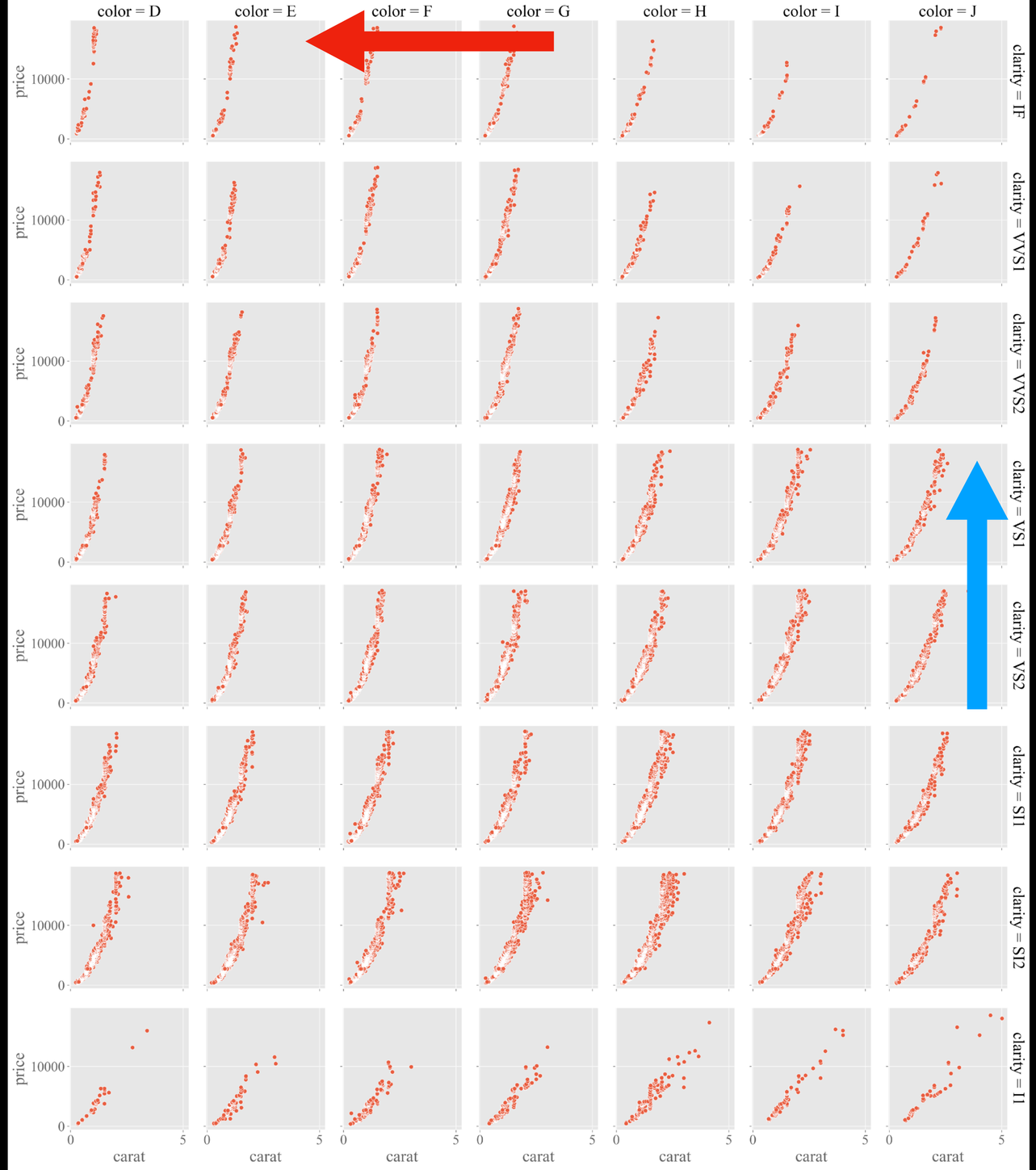

CARAT
CT
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.50
3.00
3.50
4.00
5.00
MM
4.1
5.2
5.8
6.5
7.0
7.4
7.8
8.2
8.8
9.4
9.9
10.4
11.2

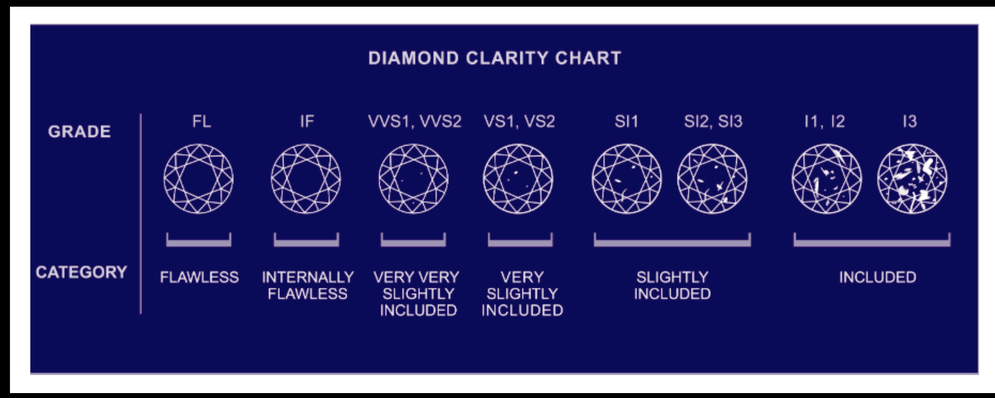

clarity
IF
VVS1
VVS2
VS1
VS2
SI1
SI2
I1
price ($)
17500
15000
12500
10000
7500
5000
2500
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
4.25
4.50
4.75
5.00
5.25
weight (Carat)

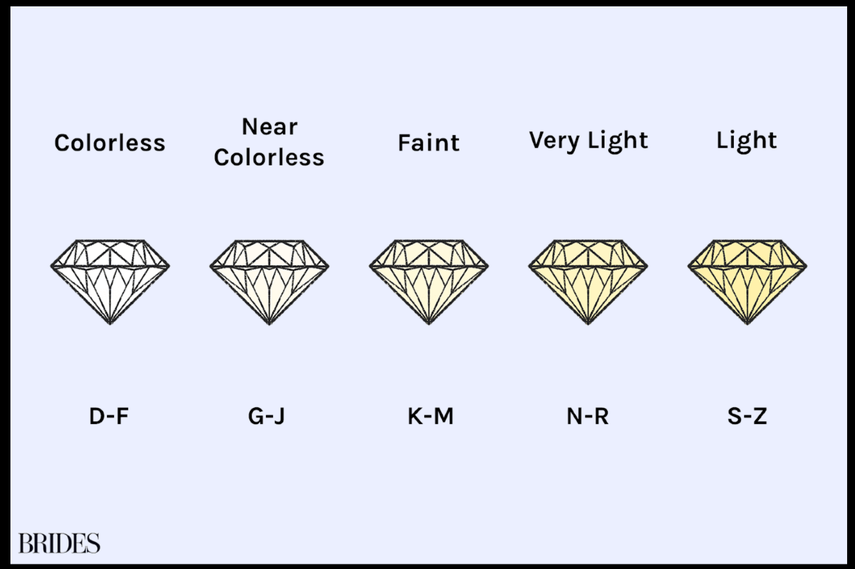

DIAMOND CLARITY CHART
GRADE
CATEGORY
FL
IF
VVS1, VVS2
VS1, VS2
SI1
SI2, SI3
I1, I2
I3
FLAWLESS
INTERNALLY FLAWLESS
VERY VERY SLIGHTLY INCLUDED
VERY SLIGHTLY INCLUDED
SLIGHTLY INCLUDED
INCLUDED


cut
Ideal
Premium
Very Good
Good
Fair
price ($)
17500
15000
12500
10000
7500
5000
2500
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
4.25
4.50
4.75
5.00
5.25
weight (Carat)


EXCELLENT
VERY GOOD
GOOD
FAIR
POOR

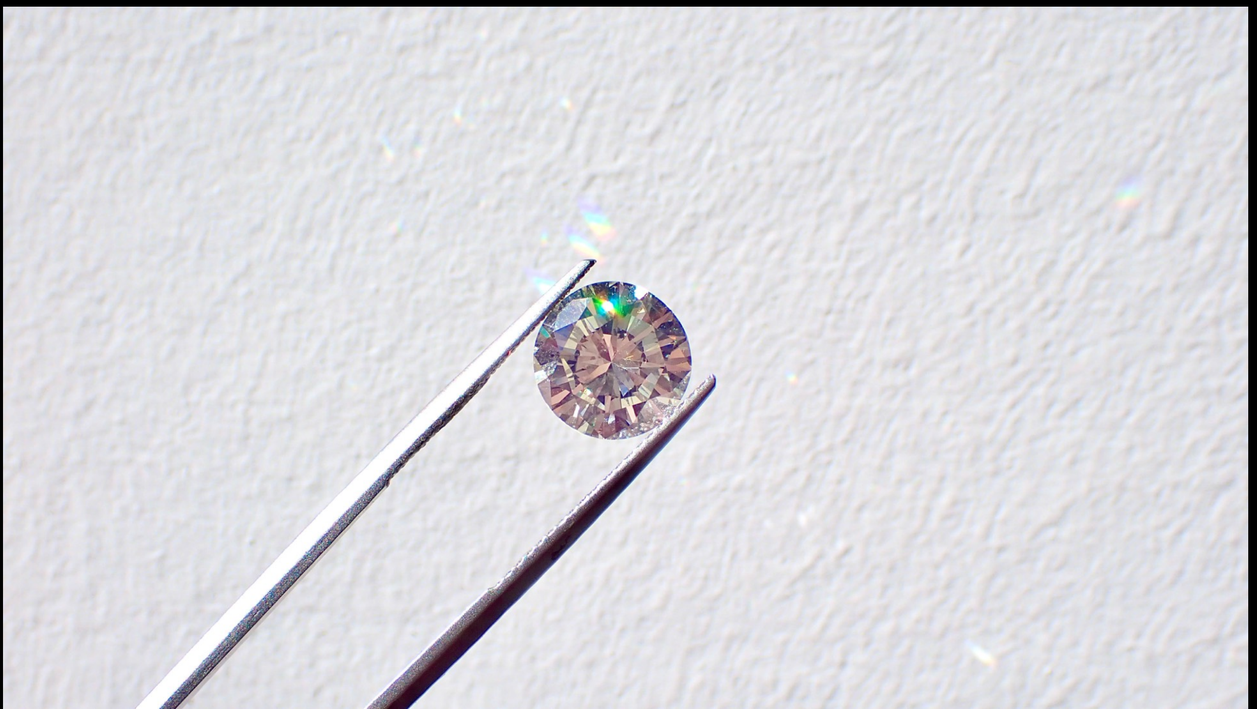

color = D
color = E
color = F
color = G
color = H
color = I
color = J
clarity = IF
clarity = VVS1
clarity = VVS2
clarity = VS1
clarity = VS2
clarity = SI1
clarity = SI2
clarity = I1
price
10000
0
carat
0
5

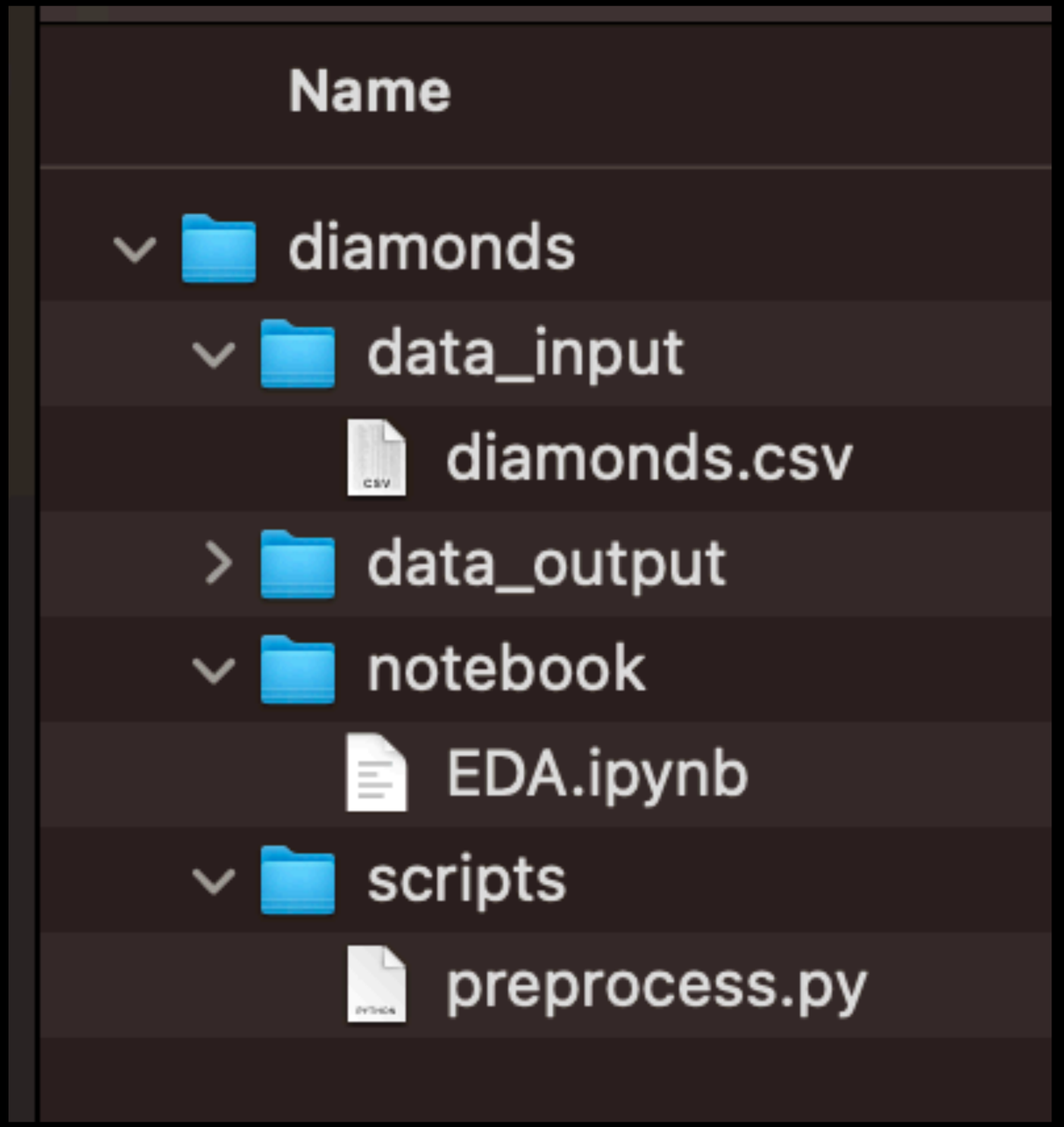

DIAMOND CLARITY CHART
GRADE
CATEGORY
FL
IF
VVS1, VVS2
VS1, VS2
SI1
SI2, SI3
I1, I2
I3
FLAWLESS
INTERNALLY FLAWLESS
VERY VERY SLIGHTLY INCLUDED
VERY SLIGHTLY INCLUDED
SLIGHTLY INCLUDED
INCLUDED


Colorless
Near Colorless
Faint
Very Light
Light
D-F
G-J
K-M
N-R
S-Z
BRIDES

新建项目(new project)

- 最简单的初始文件夹结构
  - notebook: 存储jupyter代码(.ipynb)
  - scripts: 存储python代码 (.py)
  - data_input: 输入数据
  - data_output: 输出数据



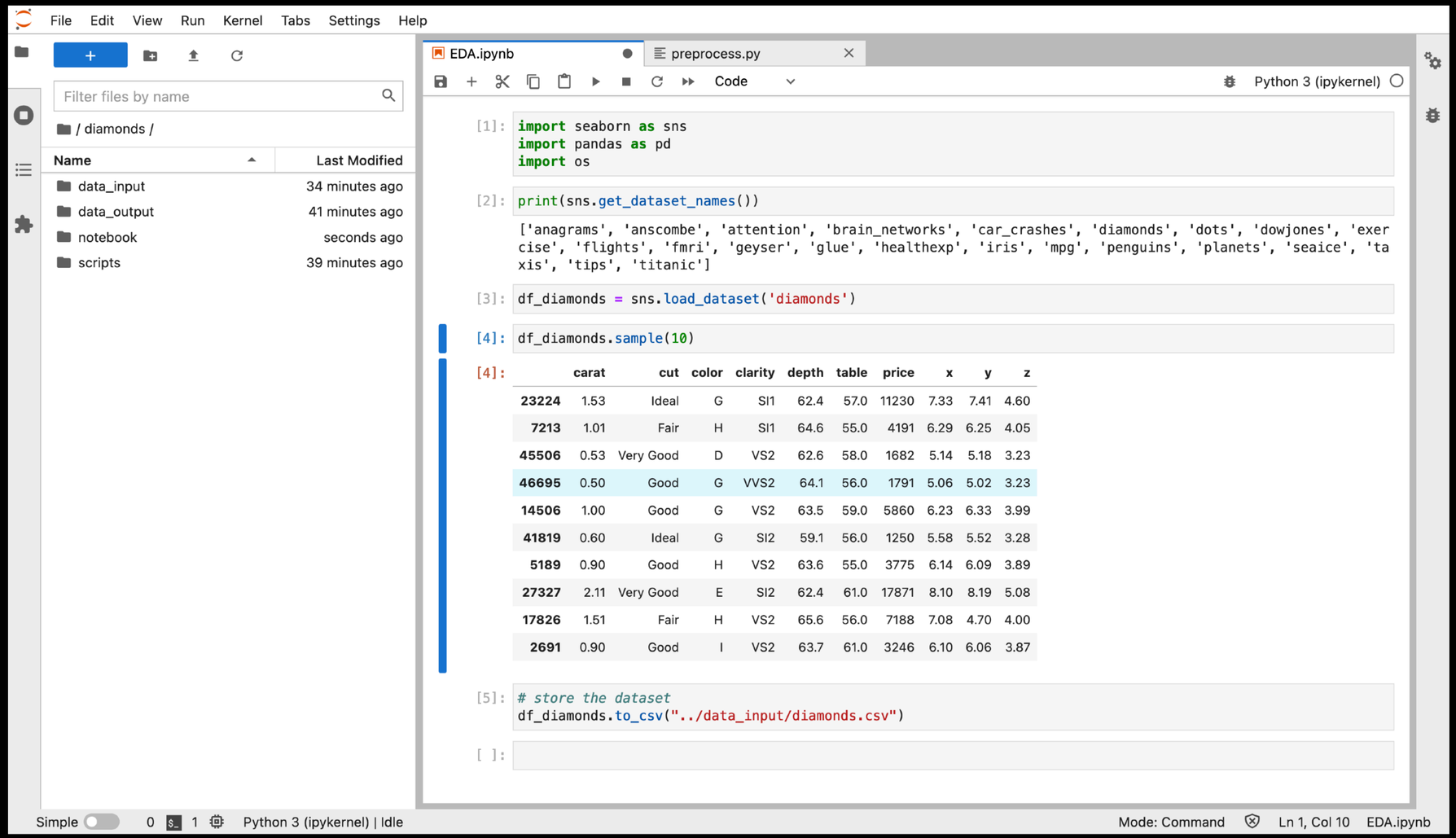


使用IDE开发(environment)

- Jupyter-Lab
  - 主要处理notebook中的代码
- PyCharm
  - Community version社区版本免费
  - 主要处理python文件中的代码
  - 匹配conda环境
- 策略：
  - 同时使用两个软件打开一个项目

开启Jupyter-lab

```
(base)                          ~ % cd                    /Documents/bilibili/diamonds/diamonds
(base)                          diamonds % conda activate ds
(ds)                            diamonds % jupyter-lab
```



PyCharm

New UI | What's New | Features | Learn | Pricing | Download

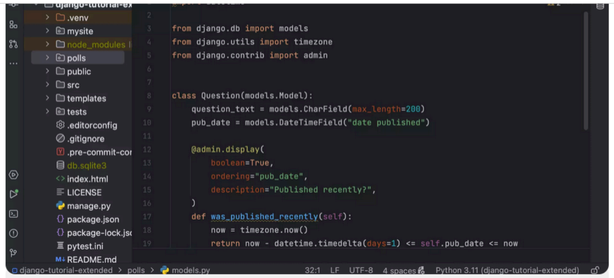
PyCharm Professional

The Python IDE for Professional Developers

Download .dmg

Free 30-day trial

Select an installer for Intel or Apple Silicon



Version: 2023.2
Build: 232.8660.197
25 July 2023

System requirements | Other versions | Installation instructions | Third-party software

We value the vibrant Python community, and that's why we proudly offer the PyCharm Community Edition for free, as our open-source contribution to support the Python ecosystem.

PyCharm Community Edition

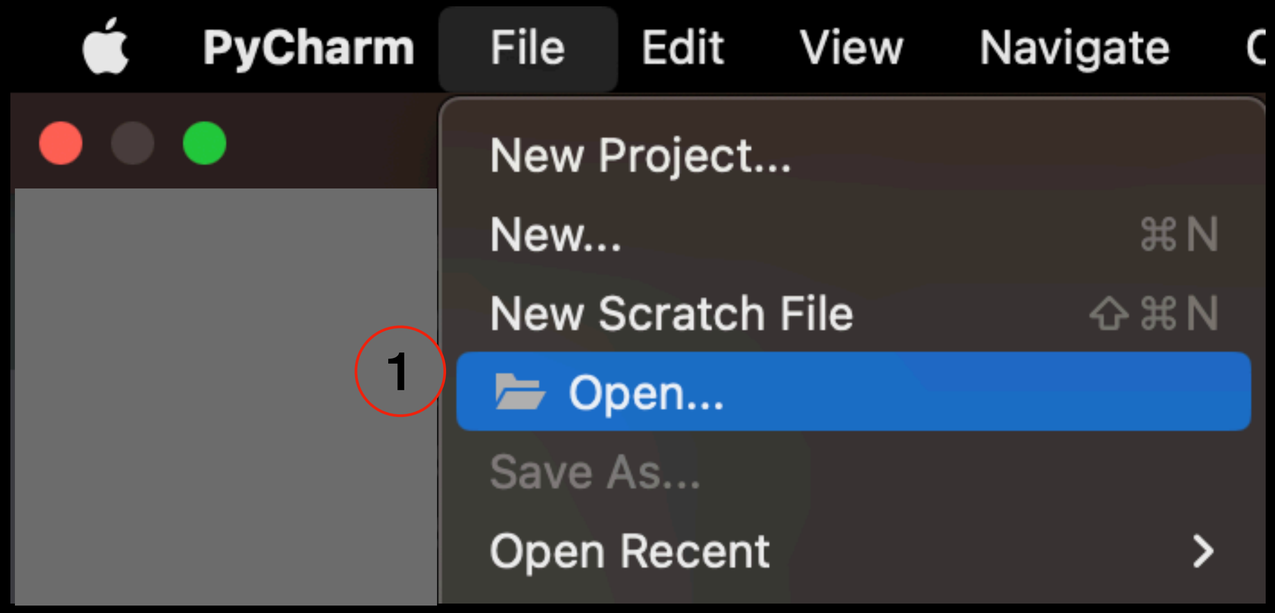
The IDE for Pure Python Development

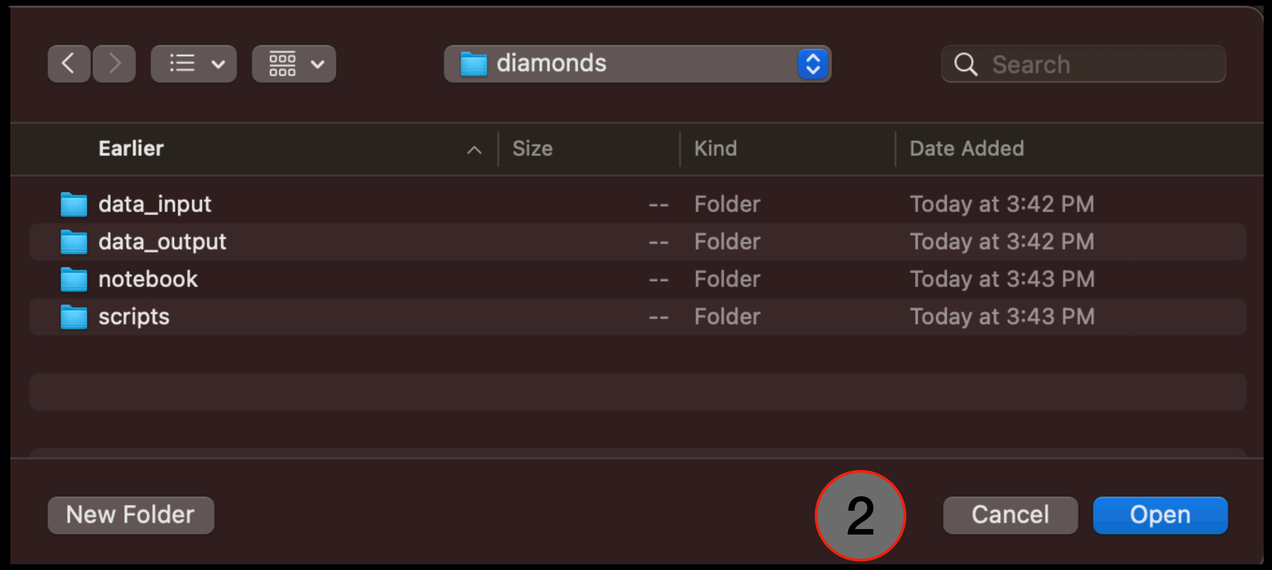
Download .dmg

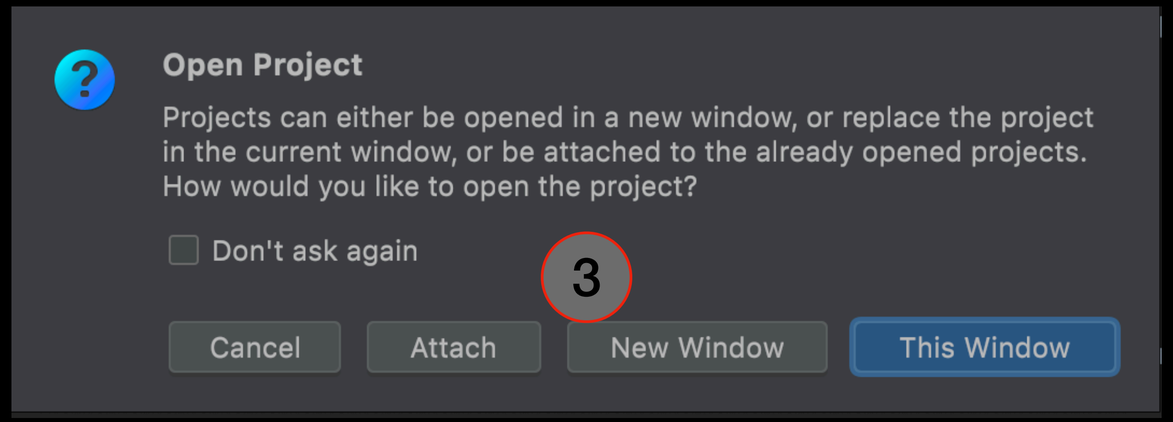
Free, built on open source

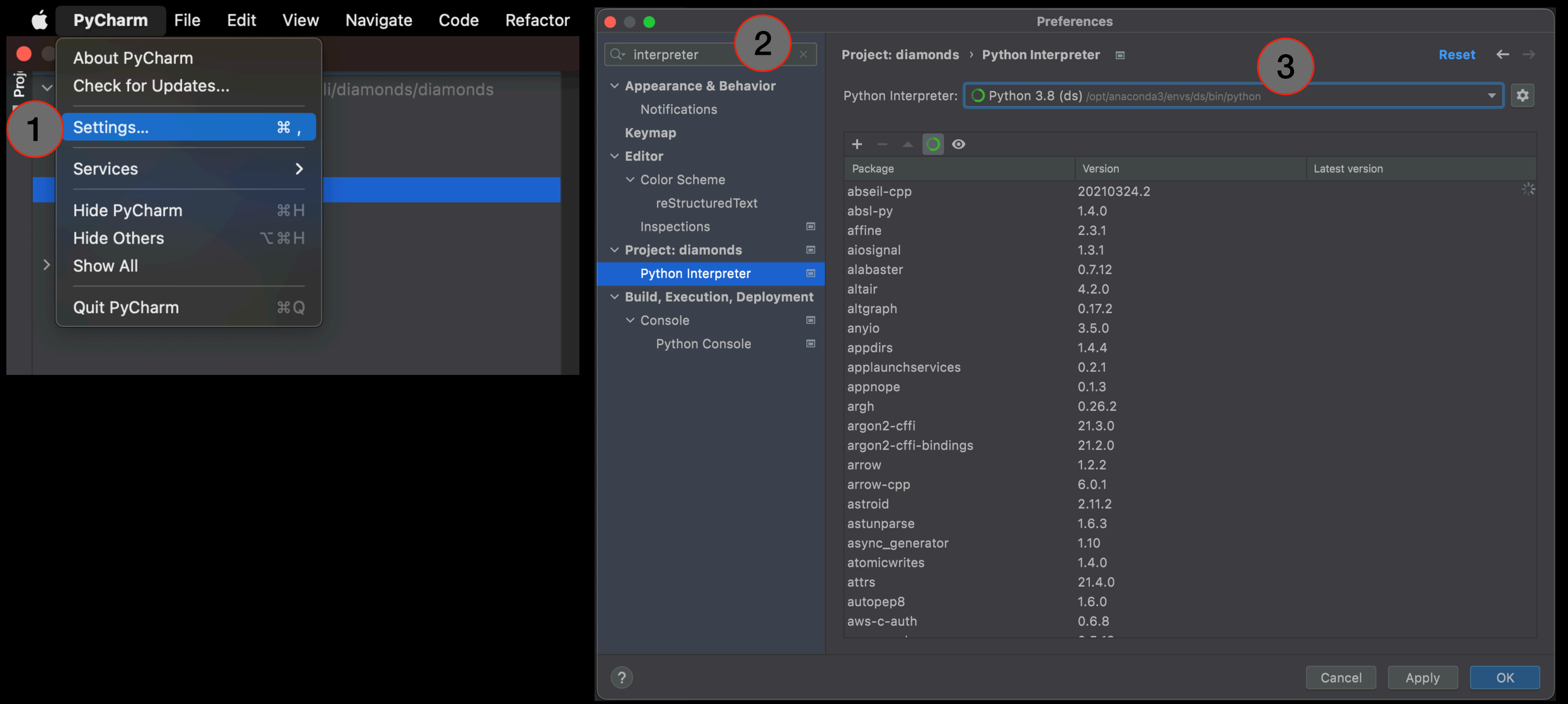
Select an installer for Intel or Apple Silicon

1


PyCharm
File
Edit
View
Navigate
New Project...
New... ⌘N
New Scratch File ⇧⌘N
Open...
Save As...
Open Recent
1


diamonds
Search
Earlier
Size
Kind
Date Added
data_input -- Folder Today at 3:42 PM
data_output -- Folder Today at 3:42 PM
notebook -- Folder Today at 3:43 PM
scripts -- Folder Today at 3:43 PM
New Folder
2
Cancel
Open


Open Project
Projects can either be opened in a new window, or replace the project in the current window, or be attached to the already opened projects. How would you like to open the project?
Don't ask again
3
Cancel
Attach
New Window
This Window


PyCharm
File
Edit
View
Navigate
Code
Refactor
About PyCharm
Check for Updates...
Settings... ⌘ ,
Services
Hide PyCharm ⌘H
Hide Others ⌥⌘H
Show All
Quit PyCharm ⌘Q
1
li/diamonds/diamonds
Preferences
interpreter
2
Appearance & Behavior
Notifications
Keymap
Editor
Color Scheme
reStructuredText
Inspections
Project: diamonds
Python Interpreter
Build, Execution, Deployment
Console
Python Console
Project: diamonds › Python Interpreter
Reset
3
Python Interpreter: Python 3.8 (ds) /opt/anaconda3/envs/ds/bin/python
Package
Version
Latest version
abseil-cpp 20210324.2
absl-py 1.4.0
affine 2.3.1
aiosignal 1.3.1
alabaster 0.7.12
altair 4.2.0
altgraph 0.17.2
anyio 3.5.0
appdirs 1.4.4
applaunchservices 0.2.1
appnope 0.1.3
argh 0.26.2
argon2-cffi 21.3.0
argon2-cffi-bindings 21.2.0
arrow 1.2.2
arrow-cpp 6.0.1
astroid 2.11.2
astunparse 1.6.3
async_generator 1.10
atomicwrites 1.4.0
attrs 21.4.0
autopep8 1.6.0
aws-c-auth 0.6.8
Cancel
Apply
OK

联合快速卡发(fast development)

✓准备步骤：

- ✓同时使用两个软件打开一个项目
- ✓新建python/notebook文件
- ✓导入数据

- 开发步骤(以Explorative Data Analysis为例)：
  - ✓在Notebook中进行各种实验（如可视化）
  - 将可视化代码写成函数
  - 将函数迁移至python文件中
  - 在notebook中添加路径，导入包和函数
  - 将notebook调整至动态加载模式

联合快速卡发(fast development)

EDA.ipynb

Markdown | Python 3 (ipykernel)

```
[1]: import seaborn as sns
     import pandas as pd
     import os
```

1

```
]: import sys
   sys.path.append("../scripts")
   import preprocess as pre
```

```
[3]: # use this method so everytime you changed your Python file (.p
     # the files are re-imported again at the time you
     # running the next cell in Jupyter Notebook (.ipynb)
     %load_ext autoreload
     %autoreload 2
```

4

```
]: pre.hello_world()
```

```
Hello World!
```

preprocess.py

```
1  import os
2
3  import matplotlib.pyplot as plt
4  import seaborn as sns
5  import numpy as np
6  import pandas as pd
7
```

2

```
def hello_world():
    print("Hello World!")
```

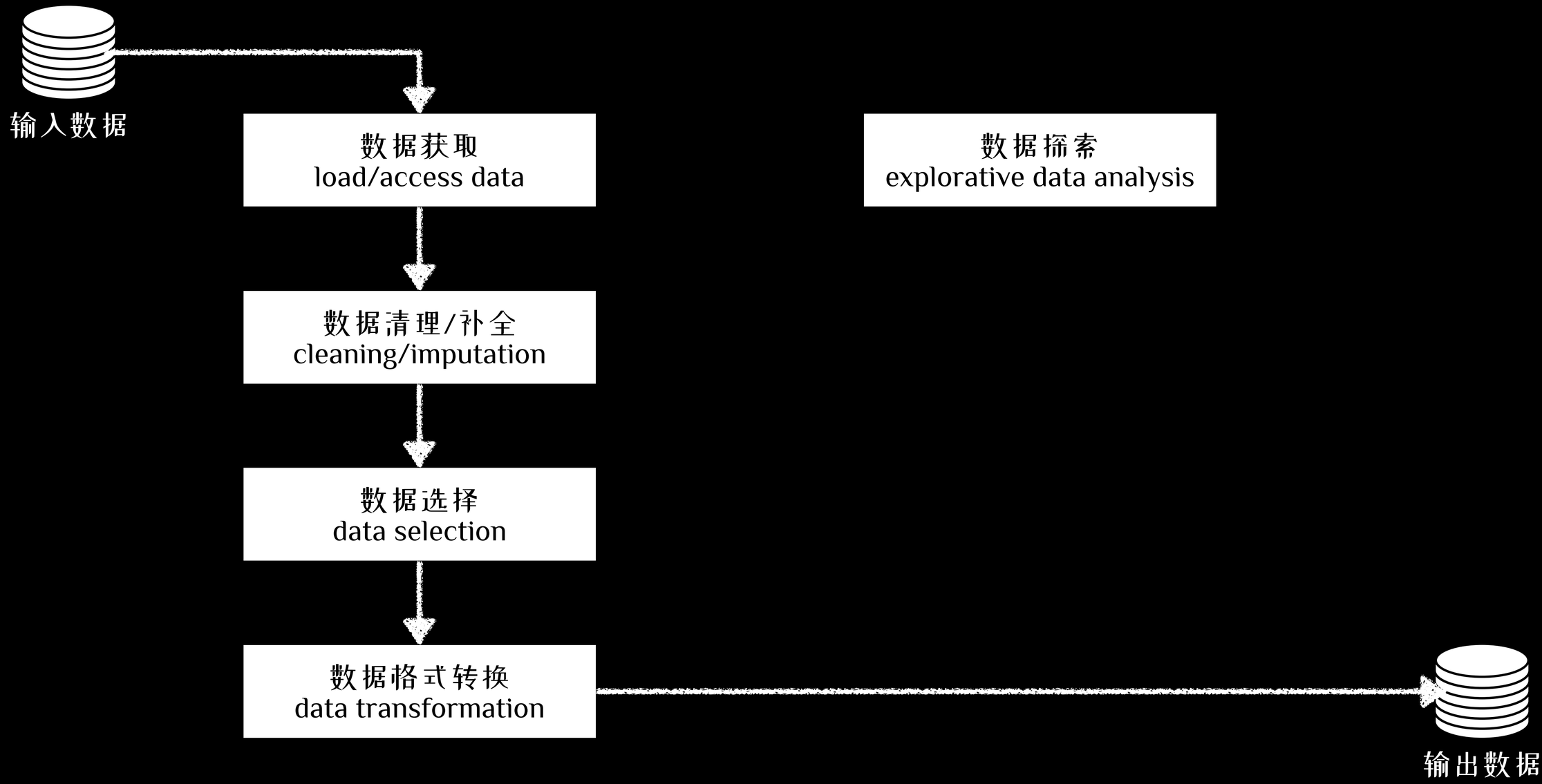
```
11
12 #------below are methods loading the diamond dataset------
13 def load_data_from_seaborn():
14     # the diamond dataset is contained in the seaborn package
15     df = sns.load_dataset("diamonds")
16     return df
17
18 def load_data_from_local(pth="../data_input/diamonds.csv"):
19     df = pd.read_csv(pth, index_col=0)
20     return df
```

3 保存

数据科学项目的具体步骤

- 数据管道 (data pipeline)
  - 管道1: 数据预处理 (pre-process)
  - 管道2: 建立/测试模型 (机器学习流程) (analysis)
  - 管道3: 评估并汇报结果 (post-analysis)

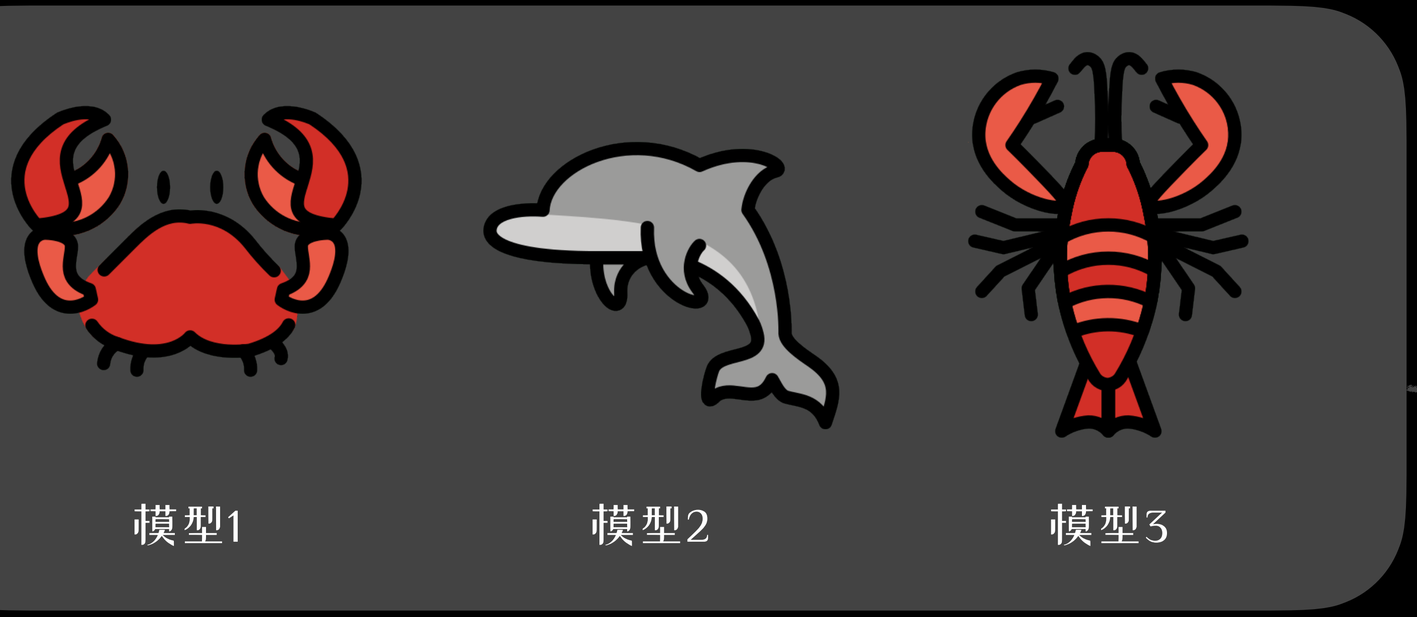
管道1/步骤1: 数据预处理



管道2/步骤2: 建立/测试模型(例:机器学习流程)

数据集

训练集 | 验证集 | 测试集



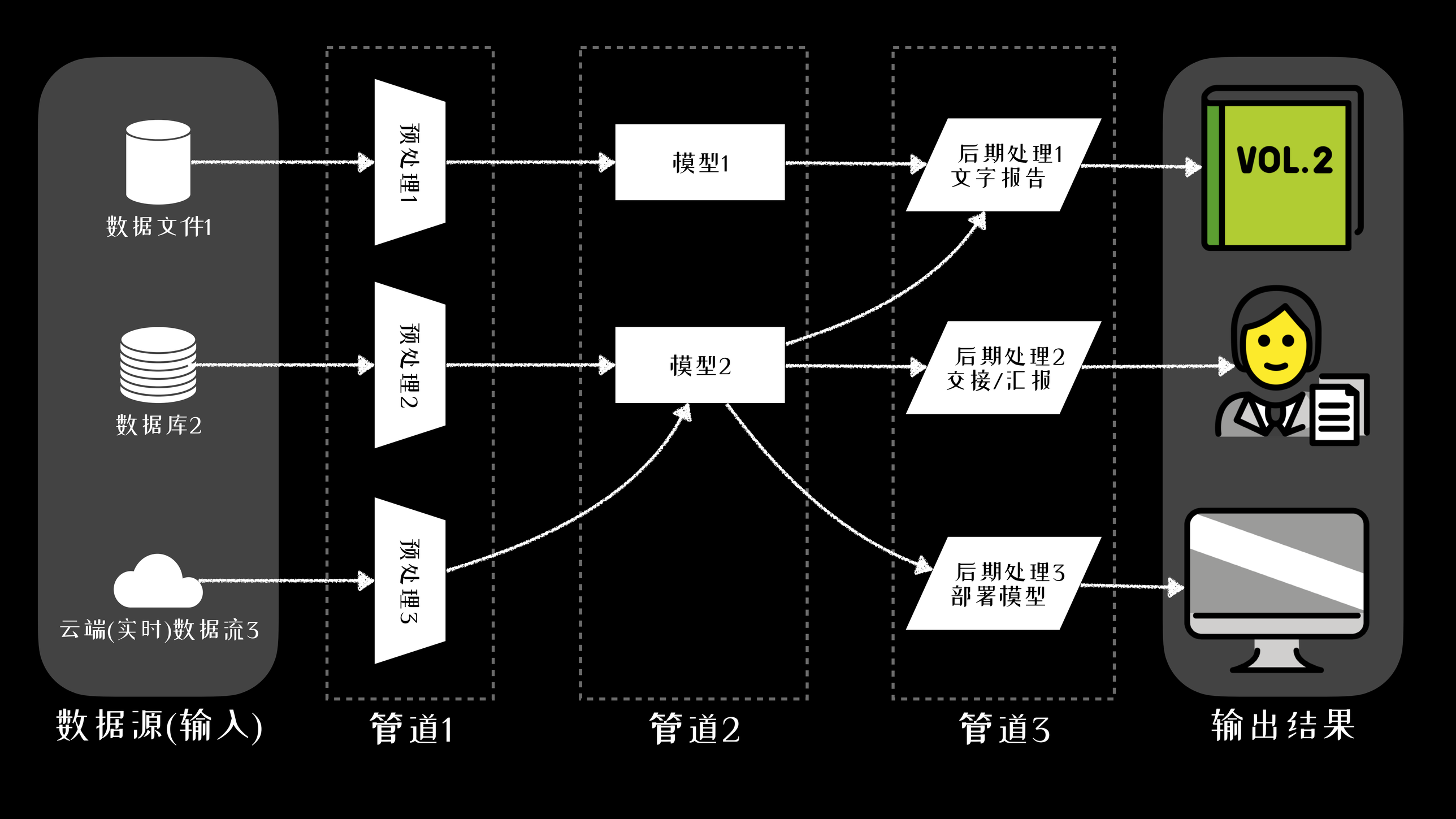
模型1 模型2 模型3

Accuracy,
MAE,
RMSE,
...

输出数据,模型,显著性...

管道3/步骤3:评估并汇报/转交成果

- 目标1：将结果呈现给他人
  - 上级，同事，甲方等等
  - 画图：需要高质量
  - 文字：写报告，论文，PPT，网页...
- 目标2：提交最终/部署模型


数据文件1
数据库2
云端(实时)数据流3
数据源(输入)
预处理1
预处理2
预处理3
管道1
模型1
模型2
管道2
后期处理1
文字报告
后期处理2
交接/汇报
后期处理3
部署模型
管道3
VOL.2
输出结果

需要尽量遵循的原则

- 原子化
- 面向对象(过程标准化)
- 配置文件
- 保存全部结果
- 实验可重复
- 数据与代码分离
- 代码规范/充分注释原则
- git/GitHub管理
- 充分利用多个程序入口
- 测试原则

```python
# ------below are methods for visualization------
def set_plot_style():
    plt.style.use('ggplot')
    # set font to Times New Roman
    plt.rcParams["font.family"] = "serif"
    plt.rcParams["font.serif"] = ["Times New Roman"]
    plt.rcParams["font.size"] = 20
    # set pixels per inches
    plt.rcParams['figure.dpi'] = 300


def plot_price_carat(df, ax=None):
    if ax is None:
        fig, ax = plt.subplots(1, 1, dpi=300, figsize=(16, 4))
    ax.plot(df.carat, df.price, "b+", alpha=0.2)
    ax.set_xlabel("weight (Carat)")
    ax.set_ylabel("price ($)")
    plt.xticks(np.arange(0, 5.5, 0.25))
    return ax


def plot_price_carat_seaborn(df, ax=None, hue="clarity"):
    if ax is None:
        fig, ax = plt.subplots(1, 1, dpi=300, figsize=(16, 4))
    # ax.plot(df.carat, df.price, "b+", alpha=0.2)
    sns.scatterplot(data=df, x="carat", y="price",
                    hue=hue, ax=ax, alpha=0.2)
    ax.set_xlabel("weight (Carat)")
    ax.set_ylabel("price ($)")
    plt.xticks(np.arange(0, 5.5, 0.25))
    return ax


def plot_price_carat_facet(df):
    sea = sns.FacetGrid(
        df, row="clarity", col="color",
        margin_titles=True)
    sea.map(sns.scatterplot, "carat", "price")
```

需要尽量遵循的原则

- 原子化
- 面向对象(过程抽象化)
- 配置文件
- 保存全部结果
- 实验可重复
- 数据与代码分离
- 代码规范/充分注释原则
- git/GitHub管理
- 充分利用多个程序入口
- 测试原则

```python
# I like to write such processing object to organize the data pipeline
class DiamondPreprocess:
    def __init__(self, config_dict=None):
        if config_dict is None:
            config_dict = {
                "source": "local",  # either "local" or "seaborn"
                "carat_range": None,
                "cuts_types": [],
                "colors": [],
            }

        self.config = config_dict
        self.df = None

    def step0_load_dataset(self):
        source = self.config["source"]
        if source == "seaborn":
            self.df = load_data_from_seaborn()
        elif source == "seaborn":
            self.df = load_data_from_local()
        else:
            print("the source is not recognized")
            # consider to raise an Error if you want
            # raise Exception(f"the data source {source} is not recognized")

        # after loading the dataset, set the column types correctly
        set_categorical_columns()

    def step1_view_basic_info(self):
        print_diamond_basic_info(df)

    def step2_data_selection(self):
        # step 1-1. filter by carat
        carat_lb, carat_ub = *self.config["carat_range"]
        df = filt_by_carat(df, carat_lb, carat_ub)

        # step 1-2. filter by cuts
        cuts_types = self.config["cuts"]
        df = filt_by_cuts(cuts_types)

        # step 2-3. filter by colors
        colors = self.config["colors"]
        df = filt_by_price(df)
        return df

    def step3_data_cleaning(self):
        pass

    def step4_data_transformation(self):
        pass

    def step5_export_outputs(self):
        self.df.to_csv("../data_output/diamonds_of_interest.csv")

    def pipeline(self):
        # just chain the methods step by step
        self.step0_load_dataset()
        self.step1_data_selection()
        self.step2_data_cleaning()
        self.step4_data_transformation()
        self.step5_export_outputs()
```

```yaml
dataset:
  script_path: ../datasets/cifar10_keras.py
model:
  script_path: ../models/optimized.py
optimizer:
  script_path: ../optimizers/adam_keras.py
  initial_lr: 0.0001
train:
  script_path: ../train/train_keras.py
  artifacts_path: ../artifacts/cifar10_opt/
  batch_size: 64
  epochs: 1000
  data_augmentation:
    samplewise_center: False
    samplewise_std_normalization: False
    rotation_range: 0
    width_shift_range: 0.1
    height_shift_range: 0.1
    horizontal_flip: True
    vertical_flip: False
    zoom_range: 0
    shear_range: 0
    channel_shift_range: 0
    featurewise_center: False
    zca_whitening: False
evaluate:
  batch_size: 1000
  augmentation_factor: 32
  data_augmentation:
    samplewise_center: False
    samplewise_std_normalization: False
    rotation_range: 0
    width_shift_range: 0.15
    height_shift_range: 0.15
    horizontal_flip: True
    vertical_flip: False
    zoom_range: 0
    shear_range: 0
    channel_shift_range: 0
    featurewise_center: False
    zca_whitening: False
```

my_configuration.yaml hosted with ❤️ by GitHub

yaml文件

```json
{
  "dataset": {
    "script_path": "../datasets/cifar10_keras.py"
  },
  "model": {
    "script_path": "../models/optimized.py"
  },
  "optimizer": {
    "script_path": "../optimizers/adam_keras.py",
    "initial_lr": 0.0001
  },
  "train": {
    "script_path": "../train/train_keras.py",
    "artifacts_path": "../artifacts/cifar10_opt/",
    "batch_size": 64,
    "epochs": 1000,
    "data_augmentation": {
      "samplewise_center": false,
      "samplewise_std_normalization": false,
      "rotation_range": 0,
      "width_shift_range": 0.1,
      "height_shift_range": 0.1,
      "horizontal_flip": true,
      "vertical_flip": false,
      "zoom_range": 0,
      "shear_range": 0,
      "channel_shift_range": 0,
      "featurewise_center": false,
      "zca_whitening": false
    }
  },
  "evaluate": {
    "batch_size": 1000,
    "augmentation_factor": 32,
    "data_augmentation": {
      "samplewise_center": false,
      "samplewise_std_normalization": false,
      "rotation_range": 0,
      "width_shift_range": 0.15,
      "height_shift_range": 0.15,
      "horizontal_flip": true,
      "vertical_flip": false,
      "zoom_range": 0,
      "shear_range": 0,
      "channel_shift_range": 0,
      "featurewise_center": false,
      "zca_whitening": false
    }
  }
}
```

JSON文件

```python
{'dataset': {'script_path': '../datasets/cifar10_keras.py'},
 'model': {'script_path': '../models/optimized.py'},
 'optimizer': {'script_path': '../optimizers/adam_keras.py',
  'initial_lr': 0.0001},
 'train': {'script_path': '../train/train_keras.py',
  'artifacts_path': '../artifacts/cifar10_opt/',
  'batch_size': 64,
  'epochs': 1000,
  'data_augmentation': {'samplewise_center': False,
   'samplewise_std_normalization': False,
   'rotation_range': 0,
   'width_shift_range': 0.1,
   'height_shift_range': 0.1,
   'horizontal_flip': True,
   'vertical_flip': False,
   'zoom_range': 0,
   'shear_range': 0,
   'channel_shift_range': 0,
   'featurewise_center': False,
   'zca_whitening': False}},
 'evaluate': {'batch_size': 1000,
  'augmentation_factor': 32,
  'data_augmentation': {'samplewise_center': False,
   'samplewise_std_normalization': False,
   'rotation_range': 0,
   'width_shift_range': 0.15,
   'height_shift_range': 0.15,
   'horizontal_flip': True,
   'vertical_flip': False,
   'zoom_range': 0,
   'shear_range': 0,
   'channel_shift_range': 0,
   'featurewise_center': False,
   'zca_whitening': False}}}
```

Python 基本数据结构：字典Dictionary

需要尽量遵循的原则

- 原子化
- 面向对象
- 配置文件
- 保存全部结果
- 实验可重复
- 数据与代码分离
- 代码规范/充分注释原则
- git/GitHub管理
- 充分利用多个程序入口
- 测试原则

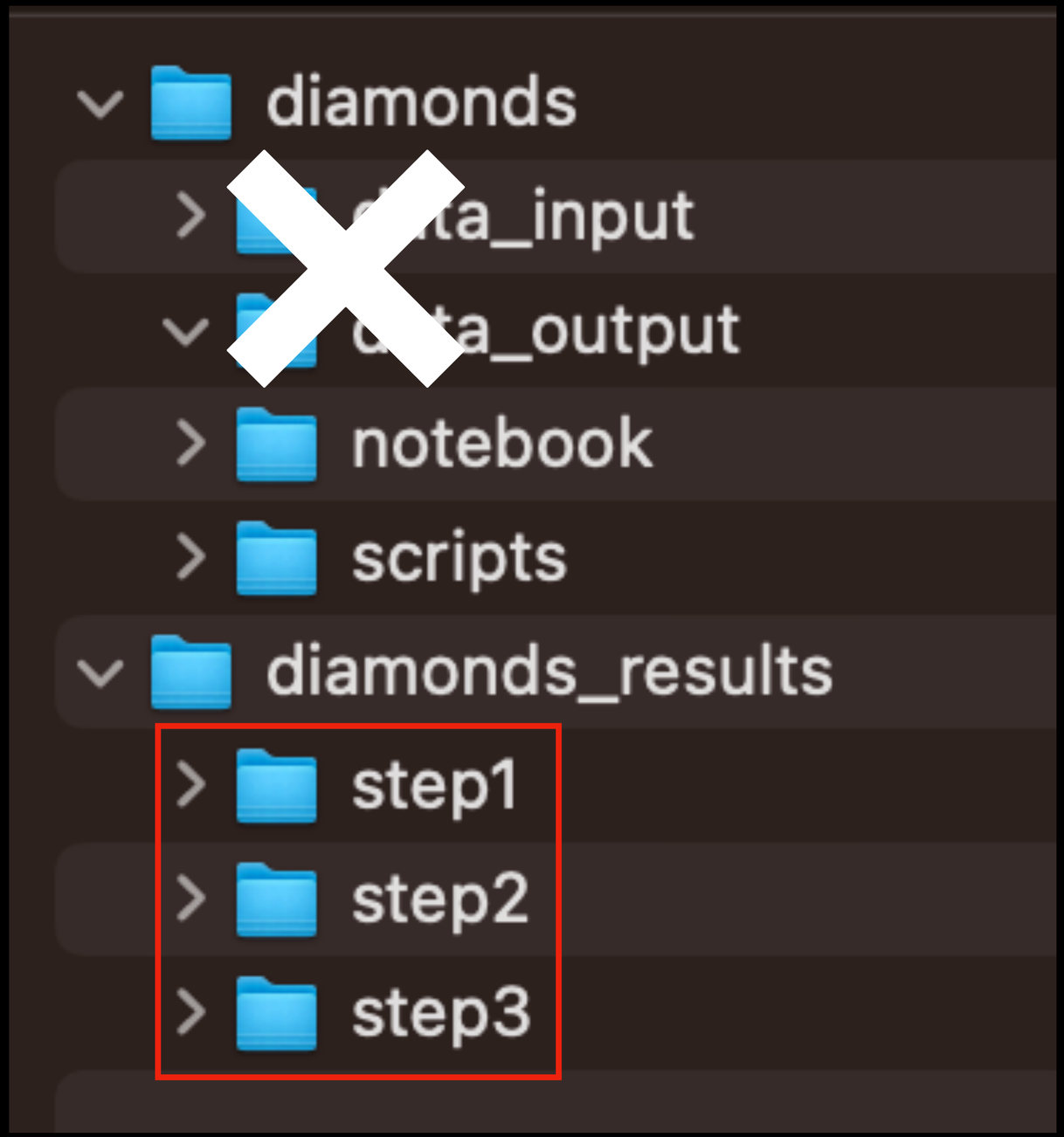
`np.random.seed(0)` makes the random numbers predictable

```
>>> numpy.random.seed(0) ; numpy.random.rand(4)
array([ 0.55,  0.72,  0.6 ,  0.54])
>>> numpy.random.seed(0) ; numpy.random.rand(4)
array([ 0.55,  0.72,  0.6 ,  0.54])
```

```
>>> numpy.random.rand(4)
array([ 0.42,  0.65,  0.44,  0.89])
>>> numpy.random.rand(4)
array([ 0.96,  0.38,  0.79,  0.53])
```

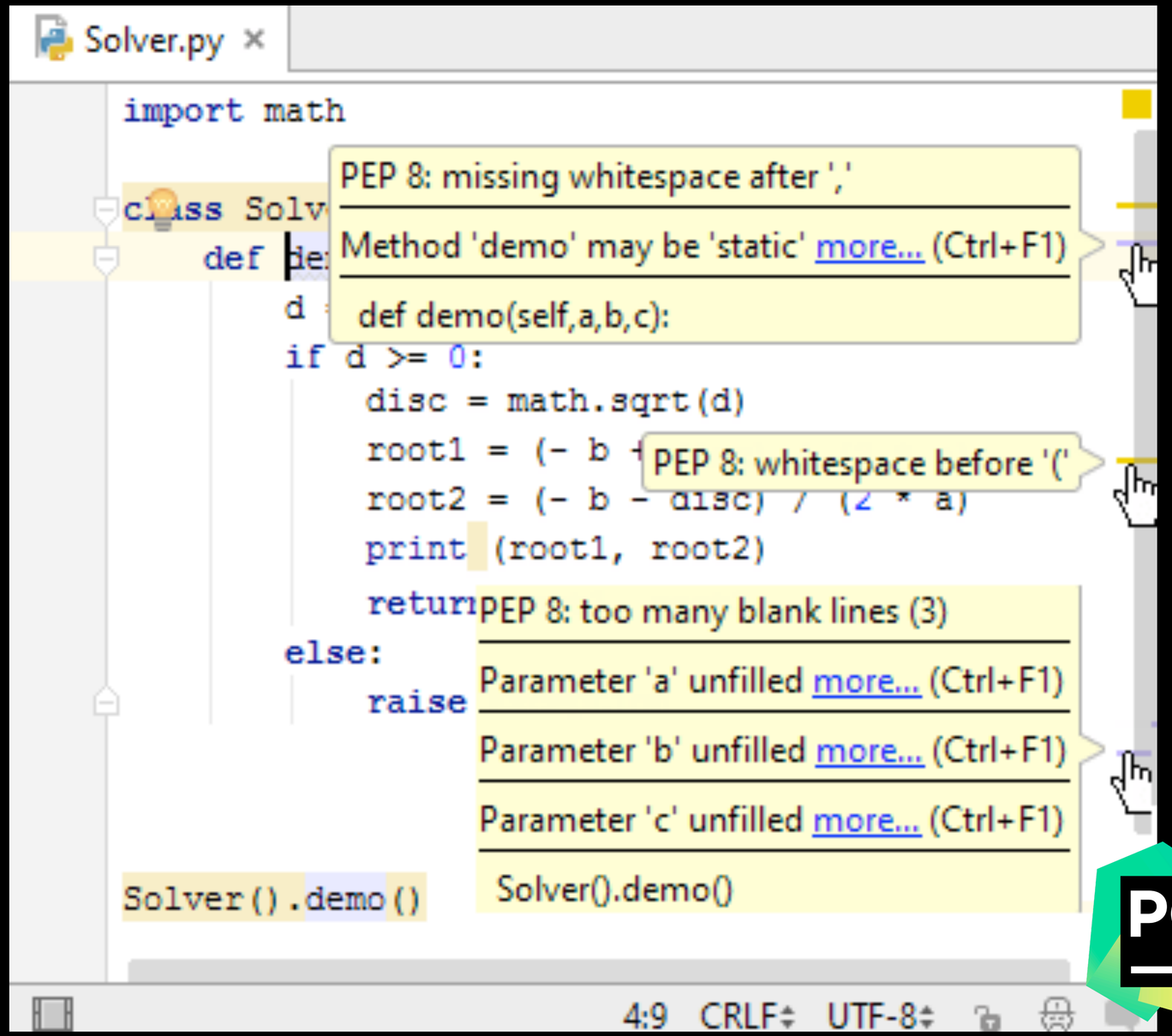
需要尽量遵循的原则

- 原子化
- 面向对象(过程抽象化)
- 配置文件
- 保存全部结果
- 实验可重复
- 数据与代码分离
- 代码规范/充分注释原则
- git/GitHub管理
- 充分利用多个程序入口
- 测试原则



需要尽量遵循的原则

- 原子化
- 面向对象(过程标准化)
- 配置文件
- 保存全部结果
- 实验可重复
- 数据与代码分离
- 代码规范/充分注释原则
- git/GitHub管理
- 充分利用多个程序入口
- 测试原则



总结

- 核心代码的质量高，则整体犯错的机会会少很多
- 要在开发效率（速度）与可靠性之间找到一个平衡点
- 没有完美的数据流，需要单独花精力重构代码
- 针对你的专业，可以不断积累你的代码库

彩蛋 1.

```
[(ds) [redacted]-MBP ~ % pip install seaborn
Requirement already satisfied: seaborn in /opt/anaconda3/envs/ds/
Requirement already satisfied: matplotlib>=2.2 in /opt/anaconda3/
Requirement already satisfied: scipy>=1.0 in /opt/anaconda3/envs/
Requirement already satisfied: pandas>=0.23 in /opt/anaconda3/env
Requireme
Requireme
Requireme
Requireme
Requireme
Requireme
Requireme
Requireme
Requireme
Requireme
```

```
[45]: !pip install seaborn

Requirement already satisfied: seaborn in /opt/anaconda3/envs/ds/lib/python3.8/site-packages (0.1
1.2)
Requirement already satisfied: scipy>=1.0 in /opt/anaconda3/envs/ds/lib/python3.8/site-packages (f
rom seaborn) (1.8.0)
Requirement already satisfied: pandas>=0.23 in /opt/anaconda3/envs/ds/lib/python3.8/site-packages
(from seaborn) (1.5.3)
Requirement already satisfied: numpy>=1.15 in /opt/anaconda3/envs/ds/lib/python3.8/site-packages
(from seaborn) (1.22.3)
Requirement already satisfied: matplotlib>=2.2 in /opt/anaconda3/envs/ds/lib/python3.8/site-packag
es (from seaborn) (3.5.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/anaconda3/envs/ds/lib/python3.8/site-pack
ages (from matplotlib>=2.2->seaborn) (1.4.2)
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/envs/ds/lib/python3.8/site-packag
es (from matplotlib>=2.2->seaborn) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/anaconda3/envs/ds/lib/python3.8/site-packa
ges (from matplotlib>=2.2->seaborn) (3.0.7)
Requirement already satisfied: cycler>=0.10 in /opt/anaconda3/envs/ds/lib/python3.8/site-packages
(from matplotlib>=2.2->seaborn) (0.11.0)
```

彩蛋 1. Google CoLab

```
!pip install torch-scatter
!pip install torch-sparse
!pip install torch-cluster
!pip install torch-spline-conv
!pip install torch-geometric
!pip install torch-geometric-temporal
```

安装一些PyTorch与GNN有关的包

彩蛋 1. Google Colab...

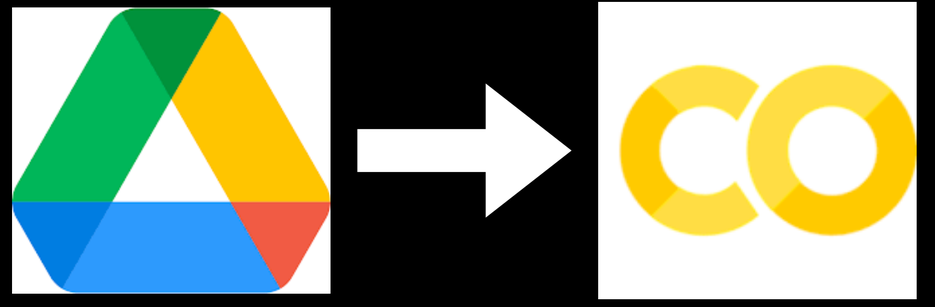
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Permit this notebook to access your Google Drive files?

This notebook is requesting access to your Google Drive files. Granting access to Google Drive will permit code executed in the notebook to modify files in your Google Drive. Make sure to review notebook code prior to allowing this access.

No thanks | Connect to Google Drive

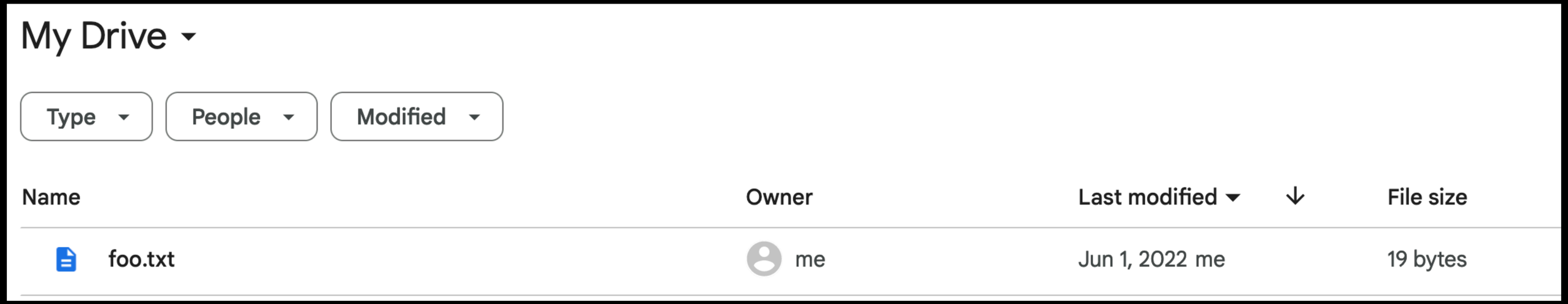


“My Drive” → /content/drive

```
# create foo.txt file in your Google Drive
# with open('/content/drive/My Drive/foo.txt', 'w') as f:
#   f.write('Hello Google Drive!')

# print the text file
# !cat /content/drive/My\ Drive/foo.txt

# disconnect from Google Drive
# drive.flush_and_unmount()
```



安装python包，并保存到Google Drive

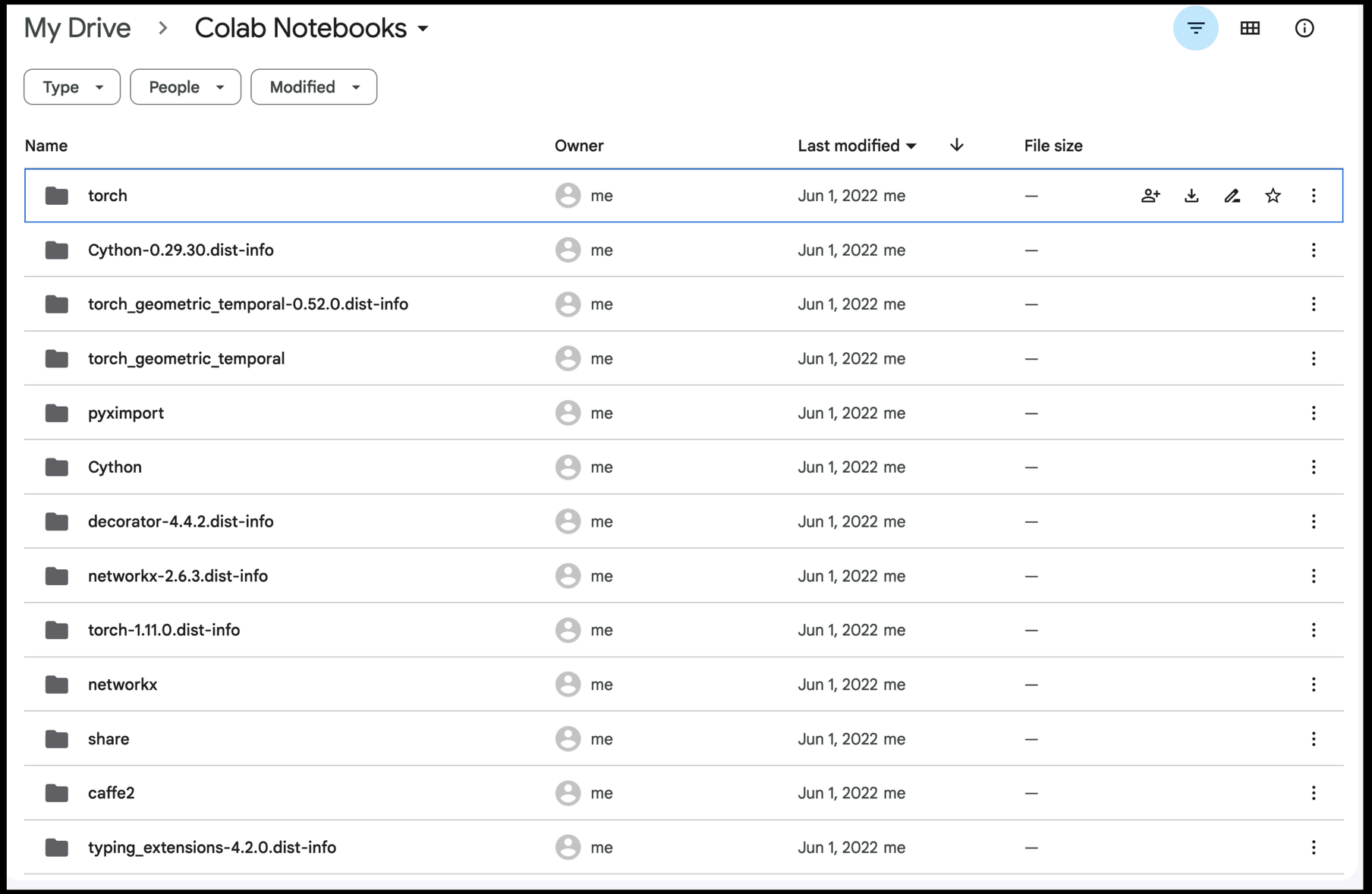
```
[6]  # How do I install a library permanently in Colab?
     # https://stackoverflow.com/questions/55253498/how-do-i-install-a-library-permanently-in-colab
     import os
     from google.colab import drive
     nb_path = '/content/notebooks'

     # create a symbolic link pointing to src named dst
     # os.symlink('/content/drive/My Drive/Colab Notebooks', nb_path)
```

```
[7]  import sys
     sys.path.insert(0,nb_path)
```

```
[8]  # install a package at the preferred location
     # (only install for once)
     !pip install --target=$nb_path torch-scatter --upgrade
     !pip install --target=$nb_path torch-sparse --upgrade
     !pip install --target=$nb_path torch-cluster --upgrade
     !pip install --target=$nb_path torch-spline-conv --upgrade
     !pip install --target=$nb_path torch-geometric --upgrade
     !pip install --target=$nb_path torch-geometric-temporal --upgrade
```

安裝python包到Google Drive路徑



再次执行．．．

```
[6]  # How do I install a library permanently in Colab?
     # https://stackoverflow.com/questions/55253498/how-do-i-install-a-library-permanently-in-colab
     import os
     from google.colab import drive
     nb_path = '/content/notebooks'

     # create a symbolic link pointing to src named dst
     # os.symlink('/content/drive/My Drive/Colab Notebooks', nb_path)
```

```
[7]  import sys
     sys.path.insert(0,nb_path)
```

```
[8]  # Install a package to the preferred location
     # Cloning into for
     !pip install --target=$nb_path torch-scatter --upgrade
     !pip install --target=$nb_path torch-sparse --upgrade
     !pip install --target=$nb_path torch-cluster --upgrade
     !pip install --target=$nb_path torch-spline-conv --upgrade
     !pip install --target=$nb_path torch-geometric --upgrade
     !pip install --target=$nb_path torch-geometric-temporal --upgrade
```

彩蛋 2. Split - Apply - Combine策略

- “分而治之”/Divide and Conquer
- Split-apply-combine
- 找到每个打磨级别中最便宜的钻石💎
  - 1. 分组(Groupby)
  - 2. 分析(Apply)
  - 3. 合并(Combine)

```python
def get_cheapest_by_group(dfg):
    idx = np.argmin(dfg.price.tolist())
    return dfg.iloc[idx, :]
```

```python
cheapest = df_diamonds.groupby(["cut"]).apply(get_cheapest_by_group)
cheapest
```

cut	carat	cut	color	clarity	depth	table	price	x	y	z
Fair	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
Good	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
Ideal	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
Premium	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
Very Good	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48

Split - Apply - Combine策略(大数据)

- Spark: MapReduce
  - 问题：统计水果的数量