

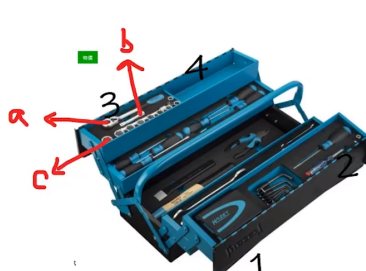
# Pytorch数据集加载

学习的两个法宝：

`dir()`: 打开、看见内部包含的各种内容

`help()` : 详细说明

package  
(pytorch)



`dir(pytorch.3.a)`  
输出:  
此扳手放在特定地方, 然后拧动

`dir():` 打开, 看见  
`help():` 说明书

`dir(pytorch)`  
输出: 1、2、3、4

`dir(pytorch.3)`  
输出: a, b, c

## Dataset类

`torch.utils.data.Dataset`提供的类: `Dataset`, 它是一个抽象类。通过继承和重写这个类就可以定义自己的数据类。需要重写的函数包括:

`__getitem__()` : 如何获取每一条数据, 以及返回相应的数据。

`__len__()` : 求数据集的长度。

下面是这个类的解释文档:

使用`conda activate pytorch` 进入`pytorch`环境, 在环境中使用`jupyter notebook`可以打开`jupyter`

```
Anaconda Prompt (ANACONDA3) - jupyter notebook

(base) C:\Users\Administrator>conda activate pytorch

(pytorch) C:\Users\Administrator>jupyter notebook
[I 21:54:38.337 NotebookApp] [nb_conda_kernels] enabled, 2 kernels found
[I 21:54:38.814 NotebookApp] [nb_conda] enabled
[I 21:54:38.814 NotebookApp] Serving notebooks from local directory: C:\Users\Administrator
[I 21:54:38.815 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 21:54:38.815 NotebookApp] http://localhost:8888/?token=15f65cf921cf6ce801cb9717666ee744836b93c1ed2b352c
[I 21:54:38.815 NotebookApp] or http://127.0.0.1:8888/?token=15f65cf921cf6ce801cb9717666ee744836b93c1ed2b352c
[I 21:54:38.816 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 21:54:38.862 NotebookApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/Administrator/AppData/Roaming/jupyter/runtime/nbserver-9852-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=15f65cf921cf6ce801cb9717666ee744836b93c1ed2b352c
    or http://127.0.0.1:8888/?token=15f65cf921cf6ce801cb9717666ee744836b93c1ed2b352c
```

查看Dataset的解释

```
In [1]: from torch.utils.data import Dataset

In [2]: help(Dataset)

Help on class Dataset in module torch.utils.data.dataset:

class Dataset(typing.Generic)
    Dataset(*args, **kwargs)

    An abstract class representing a :class:`Dataset`.

    All datasets that represent a map from keys to data samples should subclass
    it. All subclasses should overwrite :meth:`~__getitem__`, supporting fetching a
    data sample for a given key. Subclasses could also optionally overwrite
    :meth:`~__len__`, which is expected to return the size of the dataset by many
    :class:`~torch.utils.data.Sampler` implementations and the default options
    of :class:`~torch.utils.data.DataLoader`.

    .. note::
        :class:`~torch.utils.data.DataLoader` by default constructs a index
        sampler that yields integral indices. To make it work with a map-style
        dataset with non-integral indices/keys, a custom sampler must be provided.

    Method resolution order:
        Dataset
        typing.Generic
        builtins.object

    Methods defined here:

    __add__(self, other: 'Dataset[T_co]') -> 'ConcatDataset[T_co]'

    __getitem__(self, index) -> +T_co
```

使用help（）在jupyter中大段文字比较清楚些。

还可以使用后面加问号的形式出来更加详细的说明。

```
In [5]: Dataset?

In [ ]:

Init signature: Dataset(*args, **kwargs)
Docstring:
An abstract class representing a :class:`Dataset`.

All datasets that represent a map from keys to data samples should subclass
it. All subclasses should overwrite :meth:`~__getitem__`, supporting fetching a
data sample for a given key. Subclasses could also optionally overwrite
:meth:`~__len__`, which is expected to return the size of the dataset by many
:class:`~torch.utils.data.Sampler` implementations and the default options
of :class:`~torch.utils.data.DataLoader`.

.. note::
    :class:`~torch.utils.data.DataLoader` by default constructs a index
    sampler that yields integral indices. To make it work with a map-style
    dataset with non-integral indices/keys, a custom sampler must be provided.
File:
d:\anaconda3\envs\pytorch\lib\site-packages\torch\utils\data\dataset.py
Type:
type
Subclasses:
IterableDataset, TensorDataset, ConcatDataset, Subset
```

下面通过一个示例来说明这个类。

我们希望将文件中的图片读取出来并且返回该图片的标签。

代码文件如下：

```
from torch.utils.data import Dataset
from PIL import Image
import os

class MyData(Dataset):      #继承dataset继承类，之后我们再重写其中的方法

    def __init__(self,root_dir,label_dir):
        #我们创建的时候定义了两个变量
        #不同函数中的变量不可以共同使用，这里self的作用相当于将这个变量定义为
        #了全局变量
        self.root_dir = root_dir
        self.label_dir = label_dir
        self.path = os.path.join(self.root_dir,self.label_dir)
        #对两个路径进行拼接，这样不会出错
        self.img_path = os.listdir(self.path)      #获取所有图片的地址

    def __getitem__(self, idx):
        img_name = self.img_path[idx]      #获取单独图片的名称
        img_item_path =
os.path.join(self.root_dir,self.label_dir,img_name)      #获取每
张图片的相对地址
        img = Image.open(img_item_path)      #打开图片
        label = self.label_dir      #获取图片的label
        return img,label

    def __len__(self):
        return len(self.img_path)      #返回总共图片的数量

root_dir = "dataset/val"
ants_label_dir = "ants"
bees_label_dir = "bees"
ants_dataset = MyData(root_dir,ants_label_dir)
bees_dataset = MyData(root_dir,bees_label_dir)

#可以将两个数据集拼接起来
train_dataset = ants_dataset + bees_dataset
```

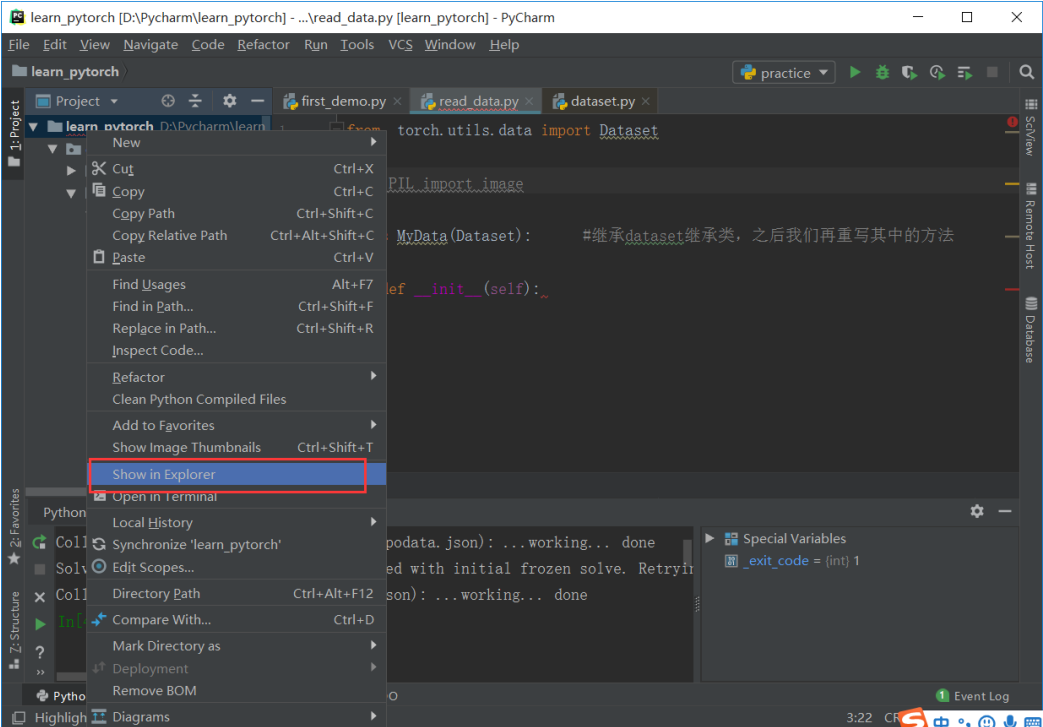
使用控制台运行：

```
Python Console
Python 3.8.6 | packaged by conda-forge | (default, Oct 7 2020, 18:22:52) [MSC v.1916 64 bit (AMD64)] on win32
In[2]: from torch.utils.data import Dataset
In[3]: import os
In[4]: from PIL import Image
In[5]: class MyData(Dataset):
In[6]:     def __init__(self, root_dir, label_dir):
In[7]:         self.root_dir = root_dir
In[8]:         self.label_dir = label_dir
In[9]:         self.path = os.path.join(root_dir, label_dir)
In[10]:         self.img_path = os.listdir(self.path)
In[11]:     def __getitem__(self, idx):
In[12]:         img_name = self.img_path[idx]
In[13]:         img_item_path = os.path.join(self.root_dir, self.label_dir, img_name)
In[14]:         img = Image.open(img_item_path)
In[15]:         img_label = self.label_dir
In[16]:         return img, img_label
In[17]:     def __len__(self):
In[18]:         return len(self.img_path)
In[19]:
```

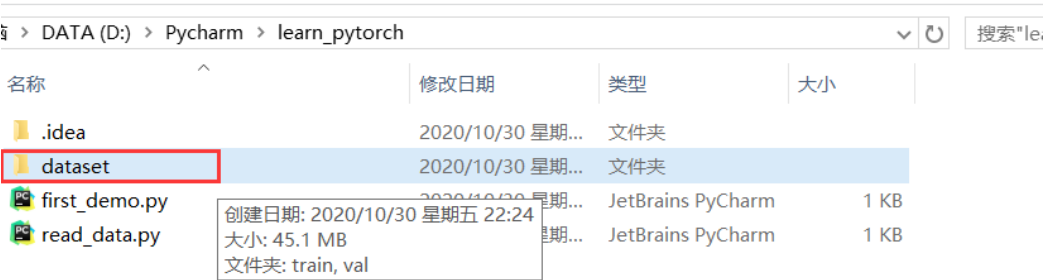
```
Python Console
In[20]:     def __len__(self):
In[21]:         return len(self.img_path)
In[22]:
In[23]: root_dir = "dataset/train"
In[24]: ants_dir = "ants_image"
In[25]: bees_dir = "bees_image"
In[26]: ants_dataset = MyData(root_dir, ants_dir)
In[27]: bees_dataset = MyData(root_dir, bees_dir)
In[28]: train_dataset = ants_dataset + bees_dataset
In[29]: len(train_dataset)
Out[6]: 245
In[30]: len(ants_dataset)
Out[7]: 124
In[31]: len(bees_dataset)
Out[8]: 121
In[32]: img, label = train_dataset[123]
In[33]: img.show()
In[34]: label
Out[11]: 'ants_image'
In[35]: img, label = train_dataset[124]
In[36]: img.show()
In[37]: label
Out[14]: 'bees_image'
In[38]:
```

将数据集复制到pycharm的脚本位置处，文件的调用。

在文件夹点击右键，选择show in explorer，会打开文件夹所在位置，



将其他地方的数据集放置在这里



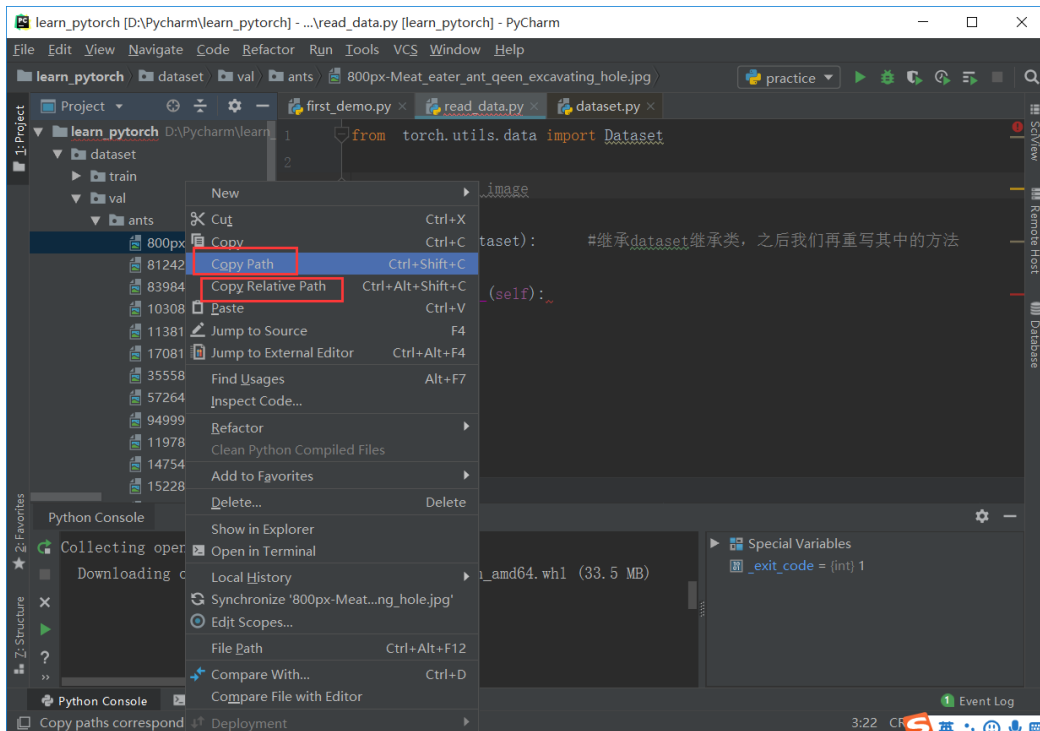
我们选择文件的路径，copy path 得到的结果是绝对路径，copy relative path是当前大文件夹下面的相对路径。

绝对路径

```
D:\Pycharm\learn_pytorch\dataset\val\ants\800pxMeat_eater_ant_queen_excavating_hole.jpg
```

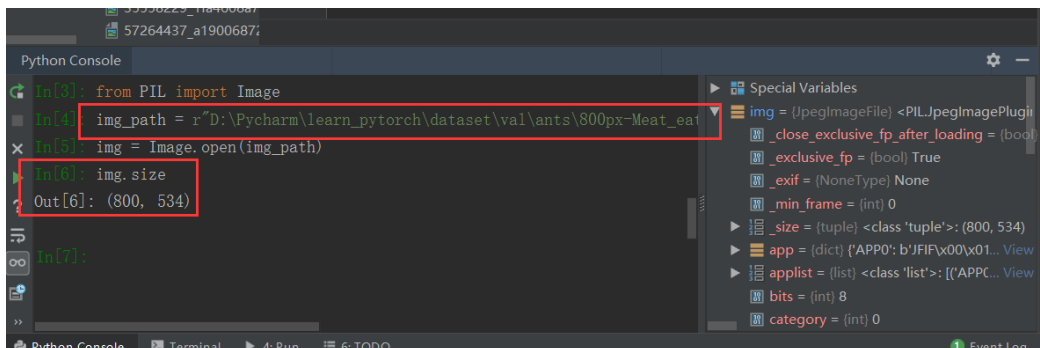
相对路径：

dataset/val/ants/800pxMeat\_eater\_ant\_queen\_excavating\_hole.jpg

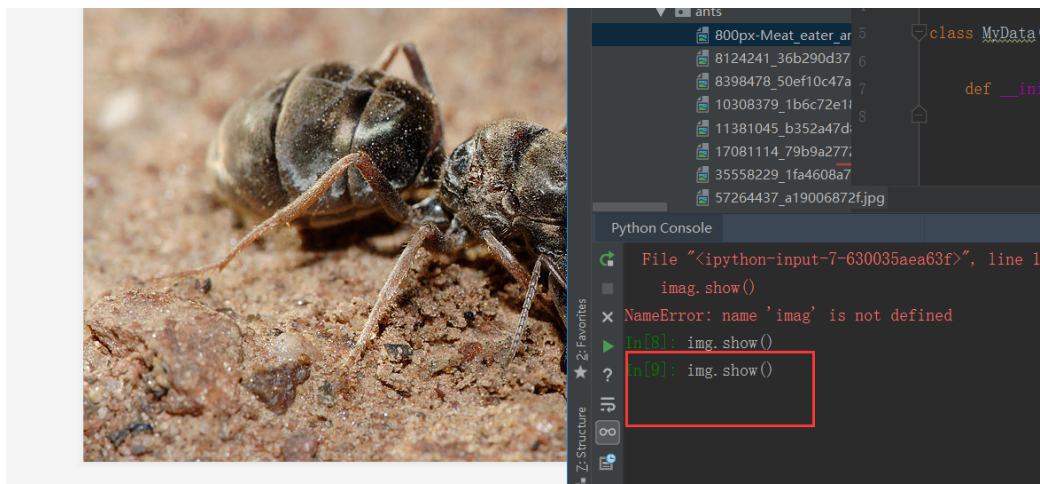


在win中图片地址的写入时，可以选择在地址前面加r，或者将地址中所有的单斜杠转为双斜杠。

在控制台可以看到img的各种属性，以及各种方法，可以很方便调用。



img.show() 可以查看图片



代码部分：

```
from torch.utils.data import Dataset
from PIL import Image
import os

class MyData(Dataset):    #继承dataset继承类，之后我们再重写其中的方法

    def __init__(self, root_dir, label_dir):
        #我们创建的时候定义了两个变量
        #不同函数中的变量不可以共同使用，这里self的作用相当于将这个变量定义为了全局变量
        self.root_dir = root_dir
        self.label_dir = label_dir
        self.path = os.path.join(self.root_dir, self.label_dir)
        #对两个路径进行拼接，这样不会出错
        self.img_path = os.listdir(self.path)    #获取所有图片的地址

    def __getitem__(self, idx):
        img_name = self.img_path[idx]    #获取单独图片的名称
        img_item_path =
os.path.join(self.root_dir, self.label_dir, img_name)    #获取每
张图片的相对地址
        img = Image.open(img_item_path)    #打开图片
        label = self.label_dir    #获取图片的label
        return img, label

    def __len__(self):
        return len(self.img_path)    #返回总共图片的数量

root_dir = "dataset/val"
ants_label_dir = "ants"
bees_label_dir = "bees"
ants_dataset = MyData(root_dir, ants_label_dir)
bees_dataset = MyData(root_dir, bees_label_dir)

#可以将两个数据集拼接起来
train_dataset = ants_dataset + bees_dataset
```

