

实验报告 3

姓名: 郭千纯

学号: 23020007033

课程: 系统开发工具基础

2024 年 9 月 9 日

目录

1	github 链接	1
2	练习内容	1
3	实例及结果	1
3.1	实例 1	1
3.2	实例 2	2
3.3	实例 3	2
3.4	实例 4	2
3.5	实例 5	3
3.6	实例 6	4
3.7	实例 7	4
3.8	实例 8	4
3.9	实例 9	4

3.10 实例 10	5
3.11 实例 11	5
3.12 实例 12	6
3.13 实例 13	6
3.14 实例 14	7
3.15 实例 15	7
3.16 实例 16	8
3.17 实例 17	8
3.18 实例 18	9
3.19 实例 19	9
3.20 实例 20	11
4 练习感悟	11

1 github 链接

2 练习内容

命令行环境

Python 入门基础

Python 视觉应用

3 实例及结果

3.1 实例 1

使用 matplotlib 相关知识绘制一个正方形

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches

# 创建一个新的图形
fig, ax = plt.subplots()

# 创建一个正方形
square = patches.Rectangle((0.2, 0.2), 0.6, 0.6, linewidth=1, edgecolor='r', facecolor='none')

# 将正方形添加到图形中
ax.add_patch(square)

# 设置图形的 x 和 y 轴的限制
ax.set_xlim(0, 1)
ax.set_ylim(0, 1)

# 设置图形的比例相同
ax.set_aspect('equal')

# 显示图形
plt.title('Square Example')
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.grid(True)
plt.show()
```

图 1: 实例 1

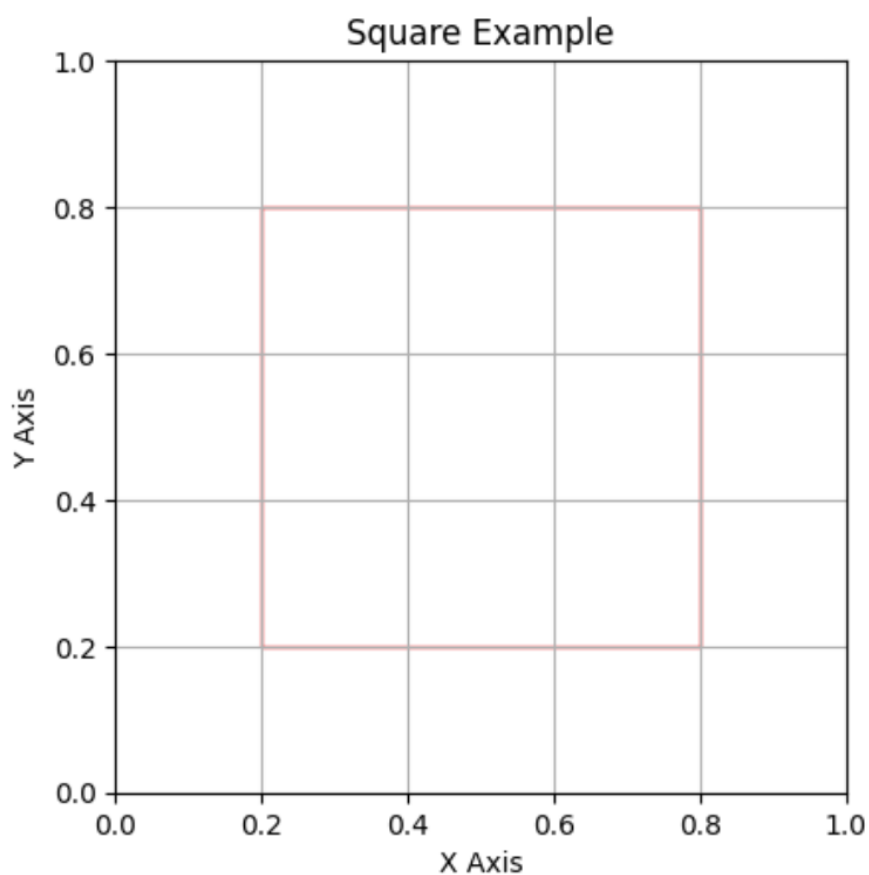


图 2: 实例 1

3.2 实例 2

使用 python 基础知识做了一个猜数字的小游戏

3.3 实例 3

使用 Pillow 库来转换图像格式

3.4 实例 4

打开一个名为 input_image.jpg 的图像文件，生成其 256x256 像素的缩略图，并保存为 thumbnail_image.jpg。

```

import random

def guess_number_game():
    # 生成一个随机的数字
    number_to_guess = random.randint(1, 100)
    attempts = 0

    print("欢迎来到猜数字游戏！")
    print("我已经选择了一个 1 到 100 之间的数字。")

    while True:
        try:
            # 获取用户的猜测
            guess = int(input("请输入你的猜测："))
            attempts += 1

            # 判断猜测的数字
            if guess < number_to_guess:
                print("太小了！再试一次。")
            elif guess > number_to_guess:
                print("太大了！再试一次。")
            else:
                print(f"恭喜你！猜对了，数字是 {number_to_guess}。")
                print(f"你总共猜了 {attempts} 次。")
                break
        except ValueError:
            print("请输入一个有效的数字。")

if __name__ == "__main__":
    guess_number_game()

```

图 3: 实例 2

3.5 实例 5

将 input_image.jpg 图像复制，并将其粘贴到 thumbnail_image.jpg 图像上，粘贴位置为 (50, 50)。最后，将合成后的图像保存为 result_image.jpg。

```
: from PIL import Image

# 打开一个图像文件
with Image.open('input_image.jpg') as img:
    # 转换图像为PNG格式
    img.save('output_image.png', format='PNG')

print("图像格式转换成功!")
```

图像格式转换成功!

图 4: 实例 3

3.6 实例 6

打开一个名为 input_image.jpg 的图像文件，将其调整为 800x600 像素，然后旋转 90 度，并将处理后的图像保存为 processed_image.jpg。

3.7 实例 7

使用 matplotlib 库绘制了一条直线和几个点，设置了图形标题和坐标轴标签，并显示了图例和图形。

3.8 实例 8

创建了一个 100x100 像素的黑色图像（所有像素值为 0），然后在图像中心绘制了一个白色的方块（像素值为 255）。最后，使用 matplotlib 显示了这个图像。

3.9 实例 9

计算一个包含数字的列表的平均值，并打印出结果。通过使用 sum(numbers) 获取列表中所有数字的总和，然后用 len(numbers) 得到列表的长度，最后

```
: from PIL import Image

# 打开一个图像文件
with Image.open('input_image.jpg') as img:
    # 创建缩略图, 指定尺寸为256x256
    img.thumbnail((256, 256))
    # 保存缩略图
    img.save('thumbnail_image.jpg')

print("缩略图创建成功!")
```

缩略图创建成功!

图 5: 实例 4

计算平均值并输出。

3.10 实例 10

定义一个函数 `is_prime`，用来检查一个数字是否为质数。如果数字大于 1 且无法被 2 到该数字平方根之间的任何整数整除，则该数字为质数。最后，代码测试了数字 29 并打印了是否为质数的结果。

3.11 实例 11

使用切片操作 `[::-1]` 来反转字符串 `text`，然后打印出反转后的字符串结果。

```
from PIL import Image

# 打开原始图像
with Image.open('input_image.jpg') as src_img:
    # 复制图像
    copied_img = src_img.copy()

    # 打开另一个图像作为粘贴目标
    with Image.open('thumbnail_image.jpg') as bg_img:
        # 将复制的图像粘贴到目标图像上, 指定粘贴位置
        bg_img.paste(copied_img, (50, 50))
        # 保存最终图像
        bg_img.save('result_image.jpg')

print("图像复制和粘贴成功!")
```

图像复制和粘贴成功!

图 6: 实例 5

3.12 实例 12

使用列表推导式生成了前 10 个自然数（从 1 到 10）的平方，并将其存储在列表 `squares` 中，最后打印出这个列表。

3.13 实例 13

使用生成器表达式计算了列表 `numbers` 中所有偶数的和，并将结果打印出来。

`num for num in numbers if num % 2 == 0` 筛选出列表中的偶数，`sum()` 函数计算这些偶数的总和。


```
: from PIL import Image

# 打开图像文件
with Image.open('input_image.jpg') as img:
    # 调整图像尺寸为800x600
    resized_img = img.resize((800, 600))

    # 旋转图像90度
    rotated_img = resized_img.rotate(90, expand=True)

    # 保存最终图像
    rotated_img.save('processed_image.jpg')

print("图像调整尺寸和旋转成功!")
```

图像调整尺寸和旋转成功!

图 7: 实例 6

3.14 实例 14

使用列表推导式将列表 words 中的每个字符串转换为大写字母，并将转换后的结果存储在 uppercase_words 中，最后打印出这些大写字母字符串

3.15 实例 15

检查了数字 number 是否为偶数。通过取模运算 $\text{number} \% 2$ ，如果结果为 0，则该数字为偶数。最后，代码打印出检查结果。

```

import matplotlib.pyplot as plt

# 创建一个新的图形
plt.figure()

# 绘制一条直线
plt.plot([0, 1, 2, 3], [0, 1, 4, 9], label='Line')

# 绘制几个点
plt.scatter([0, 1, 2, 3], [0, 1, 4, 9], color='red', label='Points')

# 设置标题和标签
plt.title('Line and Points Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# 显示图例
plt.legend()

# 显示图形
plt.show()

```

图 8: 实例 7

3.16 实例 16

使用 `sum()` 函数计算了列表 `numbers` 中所有数字的总和，并将结果打印出来。

3.17 实例 17

关闭删除会话

Tmux detach

Tmux attach -t 111

Tmux kill-session -t 111

```
plt.show()
```

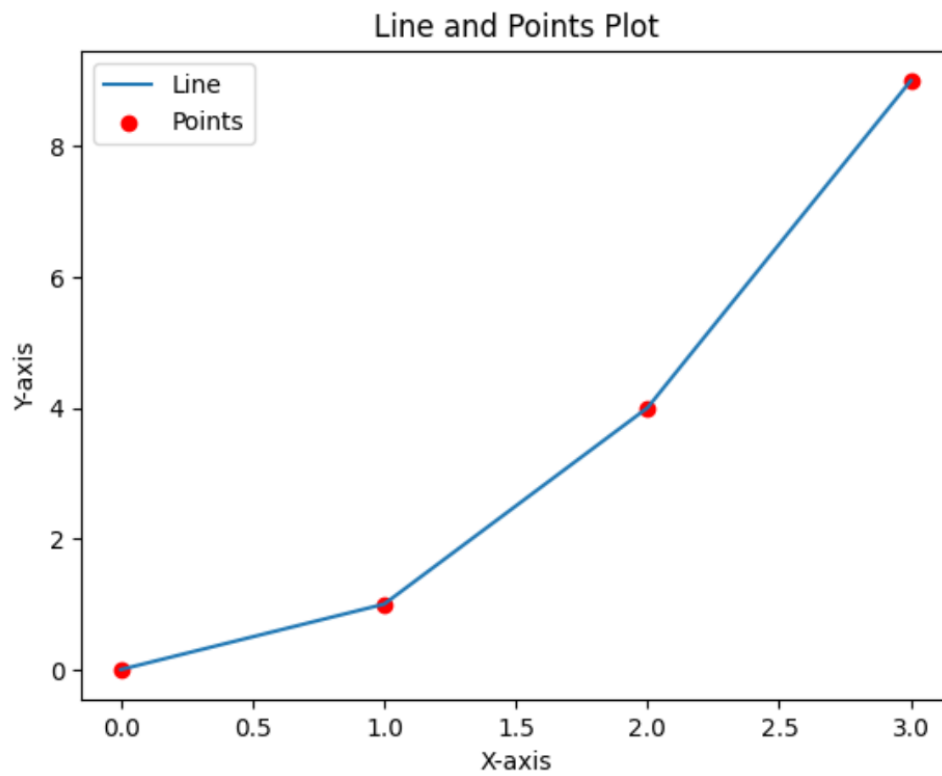


图 9: 实例 7

3.18 实例 18

重命名会话

```
Tmux rename-session -t 111 333
```

3.19 实例 19

切换会话

```
Tmux new -s 111
```

```
Tmux detach
```

```
Tmux new -s 222
```

```
import numpy as np
import matplotlib.pyplot as plt

# 创建一个100x100的黑色图像（所有像素值为0）
black_image = np.zeros((100, 100), dtype=np.uint8)

# 在图像中心绘制一个白色的方块
black_image[30:70, 30:70] = 255

# 显示图像
plt.imshow(black_image, cmap='gray')
plt.title('Image Array Representation')
plt.axis('off')
plt.show()
```

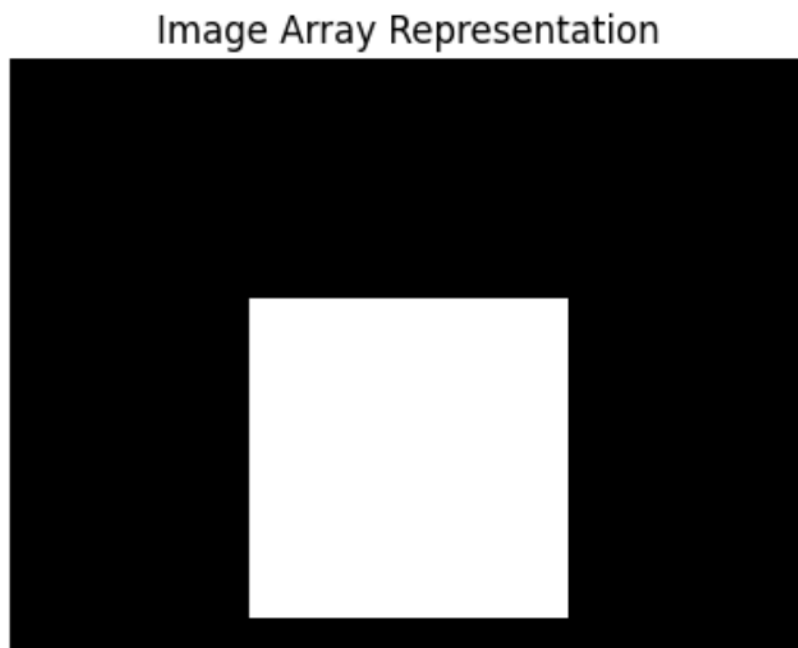


图 10: 实例 8

Tmux switch -t 111

```
: # 定义一个包含数字的列表
numbers = [10, 20, 30, 40, 50]

# 计算列表中所有数字的平均值
average = sum(numbers) / len(numbers)

# 打印结果
print("平均值:", average)

平均值: 30.0
```

图 11: 实例 9

3.20 实例 20

创建一个新的会话

Tmux

Tmux new -s 111

4 练习感悟

命令行环境

感悟：在命令行环境下操作是编程的基础技能之一，它让我们能够高效地执行各种任务。掌握基本的命令行操作，如文件导航、创建、删除和编辑文件，不仅提升了操作系统的使用效率，也帮助我们在开发中处理文件和管理项目。在实践中，我学会了如何使用命令行工具来运行 Python 脚本，这对于自动化任务和调试代码非常重要。

```

: def is_prime(n):
    """检查数字n是否为质数"""
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

# 测试数字
number = 29

# 打印结果
print(f"{number} 是质数吗?", is_prime(number))

```

29 是质数吗? True

图 12: 实例 10

Python 入门基础

感悟：Python 作为一门入门友好的编程语言，具有简洁的语法和强大的功能。通过学习 Python 的基础知识，如数据类型、控制结构、函数和模块，我对编程的核心概念有了更深入的理解。在编写简单的 Python 程序时，我体会到了编程逻辑的乐趣和解决问题的成就感。这些基础知识为后续的进阶学习打下了坚实的基础。

Python 视觉应用

```
# 定义一个字符串
text = "Hello, World!"

# 反转字符串
reversed_text = text[::-1]

# 打印结果
print("反转后的字符串:", reversed_text)
```

反转后的字符串: !dlrow ,olleH

图 13: 实例 11

```
# 生成前10个自然数的平方
squares = [x**2 for x in range(1, 11)]

# 打印结果
print("前10个自然数的平方:", squares)
```

前10个自然数的平方: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

图 14: 实例 12

感悟: Python 在视觉应用方面的能力令人惊叹。通过学习如何处理图像和视频, 我意识到了 Python 在数据处理和计算机视觉领域的强大功能。使用库如 Pillow 和 OpenCV 来处理图像和应用视觉算法, 让我感受到了 Python 在实际应用中的灵活性和强大。尤其是在实现图像变换、处理和分析时, 我体会到了编程如何将抽象的概念转化为可视化的结果。

```
# 定义一个包含数字的列表
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# 计算列表中所有偶数的和
even_sum = sum(num for num in numbers if num % 2 == 0)

# 打印结果
print("所有偶数的和:", even_sum)
```

所有偶数的和: 30

图 15: 实例 13

```
# 定义一个包含字符串的列表
words = ['hello', 'world', 'python', 'is', 'awesome']

# 将列表中的每个字符串转换为大写
uppercase_words = [word.upper() for word in words]

# 打印结果
print("转换为大写的字符串:", uppercase_words)
```

转换为大写的字符串: ['HELLO', 'WORLD', 'PYTHON', 'IS', 'AWESOME']

图 16: 实例 14


```
# 定义一个数字
number = 42

# 检查数字是否为偶数
is_even = (number % 2 == 0)

# 打印结果
print(f"{number} 是偶数吗?", is_even)
```

42 是偶数吗? True

图 17: 实例 15

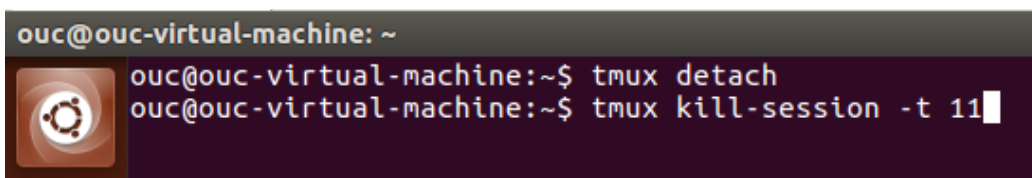
```
[18]: # 定义一个包含数字的列表
      numbers = [5, 10, 15, 20]

      # 计算列表中所有数字的总和
      total_sum = sum(numbers)

      # 打印结果
      print("数字的总和:", total_sum)
```


数字的总和: 50

图 18: 实例 16

A terminal window with a dark background. The prompt is 'ouc@ouc-virtual-machine: ~'. The first command entered is 'tmux detach', and the second is 'tmux kill-session -t 11'. A small icon of a terminal window is visible on the left side of the terminal output area.



```
ouc@ouc-virtual-machine: ~
ouc@ouc-virtual-machine:~$ tmux detach
ouc@ouc-virtual-machine:~$ tmux kill-session -t 11
```

图 19: 实例 17




```
ouc@ouc-virtual-machine:~$ tmux rename-session -t 111 333
usage: rename-session [-t target-session] new-name
ouc@ouc-virtual-machine:~$
```

图 20: 实例 18






```
ouc@ouc-virtual-machine:~$ tmux new -s 111
[detached (from session 111)]
ouc@ouc-virtual-machine:~$ tmux new -s 222
[detached (from session 111)]
```

图 21: 实例 19




```
ouc@ouc-virtual-machine: ~
ouc@ouc-virtual-machine:~$ tmux detach
ouc@ouc-virtual-machine:~$ mux rename-session -t 111 333
```

图 22: 实例 19



```
ouc@ouc-virtual-machine: ~
ouc@ouc-virtual-machine:~$ tmux new -s 11
[detached (from session 11)]
ouc@ouc-virtual-machine:~$ tmux attach -t 11
[exited]
ouc@ouc-virtual-machine:~$
```

图 23: 实例 20



```
ouc-virtual-machine: ~
ouc@ouc-virtual-machine:~$ tmux detach
```

图 24: 实例 20