

系统开发工具基础实验报告（三）

22110032031 张希敏

2024 年 9 月 12 日

目录

1	练习主题	1
2	练习内容	1
2.1	四个题目	1
2.1.1	题目一	1
2.1.2	题目二	1
2.1.3	题目三	2
2.1.4	题目四	2
2.2	20 个实例	3
3	问题及解决方案	17
4	解题感悟	17
5	github 链接	18

1 练习主题

- (1) 命令行环境
- (2) Python 入门基础
- (3) Python 视觉应用

2 练习内容

2.1 四个题目

2.1.1 题目一

我们可以使用类似 `ps aux | grep` 这样的命令来获取任务的 `pid`，然后您可以基于 `pid` 来结束这些进程。但我们其实有更好的方法来做这件事。在终端中执行 `sleep 10000` 这个任务。然后用 `Ctrl-Z` 将其切换到后台并使用 `bg` 来继续允许它。现在，使用 `pgrep` 来查找 `pid` 并使用 `pkill` 结束进程而不需要手动输入 `pid`。

```
abc@abc-virtual-machine:~/202408$ sleep 10000
^Z
[1]+  Stopped                  sleep 10000
abc@abc-virtual-machine:~/202408$ bg
[1]+  sleep 10000 &
abc@abc-virtual-machine:~/202408$ pgrep -af sleep
95087 sleep 10000
abc@abc-virtual-machine:~$ pkill -f sleep
[1]+  Terminated              sleep 10000
```

2.1.2 题目二

如果您希望某个进程结束后再开始另外一个进程，应该如何实现呢？在这个练习中，我们使用 `sleep 60 &` 作为先执行的程序。一种方法是使用 `wait` 命令。尝试启动这个休眠命令，然后待其结束后再执行 `ls` 命令。

```
abc@abc-virtual-machine:~/202408$ sleep 60 &
[1] 94902
abc@abc-virtual-machine:~/202408$ pgrep sleep
94902
abc@abc-virtual-machine:~/202408$ wait;ls
[1]+  Done                    sleep 60
buggy.sh  debug.sh  html_root  html.zip  marco.sh  out.log  pidwait.sh
```

但是，如果我们在不同的 bash 会话中进行操作，则上述方法就不起作用了。因为 wait 只能对子进程起作用。之前我们没有提过的一个特性是，kill 命令成功退出时其状态码为 0，其他状态则是非 0。kill -0 则不会发送信号，但是会在进程不存在时返回一个不为 0 的状态码。请编写一个 bash 函数 pidwait，它接受一个 pid 作为输入参数，然后一直等待直到该进程结束。您需要使用 sleep 来避免浪费 CPU 性能。

```
abc@abc-virtual-machine:~/202408$ vim pidwait.sh
abc@abc-virtual-machine:~/202408$ sleep 10 &
[1] 95101
abc@abc-virtual-machine:~/202408$ pidwait $(pgrep sleep)
abc@abc-virtual-machine:~$ alias dc=cd
abc@abc-virtual-machine:~$ dc ~/202408
abc@abc-virtual-machine:~/202408$ cd ~/202408
```

2.1.3 题目三

创建一个 dc 别名，它的功能是当我们错误的将 cd 输入为 dc 时也能正确执行。

```
abc@abc-virtual-machine:~$ alias dc=cd
abc@abc-virtual-machine:~$ dc ~/202408
abc@abc-virtual-machine:~/202408$ cd ~/202408
```

2.1.4 题目四

执行 history | awk '{ \$1="" ; print substr(\$0,2) }' | sort | uniq -c | sort -n | tail -n 10 来获取您最常用的十条命令，尝试为它们创建别名。

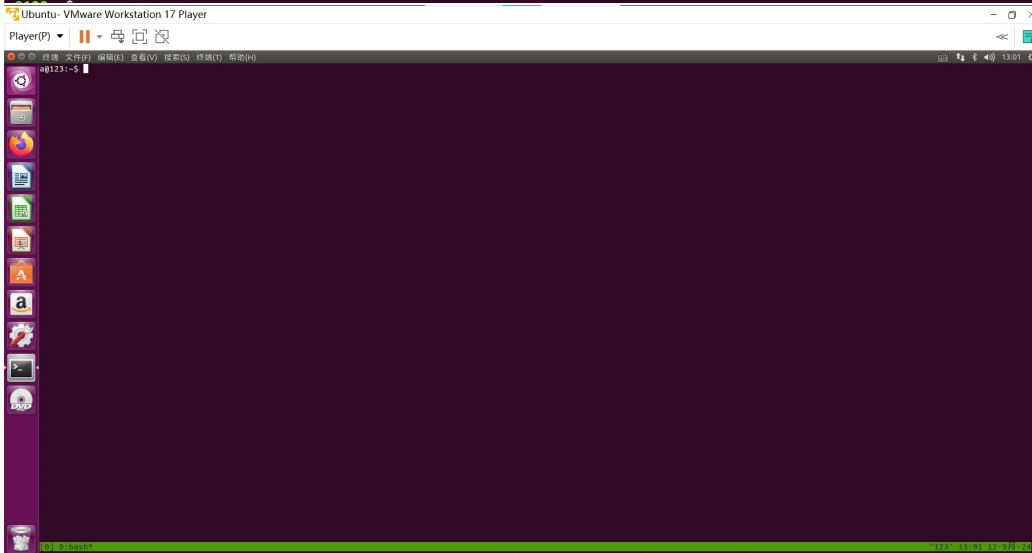
```
abc@abc-virtual-machine:~$ history | awk '{ $1="" ; print substr($0,2) }' | sort | uniq -c | sort -n | tail -n 10
  2 pkill -f sleep
  2 sleep 10000
  2 x-special/nautilus-clipboard
  3 ./ArrayParser ../test/t9.ec
  4 ./ArrayParser ../test/t1.ec
  5 ./ArrayParser ../test/t4.ec
  7 python test.py
  8 ./ArrayParser ../test/t20.ec
 32 cmake ..
 34 make
abc@abc-virtual-machine:~/Desktop$ alias ks='pkill -f sleep'
abc@abc-virtual-machine:~/Desktop$ alias gs='pgrep sleep'
abc@abc-virtual-machine:~/Desktop$ sleep 10000
^Z
[1]+  Stopped                  sleep 10000
abc@abc-virtual-machine:~/Desktop$ bg
[1]+  sleep 10000 &
abc@abc-virtual-machine:~/Desktop$ gs
1977
abc@abc-virtual-machine:~/Desktop$ ks
[1]+  Terminated              sleep 10000
abc@abc-virtual-machine:~/Desktop$
```

2.2 20 个实例

(1) 安装并启动 `tmux` 会话： 运行 `sudo apt-get install tmux` 在终端安装 `tmux`

输入 `tmux` 创建一个新的 `tmux` 会话

```
a@123:~$ sudo apt-get install tmux
[sudo] a 的密码:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列【新】软件包将被安装：
tmux
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 44 个软件包未被升级。
需要下载 223 kB 的归档。
解压缩后会消耗 601 kB 的额外空间。
获取:1 http://mirrors.cn99.com/ubuntu xenial/main amd64 tmux amd64 2.1-3build1 [223 kB]
已下载 223 kB，耗时 0秒 (237 kB/s)
正在选中未选择的软件包 tmux。
(正在读取数据库 ... 系统当前共安装有 221878 个文件和目录。)
正准备解包 .../tmux_2.1-3build1_amd64.deb ...
正在解包 tmux (2.1-3build1) ...
正在处理用于 man-db (2.7.5-1) 的触发器 ...
正在设置 tmux (2.1-3build1) ...
a@123:~$ tmux
[detached (from session 0)]
```



(2) 显示帮助： 按下组合键 `ctrl+b` ?

```
bind-key -T prefix C-b send-prefix
bind-key -T prefix C-o rotate-window
bind-key -T prefix C-e suspend-client
bind-key -T prefix Space next-layout
bind-key -T prefix | break-pane
bind-key -T prefix " split-window
bind-key -T prefix # list-buffers
bind-key -T prefix . command-prompt -I #S "rename-session '%s'"
bind-key -T prefix % split-window -h
bind-key -T prefix & confirm-before -p "kill-window #? (y/n)" kill-window
bind-key -T prefix ' command-prompt -p index "select-window -t '%s'"
bind-key -T prefix ( switch-client -p
bind-key -T prefix ) switch-client -n
bind-key -T prefix , command-prompt -I #W "rename-window '%s'"
bind-key -T prefix . delete-buffer
bind-key -T prefix . command-prompt "move-window -t '%s'"
bind-key -T prefix 0 select-window -t :0
bind-key -T prefix 1 select-window -t :1
bind-key -T prefix 2 select-window -t :2
bind-key -T prefix 3 select-window -t :3
bind-key -T prefix 4 select-window -t :4
bind-key -T prefix 5 select-window -t :5
bind-key -T prefix 6 select-window -t :6
bind-key -T prefix 7 select-window -t :7
bind-key -T prefix 8 select-window -t :8
bind-key -T prefix 9 select-window -t :9
bind-key -T prefix : command-prompt
bind-key -T prefix ; last-pane
bind-key -T prefix = choose-buffer
bind-key -T prefix ? list-keys
bind-key -T prefix D choose-client
bind-key -T prefix L switch-client -l
bind-key -T prefix M select-pane -M
bind-key -T prefix [ copy-mode
bind-key -T prefix ] paste-buffer
bind-key -T prefix C new-window
bind-key -T prefix d detach-client
bind-key -T prefix f command-prompt "find-window '%s'"
bind-key -T prefix l display-message
bind-key -T prefix l list-window
bind-key -T prefix n select-pane -n
bind-key -T prefix o next-window
bind-key -T prefix p select-pane -t :+
bind-key -T prefix P previous-window
bind-key -T prefix q display-panes
bind-key -T prefix r refresh-client
bind-key -T prefix s choose-line
bind-key -T prefix t clock-node
bind-key -T prefix w choose-window
bind-key -T prefix x confirm-before -p "kill-pane #? (y/n)" kill-pane
bind-key -T prefix z resize-pane -Z
bind-key -T prefix { swap-pane -U
```

(3) 退出当前 tmux 会话： 按下组合键 `ctrl+d` 或者显式输入 `exit` 命令

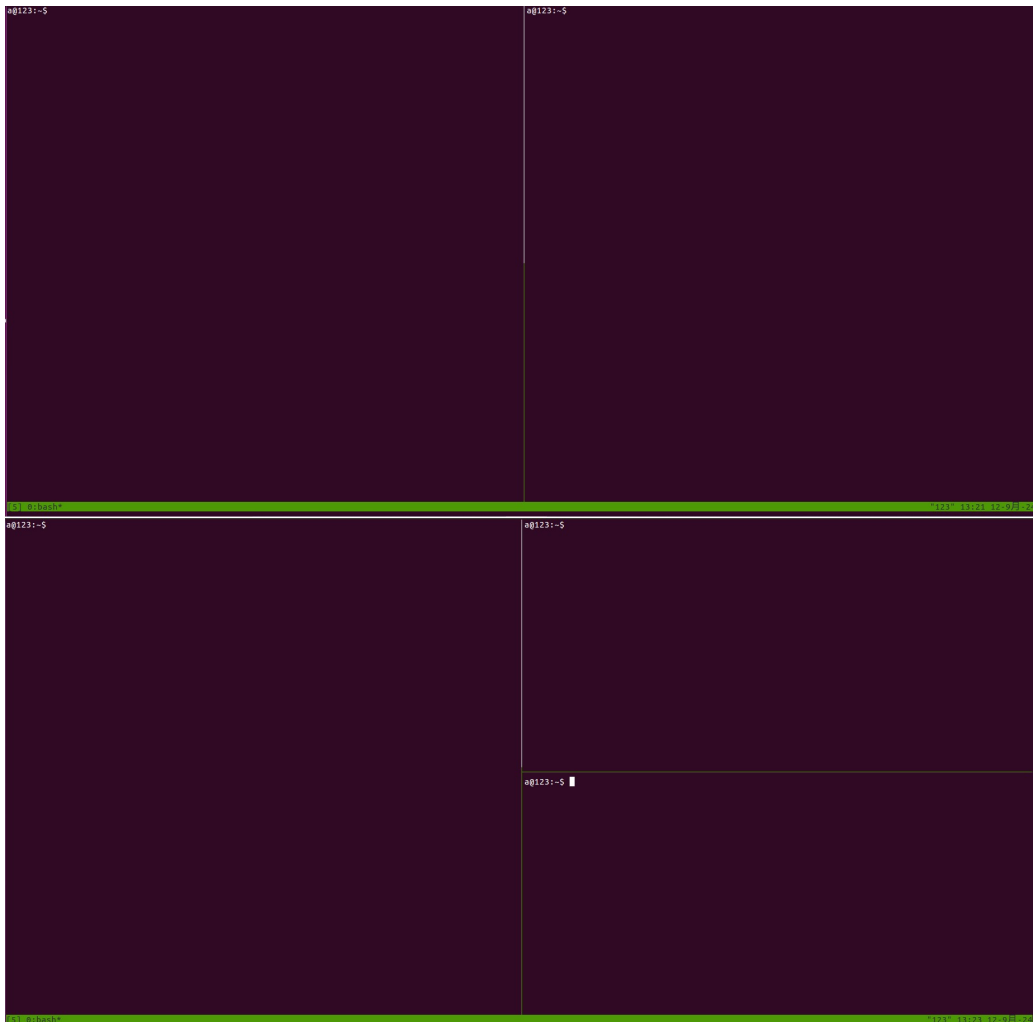
```
a@123:~$ tmux
[exited]
```

(4) `tmux new -s <session-name>`： 上面命令新建一个指定名称的会话。

```
a@123:~$ tmux new -s new1
a@123:~$
```

(5) 将窗格拆分为两个窗格： 按下组合键 `ctrl+b %` 将窗格拆分为左窗格和右窗格。

按下组合键 `ctrl+b` ” 将窗格拆分为顶部窗格和底部窗格。
要切换到不同窗格，可按下 `ctrl+b` 方向键进行切换。



(6) 分离对话： 按下组合键 `Ctrl+b d` 或者输入 `tmux detach` 命令，就会将当前会话与窗口分离。这样会退出当前 Tmux 窗口，但是会话和里面的进程仍然在后台运行。

```
a@123:~$ tmux detach
a@123:~$ tmux
[detached (from session 5)]
```

(7) **tmux ls:** 查看当前所有的 Tmux 会话。

```
a@123:~$ tmux ls
0: 1 windows (created Thu Sep 12 13:01:16 2024) [204x52]
5: 1 windows (created Thu Sep 12 13:21:15 2024) [204x52]
```

(8) **tmux attach:** 重新接入某个已存在的会话。

```
a@123:~$ tmux attach -t 0
a@123:~$
```

(9) **tmux kill-session:** 杀死某个会话。

```

a@123:~$ tmux ls
0: 1 windows (created Thu Sep 12 13:45:33 2024) [204x52]
5: 1 windows (created Thu Sep 12 13:21:15 2024) [204x52]
a@123:~$ tmux kill-session -t 0
a@123:~$ tmux ls
5: 1 windows (created Thu Sep 12 13:21:15 2024) [204x52]


```

(10) **tmux switch:** 切换会话。

```

a@123:~$ tmux switch -t 1

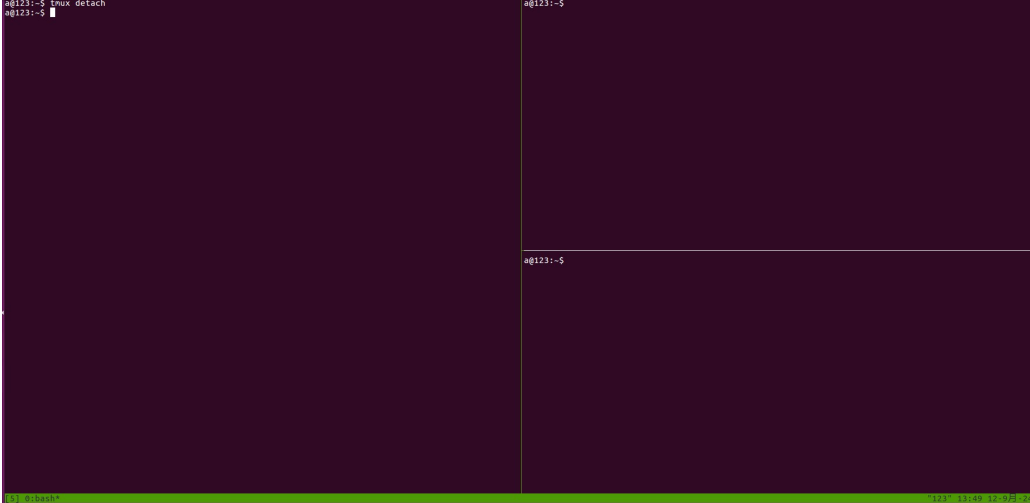
```



```

a@123:~$ tmux detach
a@123:~$

```



(11) **tmux rename-session:** 重命名会话。

```

a@123:~$ tmux ls
1: 1 windows (created Thu Sep 12 13:46:41 2024) [204x52]
5: 1 windows (created Thu Sep 12 13:21:15 2024) [204x52]
a@123:~$ tmux rename-session -t 1 2
a@123:~$ tmux ls
2: 1 windows (created Thu Sep 12 13:46:41 2024) [204x52]
5: 1 windows (created Thu Sep 12 13:21:15 2024) [204x52]

```


(12) 创建 ssh 密钥对: 输入命令 `ssh-keygen -o -a 100 -t ed25519`

```
a@123:~/ssh$ ssh-keygen -o -a 100 -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/a/.ssh/id_ed25519): sshkey
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in sshkey.
Your public key has been saved in sshkey.pub.
The key fingerprint is:
SHA256:7/WPu19vQ3osUNI71FHgXm3hC/QX5MQNLLXqabFCC0B a@123
The key's randomart image is:
+--[ED25519 256]--+
|      o*X=      |
|    ..o*        |
|  .  .o=*       |
| o  . .-o+o    |
| So .+oo.      |
| .....o+..     |
| E. .o==       |
| . .o+ *o      |
| . o+*o+       |
+-----[SHA256]-----+
```

(13) ssh 远程连接: 输入命令 `ssh <username>@<ip> [-p <port>]`

```
duc@ls1ouc-vm:~/Desktop$ ssh stui13@10.140.32.159 -p 47113
The authenticity of host '[10.140.32.159]:47113 ([10.140.32.159]:47113)' can't be established.
ECDSA key fingerprint is SHA256:Fh06li/VuwMhthYp0aIDduFLTNLLPeIM28iy3XywcIE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '[10.140.32.159]:47113' (ECDSA) to the list of known hosts.
stui13@10.140.32.159's password:
Permission denied, please try again.
stui13@10.140.32.159's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-97-generic x86_64)

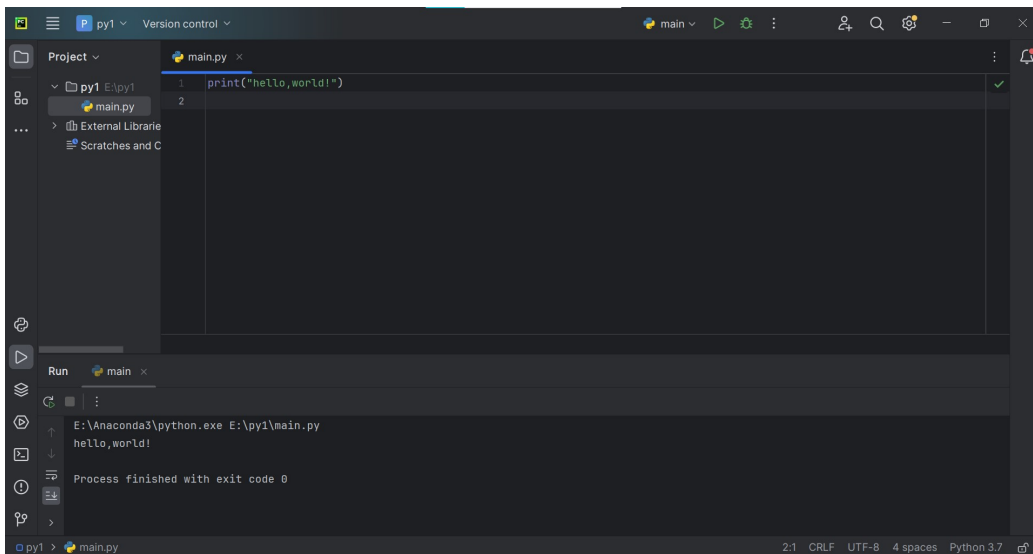
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

stui13@2998b529c935:~$ cd /etc
```

(14)python 入门: 安装并配置 pycharm, 在 pycharm 上创建一个 python 项目, 写示例输出 helloworld。



(15) python 命令行参数： 在终端命令行使用-h 参数查看各参数帮助信息。

```

C:\Users\long-wei>python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
  -b      : issue warnings about str(bytes,instance), str(bytearray,instance)
            and comparing bytes/bytearray with str. (-bb: issue errors)
  -B      : don't write .pyc files on import; also PYTHONNODONTWRITEBYTECODE=x
  -c cmd  : program passed in as string (terminates option list)
  -d      : debug output from parser; also PYTHONDEBUG=x
  -E      : ignore PYTHON* environment variables (such as PYTHONPATH)
  -h      : print this help message and exit (also --help)
  -i      : inspect interactively after running script; forces a prompt even
            if stdin does not appear to be a terminal; also PYTHONINSPECT=x
  -I      : isolate Python from the user's environment (implies -E and -s)
  -m mod  : run library module as a script (terminates option list)
  -O      : remove assert and __debug__ dependent statements; add -opt-1 before
            .pyc extension; also PYTHONOPTIMIZE=x
  -OO     : do -O changes and also discard docstrings; add -opt-2 before
            .pyc extension
  -q      : don't print version and copyright messages on interactive startup
  -s      : don't add user site directory to sys.path; also PYTHONNOUSERSITE
  -S      : don't imply 'import site' on initialization
  -u      : force the stdout and stderr streams to be unbuffered;
            this option has no effect on stdin; also PYTHONUNBUFFERED=x
  -v      : verbose (trace import statements); also PYTHONVERBOSE=x
            can be supplied multiple times to increase verbosity
  -V      : print the Python version number and exit (also --version)
            when given twice, print more information about the build
  -W arg  : warning control; arg is action:message:category:module:lineno
            also PYTHONWARNINGS=arg
  -x      : skip first line of source, allowing use of non-Unix forms of #cmd
  -X opt  : set implementation-specific option. The following options are available:
            -X faulthandler: enable faulthandler
            -X showrefcount: output the total reference count and number of used
                memory blocks when the program finishes or after each statement in the
                interactive interpreter. This only works on debug builds.
            -X tracemalloc: start tracing Python memory allocations using the
                tracemalloc module. By default, only the most recent frame is stored in a
                traceback of a trace. Use -X tracemalloc=NFRAME to start tracing with a
                traceback limit of NFRAME frames.
            -X showallcount: output the total count of allocated objects for each
                type when the program finishes. This only works when Python was built with
                COUNT_ALLOCS defined.
            -X importtime: show how long each import takes. It shows module name,
                cumulative time (including nested imports) and self time (excluding
                nested imports). Note that its output may be broken in multi-threaded
                application. Typical usage is python3 -X importtime -c 'import asyncio'
            -X dev: enable Python's "development mode" introducing additional counting

```

(16) python 实例-素数判断： 在虚拟机终端环境编写 python 程序实现素数判断的功能。

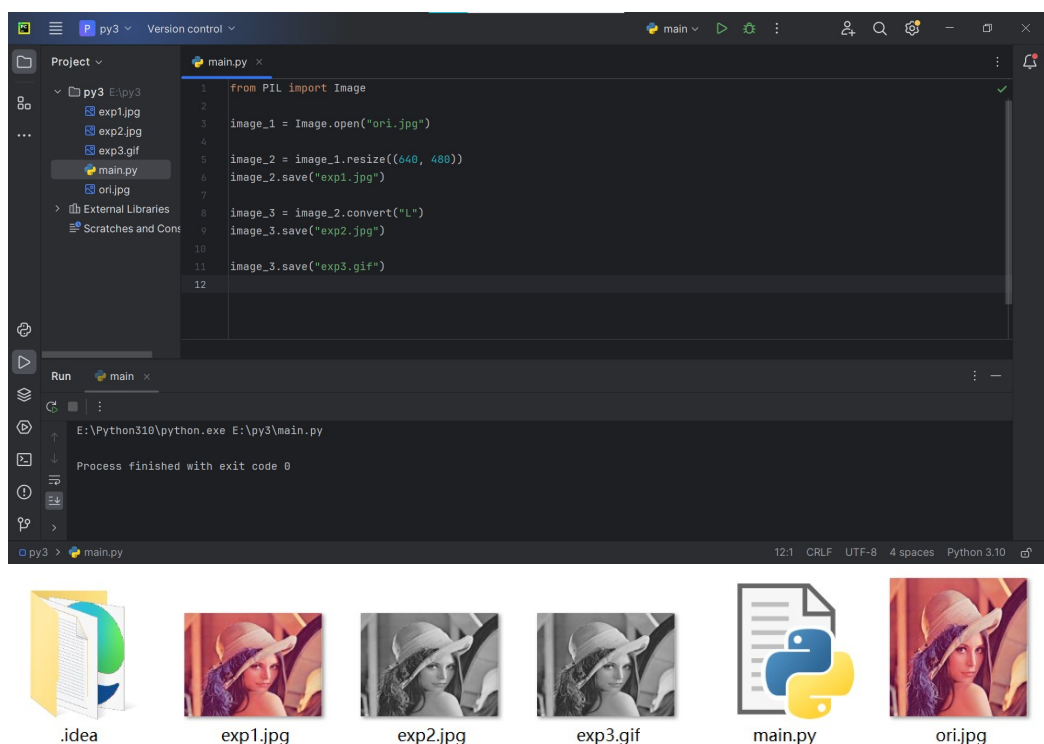
(代码已上传至 github 库-task2-py2.py)

```
1 # Python 程序用于检测用户输入的数字是否为质数
2
3 # 用户输入数字
4 num = int(input("请输入一个数字: "))
5
6 # 质数大于 1
7 if num > 1:
8     # 查看因子
9     for i in range(2, num):
10         if (num % i) == 0:
11             print(num, "不是质数")
12             print(i, "乘以", num//i, "是", num)
13             break
14         else:
15             print(num, "是质数")
16
17 # 如果输入的数字小于或等于 1, 不是质数
18 else:
19     print(num, "不是质数")
```

py2.py [3] 19, 28-24 Alt

```
ouc@tslouc-vm:~/Desktop$ vim py2.py
ouc@tslouc-vm:~/Desktop$ python py2.py
请输入一个数字: 12
12 不是质数
2 乘以 6 是 12
ouc@tslouc-vm:~/Desktop$ python py2.py
请输入一个数字: 71
71 是质数
```

(17) python 视觉-图像基本操作： 在 pycharm 写 Python 程序，引用 PIL 库，实现修改图片大小、转换灰度图像，另存为其他图像格式的功能。
(具体代码及输出结果已上传至 github 库-task2-py3)



(18) **python 视觉-频率域图像操作：** 在 pycharm 写 Python 程序对图片进行傅立叶变换、DCT 变换并查看频谱图。

实例要求： 用 Python 写程序，对目录下的图像（1.bmp，2.bmp，2.jpg，3.bmp，4.bmp），实现：

- （1）查看不同图像的傅立叶变换的图像查看不同图像的 DCT（离散余弦）变换。
- （2）对变换后得到的频谱图使用空间域图像增强的方法增强效果。
- （3）采用低通滤波器和高通滤波器对图像进行频率域滤波，设置不同的阈值，查看效果。

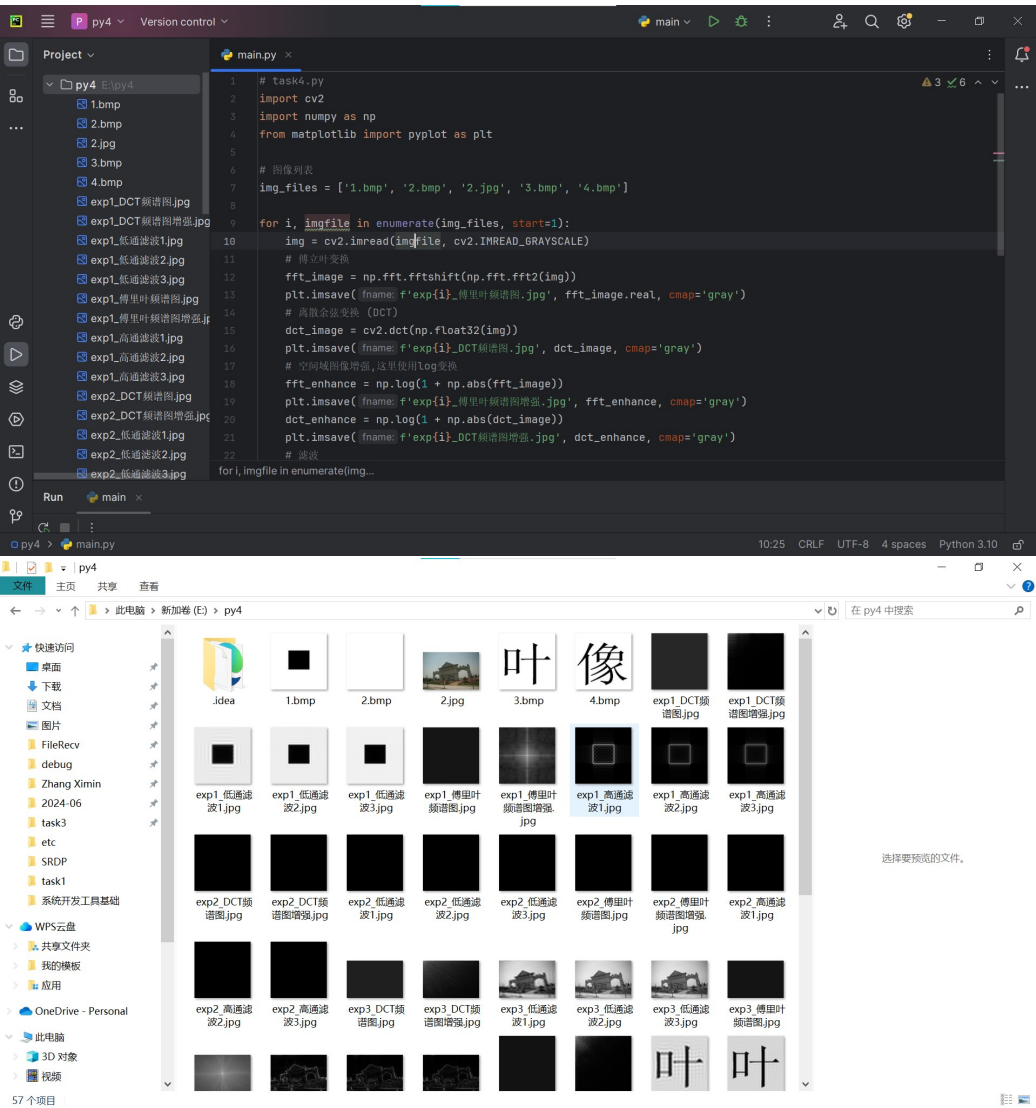
实例解答： 分如下四步：

- （1）读取图像列表（1.bmp，2.bmp，2.jpg，3.bmp，4.bmp）。
- （2）对于列表中每一个图像,用 numpy 库的 `np.fft.fft2()` 和 `np.fft.fftshift()` 函数对图像进行快速傅里叶变换，保存图像；用 opencv 库的 `cv2.dct()` 函数对图像进行离散余弦变换，保存图像。

(3) 对上一步得到的频谱图使用空间域图像增强的方法增强效果。修改图像灰度，这里用 \log 对数变换，增强图像较暗区域的细节，使频谱图可视化。保存结果。

(4) 分别采用低通滤波器和高通滤波器对图像进行频率域滤波。首先通过 $\text{dft}()$ 和 $\text{fftshift}()$ 函数对图像进行傅里叶变换，再设置掩膜 mask ，将掩膜与傅里叶变化后的图像相乘，保留低频部分，再进行傅里叶逆变换 $\text{idft}()$ 得到滤波结果。对低通滤波和高通滤波设置不同的掩膜及阈值 yuzhi ，重复上述操作，保存并查看效果。

(具体代码及输出结果已上传至 [github](#) 库-task2-py4)



(19) python 视觉-空间域图像增强 在 pycharm 写 Python 程序对图片进行直方图查看、高斯滤波磨皮操作。

实例要求： 用 Python 写一段程序，针对提供的图片 ori1.jpg 和 ori2.jpg，实现：

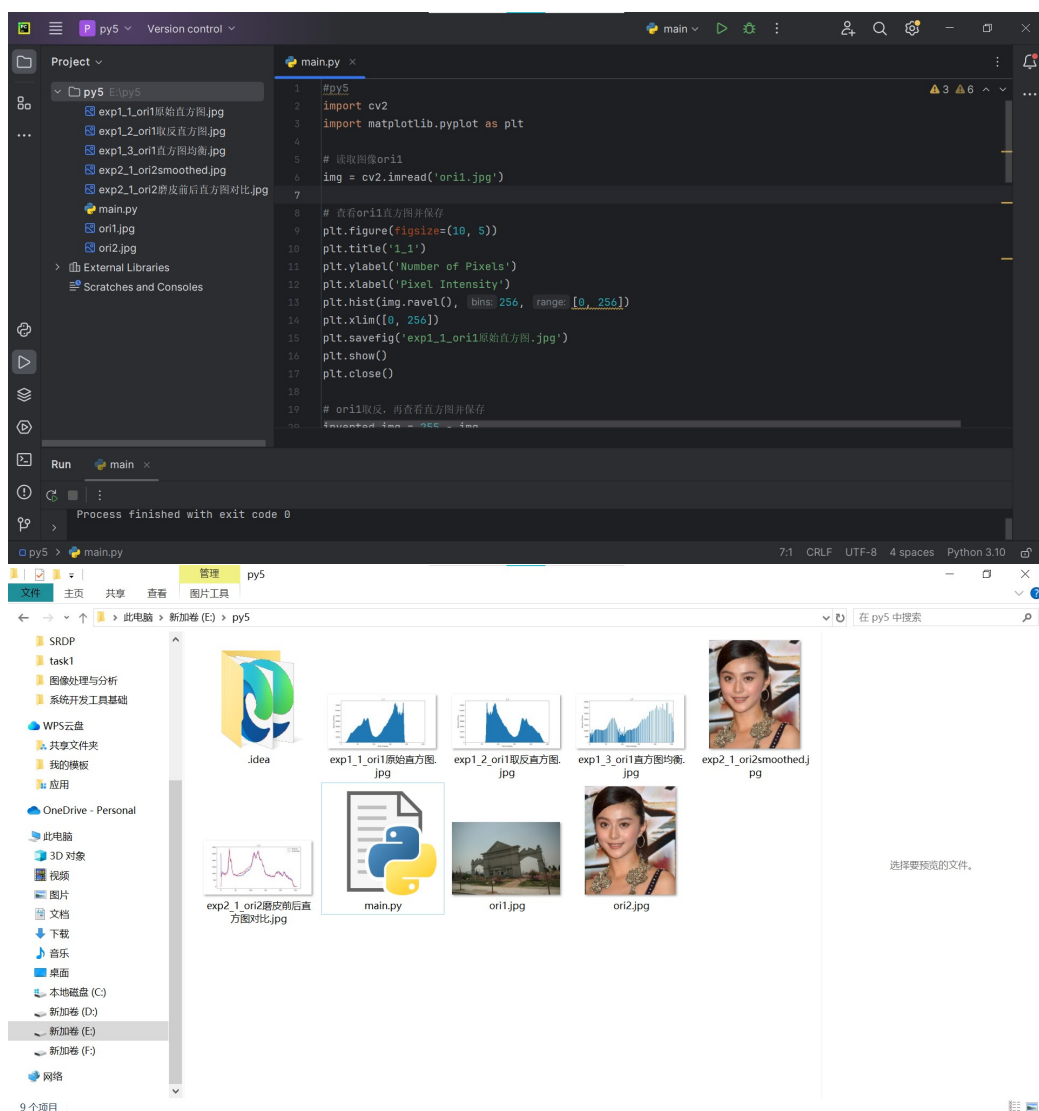
(1) 查看 ori1.jpg 的直方图并保存；对 ori1.jpg 取反，再查看直方图并保存；对 ori1.jpg 使用直方图均衡，再查看直方图并保存。

(2) 对人脸 ori2.jpg 进行滤波等操作实现类似美图秀秀的磨皮功能，并对比磨皮前后直方图变化。

实例解答： (1) 利用 opencv 库读取图像并查看直方图，plt 库展示并保存直方图。取反图 $= 255 - \text{原图}$ ，直方图均衡使用 `cv2.equalizeHist()` 函数。

(2) 磨皮使用 `cv2.GaussianBlur()` 进行高斯模糊，plt 库展示并保存直方图。

(具体代码及输出结果已上传至 github 库-task2-py5)



(20) python 视觉-油滴检测： 在 pycharm 写 Python 程序检测图片中的油滴并进行填色处理，保存结果。

实例要求： 图片为海水中漏油溢出的油滴；请标记出图片中清晰的油滴并对其进行填色处理。

实例解答： 分为如下五步：

(1) 设置图像列表。

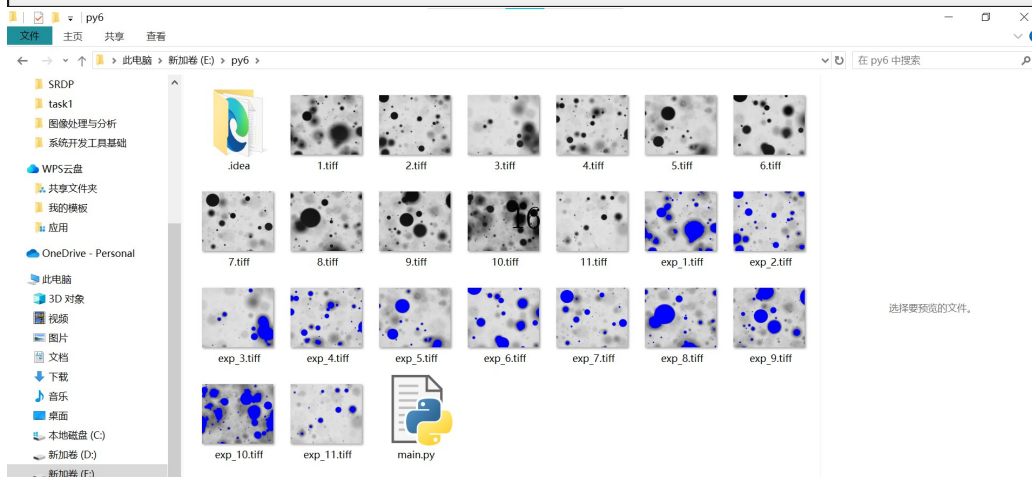
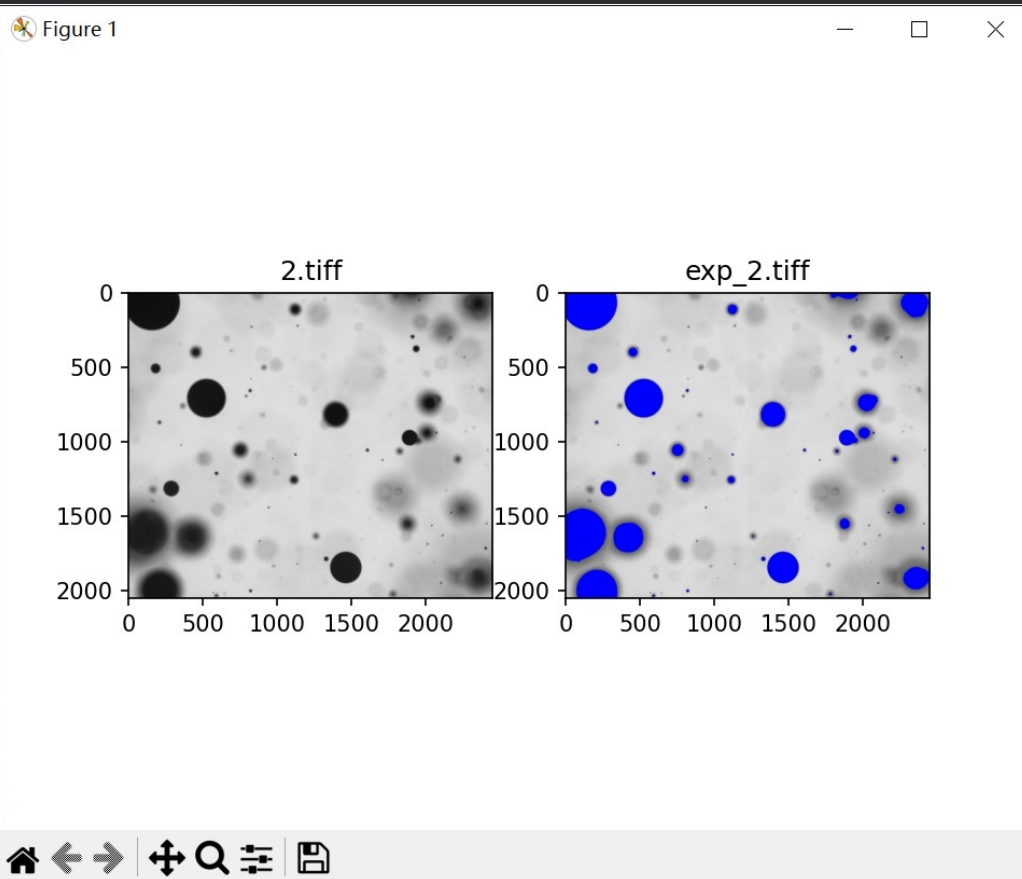
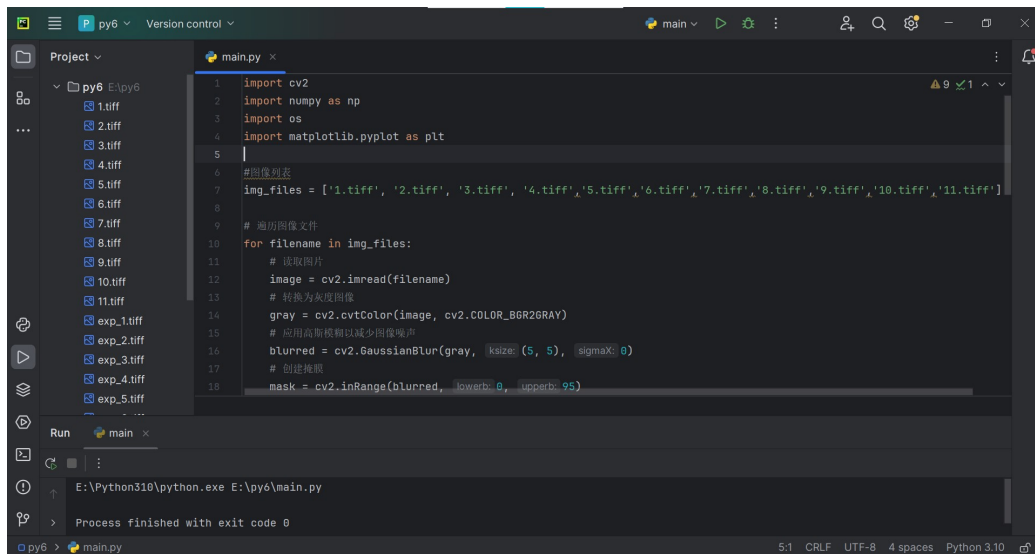
(2) 通过灰度值区分检测油滴轮廓。具体思路是：`cv2.imread()` 读取图像，对列表中的每个图像，`cv2.cvtColor()` 转换为灰度图像；应用高斯模糊 `cv2.GaussianBlur()` 减少噪声；`cv2.inRange()` 创建掩膜覆盖灰度值为 0 到 95 的区域，即油滴区域（这里也可通过二值化操作，将阈值设为 95 区分油滴区域）；进行形态学开运算操作，即先腐蚀（`cv2.erode()` 函数），再对腐蚀结果进行膨胀（`cv2.dilate()` 函数），目的是去噪、便于提取轮廓；对处理后的掩膜提取轮廓，即得到油滴轮廓。

(3) 对每个提取到的轮廓，`cv2.drawContours()` 进行填色处理。

(4) 用 `plt` 库函数将处理后的照片与原图并列展示，以体现填色效果。根据填色效果适当手动调节掩膜的灰度范围以达到更好的效果。

(5) 保存填色后的图像。

（具体代码及输出结果已上传至 [github](#) 库-task2-py6）



3 问题及解决方案

(1) 问题：安装 tmux 时报错。

解决方案：这台虚拟机的网络环境有问题，换了一台虚拟机后成功安装。

```
luc@abc-virtual-machine:~/Desktop$ sudo apt-get install tmux
[sudo] password for abc:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libutempter0
The following NEW packages will be installed:
  libutempter0 tmux
0 upgraded, 2 newly installed, 0 to remove and 65 not upgraded.
Need to get 380 kB of archives.
After this operation, 816 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Err:1 http://cn.archive.ubuntu.com/ubuntu focal/main amd64 libutempter0 amd64 1.1.6-4
Temporary failure resolving 'cn.archive.ubuntu.com'
Ign:2 http://cn.archive.ubuntu.com/ubuntu focal-updates/main amd64 tmux amd64 3.0a-2ubuntu0.4
Err:2 http://security.ubuntu.com/ubuntu focal-updates/main amd64 tmux amd64 3.0a-2ubuntu0.4
Temporary failure resolving 'cn.archive.ubuntu.com'
E: Failed to fetch http://cn.archive.ubuntu.com/ubuntu/pool/main/libu/libutempter/libutempter0_1.1.6-4_amd64.deb Temporary failure resolving 'cn.archive.ubuntu.com'
E: Failed to fetch http://security.ubuntu.com/ubuntu/pool/main/t/tmux/tmux_3.0a-2ubuntu0.4_amd64.deb Temporary failure resolving 'cn.archive.ubuntu.com'
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?
luc@abc-virtual-machine:~/Desktop$ tmux
a@123:~$ sudo apt-get install tmux
[sudo] a 的密码:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列【新】软件包将被安装：
tmux
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 44 个软件包未被升级。
需要下载 223 kB 的归档。
解压缩后会消耗 601 kB 的额外空间。
获取:1 http://mirrors.cn99.com/ubuntu xenial/main amd64 tmux amd64 2.1-3build1 [223 kB]
已下载 223 kB，耗时 0秒 (237 kB/s)
正在选中未选择的软件包 tmux。
(正在读取数据库 ... 系统当前共安装有 221878 个文件和目录。)
准备解包 .../tmux 2.1-3build1_amd64.deb ...
正在解包 tmux (2.1-3build1) ...
正在处理用于 man-db (2.7.5-1) 的触发器 ...
正在设置 tmux (2.1-3build1) ...
a@123:~$ tmux
[detached (from session 0)]
```

4 解题感悟

这次实验内容包括命令行环境、Python 入门基础和 Python 视觉应用。通过本实验我掌握了 tmux、修改配置、管理进程、编写 python 程序、引用 cv 库和 plt 库等库的库函数进行图像处理等技能。

通过练习命令行部分的习题，我掌握了使用 pgrep 和 pkill 等命令来高效管理进程、使用 tmux 进行终端多路复用以同时执行多个任务、创建别名代替一长串包含许多选项的命令以提高效率。通过学习这部分内容的知识，我对命令行环境的高效使用有了更深入的理解。

在 Python 入门基础和 Python 视觉应用方面，由于我上学期修了图像处理与分析，练习了足够多了 python 题目，所以较为熟练。python 有着丰富的库，与其他编程语言相比，python 在视觉图像处理方面有着无可比拟

的优势。在本实验中我温习了 Python 的基本语法和 Python 在图像处理领域的应用，进一步锻炼了编程能力。

5 github 链接

<https://github.com/zxm2580/xtkfgjjc-202408.git>