



中国海洋大学

# 系统开发工具基础课程实验报告

命令行环境, Python 入门基础, Python 视觉基础

Command line environment, Python introduction  
basics, Python visual basics

学生姓名 涂钦凯

学生学号 23170001085

专业班级 计算机科学与技术 1 班

指导老师 周小伟, 范浩

所在院系 信息科学与工程学部

---

# 命令行环境, Python 入门基础, Python 视觉基础

## 摘 要

本次实验掌握命令行环境的基本操作, 熟悉 Python 编程语言的基础知识, 并应用 Python 进行图像处理和视觉分析。通过实验, 学会如何在命令行中操作 Python 脚本, 理解 Python 的基本语法, 及使用 Python 处理和分析图像数据。

**关键词:** 命令行环境; Python 入门基础; Python 视觉基础

---

---

# **Command line environment, Python introduction basics, Python visual basics**

## **Abstract**

This experiment aims to master the basic operations of the command-line environment, become familiar with the fundamentals of the Python programming language, and apply Python for image processing and visual analysis. Through the experiment, students will learn how to operate Python scripts from the command line, understand Python's basic syntax, and use Python to process and analyze image data.

**Keywords:** Command line environment; Python introduction basics; Python visual basics

---

---

# 目 录

摘 要·····	I
Abstract·····	II
第 1 章 命令行环境·····	1
1.1 使用 pgrep 和 pkill 高效管理和终止后台进程·····	1
1.2 使用 wait 命令在进程结束后启动下一个任务·····	2
1.3 终端多路复用·····	2
第 2 章 Python 入门基础·····	5
2.1 基本操作和实例·····	5
第 3 章 Python 视觉应用·····	7
3.1 图片旋转·····	7
3.2 转换图像格式·····	8
3.3 创建缩略图·····	8
3.4 绘制图像、点和线·····	9
3.5 图像缩放·····	10
参考文献·····	10
附录 A 附录·····	12

---

---

# 图目录

图 1-1	.....	1
图 1-2	.....	2
图 1-3	.....	2
图 1-4	.....	3
图 1-5	.....	3
图 1-6	.....	3
图 1-7	.....	4
图 2-1	.....	5
图 2-2	.....	5
图 2-3	.....	5
图 2-4	.....	6
图 2-5	.....	6
图 2-6	.....	6
图 3-1	.....	7
图 3-2	.....	7
图 3-3	.....	8
图 3-4	.....	8
图 3-5	.....	9
图 3-6	.....	9
图 3-7	.....	10

---

## 表目录

## 第 1 章 命令行环境

### 1.1 使用 pgrep 和 pkill 高效管理和终止后台进程

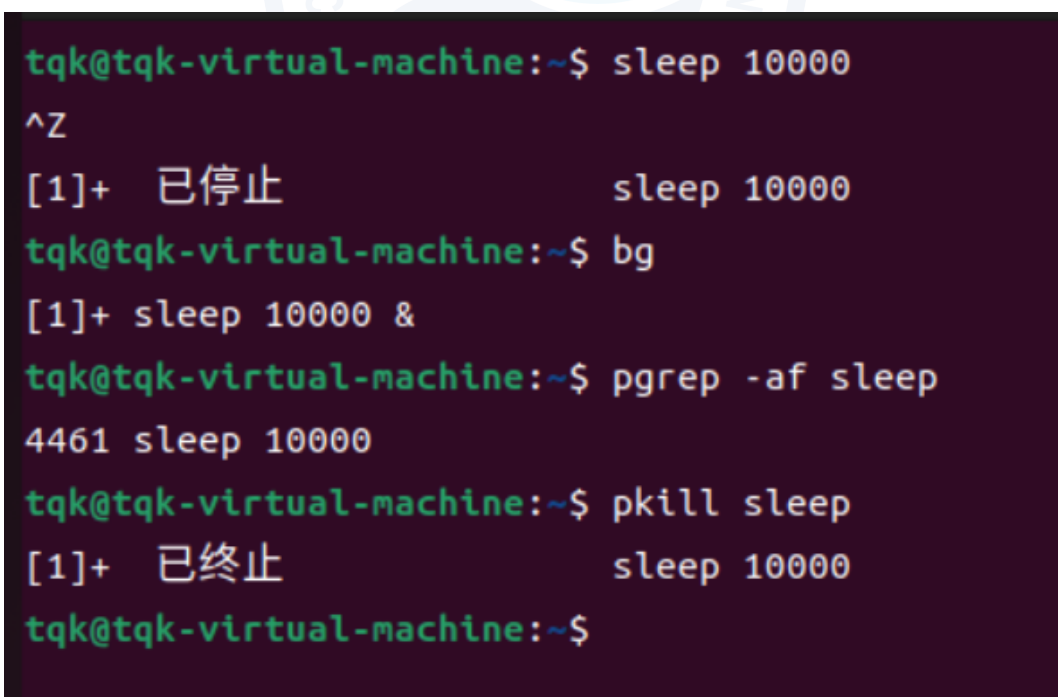
我们可以使用类似 `ps aux | grep` 这样的命令来获取任务的 pid，然后您可以基于 pid 来结束这些进程。但我们其实有更好的方法来做这件事。在终端中执行 `sleep 10000` 这个任务。然后用 `Ctrl-Z` 将其切换到后台并使用 `bg` 来继续允许它。现在，使用 `pgrep` 来查找 pid 并使用 `pkill` 结束进程而不需要手动输入 pid。

在终端执行以下命令启动一个 `sleep` 进程

然后按 `Ctrl-Z` 将其暂停，并输入 `bg` 命令让进程在后台继续运行

使用 `pgrep` 命令，并通过 `-af` 标记列出所有符合条件的进程及其完整的命令行，这将显示所有运行的 `sleep` 进程及其对应的 PID

使用 `pkill` 命令来结束所有匹配的 `sleep` 进程，而不需要手动输入 PID



```
tqk@tqk-virtual-machine:~$ sleep 10000
^Z
[1]+  已停止                  sleep 10000
tqk@tqk-virtual-machine:~$ bg
[1]+  sleep 10000 &
tqk@tqk-virtual-machine:~$ pgrep -af sleep
4461 sleep 10000
tqk@tqk-virtual-machine:~$ pkill sleep
[1]+  已终止                  sleep 10000
tqk@tqk-virtual-machine:~$
```

图 1-1

### 1.2 使用 wait 命令在进程结束后启动下一个任务

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

运行 `sleep 60 &` 命令启动一个休眠进程，并将其放到后台

使用 `wait` 命令等待这个 `sleep` 进程结束

这会阻塞终端，直到后台的 `sleep` 进程结束

当 `wait` 等到 `sleep` 进程结束时，接下来执行 `ls`

```
tqk@tqk-virtual-machine:~$ sleep 60 &
[1] 4713
tqk@tqk-virtual-machine:~$ wait
[1]+  已完成                  sleep 60
tqk@tqk-virtual-machine:~$ ls
公共的  视频  文档  音乐  heelo  hello.cpp  marco.sh  nohup.out  sigint.py
模板    图片  下载  桌面  hello  hello.txt  marco.sh.save  rve.txt  snap
tqk@tqk-virtual-machine:~$
```

图 1-2

### 1.3 终端多路复用

`tmux new -s rve` 创建一个名为 `rve` 的会话并进入，`tmux` 创建的会话会在底部依次显示会话名，窗口名，主机名和时间等信息

```
tqk@tqk-virtual-machine:~$
```



图 1-3

`ctrl+b d` 退出当前会话



```
tqk@tqk-virtual-machine:~$ tmux new -s rve  
[detached (from session rve)]
```

图 1-4

tmux ls 可以列出已创建的会话，如果已经在 tmux 创建的会话中，可以依次按下键盘 ctrl+b s 进行查看

```
tqk@tqk-virtual-machine:~$ tmux ls  
0: 1 windows (created Fri Sep 6 10:22:46 2024)  
1: 1 windows (created Fri Sep 6 10:22:58 2024)  
2: 1 windows (created Fri Sep 6 10:23:02 2024)  
3: 4 windows (created Fri Sep 6 10:24:49 2024)  
rve: 1 windows (created Thu Sep 12 19:41:04 2024)
```

图 1-5

tmux a -t rve 重新进入已存在的会话，如果已经在 tmux 创建的会话中可以使用 tmux switch -t name 来切换对话

tmux kill-session -t rve 可以关闭我们刚才创建的会话，也可以用 ctrl+d 来关闭当前会话

在会话中依次按下键盘 ctrl+b c 创建新窗口，多个窗口下底部带 \* 标记为当前活动窗口

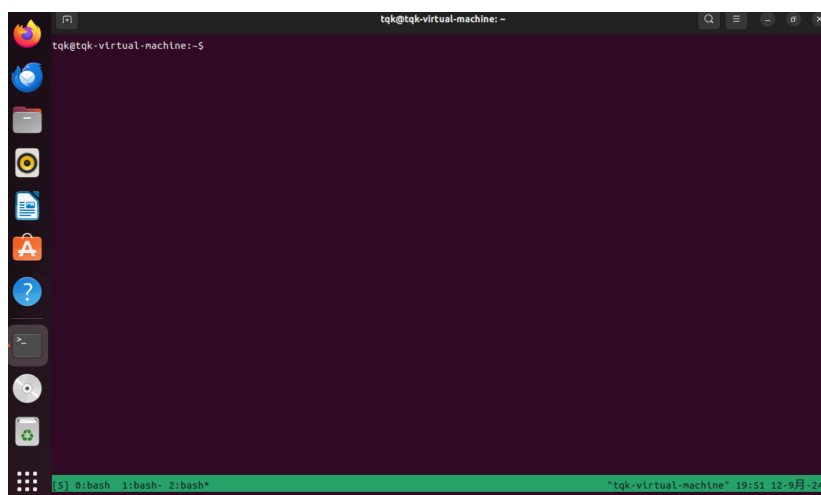


图 1-6

水平分割: Ctrl-b %

垂直分割: Ctrl-b ”

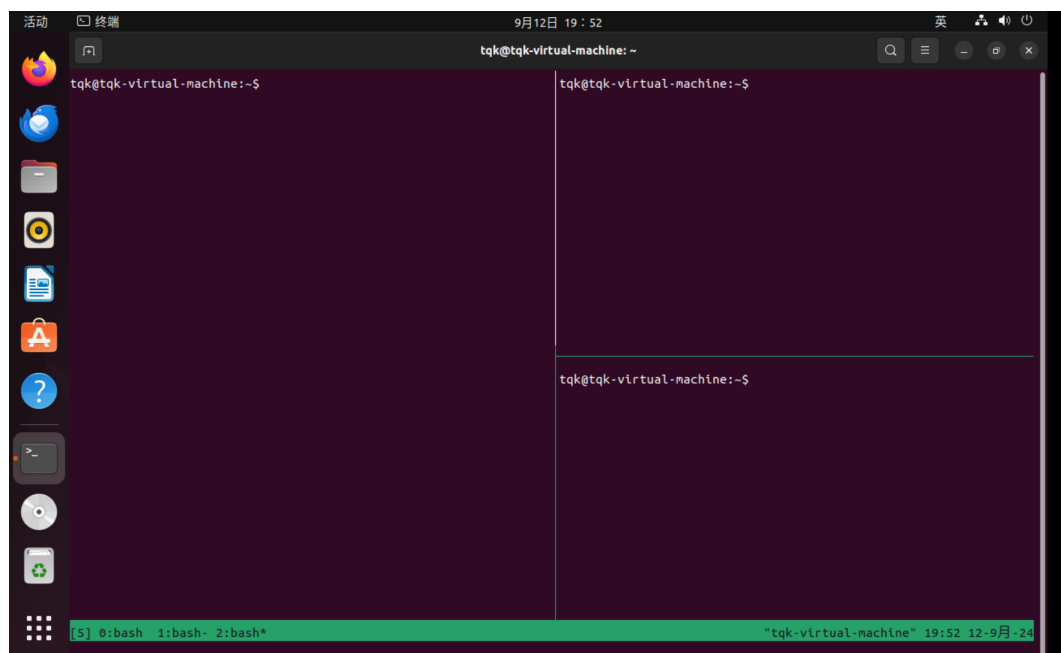


图 1-7

## 第 2 章 Python 入门基础

### 2.1 基本操作和实例

斐波那契数列

```
def fibonacci(n):  
    s = [0, 1]  
    for i in range(2, n):  
        s.append(s[i-1] + s[i-2])  
    return s  
  
n = int(input("请输入斐波那契数列长度: "))  
print(f"前 {n} 项斐波那契数列: {fibonacci(n)}")
```

图 2-1

```
PS C:\Users\Administrator\Desktop\Python> & c:/Users/Administrator/Desktop/Python/.venv/Scripts/python.exe c:/Users/Administrator/Desktop/Python/hello.py  
请输入斐波那契数列长度: 10  
前 10 项斐波那契数列: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]  
PS C:\Users\Administrator\Desktop\Python> & c:/Users/Administrator/Desktop/Python/.venv/Scripts/python.exe c:/Users/Administrator/Desktop/Python/hello.py  
请输入斐波那契数列长度: 4  
前 4 项斐波那契数列: [0, 1, 1, 2]  
PS C:\Users\Administrator\Desktop\Python>
```

图 2-2

判断质数

```
def is_prime(n):  
    if n < 2:  
        return False  
    for i in range(2, int(n ** 0.5) + 1):  
        if n % i == 0:  
            return False  
    return True  
  
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")
```

图 2-3

```
请输入一个整数: 197
197 是质数
PS C:\Users\Administrator\Desktop\Python> & c:/Users/Administrator/Desktop/Python/.venv/Scripts/python.exe c:/Users/Administrator/Desktop/Python/hello.py
请输入一个整数: 237
237 不是质数
PS C:\Users\Administrator\Desktop\Python> |
```

图 2-4

### 汉诺塔问题

```
def hanoi(n, x, y, z):
    if n == 1:
        print(f"将圆盘 1 从 {x} 移动到 {y}")
        return
    hanoi(n-1, x, z, y)
    print(f"将圆盘 {n} 从 {x} 移动到 {y}")
    hanoi(n-1, z, y, x)

n = int(input("请输入圆盘数量: "))
hanoi(n, 'A', 'C', 'B')
```

图 2-5

```
请输入圆盘数量: 3
将圆盘 1 从 A 移动到 C
将圆盘 2 从 A 移动到 B
将圆盘 1 从 C 移动到 B
将圆盘 3 从 A 移动到 C
将圆盘 1 从 B 移动到 A
将圆盘 2 从 B 移动到 C
将圆盘 1 从 A 移动到 C
PS C:\Users\Administrator\Desktop\Python> |
```

图 2-6

## 第3章 Python 视觉应用

### 3.1 图片旋转

```
from PIL import Image

# 1.打开图片
img = Image.open("1.png")
# 2.水平翻转
img1 = img.transpose(Image.FLIP_LEFT_RIGHT)
# 3.保存图片
img1.save("2.png")
# 4.垂直翻转
img2 = img.rotate(180)
# 5.保存图片
img2.save("3.png")
# 6.水平+垂直翻转
img3 = img.transpose(Image.FLIP_LEFT_RIGHT).rotate(180)
# 7.保存图片
img3.save("4.png")
```

图 3-1

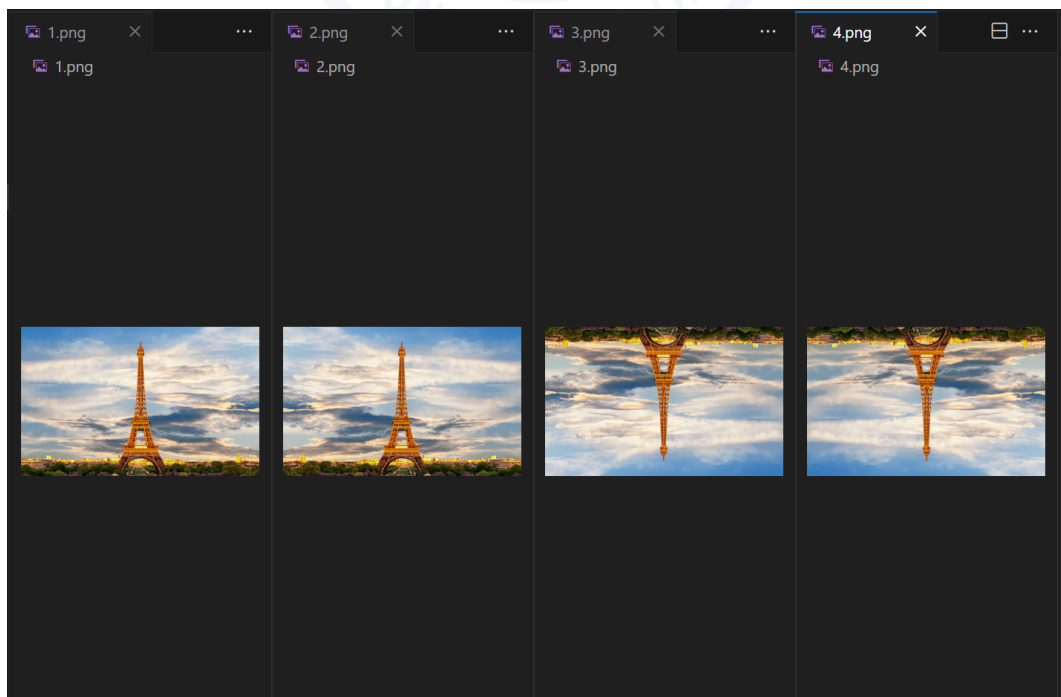


图 3-2

### 3.2 转换图像格式

将 1.png 转换为 1.jpg

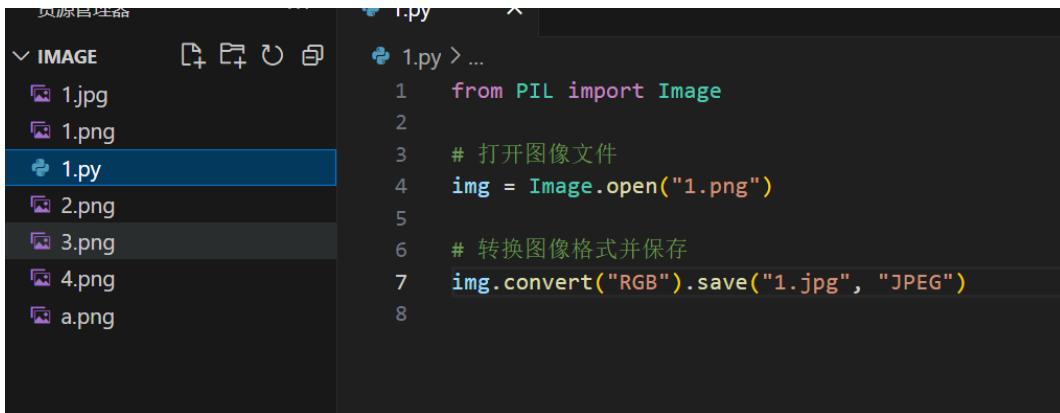


图 3-3

### 3.3 创建缩略图

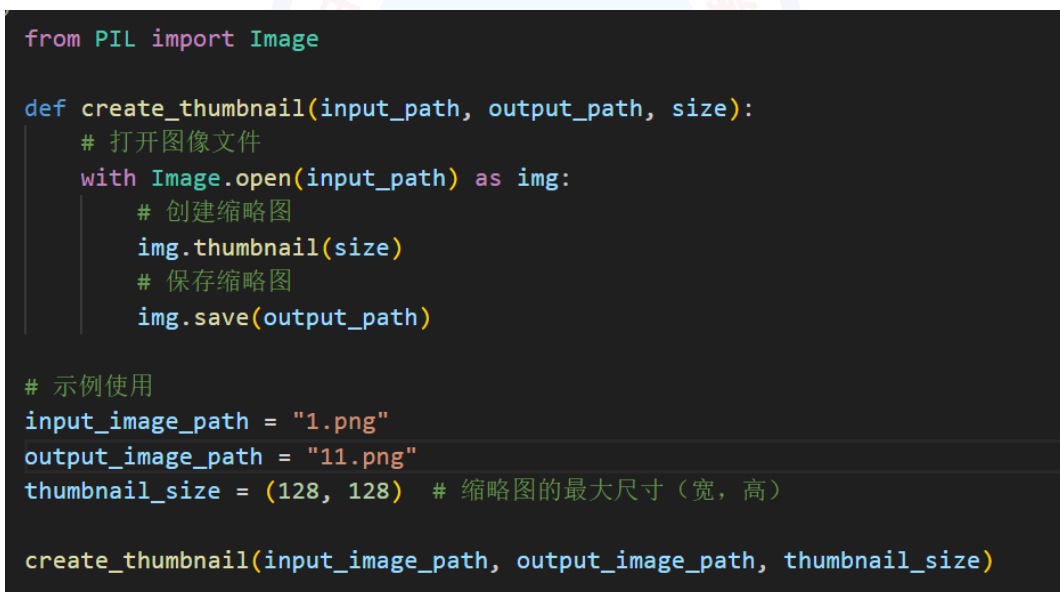


图 3-4

### 3.4 绘制图像、点和线

```
from PIL import Image, ImageDraw

# 创建一个空白图像（白色背景）
width, height = 300, 300
image = Image.new('RGB', (width, height), 'white')

# 创建一个绘图对象
draw = ImageDraw.Draw(image)

# 绘制线条
line_start = (50, 50)
line_end = (250, 50)
draw.line([line_start, line_end], fill='blue', width=5)

# 绘制点
point_position = (150, 150)
draw.point(point_position, fill='red')

# 绘制矩形
rectangle_box = [100, 100, 200, 200] # (左, 上, 右, 下)
draw.rectangle(rectangle_box, outline='green', width=3)

# 绘制椭圆
ellipse_box = [100, 100, 200, 200]
draw.ellipse(ellipse_box, fill='yellow', outline='black')

# 保存图像
image.save('aa.png')
```

图 3-5

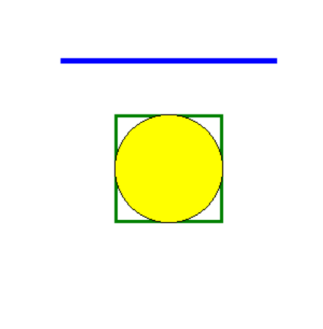


图 3-6

### 3.5 图像缩放

```
from PIL import Image

def resize_image(input_path, output_path, size):|
    # 打开原始图像
    with Image.open(input_path) as img:
        # 缩放图像
        img_resized = img.resize(size, Image.LANCZOS) # Image.LANCZOS 适用于高质量缩放

        # 保存缩放后的图像
        img_resized.save(output_path)
        print(f"Image saved to {output_path}")

# 使用示例
input_image_path = '1.jpg'
output_image_path = '2.jpg'
new_size = (800, 600) # 新的宽度和高度

resize_image(input_image_path, output_image_path, new_size)
```

图 3-7

此次报告的 github 链接: <https://github.com/RVEThr/Systems-Development-Tools-Fundamentals-Laboratory-Report>





## 参考文献



## 附录 A 附录

附录

