

第三次实验报告

——命令行环境； Python 入门基础； Python 视觉应用

杨昕昱 23020007142

September 2024

目录

1 命令行环境	3
1.1 结束进程	3
1.2 暂停和后台执行进程	4
1.3 终端多路复用	5
1.4 别名	7
1.5 配置文件	8
1.6 安装虚拟机	9
1.7 远端控制——配置私钥	11
1.8 远端控制——配置 SSH 客户端	12
2 Python 入门基础	13
2.1 Python3 下载	13
2.2 VScode 配置	15
2.3 创建一个 Python 代码文件	16
2.4 第一个程序编写	17
2.5 字符串 (String)	18
2.6 列表	19
2.7 元组	20
2.8 字典	21
3 Python 视觉应用	22
3.1 原始图像	22
3.2 PIL: Python 图像处理类库	24
3.3 Matplotlib	26
3.4 NumPy	27
3.5 SciPy	32

1 命令行环境

1.1 结束进程

当我们输入 Ctrl-C 时，shell 会发送一个 SIGINT 信号到进程。下面这个 Python 程序向您展示了捕获信号 SIGINT 并忽略它的基本操作，它并不会让程序停止。为了停止这个程序，我们需要使用 SIGQUIT 信号，通过输入 Ctrl- 可以发送该信号。

```
sigint.py  x
Ubuntu > home > yxy > sigint.py > ...
1  #!/usr/bin/env python
2  import signal, time
3
4  def handler(signum, time):
5      print("\nI got a SIGINT, but I am not stopping")
6
7  signal.signal(signal.SIGINT, handler)
8  i = 0
9  while True:
10     time.sleep(.1)
11     print("\r{}".format(i), end="")
12     i += 1
```

我们向这个程序发送两次 SIGINT，然后再发送一次 SIGQUIT，程序输出如下：

```
yxy@DESKTOP-6MF70N6:~$ python3 sigint.py
16^C
I got a SIGINT, but I am not stopping
28^C
I got a SIGINT, but I am not stopping
40^Q
```

1.2 暂停和后台执行进程

在终端中，键入 Ctrl-Z 会让 shell 发送 SIGTSTP 信号，SIGTSTP 是 Terminal Stop 的缩写（即 terminal 版本的 SIGSTOP）。我们可以使用 fg 或 bg 命令恢复暂停的工作。它们分别表示在前台继续或在后台继续。jobs 命令会列出当前终端会话中尚未完成的全部任务。可以使用百分号 + 任务编号（jobs 会打印任务编号）来选取该任务。如果要选择最近的一个任务，可以使用 \$! 这一特殊参数。命令中的 & 后缀可以让命令在直接在后台运行。

```
yxy@DESKTOP-6MF70N6:~$ sleep 1000
^Z
[3]+  Stopped                  sleep 1000
yxy@DESKTOP-6MF70N6:~$ nohup sleep 2000 &
[4] 1922
yxy@DESKTOP-6MF70N6:~$ nohup: ignoring input and appending output to 'nohup.out'

yxy@DESKTOP-6MF70N6:~$ jobs
[1]    Running                  sleep 1000 &
[2]    Running                  nohup sleep 2000 &
[3]+  Stopped                  sleep 1000
[4]-  Running                  nohup sleep 2000 &
yxy@DESKTOP-6MF70N6:~$ bg %3
[3]+ sleep 1000 &
yxy@DESKTOP-6MF70N6:~$ jobs
[1]    Running                  sleep 1000 &
[2]    Running                  nohup sleep 2000 &
[3]-  Running                  sleep 1000 &
[4]+  Running                  nohup sleep 2000 &
yxy@DESKTOP-6MF70N6:~$ kill -STOP %3

[3]+  Stopped                  sleep 1000
yxy@DESKTOP-6MF70N6:~$ jobs
[1]    Running                  sleep 1000 &
[2]    Running                  nohup sleep 2000 &
[3]+  Stopped                  sleep 1000
[4]-  Running                  nohup sleep 2000 &
yxy@DESKTOP-6MF70N6:~$ kill -SIGHUP %1
yxy@DESKTOP-6MF70N6:~$ jobs
[1]    Hangup                  sleep 1000
[2]    Running                  nohup sleep 2000 &
[3]+  Stopped                  sleep 1000
[4]-  Running                  nohup sleep 2000 &
yxy@DESKTOP-6MF70N6:~$ kill -SIGHUP %3
yxy@DESKTOP-6MF70N6:~$ jobs
[2]    Running                  nohup sleep 2000 &
[3]-  Hangup                  sleep 1000
[4]+  Running                  nohup sleep 2000 &
```

```
yxy@DESKTOP-6MF70N6:~$ kill -STOP %3
[3]+  Stopped                  sleep 1000
yxy@DESKTOP-6MF70N6:~$ jobs
[1]  Running                   sleep 1000 &
[2]  Running                   nohup sleep 2000 &
[3]+  Stopped                  sleep 1000
[4]-  Running                   nohup sleep 2000 &
yxy@DESKTOP-6MF70N6:~$ kill -SIGHUP %1
yxy@DESKTOP-6MF70N6:~$ jobs
[1]  Hangup                   sleep 1000
[2]  Running                   nohup sleep 2000 &
[3]+  Stopped                  sleep 1000
[4]-  Running                   nohup sleep 2000 &
yxy@DESKTOP-6MF70N6:~$ kill -SIGHUP %3
yxy@DESKTOP-6MF70N6:~$ jobs
[2]  Running                   nohup sleep 2000 &
[3]-  Hangup                   sleep 1000
[4]+  Running                   nohup sleep 2000 &
yxy@DESKTOP-6MF70N6:~$ kill -SIGHUP %2
yxy@DESKTOP-6MF70N6:~$ kill -SIGHUP %4
yxy@DESKTOP-6MF70N6:~$ jobs
[2]-  Running                   nohup sleep 2000 &
[4]+  Running                   nohup sleep 2000 &
yxy@DESKTOP-6MF70N6:~$ kill %4
yxy@DESKTOP-6MF70N6:~$ jobs
[2]-  Running                   nohup sleep 2000 &
[4]+  Terminated               nohup sleep 2000
yxy@DESKTOP-6MF70N6:~$ kill %2
yxy@DESKTOP-6MF70N6:~$ jobs
[2]+  Terminated               nohup sleep 2000
yxy@DESKTOP-6MF70N6:~$ jobs
yxy@DESKTOP-6MF70N6:~$ |
```

1.3 终端多路复用

tmux 这类的终端多路复用器可以允许我们基于面板和标签分割出多个终端窗口，这样可以同时与多个 shell 会话进行交互。tmux 的快捷键，类似于 $< \text{C-b} > \text{x}$ 这样的组合，即需要先按下 Ctrl+b ，松开后再按下 x 。

-tmux 开始一个新的会话

-tmux new -s NAME 以指定名称开始一个新的会话

-tmux ls 列出当前所有会话

-在 tmux 中输入 $< \text{C-b} > \text{d}$ ，将当前会话分离

-tmux a 重新连接最后一个会话。您也可以通过 -t 来指定具体的会话

```
yxy@DESKTOP-6MF70N6:~$ + ^  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
yxy@DESKTOP-6MF70N6:~$ |  
  
yxy@DESKTOP-6MF70N6:~$ tmux  
[exited]
```

- <C-b> c 创建一个新的窗口，使用 <C-d> 关闭
- <C-b> N 跳转到第 N 个窗口，注意每个窗口都是有编号的
- <C-b> p 切换到前一个窗口
- <C-b> n 切换到下一个窗口
- <C-b> , 重命名当前窗口
- <C-b> w 列出当前所有窗口

```
yxy@DESKTOP-6MF70N6:~$ + ^  
(0) /dev/pts/0: 17:12: session 1  
  
/dev/pts/0 (sort: name) —————  
yxy@DESKTOP-6MF70N6:~$ ^C  
yxy@DESKTOP-6MF70N6:~$ ^C  
yxy@DESKTOP-6MF70N6:~$ b^C  
yxy@DESKTOP-6MF70N6:~$ █  
  
[1] 0:[tmux]* "DESKTOP-6MF70N6" 17:12 08-Sep-24  
[1] 0:[tmux]* "DESKTOP-6MF70N6" 17:12 08-Sep-24
```

- <C-b> ” 水平分割
- <C-b> % 垂直分割
- <C-b> < 方向 > 切换到指定方向的面板，< 方向 > 指的是键盘上的方向

键

-<C-b> z 切换当前面板的缩放

-<C-b> [开始往回卷动屏幕。您可以按下空格键来开始选择，回车键复制选中的部分

-<C-b> < 空格 > 在不同的面板排布间切换

```
yxy@DESKTOP-6MF70N6: ~
```

```
(0) - 0: 1 windows
(1) ↳ 0: bash*
(2) - 1: 1 windows (attached)
(3) ↳ + 0: bash*
```

```
To run a command as administrator (user "root"),
use "sudo <command>".
See "man sudo_root" for details.
```

```
yxy@DESKTOP-6MF70N6:~$ |
```

```
0 (sort: index) ━━━━
ws (created Sun Sep 8 17:08:51 2024) (P-6MF70N6:~$ tmux new -s NAME
should be nested with care, unset $TMUX
P-6MF70N6:~$ ^C
P-6MF70N6:~$ )
```

```
[1] 0:bash*
```

```
"DESKTOP-6MF70N6" 17:14 08-Sep-24
```

1.4 别名

shell 的别名相当于一个长命令的缩写，shell 会自动将其替换成原本的命令。

```

yxy@DESKTOP-6MF70N6:~$ alias ll="ls -lh"
yxy@DESKTOP-6MF70N6:~$ ll
total 32K
-rw-r--r-- 1 yxy yxy      6 Aug 30 11:15 hello.txt
-rw-r--r-- 1 yxy yxy      6 Aug 30 11:16 hello2.txt
-rw-r--r-- 1 yxy yxy  141 Sep  6 09:00 marco.sh
-rw-r--r-- 1 yxy yxy   10 Sep  6 09:00 marco_history.log
drwxr-xr-x 2 yxy yxy  4.0K Sep  1 17:07 new
-rw----- 1 yxy yxy      0 Sep  8 15:59 nohup.out
-rw-r--r-- 1 yxy yxy  535 Sep  6 09:40 p.txt
-rw-r--r-- 1 yxy yxy  256 Sep  8 16:31 sigint.py
drwx----- 3 yxy yxy  4.0K Sep  1 15:15 snap
yxy@DESKTOP-6MF70N6:~$ alias gs="git status"
yxy@DESKTOP-6MF70N6:~$ gs
fatal: not a git repository (or any of the parent directories): .git
yxy@DESKTOP-6MF70N6:~$ alias v="vim"
yxy@DESKTOP-6MF70N6:~$ v
yxy@DESKTOP-6MF70N6:~$ alias la="ls -A"
yxy@DESKTOP-6MF70N6:~$ la
.bash_history  .lesshist    .swp      hello.txt          new      snap
.bash_logout    .motd_shown  .vim      hello2.txt        nohup.out
.bashrc         .p.txt.swp   .viminfo  marco.sh        p.txt
.cache          .profile     .vimrc    marco_history.log  sigint.py
yxy@DESKTOP-6MF70N6:~$ alias ll
alias ll='ls -lh'
yxy@DESKTOP-6MF70N6:~$ |

```

1.5 配置文件

首先，在本地创造一个新文件夹，并用 git init 来初始化为 Git 仓库。
然后将本机的配置文件如.vimrc/.bashrc/复制进该目录。

```

yxy@DESKTOP-6MF70N6:~$ mkdir ~/gits
yxy@DESKTOP-6MF70N6:~$ mkdir ~/gits/dotfiles
yxy@DESKTOP-6MF70N6:~$ git init ~/gits/dotfiles
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/yxy/gits/dotfiles/.git/
yxy@DESKTOP-6MF70N6:~$ ls -a ~/gits/dotfiles
.  ..  .git
yxy@DESKTOP-6MF70N6:~$ cp ~/.vimrc ~/gits/dotfiles/
yxy@DESKTOP-6MF70N6:~$ ls -a ~/gits/dotfiles
.  ..  .git  .vimrc
yxy@DESKTOP-6MF70N6:~$ cp ~/.bashrc ~/gits/dotfiles/
yxy@DESKTOP-6MF70N6:~$ cp ~/.tmux.conf ~/gits/dotfiles/
cp: cannot stat '/home/yxy/.tmux.conf': No such file or directory
yxy@DESKTOP-6MF70N6:~$ ls -a ~/gits/dotfiles
.  ..  .bashrc  .git  .vimrc
yxy@DESKTOP-6MF70N6:~$ |

```

再运用第一次课学习的 git 操作，将这个仓库 push 到 Github 上。

```
yxy@DESKTOP-6MF70N6:~/gits/dotfiles
yxy@DESKTOP-6MF70N6:~/gits/dotfiles$ git remote add origin git@github.com:iris-inu/git
.git
yxy@DESKTOP-6MF70N6:~/gits/dotfiles$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .bashrc
    .vimrc

nothing added to commit but untracked files present (use "git add" to track)
yxy@DESKTOP-6MF70N6:~/gits/dotfiles$ git add .
yxy@DESKTOP-6MF70N6:~/gits/dotfiles$ git status
On branch master

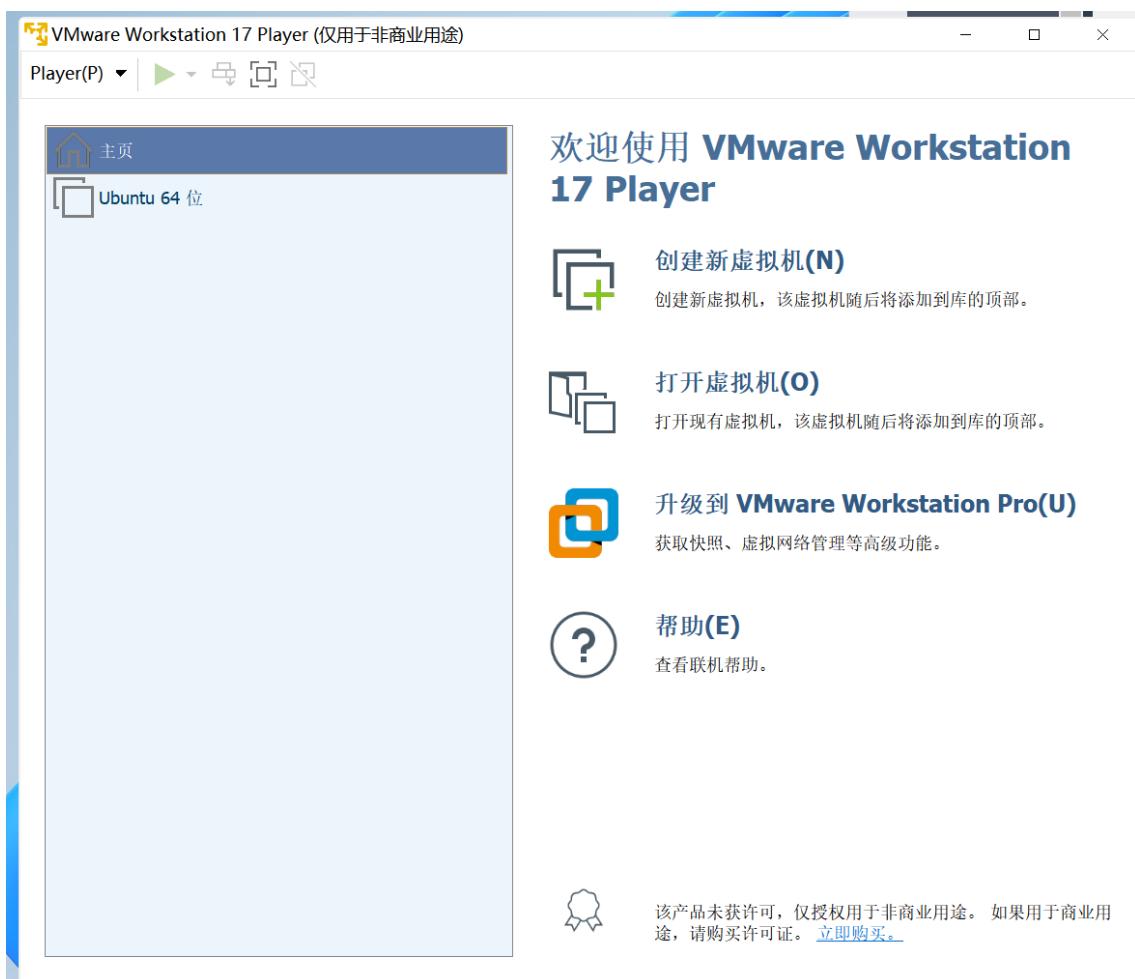
No commits yet

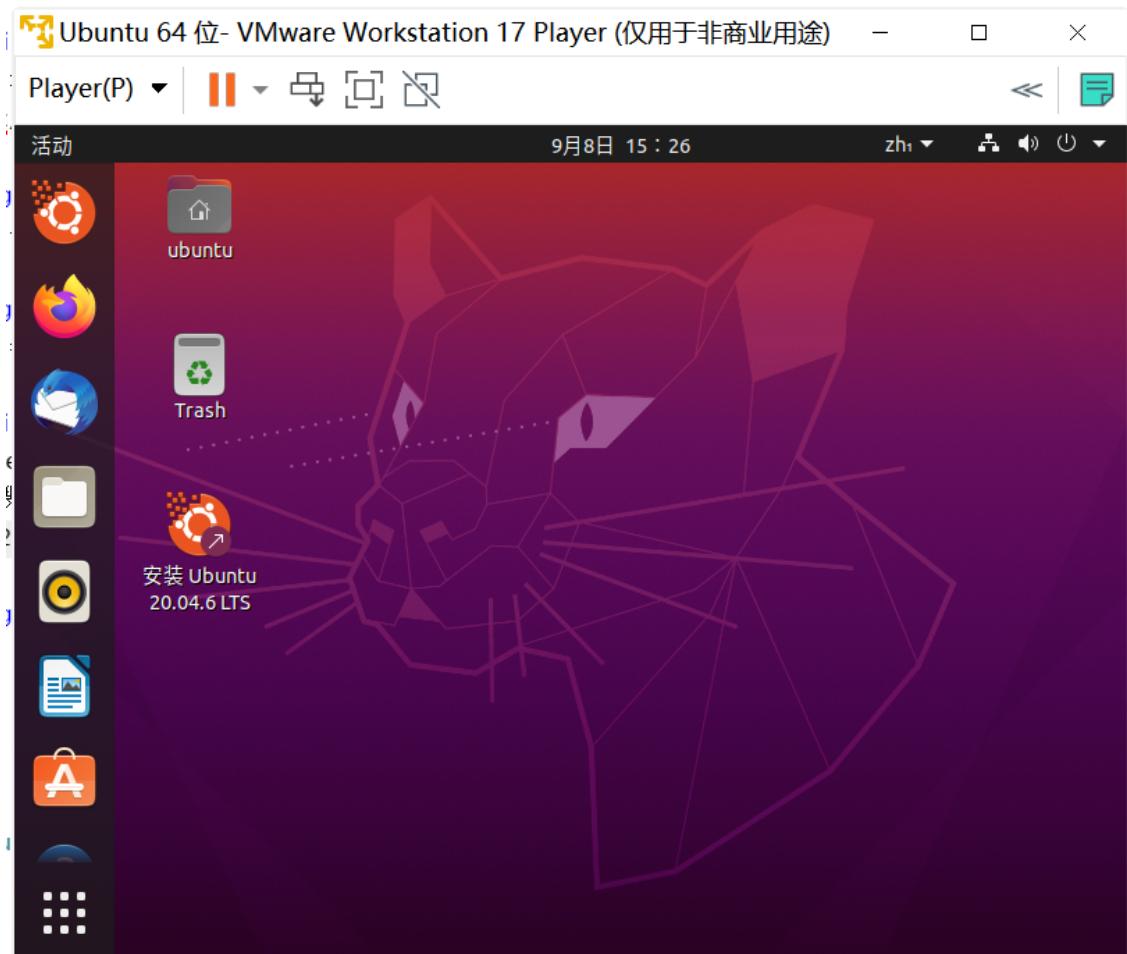
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .bashrc
    new file:   .vimrc
```

在另一台机器，或虚拟机上，将上面的 Github 仓库复制下来。

1.6 安装虚拟机

下载 VMware Workstation Player 17。找到清华镜像：清华大学开源软件镜像站 | Tsinghua Open Source Mirror，搜索框输入 ubuntu，然后找到 ubuntu-releases。找到 20.04/文件下，点击箭头所指的蓝色字体“ubuntu-20.04.1-desktop-amd64.iso”即可下载。下载完毕后可以创建一个新虚拟机





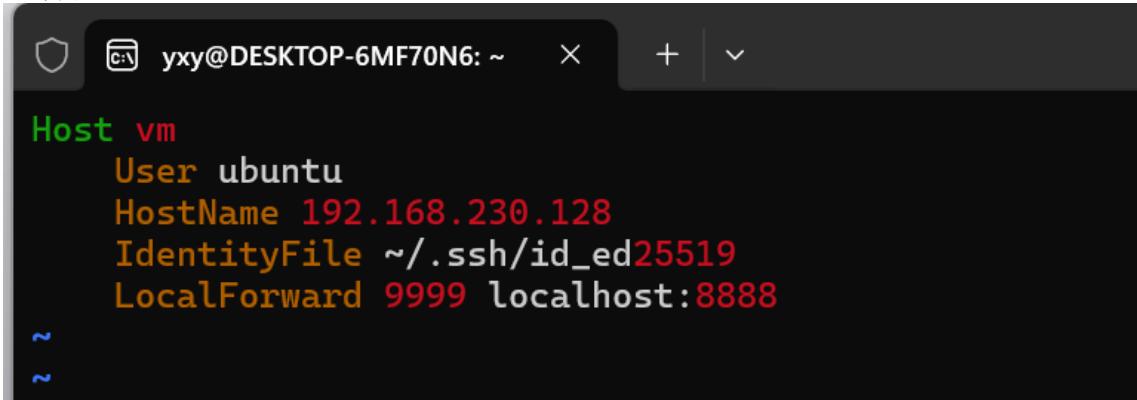
1.7 远端控制——配置私钥

前往 `/ssh/`，查看发现没有存在 SSH 密钥对。然后使用 `ssh-keygen -o -a 100 -t ed25519` 来创建一个私钥。为密钥设置密码后使用 `ssh-agent`。

```
yxy@DESKTOP-6MF70N6:~/gits/dotfiles$ cd ~/.ssh/
yxy@DESKTOP-6MF70N6:~/.ssh$ ls
known_hosts
yxy@DESKTOP-6MF70N6:~/.ssh$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/yxy/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/yxy/.ssh/id_ed25519
Your public key has been saved in /home/yxy/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:mCh2K/WTuY7y2sXgOB6jc8AkTA2/iDDcAI4+BNAVyfQ yxy@DESKTOP-6MF70N6
The key's randomart image is:
+--[ED25519 256]--+
|*+o++o
|=.=.o.
|*+ o E
|*+. . o
|=+o.= o S
|.o.* = o
| .* o B
| .oo* o o
| .oo++.o
+---[SHA256]----+
yxy@DESKTOP-6MF70N6:~/.ssh$ ls
id_ed25519  id_ed25519.pub  known_hosts
yxy@DESKTOP-6MF70N6:~/.ssh$ eval "$(ssh-agent -s)"
Agent pid 86543
yxy@DESKTOP-6MF70N6:~/.ssh$ ssh-add ~/ssh/id_ed25519
Identity added: /home/yxy/.ssh/id_ed25519 (yxy@DESKTOP-6MF70N6)
```

1.8 远端控制——配置 SSH 客户端

编辑 SSH 配置文件，在文件中添加以下内容，写入远端机器实际的用户名和 IP 地址。



```
Host vm
  User ubuntu
  HostName 192.168.230.128
  IdentityFile ~/.ssh/id_ed25519
  LocalForward 9999 localhost:8888
```

2 Python 入门基础

2.1 Python3 下载

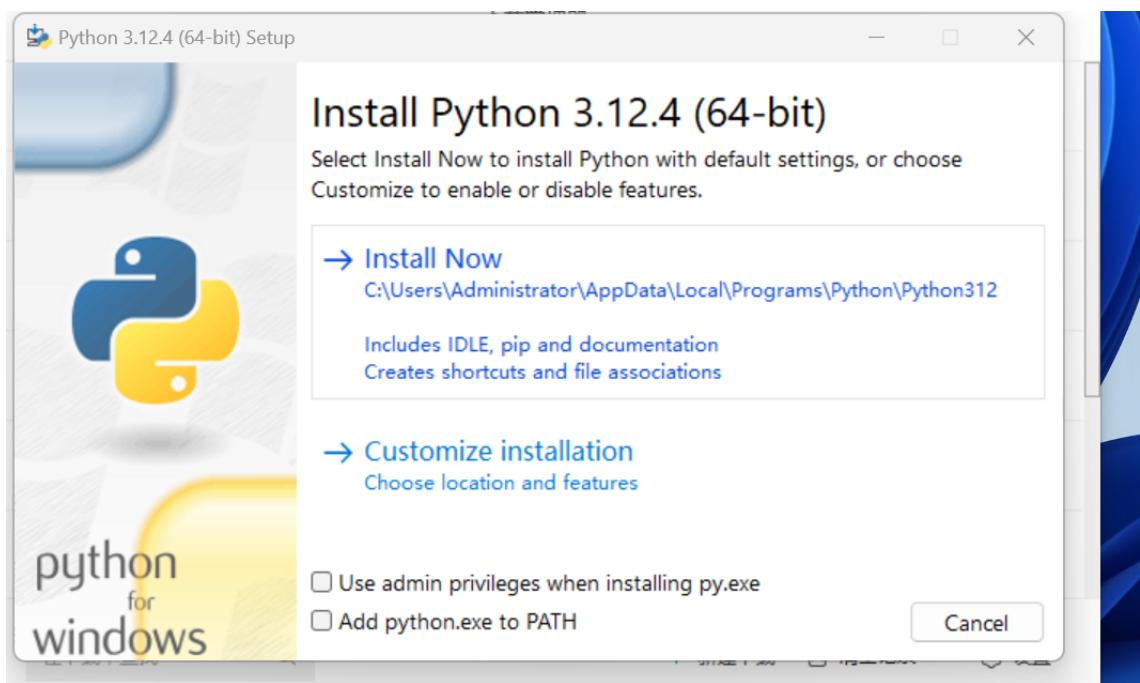
打开 WEB 浏览器访问 <https://www.python.org/downloads/windows/>，安装时勾选 Add Python 3.6 to PATH。按 Win+R 键，输入 cmd 调出命令提示符，输入 python。

Stable Releases

- [Python 3.12.5 - Aug. 6, 2024](#)

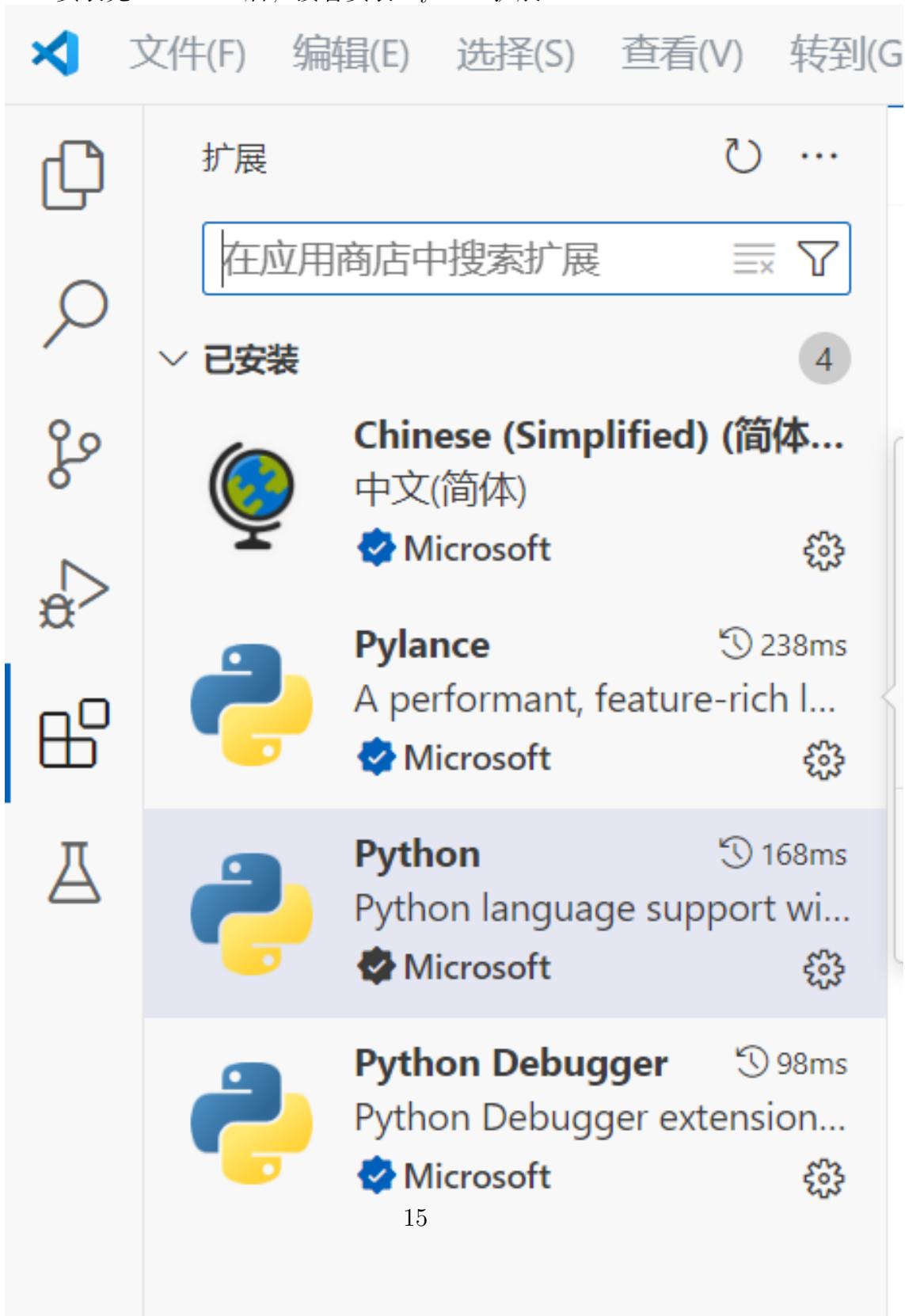
Note that Python 3.12.5 cannot be used on Windows 7 or earlier.

- Download [Windows installer \(64-bit\)](#)
- Download [Windows installer \(32-bit\)](#)
- Download [Windows installer \(ARM64\)](#)
- Download [Windows embeddable package \(64-bit\)](#)
- Download [Windows embeddable package \(32-bit\)](#)
- Download [Windows embeddable package \(ARM64\)](#)
- [Python 3.12.4 - June 6, 2024](#)



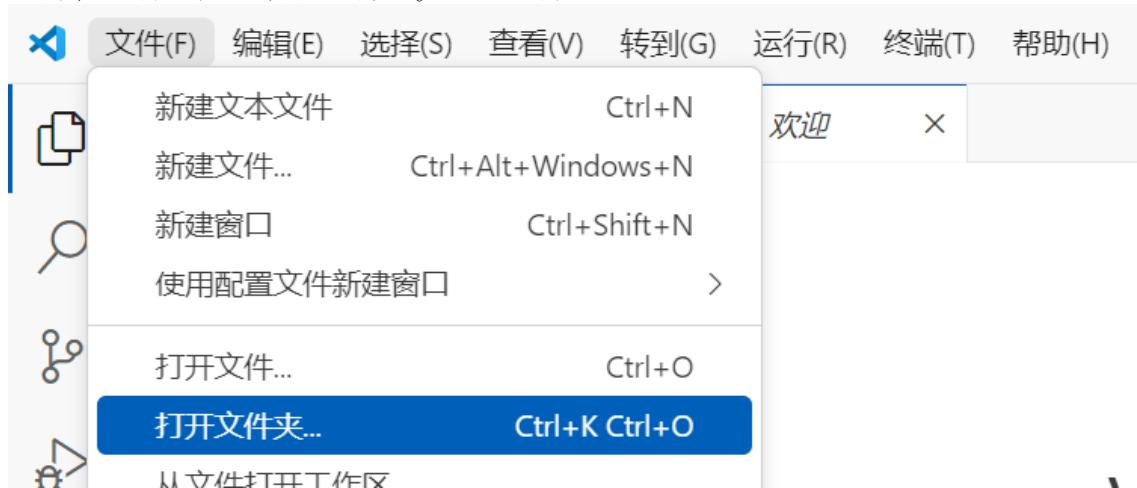
2.2 VScode 配置

安装完 VS Code 后，接着安装 Python 扩展。



2.3 创建一个 Python 代码文件

打开 VScode，然后点击新建文件。也可以打开一个已存在的文件或目录（文件夹），然后我们创建一个 test.py 文件，点击下面新建文件图标，输入文件名 test.py。编辑完后点击右上角绿色图标，即可运行。也可以右击文件，选择“在终端中运行 Python 文件”。





2.4 第一个程序编写

编写第一个 Python 程序，输出“Hello, world!”。

```
D: > 学习与作业 > python学习 > Untitled-1.py
1 print("Hello,world!")
2
```

输出结果如下：

```
PS C:\Users\Administrator> & C:/Users/Administrator/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/学习与作业/python学习/Untitled-1.py
Hello,world!
```

2.5 字符串 (String)

- Python 中单引号' 和双引号" 使用完全相同。
- 使用三引号 (''' 或 """') 可以指定一个多行字符串。
- 转义符 \。
- 反斜杠可以用来转义，使用 r 可以让反斜杠不发生转义。如 r"this is a line with \n" 则 \n 会显示，并不是换行。
- 按字面意义级联字符串，如"this " "is " "string" 会被自动转换为 this is string。
- 字符串可以用 + 运算符连接在一起，用 * 运算符重复。
- Python 中的字符串有两种索引方式，从左往右以 0 开始，从右往左以 -1 开始。
- Python 中的字符串不能改变。
- Python 没有单独的字符类型，一个字符就是长度为 1 的字符串。
- 字符串切片 str[start:end]，其中 start（包含）是切片开始的索引，end（不包含）是切片结束的索引。
- 字符串的切片可以加上步长参数 step，语法格式如下：str[start:end:step]



```
Untitled-1.py  字符  ●

  字符
1  str='123456789'
2
3  print(str)          # 输出字符串
4  print(str[0:-1])    # 输出第一个到倒数第二个的所有字符
5  print(str[0])       # 输出字符串第一个字符
6  print(str[2:5])    # 输出从第三个开始到第六个的字符（不包含）
7  print(str[2:])      # 输出从第三个开始后的所有字符
8  print(str[1:5:2])   # 输出从第二个开始到第五个且每隔一个的字符（步长为2）
9  print(str * 2)      # 输出字符串两次
10 print(str + '你好') # 连接字符串
11
12 print('-----')
13
14 print('hello\nrunoob') # 使用反斜杠(\)+n转义特殊字符
15 print(r'hello\nrunoob') # 在字符串前面添加一个 r, 表示原始字符串, 不会发生转义
```

```
123456789
12345678
1
345
3456789
24
123456789123456789
123456789你好
-----
hello
runoob
hello\nrunoob
```

2.6 列表

与字符串的索引一样，列表索引从 0 开始，第二个索引是 1，依此类推。索引也可以从尾部开始，最后一个元素的索引为 -1，往前一位为 -2，以此类推。

```
列表.py  ×
+
列表.py > ...
1  list = ['red', 'green', 'blue', 'yellow', 'white', 'black']
2  print( list[0] )
3  print( list[1] )
4  print( list[2] )
5  print( list[-1] )
6  print( list[-2] )
7  print( list[-3] )
8
```

输出结果如下：

```
PS D:\学习与作业\python学习> & C:  
red  
green  
blue  
black  
white  
yellow
```

可以对列表的数据项进行修改或更新也可以使用 append() 方法来添加列表项。

```
9  list = ['Google', 'Runoob', 1997, 2000]  
10  
11  print ("第三个元素为 : ", list[2])  
12  list[2] = 2001  
13  print ("更新后的第三个元素为 : ", list[2])  
14  
15  list1 = ['Google', 'Runoob', 'Taobao']  
16  list1.append('Baidu')  
17  print ("更新后的列表 : ", list1)
```

输出结果如下：

```
第三个元素为 : 1997  
更新后的第三个元素为 : 2001  
更新后的列表 : ['Google', 'Runoob', 'Taobao', 'Baidu']
```

2.7 元组

元组可以使用下标索引来访问元组中的值。元组中的元素值是不允许修改的，但可以对元组进行连接组合。

元组.py > ...

```
1  tup1 = ('Google', 'Runoob', 1997, 2000)
2  tup2 = (1, 2, 3, 4, 5, 6, 7 )
3
4  print ("tup1[0]: ", tup1[0])
5  print ("tup2[1:5]: ", tup2[1:5])
6
7  tup3 = tup1 + tup2
8  print (tup3)
```

输出结果如下：

```
PS D:\学习与作业\python学习> & C:/Users/Administrator/AppData/
tup1[0]: Google
tup2[1:5]: (2, 3, 4, 5)
('Google', 'Runoob', 1997, 2000, 1, 2, 3, 4, 5, 6, 7)
```

2.8 字典

把相应的键放入到方括号中，可访问字典的值。向字典添加新内容的方法是增加新的键/值对，修改或删除已有键/值对。

字典.py > ...

```
1  tinydict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}
2
3  print ("tinydict['Name']: ", tinydict['Name'])
4  print ("tinydict['Age']: ", tinydict['Age'])
5
6  tinydict['Age'] = 8           # 更新 Age
7  tinydict['School'] = "菜鸟教程" # 添加信息
8
9  print ("tinydict['Age']: ", tinydict['Age'])
10 print ("tinydict['School']: ", tinydict['School'])
```

输出结果如下：

```
PS D:\学习与作业\python学习> & C:/Users/Administrator  
tinydict['Name']: Runoob  
tinydict['Age']: 7  
tinydict['Age']: 8  
tinydict['School']: 菜鸟教程
```

3 Python 视觉应用

3.1 原始图像

原始图像如下：





3.2 PIL: Python 图像处理类库

在终端中输入“pip install pillow”下载 PIL 库。

```
PS C:\Users\Administrator\Desktop> pip install pillow
Collecting pillow
  Downloading pillow-10.4.0-cp312-cp312-win_amd64.whl.metadata (9.3 kB)
  Downloading pillow-10.4.0-cp312-cp312-win_amd64.whl (2.6 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 8.1 MB/s eta 0:00:00
Installing collected packages: pillow
Successfully installed pillow-10.4.0
```

要读取一幅图像，可以使用：

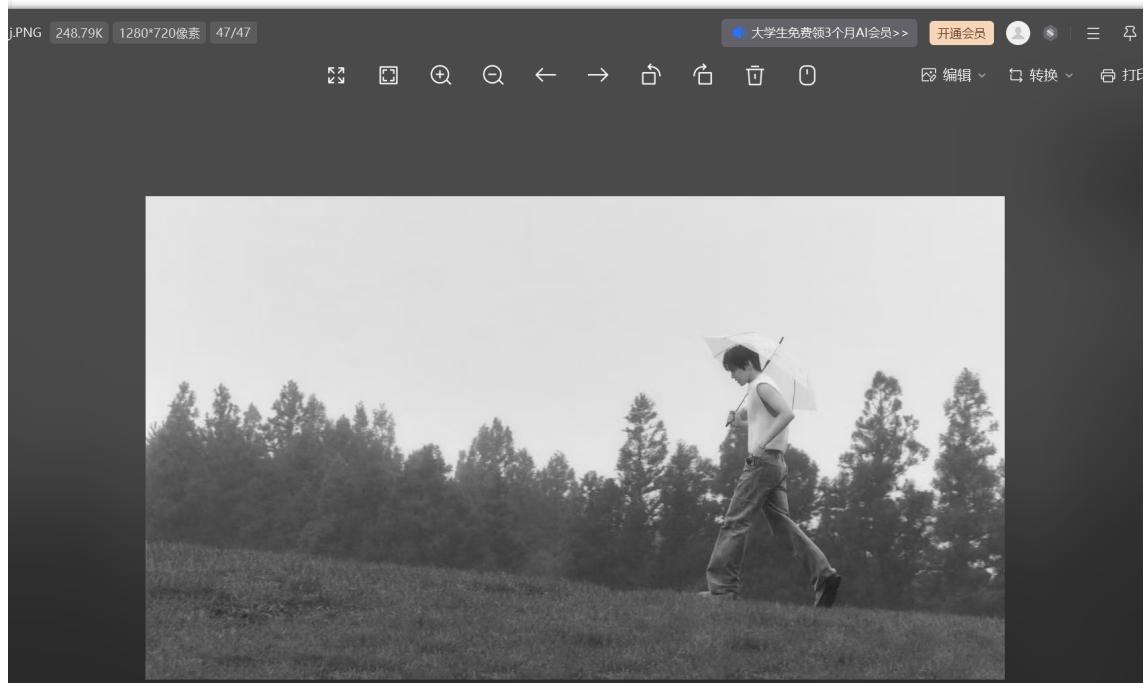
```
rom PIL import Image
pil_im = Image.open('empire.jpg')
```

图像的颜色转换可以使用 convert() 方法来实现。要读取一幅图像，并将其转换成灰度图像，只需要加上 convert('L')。

要展示图片可以用 show() 命令。

输入和输出结果如下：

```
1.py > ...
1  from PIL import Image
2  pil_im = Image.open('a.jpg')
3  pil_im = Image.open('a.jpg').convert('L')
4  pil_im.show ()
```



使用 crop() 方法可以从一幅图像中裁剪指定区域：

```
box = (100,100,400,400)
```

```
region = pil_im.crop(box)
```

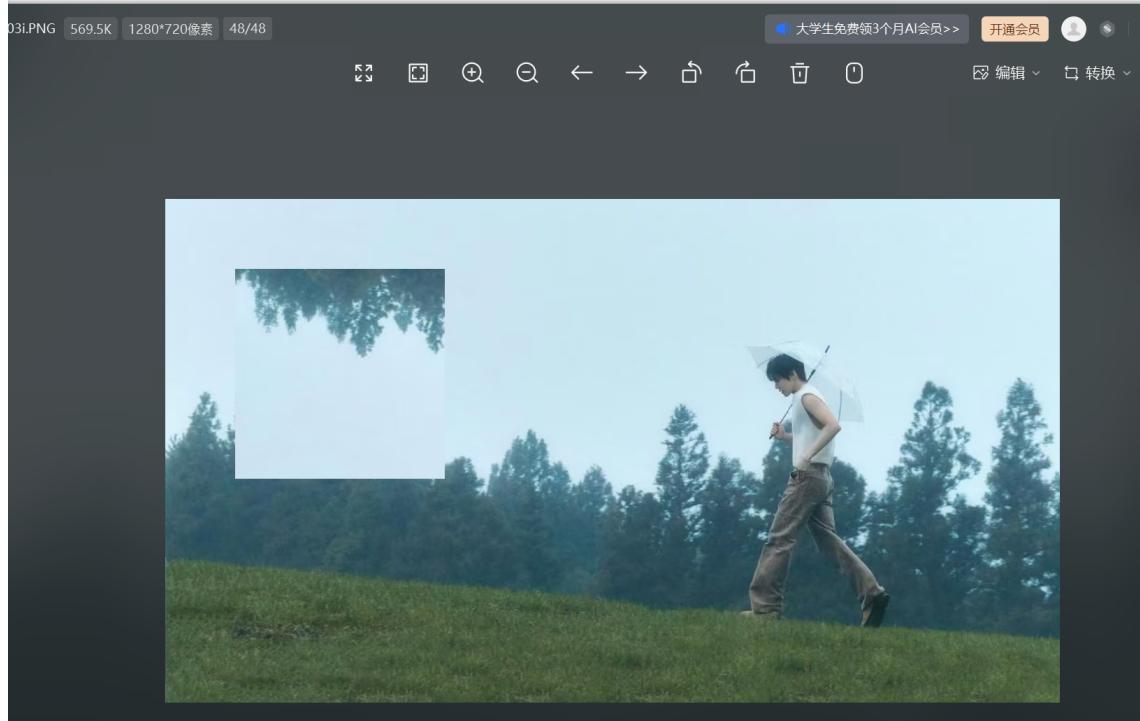
该区域使用四元组来指定。四元组的坐标依次是（左，上，右，下）。PIL 中指定坐标系的左上角坐标为（0，0）。我们可以旋转上面代码中获取的区域，然后使用 paste() 方法将该区域放回去，具体实现如下：

```
region = region.transpose(Image.ROTATE_180)
```

```
pil_im.paste(region,box)
```

输入和输出结果如下：

```
1.py > ...
1  from PIL import Image
2  pil_im = Image.open('a.jpg')
3
4  box = (100,100,400,400)
5  region = pil_im.crop(box)
6
7  region = region.transpose(Image.ROTATE_180)
8  pil_im.paste(region,box)
9  pil_im.show ()
```



3.3 Matplotlib

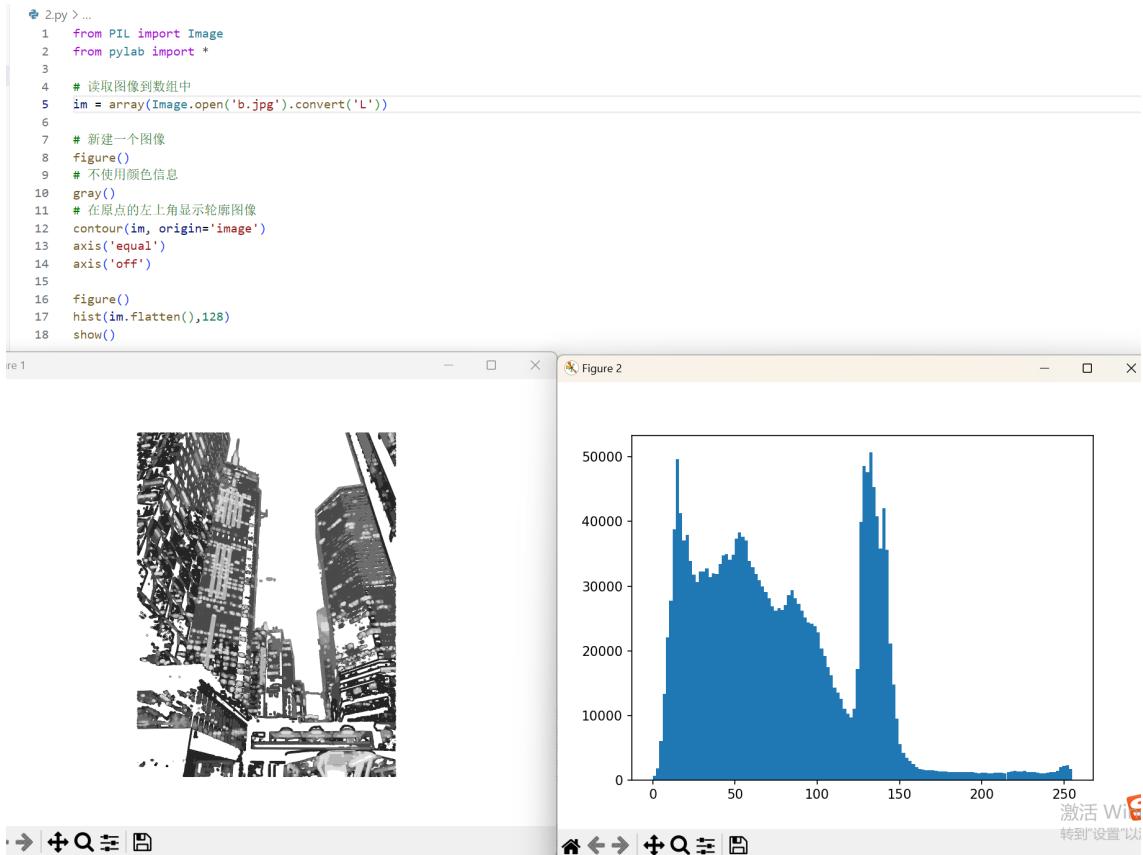
在终端中输入“pip install matplotlib” 下载 Matplotlib 库。

```
PS C:\Users\Administrator\Desktop> pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.9.2-cp312-cp312-win_amd64.whl.metadata (11 kB)
Collecting contourpy==1.3.0 (from matplotlib)
  Downloading contourpy-1.3.0-cp312-cp312-win_amd64.whl.metadata (5.4 kB)
Collecting cycler==0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools==4.22.0 (from matplotlib)
  Downloading fonttools-4.53.1-cp312-cp312-win_amd64.whl.metadata (165 kB)
Collecting kiwisolver==1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.7-cp312-cp312-win_amd64.whl.metadata (6.4 kB)
Collecting numpy==1.23.4 (from matplotlib)
  Downloading numpy-2.1.1-cp312-cp312-win_amd64.whl.metadata (59 kB)
Collecting packaging==24.1 (from matplotlib)
  Downloading packaging-24.1-py3-none-any.whl.metadata (3.2 kB)
Requirement already satisfied: pillow==8 in c:\users\administrator\appdata\local\programs\python\python312\lib\site-packages (from matplotlib) (10.4.0)
Collecting pyparsing==3.1.1 (from matplotlib)
  Downloading pyparsing-3.1.4-py3-none-any.whl.metadata (5.1 kB)
Collecting python-dateutil==2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting six==1.5 (from python-dateutil==2.7>matplotlib)
  Downloading six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
  Downloading matplotlib-3.9.2-cp312-cp312-win_amd64.whl (7.8 MB)
  100%|██████████| 7.8MB 15.1 MB/s eta 0:00:00
  Downloading contourpy-1.3.0-cp312-cp312-win_amd64.whl (210 kB)
  Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
  Downloading fonttools-4.53.1-cp312-cp312-win_amd64.whl (2.2 MB)
  2.3/2.3 MB 17.6 MB/s eta 0:00:00
  Downloading kiwisolver-1.4.7-cp312-cp312-win_amd64.whl (55 kB)
  Downloading numpy-2.1.1-cp312-cp312-win_amd64.whl (12.6 MB)
  12.6/12.6 MB 4.8 MB/s eta 0:00:00
  Downloading packaging-24.1-py3-none-any.whl (53 kB)
  Downloading pyparsing-3.1.4-py3-none-any.whl (104 kB)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
  Downloading six-1.16.0-py2.py3-none-any.whl (1.8 kB)
Installing collected packages: six, pyparsing, packaging, numpy, kiwisolver, fonttools, cycler, python-dateutil, contourpy, matplotlib
Successfully installed contourpy-1.3.0 cycler-0.12.1 fonttools-4.53.1 kiwisolver-1.4.7 matplotlib-3.9.2 numpy-2.1.1 packaging-24.1 pyparsing-3.1.4 python-dateutil-2.9.0.post0 six-1.16.0
```

绘制图像的轮廓（或者其他二维函数的等轮廓线）在工作中非常有用。因为绘制轮廓需要对每个坐标 [x, y] 的像素值施加同一个阈值，所以首先需要将图像灰度化：

```
from PIL import Image
from pylab import *
# 读取图像到数组中
im = array(Image.open('empire.jpg').convert('L'))
# 新建一个图像
figure()
# 不使用颜色信息
gray()
# 在原点的左上角显示轮廓图像
contour(im, origin='image')
axis('equal')
axis('off')
figure()
hist(im.flatten(),128)show()
```

输入和输出结果如下：



3.4 NumPy

在终端中输入“`pip install numpy`”下载 numpy 库。

将图像读入 NumPy 数组对象后，我们可以对它们执行任意数学操作。一个简单的例子就是图像的灰度变换。考虑任意函数 f ，它将 $0\dots 255$ 区间（或者 $0\dots 1$ 区间）映射到自身（意思是说，输出区间的范围和输入区间的范围相同）。下面是关于灰度变换的一些例子：

```
from PIL import Image
from numpy import *
im = array(Image.open('empire.jpg').convert('L'))
im2 = 255 - im # 对图像进行反相处理
im3 = (100.0/255) * im + 100 # 将图像像素值变换到 100...200 区间
im4 = 255.0 * (im/255.0)**2 # 对图像像素值求平方后得到的图像
```

第一个例子将灰度图像进行反相处理；第二个例子将图像的像素值变换到 100...200 区间；第三个例子对图像使用二次函数变换，使较暗的像素值变得更大。

输入和输出结果如下：

```
3.py > ...
1  from PIL import Image
2  import numpy as np
3
4  # 打开并转换图像为灰度
5  im = np.array(Image.open('b.jpg').convert('L'), dtype=np.uint8)
6
7  # 对图像进行反相处理
8  im2 = 255 - im
9  im2_image = Image.fromarray(im2) # 将 NumPy 数组转换回 PIL Image 对象
10 im2_image.show() # 显示反相后的图片
11
12 # 注意：对于 im3 和 im4，你需要确保像素值在 0-255 范围内
13 # 因为直接计算可能会产生超出这个范围的值
14 im3 = (100.0/255) * im + 100
15 im3_clipped = np.clip(im3, 0, 255).astype(np.uint8) # 裁剪到 0-255 范围并转换数据类型
16 im3_image = Image.fromarray(im3_clipped)
17 im3_image.show() # 显示调整后的图片
18
19 im4 = 255.0 * (im / 255.0) ** 2
20 im4_clipped = np.clip(im4, 0, 255).astype(np.uint8) # 裁剪到 0-255 范围并转换数据类型
21 im4_image = Image.fromarray(im4_clipped)
22 im4_image.show() # 显示平方后的图片
```







3.5 SciPy

在终端中输入“pip install SciPy” 下载 numpy 库。

```
PS C:\Users\Administrator\Desktop> pip install scipy
Collecting scipy
  Downloading scipy-1.14.1-cp312-cp312-win_amd64.whl.metadata (60 kB)
Requirement already satisfied: numpy<2.3,>=1.23.5 in c:\users\administrator\appdata\local\programs\python\python312\lib\site-packages (from scipy) (2.1.1)
  Downloading scipy-1.14.1-cp312-cp312-win_amd64.whl (44.5 MB)
    44.5/44.5 MB 5.2 MB/s eta 0:00:00
Installing collected packages: scipy
Successfully installed scipy-1.14.1
```

SciPy 有用来做滤波操作的 `scipy.ndimage.filters` 模块。该模块使用快速一维分离的方式来计算卷积。你可以像下面这样来使用它：

```
from PIL import Image
from numpy import *
from scipy.ndimage import filters
im = array(Image.open('empire.jpg').convert('L'))
im2 = filters.gaussian_filter(im,5)
```

上面 `guassian_filter()` 函数的最后一个参数表示标准差。

如果打算模糊一幅彩色图像，只需简单地对每一个颜色通道进行高斯模糊：

```
im = array(Image.open('empire.jpg'))
im2 = zeros(im.shape)
for i in range(3):
    im2[:, :, i] = filters.gaussian_filter(im[:, :, i], 5)
im2 = uint8(im2)
```

输入和输出结果如下：

```
4.py > ...
1  from PIL import Image
2  import numpy as np
3  from scipy.ndimage import filters
4
5  # 加载并转换为灰度图像
6  im_gray = np.array(Image.open('b.jpg').convert('L'))
7  im2_gray = filters.gaussian_filter(im_gray, 5)
8  # 将NumPy数组转换为PIL Image对象以显示
9  Image.fromarray(im2_gray).show()
10
11 # 加载彩色图像
12 im_color = np.array(Image.open('b.jpg'))
13 im2_color = np.zeros(im_color.shape, dtype=np.float64) # 使用float64以避免在计算过程中溢出
14 for i in range(3): # 遍历RGB三个通道
15     im2_color[:, :, i] = filters.gaussian_filter(im_color[:, :, i], 5)
16 # 将处理后的图像数据转换回uint8类型（因为PIL Image需要这种类型）
17 im2_color = np.clip(im2_color, 0, 255).astype(np.uint8)
18 # 显示彩色图像
19 Image.fromarray(im2_color).show()
```



