



中国海洋大学

---

# 实验报告

计算机科学与技术

23020007034-郭长挺

<https://github.com/luoshenzjtywxw/-.git>

## 1 Python 入门基础

### 1.1 练习

冒泡排序：

```
def bubble_sort(arr):  
    n = len(arr)  
    for i in range(n):  
        for j in range(0, n-i-1):  
            if arr[j] > arr[j+1]:  
                arr[j], arr[j+1] = arr[j+1], arr[j] #元组交换语法  
    return arr
```

#示例

```
array = [64, 34, 25, 12, 28, 11, 90]  
sorted_array = bubble_sort(array)  
print(f"排序后的数组：{sorted_array}")
```

```
排序后的数组：[11, 12, 25, 28, 34, 64, 90]
```

```
进程已结束，退出代码为 0
```

斐波那契数列：

```
def cal(n):  
    if n==1:  
        return 1  
    elif n==2:  
        return 1  
    elif n>=3:  
        return cal(n-1)+cal(n-2)
```

#示例

---

---

```
num = int(input("请输入一个整数: "))
print(f"{cal(num)} ")
```

```
请输入一个整数: 10
```

```
55
```

```
进程已结束，退出代码为 0
```

质数判断:

```
def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    i = 2
    while i*i<=n:
        if n%i==0:
            return False
        i +=1
    return True
#示例
num = int(input("请输入一个整数: "))
if is_prime(num):
    print(f"{num} 是质数")
else:
    print(f"{num} 不是质数")
```

```
请输入一个整数: 6
```

```
6 不是质数
```

```
进程已结束，退出代码为 0
```

二分查找:

```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1
#示例
```

---

---

```
print(binary_search([1, 2, 3, 4, 5, 6, 7, 8, 9], 8))
```

```
7
```

```
进程已结束，退出代码为 0
```

选择排序：

```
def selection_sort(arr):
    # 遍历数组的每个元素
    for i in range(len(arr)):
        # 假设当前索引的元素是最小的
        min_index = i
        # 遍历未排序部分的元素
        for j in range(i + 1, len(arr)):
            # 如果找到更小的元素，则更新最小元素的索引
            if arr[j] < arr[min_index]:
                min_index = j
        # 交换当前元素和找到的最小元素
        arr[i], arr[min_index] = arr[min_index], arr[i]

# 示例
if __name__ == "__main__":
    array = [64, 25, 12, 22, 11]
    print("原始数组:", array)
    selection_sort(array)
    print("排序后的数组:", array)
```

```
原始数组: [64, 25, 12, 22, 11]
排序后的数组: [11, 12, 22, 25, 64]
```

```
进程已结束，退出代码为 0
```

插入排序：

```
def climb_stairs(n):
    if n <= 2:
        return n
    dp = [0] * (n + 1)
    dp[1] = 1
    dp[2] = 2
    for i in range(3, n + 1):
        dp[i] = dp[i - 1] + dp[i - 2]
    return dp[n]

# 示例
if __name__ == "__main__":
    n = 8
    print(f"爬 {n} 阶楼梯的方法数为: {climb_stairs(n)}")
```

---

---

```
原始数组: [64, 25, 12, 22, 11]
排序后的数组: [11, 12, 22, 25, 64]

进程已结束, 退出代码为 0
```

爬楼梯:

```
def climb_stairs(n):
    if n <= 2:
        return n
    dp = [0] * (n + 1)
    dp[1] = 1
    dp[2] = 2
    for i in range(3, n + 1):
        dp[i] = dp[i - 1] + dp[i - 2]
    return dp[n]

#示例
if __name__ == "__main__":
    n = 8
    print(f"爬 {n} 阶楼梯的方法数为: {climb_stairs(n)}")
```

```
爬 8 阶楼梯的方法数为: 34
```

```
进程已结束, 退出代码为 0
```

## 1.2 心得

python 的语法很灵活, 感觉上手非常快, 适合初学者。没有非常讲究格式, 非常简洁, 提高开发效率和维护性。而且没有主函数这个概念。

## 2 python 视觉应用

### 2.1 练习

改变一个图片的颜色并且保存成另外一种格式:

```
from PIL import Image
im = Image.open('scenery.png')
print(im.format, im.size, im.mode)
r, g, b = im.split()
om = Image.merge('RGB', (b, r, g))
om.save('newscenery.jpg')
```

---



将两张图片合成为一张图片

```
from PIL import Image
```

```
background = Image.open('blackboard.png')
```

```
problem = Image.open('store.png')
```

```
problem = problem.resize((int(problem.size[0] / problem.size[1] * 200), 200))
```

```
width, height = problem.size
```

```
for i in range(0, width):
```

```
    for j in range(0, height):
```

```
        if problem.getpixel((i, j)) != (255, 255, 255):
```

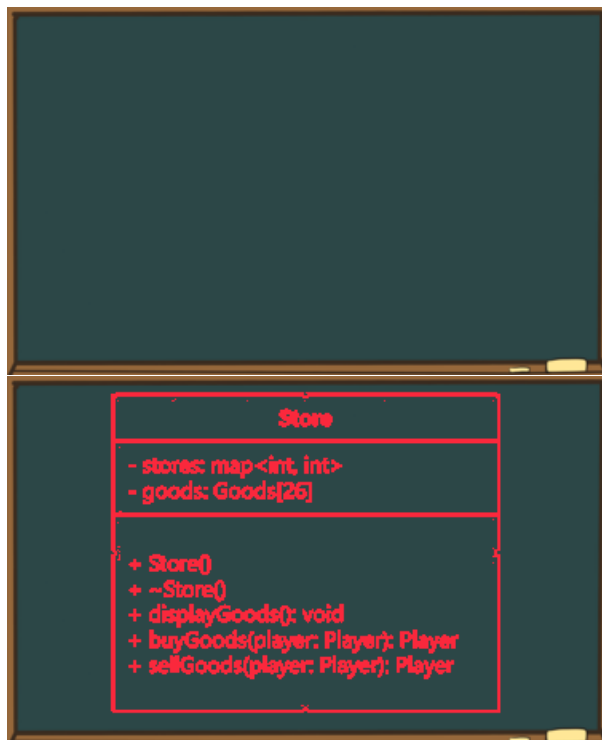
```
            background.putpixel((i + 65, j + 10), (250, 40, 60))
```

```
background.show()
```

```
background.save("union.png")
```

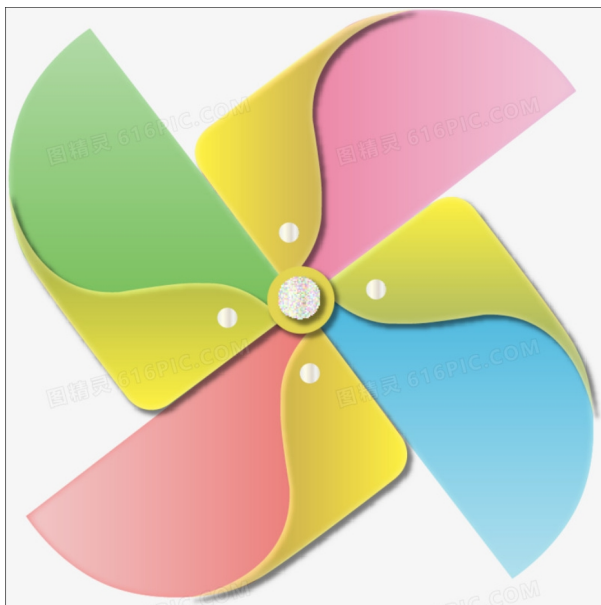
Store
- stores: map<int, int> - goods: Goods[26]
+ Store() + ~Store() + displayGoods(): void + buyGoods(player: Player): Player + sellGoods(player: Player): Player

---



旋转图片:

```
from PIL import Image  
im = Image.open("rotate.png")  
out=im.rotate(90)  
out.save("new_rotate.png")  
out.show()
```



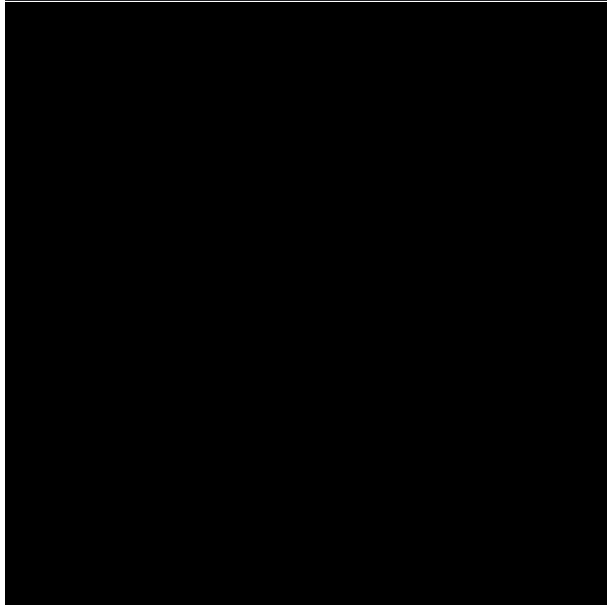


生成纯黑并且打印图片信息：

```
from PIL import Image
image=Image.new('RGBA', (500,500), (0,0,0))
print (image. format, image.size, image.mode)
image.show()
image.save("black.png")
```

None (500, 500) RGBA

进程已结束，退出代码为 0



---

## 2.2 心得

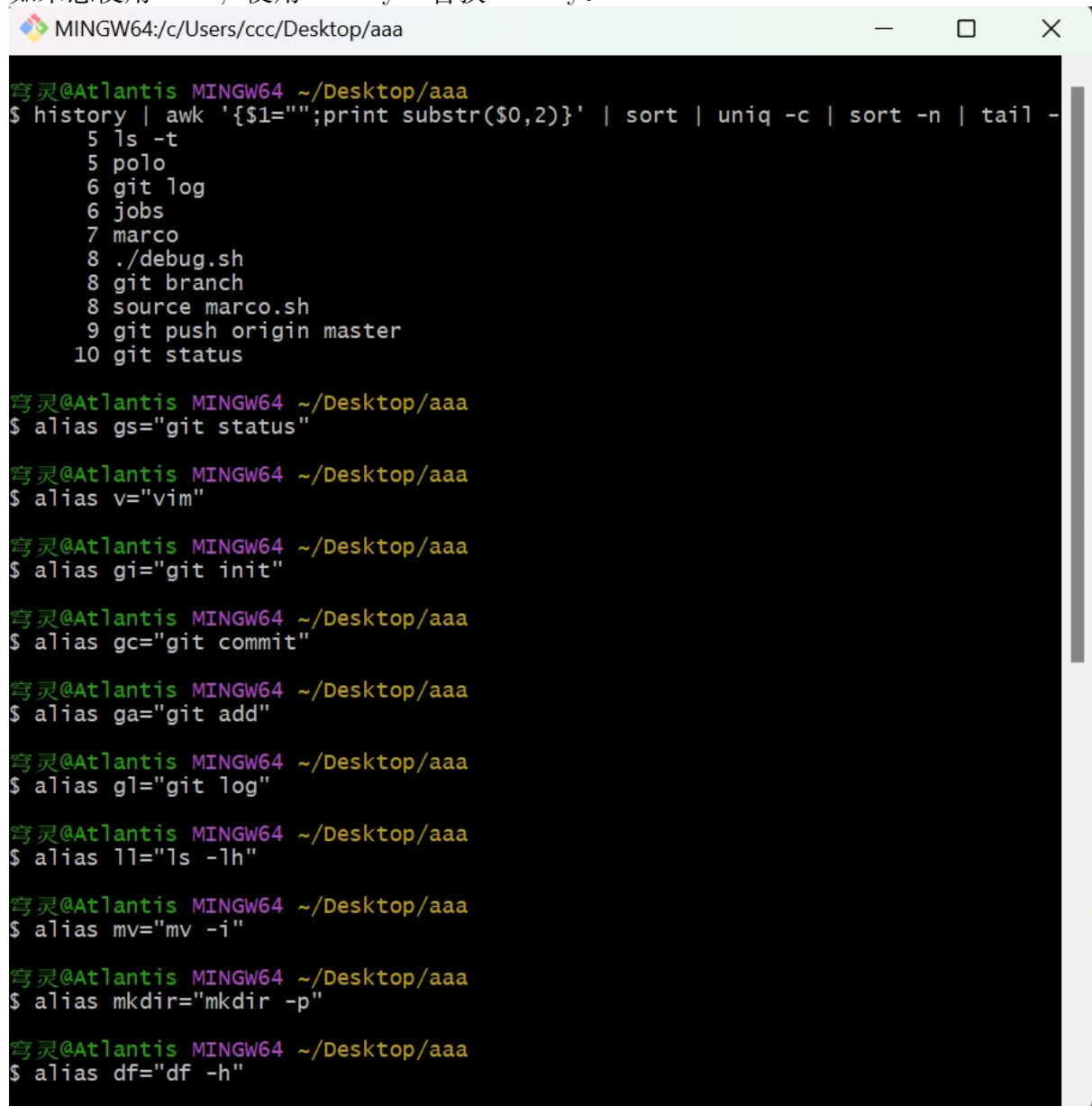
python 的视觉应用感觉非常神奇，也非常快捷，pillow 中提供了非常简单的 API 进行图像处理，适合快速开发和原型设计，支持图像裁剪、调整大小、旋转、滤镜、绘图等操作，同时兼容多种图像格式。

## 3 命令行环境

### 3.1 练习

别名：

1. 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。
2. 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。注意：这个命令只在 Bash 中生效，如果您使用 ZSH，使用 `history 1` 替换 `history`。



```
MINGW64:/c/Users/ccc/Desktop/aaa
穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ history | awk '{1=""}; print substr($0,2)}' | sort | uniq -c | sort -n | tail -
  5 ls -t
  5 polo
  6 git log
  6 jobs
  7 marco
  8 ./debug.sh
  8 git branch
  8 source marco.sh
  9 git push origin master
 10 git status

穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ alias gs="git status"

穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ alias v="vim"

穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ alias gi="git init"

穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ alias gc="git commit"

穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ alias ga="git add"

穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ alias gl="git log"

穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ alias ll="ls -lh"

穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ alias mv="mv -i"

穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ alias mkdir="mkdir -p"

穹灵@Atlantis MINGW64 ~/Desktop/aaa
$ alias df="df -h"
```



```

穹灵@Atlantis MINGW64 ~/Desktop/aaa (master)
$ ga a.txt

穹灵@Atlantis MINGW64 ~/Desktop/aaa (master)
$ gc -m "提交了a.txt" a.txt
[master (root-commit) 53cef1f] 提交了a.txt
1 file changed, 109 insertions(+)
create mode 100644 a.txt

穹灵@Atlantis MINGW64 ~/Desktop/aaa (master)
$ gl
commit 53cef1fcf34dd117e25c28fef60449a37523887f (HEAD -> master)
Author: luoshen <2021820292@qq.com>
Date: Sun Sep 8 22:33:04 2024 +0800

    提交了a.txt

```

```

MINGW64:/c/Users/ccc
穹灵@Atlantis MINGW64 ~/Desktop/aaa (master)
$ alias dc=cd

穹灵@Atlantis MINGW64 ~/Desktop/aaa (master)
$ dc

穹灵@Atlantis MINGW64 ~
$ |

```

配置文件让我们帮助您进一步学习配置文件：

1. 为您的配置文件新建一个文件夹，并设置好版本控制
2. 在其中添加至少一个配置文件，比如说您的 shell，在其中包含一些自定义设置（可以从设置 \$PS1 开始）。
3. 建立一种在新设备进行快速安装配置的方法（无需手动操作）。最简单的方法是写一个 shell 脚本对每个文件使用 `ln -s`，也可以使用专用工具
4. 在新的虚拟机上测试该安装脚本。
5. 将您现有的所有配置文件移动到项目仓库里。
6. 将项目发布到 GitHub。

```

穹灵@Atlantis MINGW64 ~/Desktop
$ mkdir dotfiles

穹灵@Atlantis MINGW64 ~/Desktop
$ git init dotfiles
Initialized empty Git repository in C:/Users/ccc/Desktop/dotfiles/.git/

穹灵@Atlantis MINGW64 ~/Desktop
$ cd dotfiles

穹灵@Atlantis MINGW64 ~/Desktop/dotfiles (master)
$ git config --global core.autocrlf true

穹灵@Atlantis MINGW64 ~/Desktop/dotfiles (master)
$ git add .

穹灵@Atlantis MINGW64 ~/Desktop/dotfiles (master)
$ git commit -m "提交了配置文件" .vimrc
warning: in the working copy of '.vimrc', LF will be replaced by CRLF the next time Git touches it
[master (root-commit) 2b5653f] 提交了配置文件
1 file changed, 80 insertions(+)
create mode 100644 .vimrc

```

---

 **luoshenzjtywxw** 提交了配置文件

a184ad3 · 10 minutes ago

 **1 Commit**

 **.vimrc**

提交了配置文件

10 minutes ago

## 3.2 心得

命令行环境的内容相当丰富，可以通过命令行控制远端的服务器，控制计算机任务的运行，还可以依据自己的工作习惯进行个性化定制。

tmux 是一个相当有用的工具，可以允许我们基于面板和标签分割出多个终端窗口，这样就可以同时与多个 shell 会话进行交互。