

实验报告——命令行环境，python 入门基础，python 视觉应用

姓名：徐亚齐 学号：23020007136

2024 年 9 月 6 日

1 实验实例

1.1 获取最常用的 10 条命令，并选择一条为它起别名

要查询最常用的命令,需要用到 `history | awk '{ $1=""; print substr($0,2)}' | sort | uniq -c | sort -n | tail -n 10`。起别名用 `alias`

```
→ / (master) history | awk '{ $1=""; print substr($0,2)}' | sort | uniq -c | sort -n | tail -n 10
  7 curl -fLo ~/.vim/autoload/plug.vim --create-dirs https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
  7 vim
  7 vimtutor
  8 ./semester | grep last-modified > ~/last-modified.txt
10 git add .
11 git branch -M main
12 ls
13 git init
20 git status
23 git push -u origin main

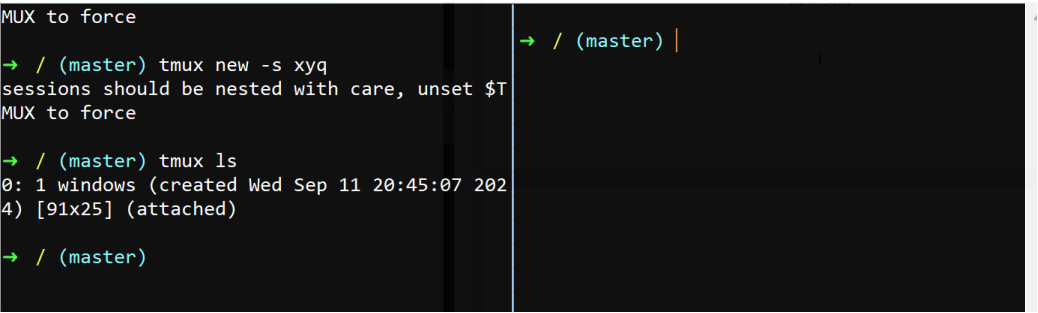
→ / (master) alias sl=ls

→ / (master) sl
Git/                               cmd/                               hello.txt                         tmux.exe*                         usr/
Git-2.46.0-64-bit.zip             dev/                               mingw64/                         tmux.exe.stackdump
LICENSE.txt                       etc/                               msys-event-2-1-7.dll*           unins000.dat
ReleaseNotes.html                git-bash.exe*                     proc/                             unins000.exe*
bin/                               git-cmd.exe*                     tmp/                             unins000.msg
```

图 1: 实例

1.2 练习使用 tmux

练习以下命令：`tmux` 表示开始一个新的会话，`tmux new -s NAME` 表示以指定名称开始一个新的会话，`tmux ls` 表示列出当前所有会话，`<C-b> %` 表示垂直分割，`<C-b> w` 表示列出当前所有窗口

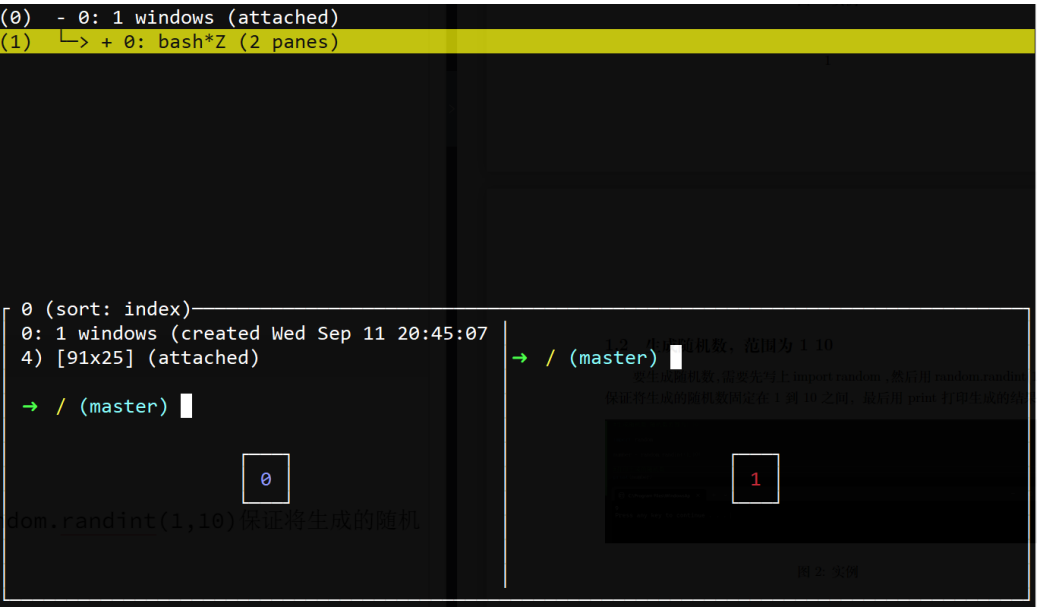


```
MUX to force
→ / (master) tmux new -s xyq
sessions should be nested with care, unset $T
MUX to force

→ / (master) tmux ls
0: 1 windows (created Wed Sep 11 20:45:07 202
4) [91x25] (attached)

→ / (master)
```

图 2: 垂直分割后的界面



```
(0) - 0: 1 windows (attached)
(1) ↵ + 0: bash*Z (2 panes)

0 (sort: index)
0: 1 windows (created Wed Sep 11 20:45:07
4) [91x25] (attached)

→ / (master)

dom.randint(1,10) 保证将生成的随机
```

图 3: 列出窗口

1.3 创建密钥并查看其相关信息

使用 `ssh-keygen -o -a 100 -t ed25519` 来创建密钥，创建时需要输入密码，然后用 `ls` 命令就可以输出相关的信息。

```
→ / (master) ~/.ssh
-bash: /c/Users/Lenovo/.ssh: Is a directory

→ / (master) ssh-keygen -o -a 100 -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/Lenovo/.ssh/id_ed25519): miyao
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in miyao
Your public key has been saved in miyao.pub
The key fingerprint is:
SHA256:Sufh2tsMMNLyrrp6n4Jkd28chRiUfHUVoi4wY65DT2hc Lenovo@LAPTOP-FGVNVP6K
The key's randomart image is:
+--[ED25519 256]--+
|          ..=O          |
|       . . o +         |
|      o + * + .        |
|    + +.E + .          |
|   *o.=.S              |
|  . o=+B..             |
|  . .000+.             |
|1. = *.00+             |
|8.+.B++.O.O            |
+-----[SHA256]-----+

→ / (master) ls ~/.ssh
all.txt  authorized_keys  id_ed25519  id_ed25519.pub  id_rsa.pub/
```

图 4: 实例

1.4 练习后台运行的有关命令，获取任务，打印任务编号，开始运行与结束运行等

`jobs` 命令会列出当前终端会话中尚未完成的全部任务，百分号 + 任务编号可以获取任务，命令中的 `&` 后缀可以让命令在直接在后台运行等。

```

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ sleep 1000

[1]+  Stopped                  sleep 1000

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ nohup sleep 2000 &
[2] 1590
nohup: ignoring input and appending output to 'nohup.out'

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ jobs
[1]+  Stopped                  sleep 1000
[2]-  Running                  nohup sleep 2000 &

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ bg %1
[1]+ sleep 1000 &

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ jobs
[1]-  Running                  sleep 1000 &
[2]+  Running                  nohup sleep 2000 &

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ kill -STOP %1

[1]+  Stopped                  sleep 1000

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ jobs
[1]+  Stopped                  sleep 1000
[2]-  Running                  nohup sleep 2000 &

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ kill -SIGHUP %1
[1]+  Hangup                  sleep 1000

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ jobs
[2]+  Running                  nohup sleep 2000 &

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ kill -SIGHUP %2

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ jobs
[2]+  Running                  nohup sleep 2000 &

Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ kill %2
[2]+  Terminated             nohup sleep 2000

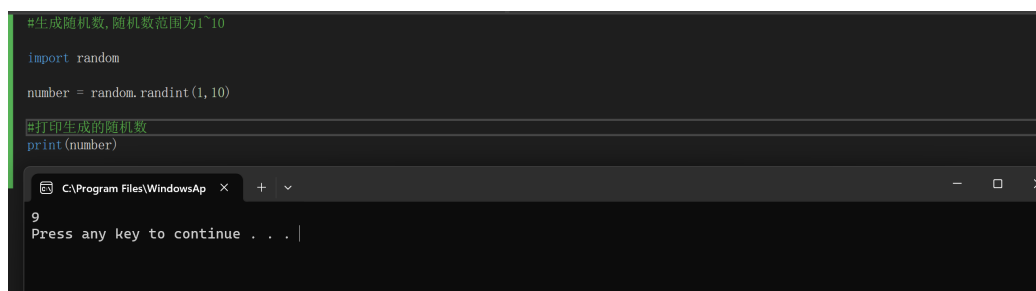
Lenovo@LAPTOP-FGVNVP6K MINGW64 ~/Desktop (main)
$ jobs

```

图 5: 实例

1.5 生成随机数，范围为 1 10

要生成随机数,需要先写上 `import random` ,然后用 `random.randint(1,10)` 保证将生成的随机数固定在 1 到 10 之间，最后用 `print` 打印生成的结果。



```
#生成随机数,随机数范围为1~10
import random
number = random.randint(1,10)

#打印生成的随机数
print(number)
```

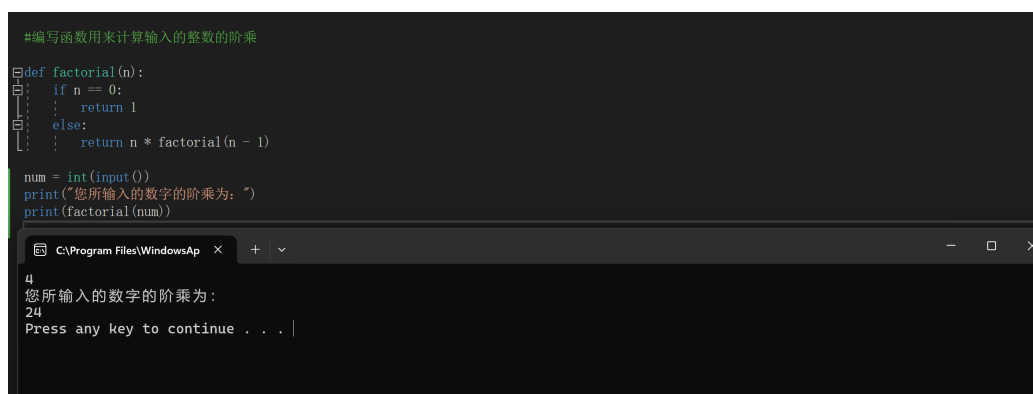
C:\Program Files\WindowsAp x + v - □ x

9
Press any key to continue . . . |

图 6: 实例

1.6 用 python 编写函数，用来计算整数的阶乘

首先要用到 `def`，在函数中用 `if` 语句将输入的整数分为 0 和非 0 两类，分别进行计算。最后就可以使用 `int (input ())` 输入一个整数，并调用函数，输出函数的返回值。



```
#编写函数用来计算输入的整数的阶乘
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

num = int(input())
print("您所输入的数字的阶乘为: ")
print(factorial(num))
```

C:\Program Files\WindowsAp x + v - □ x

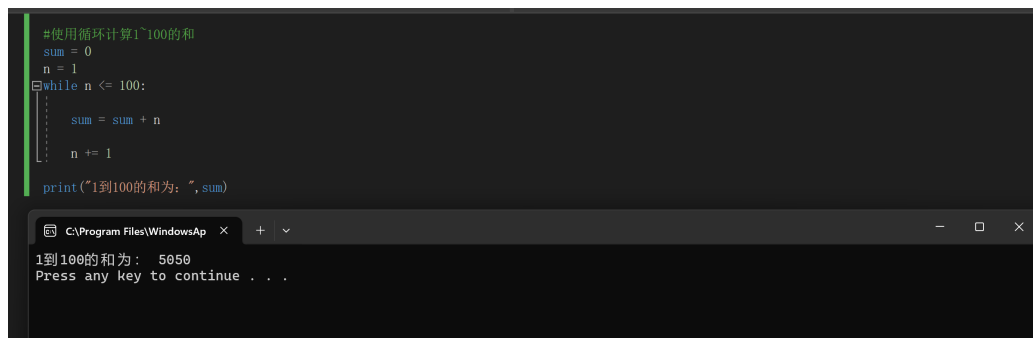
4
您所输入的数字的阶乘为:
24
Press any key to continue . . . |

图 7: 实例

1.7 使用循环计算 1 100 的和

首先将一个变量 `sum` 赋值为 0，用来表示计算的和。再将 `n` 的初始值设置为 1，用 `while` 循环，每次循环 `sum` 都要加上 `n`，`n` 本身加 1，直到 `n`

的值大于 100 后停止循环，用 `print` 输出 `sum` 即可获得 1 100 的和。



```
#使用循环计算1~100的和
sum = 0
n = 1
while n <= 100:
    sum = sum + n
    n += 1
print("1到100的和为: ",sum)
```

C:\Program Files\WindowsAp x + v - □ x

1到100的和为: 5050
Press any key to continue . . .

图 8: 实例

1.8 尝试截取字符串一部分与一段字符拼接

先定义一个字符串，用 `print` 输出时，使用 `[]` 加上 `+'tom'`，就可以指定截取部分，并在截取后的字符串后面加上 `tom` 字符。



```
#尝试截取字符串一部分与一段字符拼接
str = 'Hello World!'
print("拼接后的字符串: ",str[0:6]+'tom')
```

C:\Program Files\WindowsAp x + v - □ x

拼接后的字符串: Hello tom
Press any key to continue . . . |

图 9: 实例

1.9 将字符串 a = “This is string example” 全部转成大写，字符串 b = “Welcome To My World” 全部转成小写

首先定义字符串 `a = “This is string example”` 和 `b = “Welcome To My World”`，然后字符串 `a` 用 `upper()` 将字母变为大写，`b` 字符用 `lower()` 将字符变为小写，再用 `print` 输出结果。

```
#将字符串 a = "This is string example" 全部转成大写, 字符串 b = "Welcome To My World" 全部转成小写
a = 'This is string example'
b = 'Welcome To My World'

print(a.upper())
print(b.lower())
```

C:\Program Files\WindowsAp x + v - □ x

THIS IS STRING EXAMPLE
welcome to my world
Press any key to continue . . . |

图 10: 实例

1.10 比较给出的列表是否相同

因为比较两个列表需要用到 `operator.eq`, 所以需要先加上 `import operator`。定义两个列表, 并用 `operator.eq` 将定义好的列表进行比较。用 `if` 语句进行判断, 如果返回值是 `true`, 则输出“列表相同”, 否则输出“列表不相同”。

```
#比较给出的列表是否相同

import operator
a = [1, 2, 3]
b = [2, 3, 4]

if operator.eq(a, b):
    print("两个列表相同")
else:
    print("两个列表不相同")
```

C:\Program Files\WindowsAp x + v - □ x

两个列表不相同
Press any key to continue . . . |

图 11: 实例

1.11 将列表 [3,0,8,5,7] 中大于 5 的元素置为 1, 其余元素置为 0

先定义一个列表 `a=[3,0,8,5,7]`, 用 `for` 循环和 `[]` 遍历整个列表中的元素。并用 `if` 语句判断, 若大于 5, 则将元素赋值为 1, 否则为 0。

```
#将列表 [3, 0, 8, 5, 7] 中大于 5 元素置为1, 其余元素置为0
a = [3, 0, 8, 5, 7]
for i in range(len(a)):
    if a[i] > 5:
        a[i] = 1
    else:
        a[i] = 0
print(a)
```

```
C:\Program Files\WindowsAp  x  +  v  -  □  x
[0, 0, 1, 0, 1]
Press any key to continue . . . |
```

图 12: 实例

1.12 向字典 ‘Alice’ : 20, ‘Beth’ : 18, ‘Cecil’ : 21 中追加 ‘David’ :19 键值对, 更新 Cecil 的值为 17

定义字典需要用, 先定义一个字典 a = ‘Alice’:20,’Beth’:18,’Cecil’:21, 不管是追加键值对还是修改本来存在的值, 都要用到 []。最后用 print 打印修改后的字典查看结果。

```
#向字典 { 'Alice' : 20, 'Beth' : 18, 'Cecil' : 21} 中追加 'David' :19 键值对, 更新Cecil的值为17
a = {'Alice':20,'Beth':18,'Cecil':21}
a['David'] = 19
a['Cecil'] = 17
print(a)
```

```
C:\Program Files\WindowsAp  x  +  v  -  □  x
{'Alice': 20, 'Beth': 18, 'Cecil': 17, 'David': 19}
Press any key to continue . . . |
```

图 13: 实例

1.13 以列表 [‘A’ , ‘B’ , ‘C’ , ‘D’ , ‘E’ , ‘F’ , ‘G’ , ‘H’] 中的每一个元素为键, 默认值都是 0, 创建一个字典

首先定义一个列表, a=[‘A’ , ‘B’ , ‘C’ , ‘D’ , ‘E’ , ‘F’ , ‘G’ , ‘H’] 和一个字典 b, 使用 for 循环, 在循环中用 b[i]=0, 最后用 print 打印字典 b, 就可以得到想要的结果。


```
#以列表 ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'] 中的每一个元素为键，默认值都是0，创建一个字典
a = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
b = {}
for i in a:
    b[i] = 0
print("创建的字典: ")
print(b)
```

C:\Program Files\WindowsAp x + | v - □ x

创建的字典:
{ 'A': 0, 'B': 0, 'C': 0, 'D': 0, 'E': 0, 'F': 0, 'G': 0, 'H': 0}
Press any key to continue . . . |

图 14: 实例

1.14 创建一个空集合，增加 ‘x’，‘y’，‘z’ 三个元素

用 set() 创建一个空集合（不能用创建），再利用 update() 函数将 x,y,z 三个元素添加到新创建的空集合中，再用 print 打印结果。

```
#创建一个空集合，增加 ['x', 'y', 'z'] 三个元素
a = set() #空集合只能用set()创建
b = ['x', 'y', 'z']
a.update(b)
print(a)
```

C:\Program Files\WindowsAp x + | v - □ x

{ 'y', 'z', 'x' }
Press any key to continue . . . |

图 15: 实例

1.15 利用 PIL 截取一张图像的一部分，然后将截取的部分旋转 180 度后，粘贴回图像原来的位置

首先需要下载 PIL，下载后输入语句 from PIL import Image，用 open() 打开图像，然后使用 crop() 方法，从一幅图像中裁剪指定区域。在将指定区域旋转后，使用 paste() 方法将该区域放回去，最后用 show()，来查看变换后的结果。

```
from PIL import Image
pil_im = Image.open('图片2.jpg')

#裁剪指定区域
box = (100, 100, 400, 400)
region = pil_im.crop(box)

#将裁剪的区域放回
region = region.transpose(Image.ROTATE_180)
pil_im.paste(region, box)

pil_im.show()
```

图 16: 代码



图 17: 原图



图 18: 修改后

1.16 用 PIL 绘制图像点和线

先读取一个图像到数组中，再使用 `imshow()` 来绘制图像，然后取一些点，使用 `plot(x,y,'r*')` 语句来表示用红色星状标记绘制点，`plot(x[:2],y[:2])` 表示绘制前两个点的线。最后用 `title()` 添加标题，显示出绘制后的图像。

```
from PIL import Image
from pylab import *
# 读取图像到数组中
im = array(Image.open('图片2.jpg'))
# 绘制图像
imshow(im)
# 一些点
x = [100, 100, 400, 400]
y = [200, 500, 200, 500]
# 使用红色星状标记绘制点
plot(x, y, 'r*')
# 绘制连接前两个点的线
plot(x[:2], y[:2])
# 添加标题，显示绘制的图像
title('Plotting: "empire.jpg"')
show()
```

图 19: 代码

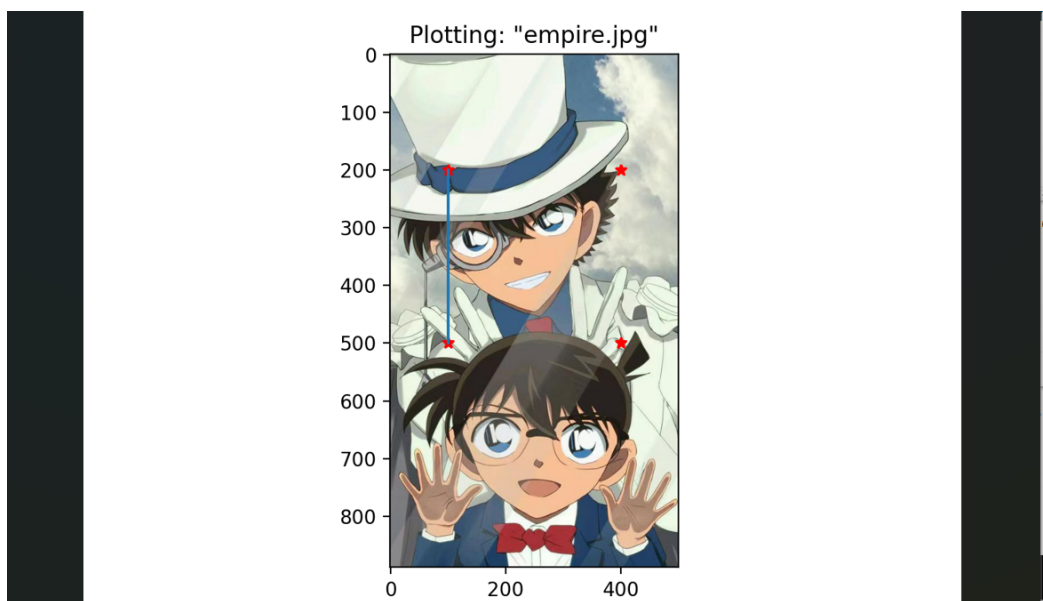


图 20: 绘制后

1.17 绘制一个图像的等轮廓线和直方图

需要先读取一个图片到数组中，读取时需要将图像灰度化。然后新建一个图像，将轮廓图像在原点的左上角显示。在绘制直方图时可以使用 `hist()` 函数绘制，不过要注意，`hist()` 只接受一维数组作为输入，所以在绘制前，必须先用 `flatten()` 方法将任意数组按照行优先准则转换成一维数组，对图像进行压平处理后才能进行接下来的操作。

```

from PIL import Image
from pylab import *
# 读取图像到数组中
im = array(Image.open('图片4.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()

```

图 21: 代码

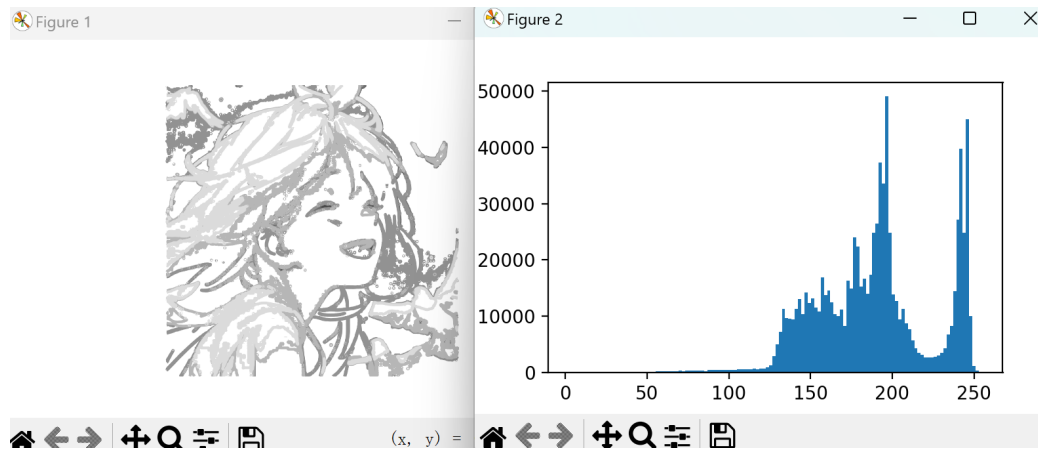


图 22: 等轮廓线和直方图

1.18 绘制一幅图像，在绘图窗口的图像区域点击三次，程序将这些点击的坐标 $[x, y]$ 自动保存在 x 列表里。

为了实现交互式标注，需要用到 PyLab 库中的 `ginput()` 函数，需要有 `from pylab import *`，然后用 `x=ginput(3)` 来实现点击三次，最后输出结果。

```
from PIL import Image
from pylab import *
im = array(Image.open('图片2.jpg'))
imshow(im)
print('Please click 3 points')
x = ginput(3)
print('you clicked:',x)
show()
```

C:\Program Files\WindowsAp x + v - □ x

Please click 3 points
you clicked: [(np.float64(274.75568181818176), np.float64(346.8257575757576)), (np.float64(425.08712121212113), np.float64(344.4204545454545)), (np.float64(456.3560606060605), np.float64(580.1401515151515))]

图 23: 实例

1.19 尝试输出图像数据信息

用 `im.shape`, `im.dtype` 可以输出图像数组的大小, 以及图像数组元素的数据类型

```
from PIL import Image
from numpy import *
im = array(Image.open('图片1.jpg'))
print(im.shape, im.dtype)
im = array(Image.open('图片1.jpg').convert('L'),'f')
print(im.shape, im.dtype)
```

C:\Program Files\WindowsAp x + v - □ x

(1082, 500, 3) uint8
(1082, 500) float32
Press any key to continue . . . |

图 24: 实例

1.20 将一张灰度图像的像素值变换到 100...200 区间, 并查看此时图像中的最小和最大像素值

首先用 `convert('L')` 将图像变灰, 然后将图像的像素值变换区间, 最后用 `min()` 和 `max()` 求出图像的最大和最小像素值, 并用 `print` 输出结果。

```
from PIL import Image
from numpy import *
im = array(Image.open('图片1.jpg').convert('L'))
im3 = (100.0/255) * im + 100 # 将图像像素值变换到 100...200 区间
print (int(im3.min()), int(im3.max()))
```

C:\Program Files\WindowsAp x + v - □ x

100 200
Press any key to continue . . . |

图 25: 实例

2 实验结果

实验链接: <https://github.com/XY568/tired.git>

3 心得体会

对命令行环境的学习让我初步了解了后台运行, 终端多路复用, 起别名, 配置文件等操作, 对计算机有了更加全面的认识, 也见识到了一些新的命令和东西, 对命令行环境的学习让我感受到自己依旧有着许多知识上的欠缺, 激发了今后的热情。虽然刚开始接触 Python, 但我已经感受到了它的简洁与方便。Python 中有许多的函数, 帮助我们实现各种功能, 如果能够充分利用它本身的函数, 可以快速解决问题, 提升工作效率。除此之外, Python 中的许多语句通俗易懂, 我在学习的过程中, 能够较为轻松的掌握基础的语法, 这也使得 Python 的应用更加简单方便。在 Python 的视觉应用中, 我学会了利用 numpy, scipy 等库对图像进行修改, 这种对图像编辑的方法更加简单, 容易操作, 能显著提高图像的编辑和修改效率, 是一个难得的处理图像的工具。

无论是命令行环境还是 Python 及其应用, 在对它们的学习中, 我都看到了自己今后还有很长的路要走, 还有许多知识没有掌握。未来我也会继续学习它们, 努力熟练应用每一个可以提高工作效率的工具, 逐步加强学习, 提升自我。