开发环境设置

- PyTorch, TensorFlow, NumPy, Jupyter, & VS Code...

Python Packages
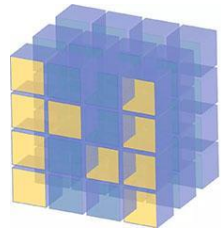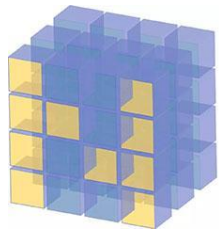
环境安装（pip install）

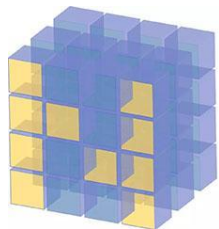# （pip install）镜像加速

-i https://pypi.tuna.tsinghua.edu.cn/simple

开发环境 VS Code

VS Code 插件
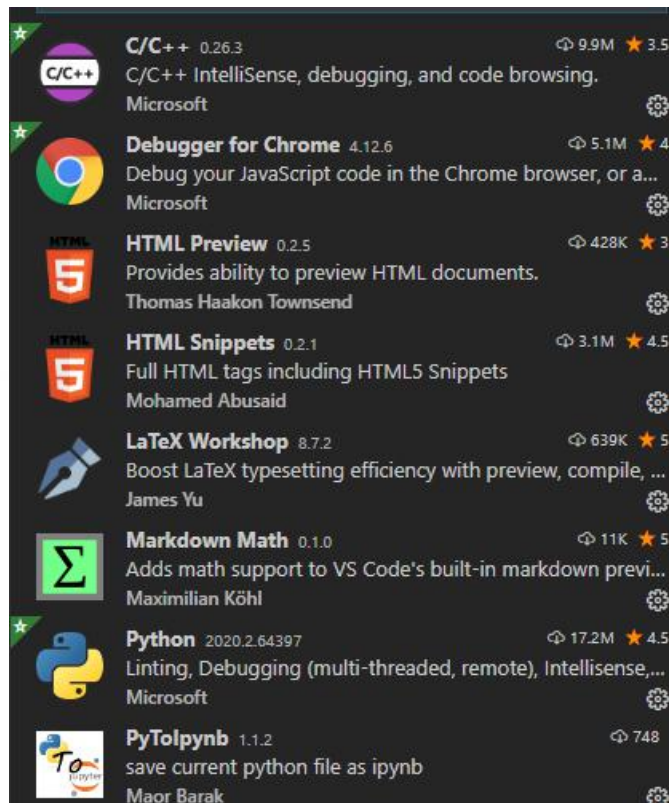
```python
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

```python
import torch
print(torch.__version__)
```

```
1.4.0
```

```python
# Construct a 5x3 matrix, uninitialized
x = torch.empty(5, 3)

# Construct a randomly initialized matrix
x = torch.rand(5, 3)
```

File   Edit   Selection   View   Go   Run   Terminal   Help

RUN   ▶ No Configurations ▾

VARIABLES

Locals
> nn: <module 'torch.nn' from 'E:\\Python37\\lib\\site-...
> np: <module 'numpy' from 'E:\\Python37\\lib\\site-pac...
> torch: <module 'torch' from 'E:\\Python37\\lib\\site-...
> torchvision: <module 'torchvision' from 'E:\\Python37...
> transforms: <module 'torchvision.transforms' from 'E:...
> __builtins__: {'ArithmeticError': <class 'ArithmeticE...
  __cached__: None
  __doc__: None
  __file__: 'f:\\class-20\\web\\pytorch-tutorial\\tutor...
  __loader__: None

WATCH

CALL STACK   PAUSED ON BREAKPOINT
  <module>   main.py 26:1

BREAKPOINTS
☐ Raised Exceptions
☑ Uncaught Exceptions
● ☑ main.py 01-basics\pytorch_basics   26

pytorch-tut.ipynb*   ● main.py

01-basics › pytorch_basics ›

```python
1   import torch
2   import torchvision
3   import torch.nn as nn
4   import numpy as np
5   import torchvision.transforms as transforms
6
7
8   # ================================================================ #
9   #                        Table of Contents                         #
10  # ================================================================ #
11
12  # 1. Basic autograd example 1               (Line 25 to 39)
13  # 2. Basic autograd example 2               (Line 46 to 83)
14  # 3. Loading data from numpy                (Line 90 to 97)
15  # 4. Input pipline                          (Line 104 to 129)
16  # 5. Input pipline for custom dataset       (Line 136 to 156)
17  # 6. Pretrained model                       (Line 163 to 176)
18  # 7. Save and load model                    (Line 183 to 189)
19
20
21  # ================================================================ #
22  #                   1. Basic autograd example 1                    #
23  # ================================================================ #
24
25  # Create tensors.
26  x = torch.tensor(1., requires_grad=True)
27  w = torch.tensor(2., requires_grad=True)
28  b = torch.tensor(3., requires_grad=True)
29
30  # Build a computational graph.
31  y = w * x + b    # y = 2 * x + 3
32
```

OUTPUT   TERMINAL   DEBUG CONSOLE   PROBLEMS 7   1: Python Debug Console ▾

ptvsd\launcher f:\class-20\web\pytorch-tutorial\tutorials\01-basics\pytorch_basics\main.py "