



中国海洋大学

实验报告

——命令行环境与 python 基础及视觉应用的学习

学号： 23060021010

姓名： 郭晓伟

班级： 23 级软件工程五八班

1 实验要求

- 1.1 学习命令行环境的任务控制，终端多路复用，别名，配置文件，远端设备
- 1.2 学习 Python 的基础入门及其视觉应用
- 1.3 完成 4 个课堂练习与 20 个与命令行环境和 Python 有关的实例

2 实验内容

2.1 命令行环境的学习

- 2.1.1 任务控制是指在命令行环境下管理和控制任务的执行状态（前台、后台、暂停等）。常用的任务控制命令有：

Ctrl + C: 终止当前前台任务。

Ctrl + Z: 暂停当前前台任务并将其放到后台。

jobs: 列出当前所有的后台任务。

fg [任务号]: 将后台任务恢复到前台执行。

bg [任务号]: 在后台继续执行暂停的任务。

kill [任务号或 PID]: 终止指定的后台任务或进程。

- 2.1.2 终端多路复用是指在一个终端中管理多个会话或窗口，允许用户在单一终端窗口中并行运行多个会话。最常用的多路复用工具包括 tmux 和 screen。

tmux new -s <session-name>: 创建一个新的 tmux 会话。

Ctrl + b + d: 分离当前 tmux 会话（后台运行）。

tmux ls: 列出所有 tmux 会话。

tmux attach -t <session-name>: 恢复连接到指定的 tmux 会话。

多路复用非常适合在远程会话中保持任务持续运行，即使断开连接，任务依然在后台执行。

- 2.1.3 别名是为命令设置的快捷方式，用于简化复杂或常用命令的输入。可以通过在 shell 配置文件（如 ~/.bashrc 或 ~/.zshrc）中定义别名。

配置文件用于设置命令行环境的个性化和自动化配置。常见的配置文件包括：

~/.bashrc: Bash shell 的配置文件，通常用于定义别名、环境变量和自定义函数。

~/.zshrc: Zsh shell 的配置文件，类似于 .bashrc。

~/.bash_profile 或 ~/.bash_login: 在用户登录时执行的配置文件，通常用于设置环境变量。

~/.vimrc: Vim 文本编辑器的配置文件，用于自定义编辑器的行为和外观。

2.2 Python 基础入门及计算机视觉应用

2.2.1 Python 是一种高级的、解释型、面向对象的编程语言，由 Guido van Rossum 于 1991 年发布。由于其语法简洁、可读性强和庞大的标准库，Python 已成为应用广泛的编程语言，特别是在数据分析、人工智能、Web 开发、自动化等领域。

2.2.2 python 的特点：

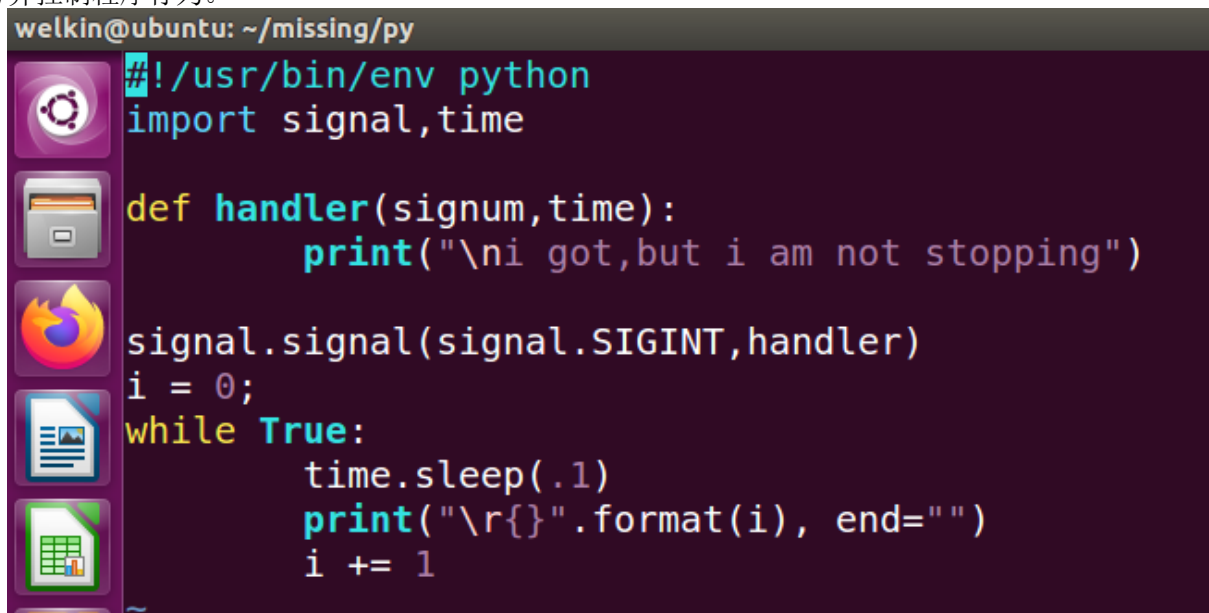
1. 简洁易学：Python 语法简单，代码可读性强。它允许开发者用更少的代码完成同样的任务，因此非常适合初学者学习编程。
2. 跨平台：Python 是跨平台的，支持 Windows、MacOS、Linux 等操作系统，Python 代码在不同的操作系统上几乎不需要更改。
3. 解释型语言：Python 是解释型语言，代码可以逐行运行，无需编译，方便调试和快速开发。
4. 丰富的库和社区支持：Python 拥有丰富的标准库和第三方库，几乎可以实现任何应用场景的需求。比如科学计算的 NumPy 和 SciPy，数据分析的 Pandas，机器学习的 TensorFlow，Web 开发的 Django 等。
5. 强大的集成性：Python 能够轻松与其他语言（如 C、C++、Java）进行集成，支持扩展和调用外部库，适合多种应用开发。

2.2.3 Python 在计算机视觉（Computer Vision, CV）领域具有广泛的应用，得益于其简洁的语法、强大的库支持以及活跃的社区。计算机视觉旨在让计算机能够“看见”并“理解”图像或视频内容，从而执行各种任务，如图像识别、对象检测、图像分割等。常用的 Python 计算机视觉库有 OpenCV, TensorFlow 和 PyTorch 等。

3 实例练习

3.1 编写捕获 SIGINT 信号的 py 脚本

定义信号处理函数 handler，将 SIGINT 信号处理程序设置为自定义的 handler 函数，实现了捕获此信号并控制程序行为。



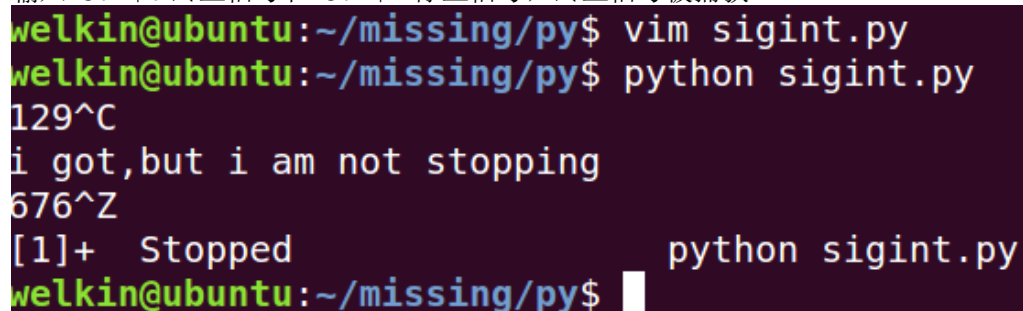
```
welkin@ubuntu: ~/missing/py
#!/usr/bin/env python
import signal,time

def handler(signum,time):
    print("\ni got,but i am not stopping")

signal.signal(signal.SIGINT,handler)
i = 0;
while True:
    time.sleep(.1)
    print("\r{}".format(i), end="")
    i += 1
```

3.2 任务控制

输入 Ctrl+c 终止信号和 Ctrl+z 停止信号，终止信号被捕获



```
welkin@ubuntu:~/missing/py$ vim sigint.py
welkin@ubuntu:~/missing/py$ python sigint.py
129^C
i got,but i am not stopping
676^Z
[1]+  Stopped                  python sigint.py
welkin@ubuntu:~/missing/py$
```

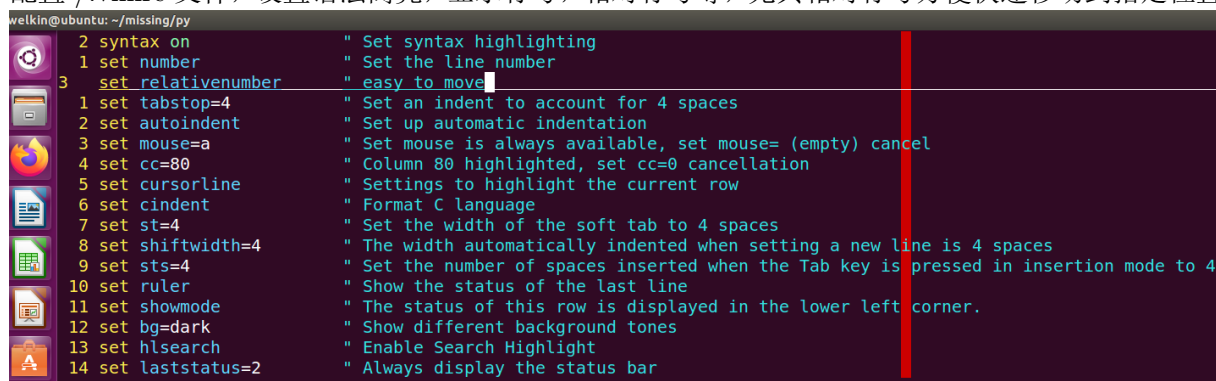
3.3 创建别名

利用 alias 命令，创建别名，mkd=mkdir

```
welkin@ubuntu:~/missing/py$ alias mkd=mkdir
welkin@ubuntu:~/missing/py$ mkd a
welkin@ubuntu:~/missing/py$ ll
total 16
drwxrwxr-x 3 welkin welkin 4096 Sep 11 21:05 ./
drwxrwxr-x 5 welkin welkin 4096 Sep  6 09:43 ../
drwxrwxr-x 2 welkin welkin 4096 Sep 11 21:05 a/
-rw-rw-r-- 1 welkin welkin  222 Sep  6 10:29 sigint.py
welkin@ubuntu:~/missing/py$
```

3.4 配置 vimrc 文件

配置 `./vimrc` 文件，设置语法高亮，显示行号，相对行号等，尤其相对行号方便快速移动到指定位置



```
welkin@ubuntu: ~/missing/py
2 syntax on          " Set syntax highlighting
1 set number         " Set the line number
3 set relativenumber " easy to move
1 set tabstop=4      " Set an indent to account for 4 spaces
2 set autoindent     " Set up automatic indentation
3 set mouse=a        " Set mouse is always available, set mouse= (empty) cancel
4 set cc=80          " Column 80 highlighted, set cc=0 cancellation
5 set cursorline     " Settings to highlight the current row
6 set cindent        " Format C language
7 set st=4           " Set the width of the soft tab to 4 spaces
8 set shiftwidth=4   " The width automatically indented when setting a new line is 4 spaces
9 set sts=4          " Set the number of spaces inserted when the Tab key is pressed in insertion mode to 4
10 set ruler         " Show the status of the last line
11 set showmode      " The status of this row is displayed in the lower left corner.
12 set bg=dark       " Show different background tones
13 set hlsearch      " Enable Search Highlight
14 set laststatus=2  " Always display the status bar
```

3.5 python 计算日期

输入某年某月某日，判断这一天是这一年的第几天，代码及运行结果如下

```
year = int(input('year:\n'))
month = int(input('month:\n'))
day = int(input('day:\n'))

months = (0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334)
if 0 < month <= 12:
    sum = months[month - 1]
else:
    print('data error')
sum += day
leap = 0
if (year % 400 == 0) or ((year % 4 == 0) and (year % 100 != 0)):
```

```

    leap = 1
if (leap == 1) and (month > 2):
    sum += 1
print('it is the %dth day.' % sum)

```

```

C:\Users\23241\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\23241\PycharmProjects\pythonProject\day.py
year:
2024
month:
9
day:
12
it is the 256th day.
Process finished with exit code 0

```

3.6 斐波那契数列

使用递归的方法输出斐波那契数列（展示第 11 个）

```

def fib(n):
    if n == 1 or n == 2:
        return 1
    return fib(n - 1) + fib(n - 2)

print(fib(11))

```

```

C:\Users\23241\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\23241\PycharmProjects\pythonProject\fibonacci.py
89

```

3.7 乘法口诀表

使用 python 的 for 循环输出乘法表

```

for i in range(1, 10):
    print()
    for j in range(1, i+1):
        print ("%d*%d=%d" % (i, j, i*j), end=" ")

```

```

C:\Users\23241\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\23241\PycharmProjects\pythonProject\table.py
1*1=1
2*1=2 2*2=4
3*1=3 3*2=6 3*3=9
4*1=4 4*2=8 4*3=12 4*4=16
5*1=5 5*2=10 5*3=15 5*4=20 5*5=25
6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81

```

3.8 停顿输出

使用 time 模块的 sleep() 函数，实现停顿一秒再输出

```
import time

myD = {1: 'a', 2: 'b'}
for key, value in dict.items(myD):
    print(key, value)
    time.sleep(1)
```

```
C:\Users\23241\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\23241\PycharmProjects\pythonProject\sleep.py
1 a
2 b
```

3.9 输出素数

输出 201 到 301 的素数，方法为分别用 2 到此数的平方根去除此数，若能整除则不是，反之为素数

```
h = 0
leap = 1
from math import sqrt
from sys import stdout
for m in range(201,301):
    k = int(sqrt(m + 1))
    for i in range(2,k + 1):
        if m % i == 0:
            leap = 0
            break
    if leap == 1:
        print ('%-4d' % m)
        h += 1
    leap = 1
print ('The total is %d' % h)
```

```
C:\Users\23241\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\23241\PycharmProjects\pythonProject\isPrime.py
211
223
227
229
233
239
241
251
257
```

3.10 反顺序打印字符

利用递归函数调用方式反顺序输出

```
def output(s, l):
    if l == 0:
        return
    print(s[l - 1], end=" ")
    output(s, l - 1)

s = input('input:')
l = len(s)
output(s, l)
```

```
C:\Users\23241\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\23241\PycharmProjects\pythonProject\reverse.py
input:hello
olleh
```

3.11 选择排序

采用选择法，依次选择最小的数交换，以此类推

```
l = [9,5,2,8,3,1,7,3]
N = len(l)

for i in range(N - 1):
    min = i
    for j in range(i + 1, N):
        if l[min] > l[j]: min = j
    l[i], l[min] = l[min], l[i]
print('排序后: ')
for i in range(N):
```



```
print(l[i])
```

```
C:\Users\23241\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\23241\PycharmProjects\pythonProject\sort.py
排序后:
1
2
3
3
5
7
8
9
```

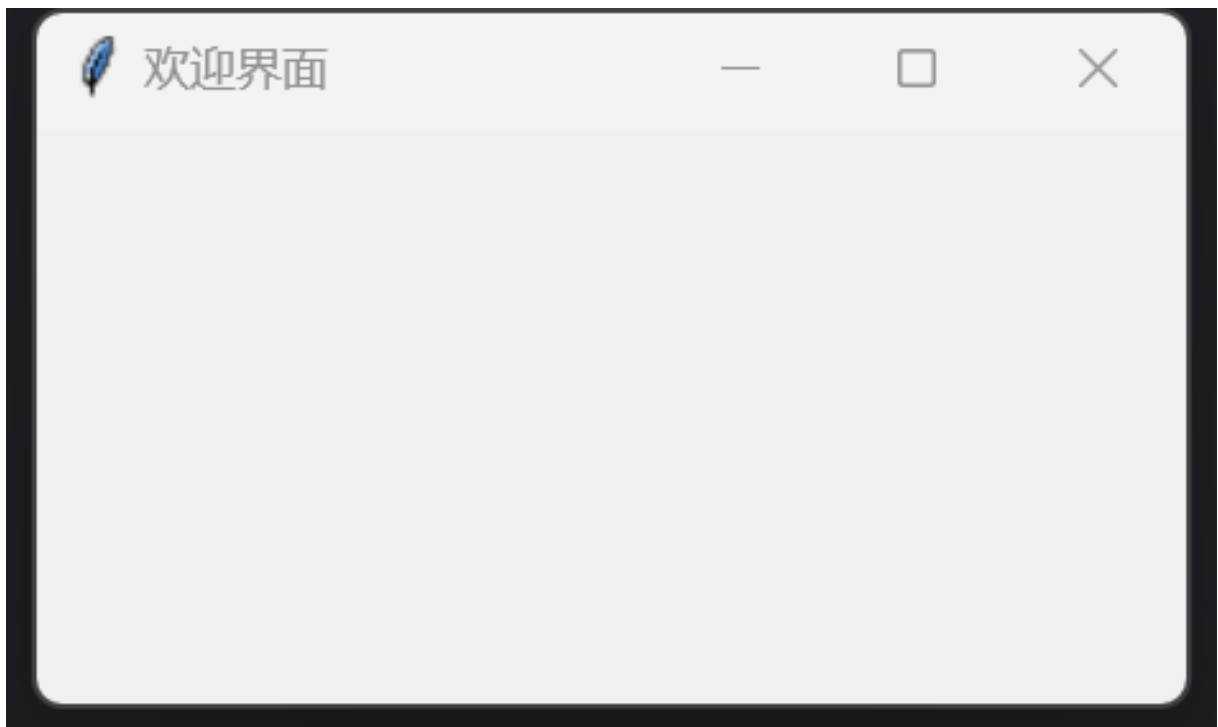
3.12 实现简单 GUI 程序

导入 Tkinter 模块。创建主窗口，在主窗口中添加组件（如按钮、标签、文本框等），设置布局和事件处理程序，启动主事件循环（mainloop()），从而构建简单的 GUI 应用

```
import tkinter as tk

root = tk.Tk()
root.title("欢迎界面")
root.geometry("300x150")

root.mainloop()
```



3.13 在 GUI 应用中实现按钮等

导入 Tkinter 模块。创建主窗口，在主窗口中添加组件（如按钮、标签、文本框等），设置布局 and 事件处理程序，启动主事件循环（mainloop()），从而构建简单的 GUI 应用

```
import tkinter as tk

def greet():
    user_name = entry.get()
    label.config(text=f"Hello, {user_name}!")

root = tk.Tk()
root.title("欢迎界面")
root.geometry("300x150")

label = tk.Label(root, text="请输入你的名字:", font=("Arial", 12))
label.pack(pady=10)

entry = tk.Entry(root)
entry.pack(pady=5)

button = tk.Button(root, text="确认", command=greet)
button.pack(pady=5)

root.mainloop()
```



3.14 求最大公约数

采用欧几里得算法，计算最大公约数

```
def gcd(a, b):  
    while b:  
        a, b = b, a % b  
    return a  
  
print(gcd(48, 18)) # 输出 6
```

3.15 判断回文字符串

利用 python 的切片操作，生成字符串的反转版本

```
def is_palindrome(s):  
    return s == s[::-1]  
  
print(is_palindrome("racecar")) # 输出: True  
print(is_palindrome("hello"))  # 输出: False
```

3.16 列表去重

利用 set 的特性，和其高效的查找操作，检测重复的元素

```
def remove_duplicates(lst):
    seen = set()
    unique_lst = []
    for item in lst:
        if item not in seen:
            unique_lst.append(item)
            seen.add(item)
    return unique_lst

print(remove_duplicates([1, 2, 3, 2, 1, 4, 5]))
# 输出: [1, 2, 3, 4, 5]
```

3.17 统计字符出现次数

利用字典和其 get() 方法统计字符的出现次数

```
def ccount(s):
    count = {}
    for char in s:
        count[char] = count.get(char, 0) + 1
    return count

print(ccount("hello world"))
# 输出: {'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'w': 1, 'r': 1, 'd': 1}
```

3.18 冒泡排序

使用冒泡排序对数字进行排序

```
def sort(lst):
    n = len(lst)
    for i in range(n):
        for j in range(0, n-i-1):
```

```
        if lst[j] > lst[j+1]:
            lst[j], lst[j+1] = lst[j+1], lst[j]

    return lst

print(sort([64, 34, 25, 12, 22, 11, 90]))
# 输出: [11, 12, 22, 25, 34, 64, 90]
```

3.19 合并两个有序列表

编写 merge 函数，合并两个有序列表并保持排序

```
def merge(list1, list2):
    return sorted(list1 + list2)

print(merge([1, 3, 5], [2, 4, 6]))
# 输出: [1, 2, 3, 4, 5, 6]
```

3.20 连续子数组的最大和

使用卡丹算法，关键点在于判断对于当前元素，是加上前面的子数组后和最大，还是单独成为新的子数组最大，从而不断更新最大和

```
def max_sum(nums):
    # 初始化两个变量
    max_current = nums[0] # 当前子数组的最大和
    max_global = nums[0] # 全局最大和

    # 从第二个元素开始遍历数组
    for i in range(1, len(nums)):
        # 对于当前元素，判断是加上前面的子数组和更大，还是单独成为新的子数组更大
        max_current = max(nums[i], max_current + nums[i])

        # 更新全局最大和
        if max_current > max_global:
            max_global = max_current
```

```
    return max_global

nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]
print(max_sum(nums)) # 输出: 6
```

4 实验收获与感悟

在学习命令行环境、任务控制、终端多路复用、别名、配置文件、远端设备与 Python 基础入门及其视觉应用后，我对工具的灵活性和高效性有了更深的理解。命令行是管理系统的关键，通过任务控制命令（如 ‘fg’、‘bg’、‘kill’），我学会了如何暂停、恢复和终止进程，体会到了在复杂系统中掌控任务的能力。同时，终端多路复用工具如 ‘tmux’ 极大提升了操作效率，尤其在远程工作或系统管理中，它允许我在不中断工作的情况下同时管理多个会话，减少了上下文切换的时间浪费。

别名和配置文件是日常操作效率的倍增器，通过自定义 ‘.bashrc’ 或 ‘.vimrc’ 文件，我能够简化复杂命令的执行并优化工作环境。在学习的过程中，我意识到一个高效的开发环境不仅仅依赖于工具的使用，更在于合理的配置和定制，这有助于长时间提高工作效率。与此同时，学习远程设备管理让我掌握了 SSH、SCP 等工具，能够无缝管理和维护远程设备，确保远程会话的安全性和稳定性。

在 Python 编程的学习中，我感受到了其简单易用性以及多个领域的广泛应用，特别是在视觉应用方面，如使用 ‘matplotlib’ 和 ‘OpenCV’ 进行图像处理和数据可视化。通过掌握 Python 的基础数据结构和控制流，我不仅能够解决问题，还能在实践中理解编程与现实应用的结合。整体学习过程让我意识到，技术学习不仅仅是掌握工具，更是培养解决实际问题的能力，而实践在这个过程中起到了至关重要的作用。

Github 仓库链接: <https://github.com/xwelkin/lab.git>