

# 第三次实验报告

宫怡

学号：24020007033

2025.9.14

## 目录

<b>1 命令行环境</b>	<b>3</b>
1.1 任务控制 . . . . .	3
1.1.1 结束进程 . . . . .	3
1.1.2 wait 命令 . . . . .	3
1.2 别名 . . . . .	4
1.2.1 创建别名 . . . . .	4
1.2.2 为多条命令创建别名 . . . . .	4
1.3 配置文件 . . . . .	4
1.3.1 为配置文件新建文件夹 . . . . .	4
1.3.2 添加配置文件 . . . . .	5
1.3.3 建立进行快速安装配置的方法 . . . . .	5
1.4 远端设备 . . . . .	6
1.4.1 SSH 密钥对 . . . . .	6
1.4.2 虚拟机配置 ssh 远程连接 . . . . .	6
<b>2 Python 入门基础</b>	<b>7</b>
2.1 字符串练习 . . . . .	7
2.2 列表练习 . . . . .	7
2.3 元组练习 . . . . .	8
2.4 字典练习 . . . . .	8
2.5 时间练习 . . . . .	8
2.6 日期练习 . . . . .	9
2.7 文件 I/O 练习 . . . . .	9
<b>3 Python 视觉应用</b>	<b>9</b>
3.1 Pillow 库 . . . . .	9
3.2 Numpy 库 . . . . .	10
3.3 Scipy 库 . . . . .	10
3.4 OpenCV 库 . . . . .	11

<b>4 心得体会</b>	<b>12</b>
4.1 命令行环境的学习体会 . . . . .	12
4.2 Python 入门基础的学习体会 . . . . .	12
4.3 Python 在视觉应用中的初步探索 . . . . .	13

**实验内容:** 1. 命令行环境

2.python 入门基础

3.python 视觉应用

# 1 命令行环境

## 1.1 任务控制

### 1.1.1 结束进程

sleep 10000 启动休眠任务， Ctrl+Z 挂起前台正在运行的任务， bg 挂起的任务放到后台继续运行， pgrep -af sleep 查找进程, pkill -f sleep 结束进程

```
gy@gy-virtual-machine:~$ sleep 10000
^Z
[1]+  Stopped                  sleep 10000
gy@gy-virtual-machine:~$ bg
[1]+ sleep 10000 &
gy@gy-virtual-machine:~$ jobs
[1]+  Running                  sleep 10000 &
gy@gy-virtual-machine:~$ pgrep -af sleep
11681 sleep 10000
gy@gy-virtual-machine:~$ pkill -f sleep
[1]+  Terminated                sleep 10000
gy@gy-virtual-machine:~$ pgrep -af sleep
```

### 1.1.2 wait 命令

直接输入 wait, 等待全部进程结束，等到 sleep 结束后再执行 ls。

```
gy@gy-virtual-machine:~$ sleep 60 &
[1] 15682
gy@gy-virtual-machine:~$ wait
[1]+  Done                    sleep 60
gy@gy-virtual-machine:~$ ls
Desktop  Downloads  last-modified.txt  Pictures  snap  Templates  test2.html  Videos  vimrc.1
Documents  html_files.zip  Music  Public  subdir  test1.html  test_project  vimrc
gy@gy-virtual-machine:~$
```

编写 pidwait 函数，可以不用管是否是当前 shell 的子进程，等待任意 PID

```
pidwait(){
    local pid=$1
    if[ -z "$pid" ]; then
        echo "pidwait <PID>"
        return 1
    fi
    while kill -0 "$pid" 2>/dev/null; do
        sleep 1
    done
}
```

```
gy@gy-virtual-machine:~$ nano ~/.bashrc
gy@gy-virtual-machine:~$ source ~/.bashrc
gy@gy-virtual-machine:~$ sleep 30 &
[1] 36960
gy@gy-virtual-machine:~$ pidwait <36960>
bash: syntax error near unexpected token `36960'
gy@gy-virtual-machine:~$ pidwait 36960
[1]+  Done                  sleep 30
gy@gy-virtual-machine:~$ ls
Desktop  Downloads  last-modified.txt  Pictures  snap  Templates  test2.html  Videos  vimrc.i
Documents  html_files.zip  Music          Public    subdir  test1.html  test_project  vimrc
gy@gy-virtual-machine:~$
```

## 1.2 别名

### 1.2.1 创建别名

在`~/.bashrc`的文件中添加“alias dc='cd'”，然后使修改生效，使用 dc 定位到 tmp 文件夹中

```
gy@gy-virtual-machine:~$ nano ~/.bashrc
gy@gy-virtual-machine:~$ source ~/.bashrc
gy@gy-virtual-machine:~$ dc /tmp
gy@gy-virtual-machine:/tmp$
```

### 1.2.2 为多条命令创建别名

执行 `|history | awk '1 = """;print substr(0,2)' | sort | uniq -c | sort -n | tail -n 10`

获取最常用的 10 条命令，在 bashrc 文件中为其更换别名，并使修改生效

```
gy@gy-virtual-machine:~$ history | awk '{$1=""};print substr($0,2)' | sort | uniq -c | sort -n | tail -n 10
3 git clone https://github.com/ctrlpvim/ctrlpvim
3 nano marco.sh
3 nano target.sh
3 pgrep -af sleep
3 ./runner.sh
3 sudo apt install git
4 source marco.sh
5 ./semester
7 python3 fizzbuzz.py 20
8 vim fizzbuzz.py
```

## 1.3 配置文件

### 1.3.1 为配置文件新建文件夹

新建目录，初始化 git 仓库，把配置文件复制放到新的.dotfiles 文件夹中，建立符号链接，将文件夹.dotfiles 提交到 git

```
gy@gy-virtual-machine:~$ mkdir ~/.dotfiles
gy@gy-virtual-machine:~$ cd ~/.dotfiles
gy@gy-virtual-machine:~/dotfiles$ git init
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/gy/.dotfiles/.git/
gy@gy-virtual-machine:~/dotfiles$ cp ~/bashrc ~/.dotfiles/bashrc
gy@gy-virtual-machine:~/dotfiles$ cp ~/vimrc ~/.dotfiles/vimrc
gy@gy-virtual-machine:~/dotfiles$ ln -s ~/.dotfiles/bashrc ~/bashrc
ln: failed to create symbolic link '/home/gy/.bashrc': File exists
gy@gy-virtual-machine:~/dotfiles$ mv ~/bashrc ~/bashrc.backup
gy@gy-virtual-machine:~/dotfiles$ ln -s ~/.dotfiles/bashrc ~/bashrc
gy@gy-virtual-machine:~/dotfiles$ ln -s ~/.dotfiles/vimrc ~/vimrc
ln: failed to create symbolic link '/home/gy/.vimrc': File exists
gy@gy-virtual-machine:~/dotfiles$ mv ~/vimrc ~/vimrc.backup
gy@gy-virtual-machine:~/dotfiles$ ln -s ~/.dotfiles/vimrc ~/vimrc
gy@gy-virtual-machine:~/dotfiles$ cd ../dotfiles
gy@gy-virtual-machine:~/dotfiles$ git add .
gy@gy-virtual-machine:~/dotfiles$ git commit -m "add bashrc & vimrc"
[master (root-commit) 46c965f] add bashrc & vimrc
 2 files changed, 208 insertions(+)
 create mode 100644 bashrc
 create mode 100644 vimrc
```

### 1.3.2 添加配置文件

在 dotfiles 的目录里新建配置文件，添加自定义配置，PS1='\\$@\\$h:\\$w\\$'链接到 ~/.bashrc，让配置生效，发现提示符改变。

```
gy@gy-virtual-machine:~$ ls -l ~/.bashrc
lrwxrwxrwx 1 gy gy 25 9月 16 14:22 /home/gy/.bashrc -> /home/gy/.dotfiles/bashrc
gy@gy-virtual-machine:~$ source ~/.bashrc
gy@gy-virtual-machine:~$ source ~/.bashrc
gy@gy-virtual-machine:~$ 
gy@gy-virtual-machine:~/dotfiles$ nano ~/.dotfiles/bashrc
gy@gy-virtual-machine:~/dotfiles$ ls -l ~/.dotfiles/bashrc
-rw-r--r-- 1 gy gy 3928 9月 16 14:03 /home/gy/.dotfiles/bashrc
gy@gy-virtual-machine:~/dotfiles$ nano ~/.dotfiles/bashrc
gy@gy-virtual-machine:~/dotfiles$ mv ~/bashrc ~/bashrc.backup
gy@gy-virtual-machine:~/dotfiles$ ln -s ~/.dotfiles/bashrc ~/bashrc
gy@gy-virtual-machine:~/dotfiles$ source ~/.bashrc
gy@gy-virtual-machine:~/dotfiles$ nano ~/.dotfiles/bashrc
```

### 1.3.3 建立进行快速安装配置的方法

将所有配置文件移至 /dotfiles 目录，写 install.sh 脚本，在新设备上执行 git clone 拷贝 /dotfiles，执行 install.sh 脚本

```
gy@gy-virtual-machine:~/dotfiles$ mv ~/bashrc ~/vimrc ~/gitconfig ~/dotfiles
gy@gy-virtual-machine:~/dotfiles$ cd ~/dotfiles
gy@gy-virtual-machine:~/dotfiles$ nano install.sh
gy@gy-virtual-machine:~/dotfiles$ chmod +x install.sh
```

```
GNU nano 6.2                               install.sh
#!/bin/bash

DOTFILES=~/dotfiles

FILES=(.bashrc .vimrc .gitconfig)

for file in "${FILES[@]}"; do
    if [ -f "$HOME/$file" ] || [ -L "$HOME/$file" ]; then
        mv "$HOME/$file" "$HOME/$file.backup"
    fi
    ln -s "$DOTFILES/$file" "$HOME/$file"
    echo "Linked $file"
done
echo "finished"
```

## 1.4 远端设备

### 1.4.1 SSH 密钥对

.ssh 文件夹不存在，新建目录，生成 SSH 密钥，启动 ssh-agent 并添加密钥，查看密钥

```
gy@gy-virtual-machine:~$ cd ~/.ssh
bash: cd: /home/gy/.ssh: No such file or directory
gy@gy-virtual-machine:~$ mkdir -p ~/.ssh
gy@gy-virtual-machine:~$ chmod 700 ~/.ssh
gy@gy-virtual-machine:~$ ssh-keygen -o -a 100 -t ed25519 -f ~/.ssh/id_ed25519
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/gy/.ssh/id_ed25519
Your public key has been saved in /home/gy/.ssh/id_ed25519.pub
The key's randomart image is:
+--[ED25519 256]--+
|          .B00 .|
|          ..+++=..|
|          .+...+...+|
|o + o . + .. |
| .E.o . o   o |
| . . o . . . |
| .o . o . . .|
| . . . . . . .|
+----[SHA256]----+
gy@gy-virtual-machine:~$ eval "$(ssh-agent -s)"
gy@gy-virtual-machine:~$ ssh-add ~/.ssh/id_ed25519
Enter passphrase for /home/gy/.ssh/id_ed25519:
Identity added: /home/gy/.ssh/id_ed25519 (gy@gy-virtual-machine)
gy@gy-virtual-machine:~$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAACNzaC1ZbDQTE5AAAImgr4AL0LkmM28Iq3rSx2ljl242rLPUh2mfGRh0Qfqc gy@gy-virtual-machine
gy@gy-virtual-machine:~$
```

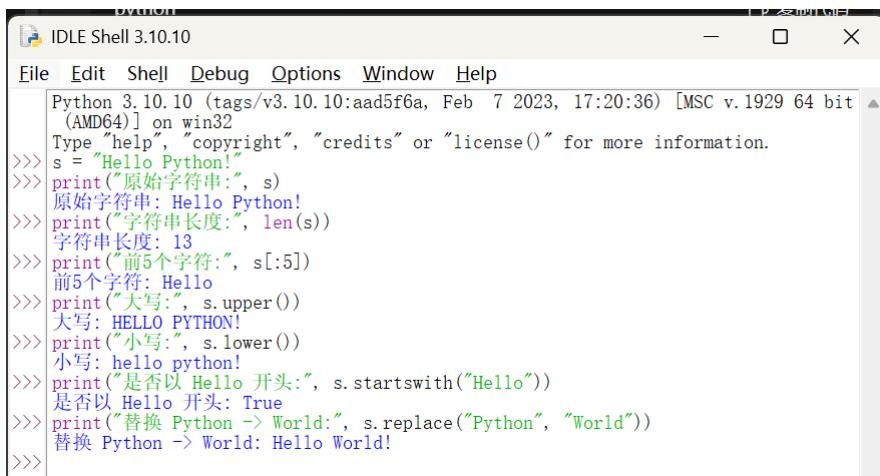
### 1.4.2 虚拟机配置 ssh 远程连接

配置虚拟机的网络连接方式是 NAT，安装`openssh_server`，安装 xshell，新建对话连接虚拟机。

```
gy@gy-virtual-machine:~$ /etc/init.d/iptables start
bash: /etc/init.d/iptables: No such file or directory
gy@gy-virtual-machine:~$ sudo apt install openssh-server
[sudo] password for gy:
Reading package lists... Done
gy@gy-virtual-machine:~$ sudo service ssh restart
gy@gy-virtual-machine:~$ sudo vi/etc/ssh/sshd_config
sudo: vi/etc/ssh/sshd_config: command not found
gy@gy-virtual-machine:~$ sudo vi /etc/ssh/sshd_config
```

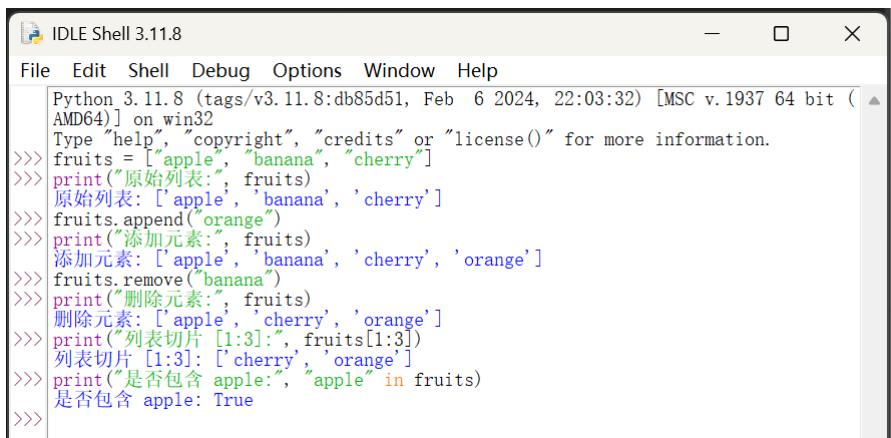
## 2 Python 入门基础

### 2.1 字符串练习



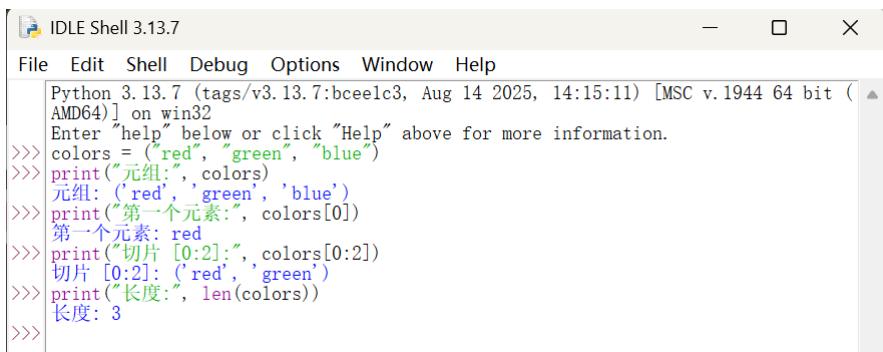
```
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb  7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> s = "Hello Python!"
>>> print("原始字符串:", s)
原始字符串: Hello Python!
>>> print("字符串长度:", len(s))
字符串长度: 13
>>> print("前5个字符:", s[:5])
前5个字符: Hello
>>> print("大写:", s.upper())
大写: HELLO PYTHON!
>>> print("小写:", s.lower())
小写: hello python!
>>> print("是否以 Hello 开头:", s.startswith("Hello"))
是否以 Hello 开头: True
>>> print("替换 Python -> World:", s.replace("Python", "World"))
替换 Python -> World: Hello World!
```

### 2.2 列表练习



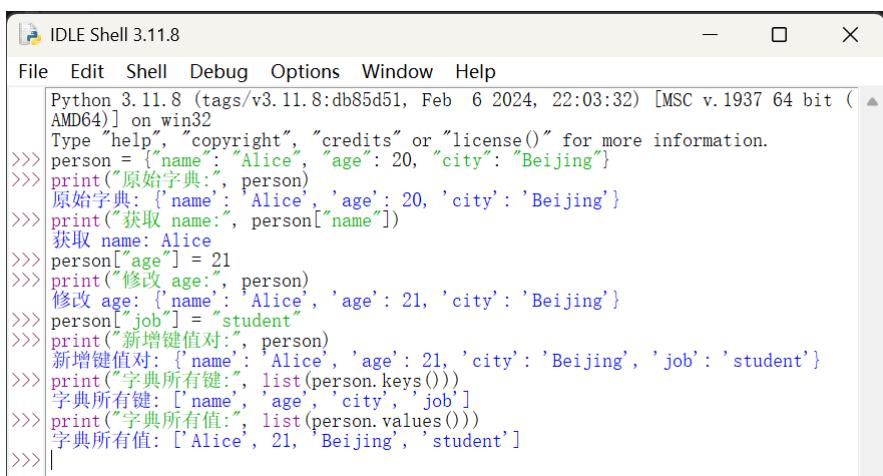
```
Python 3.11.8 (tags/v3.11.8:db85d51, Feb  6 2024, 22:03:32) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> fruits = ["apple", "banana", "cherry"]
>>> print("原始列表:", fruits)
原始列表: ['apple', 'banana', 'cherry']
>>> fruits.append("orange")
>>> print("添加元素:", fruits)
添加元素: ['apple', 'banana', 'cherry', 'orange']
>>> fruits.remove("banana")
>>> print("删除元素:", fruits)
删除元素: ['apple', 'cherry', 'orange']
>>> print("列表切片 [1:3]:", fruits[1:3])
列表切片 [1:3]: ['cherry', 'orange']
>>> print("是否包含 apple:", "apple" in fruits)
是否包含 apple: True
```

## 2.3 元组练习



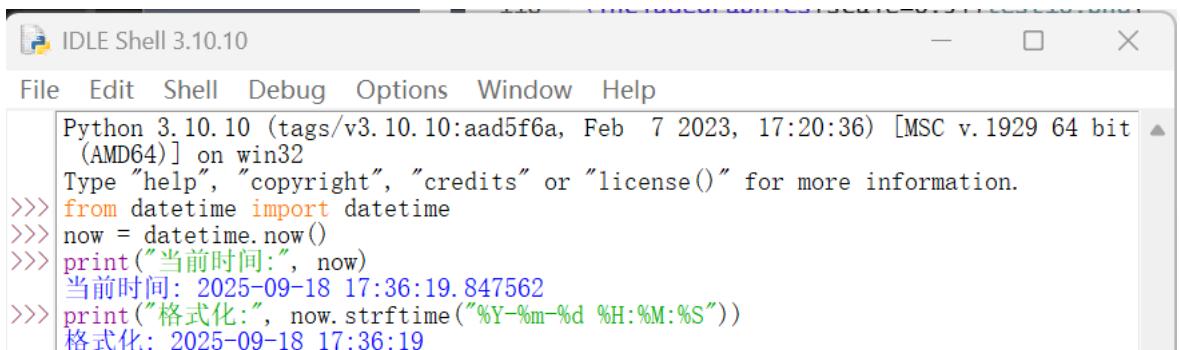
```
Python 3.13.7 (tags/v3.13.7:bce1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>> colors = ("red", "green", "blue")
>>> print("元组:", colors)
元组: ('red', 'green', 'blue')
>>> print("第一个元素:", colors[0])
第一个元素: red
>>> print("切片 [0:2]:", colors[0:2])
切片 [0:2]: ('red', 'green')
>>> print("长度:", len(colors))
长度: 3
>>>
```

## 2.4 字典练习



```
Python 3.11.8 (tags/v3.11.8:db85d51, Feb 6 2024, 22:03:32) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> person = {"name": "Alice", "age": 20, "city": "Beijing"}
>>> print("原始字典:", person)
原始字典: {'name': 'Alice', 'age': 20, 'city': 'Beijing'}
>>> print("获取 name:", person["name"])
获取 name: Alice
>>> person["age"] = 21
>>> print("修改 age:", person)
修改 age: {'name': 'Alice', 'age': 21, 'city': 'Beijing'}
>>> person["job"] = "student"
>>> print("新增键值对:", person)
新增键值对: {'name': 'Alice', 'age': 21, 'city': 'Beijing', 'job': 'student'}
>>> print("字典所有键:", list(person.keys()))
字典所有键: ['name', 'age', 'city', 'job']
>>> print("字典所有值:", list(person.values()))
字典所有值: ['Alice', 21, 'Beijing', 'student']
>>>
```

## 2.5 时间练习



```
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb 7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from datetime import datetime
>>> now = datetime.now()
>>> print("当前时间:", now)
当前时间: 2025-09-18 17:36:19.847562
>>> print("格式化:", now.strftime("%Y-%m-%d %H:%M:%S"))
格式化: 2025-09-18 17:36:19
```

## 2.6 日期练习

```
File Edit Shell Debug Options Window Help
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb 7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from datetime import datetime
>>> date1=datetime(2025,1,1)
>>> date2=datetime(2025,9,17)
>>> delta=date2-date1
>>> print("两个日期之间的天数差",delta.days)
两个日期之间的天数差 259
```

## 2.7 文件 I/O 练习

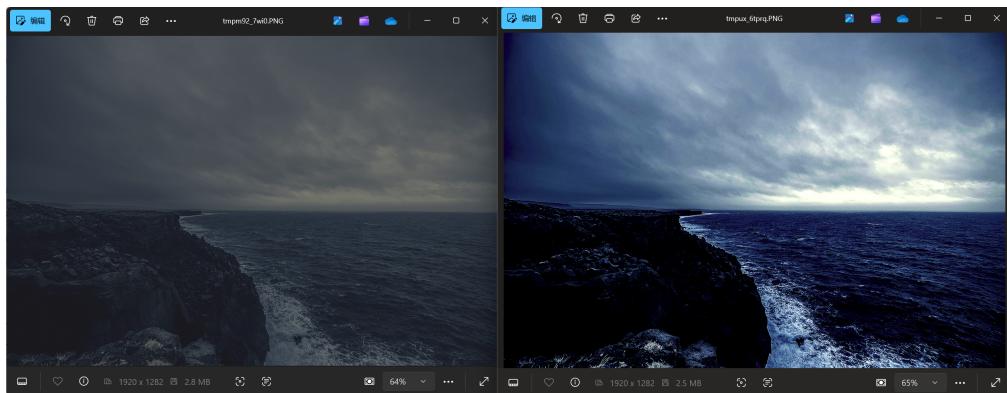
```
File Edit Shell Debug Options Window Help
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb 7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> with open("example.txt","w",encoding="utf-8") as f:
...     f.write("只是第一行\n")
...     f.write("Hello")
...
6
5
>>> |
```

# 3 Python 视觉应用

## 3.1 Pillow 库

采用 pillow 中的 enhance 增加对比度，使暗的图片变得明亮，左侧为原图，右侧为修改过后的图片

```
pillow.py ×
1  from PIL import Image,ImageEnhance
2  img_original = Image.open("dark.jpg")
3  img_original.show("Original Image")
4  img = ImageEnhance.Contrast(img_original)
5  img.enhance(3.8).show("Image With More Contrast")
```



### 3.2 Numpy 库

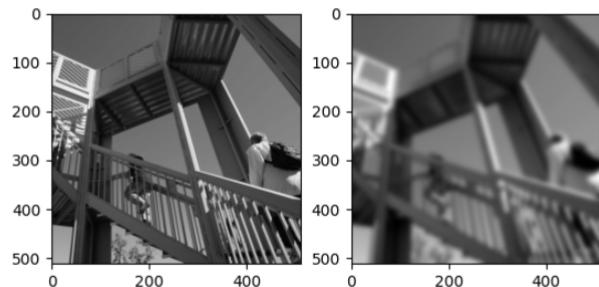
操纵图像的 RGB 值，进行图像处理。



### 3.3 Scipy 库

使用 scikit-image，使用 scipy 库进行高斯模糊。

```
Scipy.py ×  
1  from scipy.datasets import ascent  
2  import matplotlib.pyplot as plt  
3  from scipy.ndimage import gaussian_filter  
4  
5  fig = plt.figure()  
6  plt.gray()  
7  ax1 = fig.add_subplot(121)  
8  ax2 = fig.add_subplot(122)  
9  
10 img = ascent() # 新接口  
11 result = gaussian_filter(img, sigma=5)  
12  
13 ax1.imshow(img)  
14 ax2.imshow(result)  
15 plt.show()
```

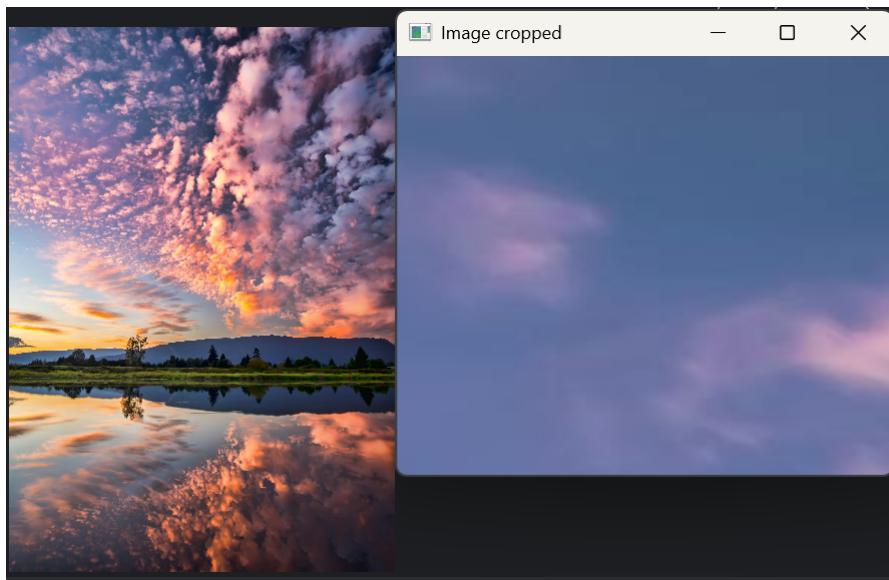


### 3.4 OpenCV 库

读取照片，从原图里裁剪出一张小图，然后对裁剪图按照目标尺寸进行缩放。  
左侧为原图，右侧为小图。

```
openCV.py ×
1 import cv2
2 img = cv2.imread("0.jpg")
3 imgCropped = img[50:283, 25:190]
4 shape = imgCropped.shape
5 print(shape[0])
6 imgCropped = cv2.resize(imgCropped, dsize=(shape[1]*2, shape[0]*12//10))
7 cv2.imshow( winname: "Image cropped", imgCropped)
8 cv2.imshow( winname: "Image", img)
9 cv2.waitKey(0)
10 |
```

The screenshot shows a code editor window with a dark theme. The file is named 'openCV.py'. The code imports cv2, reads an image named '0.jpg', crops a portion of it, prints its shape, resizes it, and then displays both the original and the cropped/resized images in separate windows using cv2.imshow(). A cv2.waitKey(0) call is at the end to keep the windows open.



## 4 心得体会

### 4.1 命令行环境的学习体会

最初接触命令行时，觉得黑底白字的界面比较“古老”，但随着学习的深入，我认识到命令行是一个强大而高效的工具。通过命令行可以快速完成文件管理、程序运行和环境配置等操作，而这些往往比图形化界面更灵活、更专业。

### 4.2 Python 入门基础的学习体会

在入门阶段，我掌握了 Python 的数据类型（字符串、列表、字典等）、控制语句（循环与条件判断）、函数与模块的基本用法。

通过动手写小程序，例如简单的日期计算器、文本处理脚本，我逐渐体会到编程的逻辑性和创造性。

我发现 Python 的语法简洁、功能强大，这为后续学习数据结构、算法以及各类应用打下了扎实的基础。

### 4.3 Python 在视觉应用中的初步探索

我尝试使用了 Python 的一些图像处理库（如 OpenCV、Pillow、Numpy 等），初步了解了计算机视觉的应用。我学会了读取、显示和保存图像，并尝试进行图像缩放、裁剪、旋转等基本操作。

GitHub 仓库链接：<https://github.com/gy-sun-enfp/System-tool-development>