

Python入门与视觉应用

23020007073 刘畅

2024 年 9 月 14 日

1 实验目的

掌握Python的基本编程方法，能够编写简单的程序。

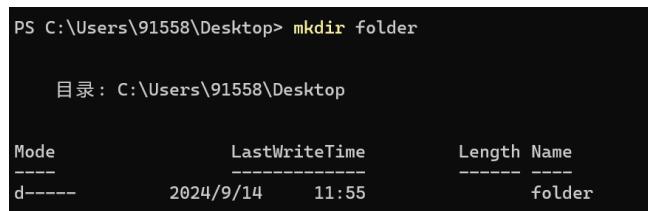
了解图像处理的基本概念，能够通过Python进行图像读取、显示和保存。

2 练习内容

2.1 命令行学习5个例子

1. 创建文件夹：

```
mkdir new_folder
```

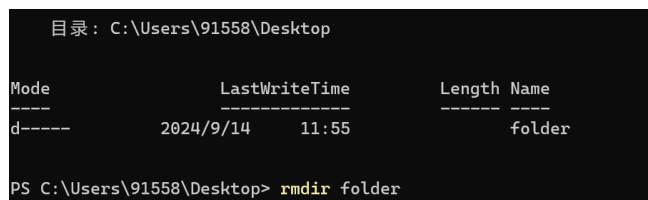


PS C:\Users\91558\Desktop> mkdir folder
目录：C:\Users\91558\Desktop
Mode LastWriteTime Length Name
---- ----- ----- ----
d---- 2024/9/14 11:55 folder

图 1: 创建文件夹

2. 删除文件夹：

```
rmdir /s /q folder_name
```



PS C:\Users\91558\Desktop> rmdir folder
目录：C:\Users\91558\Desktop
Mode LastWriteTime Length Name
---- ----- ----- ----
d---- 2024/9/14 11:55 folder

图 2: 删除文件夹

3. 文本文件的创立与写入

```
echo "Hello, World!" > hello.txt
```

```
PS C:\Users\91558\Desktop> echo "Hello, World!" > hello.txt
```

图 3: 文本文件的创立与写入

4.type example.txt 查看文件内容

```
PS C:\Users\91558\Desktop> type hello.txt
Hello, World!
```

图 4: 列表的输出结果

5.wmic memorychip list brief 查看内存信息

```
PS C:\Users\91558\Desktop> wmic memorychip list brief
Capacity DeviceLocator MemoryType Name Tag TotalWidth
4294967296 Controller0-ChannelA 0 物理内存 Physical Memory 0 16
4294967296 Controller0-ChannelB 0 物理内存 Physical Memory 1 16
4294967296 Controller0-ChannelC 0 物理内存 Physical Memory 2 16
4294967296 Controller0-ChannelD 0 物理内存 Physical Memory 3 16
4294967296 Controller1-ChannelA 0 物理内存 Physical Memory 4 16
4294967296 Controller1-ChannelB 0 物理内存 Physical Memory 5 16
4294967296 Controller1-ChannelC 0 物理内存 Physical Memory 6 16
4294967296 Controller1-ChannelD 0 物理内存 Physical Memory 7 16
```

图 5: 内存信息

2.2 Python样例10个

1.print()打印的意思 print("hello,world")

```
Hello, World!

进程已结束，退出代码为 0
```

图 6: 用print来打印内容

2.条件语句的使用

```
x=1 y=8
if x > y:
    print("x is bigger than y")
else:
    print("x is not bigger than y")
```

```
x is not bigger than y
```

图 7: 条件语句输出结果

3.循环语句的使用

```
for i in [1, 2, 3, 4, 5]:
    print(i)
```

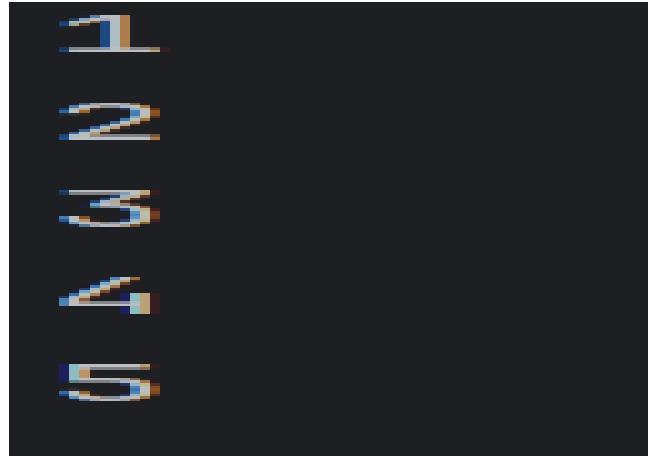


图 8: 循环语句的使用

4. 函数的定义与使用

```
def greet():
    print("Hello, Python")
greet()
```

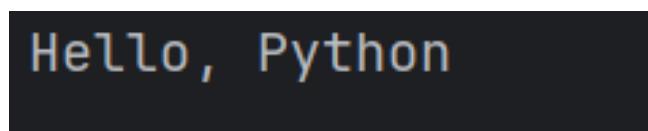


图 9: 函数的定义与使用

5. 列表的使用

```
list = [1, 2, 3, 4, 5]
print(list[0])
```

```
list = [1, 2, 3, 4, 5]
print(list[0])
```

```
D:\py项目及代码\项目\pythonProject\.venv\Scripts\python.exe D:\py项
1

进程已结束，退出代码为 0
```

图 10: 列表的输出结果

6.字典的使用:

```
person = {
    '名字': '李华',
    '年龄': 20,
    '城市': '青岛',
    '职业': '学生'
}

# 访问字典中的数据
print(person['名字'])
print(person['职业'])
```

```
pythonProject D:\py项目及代码\项目\pythonProject
  venv
  测试.py
外部库
临时文件和控制台
```

```
测试.py ×
```

```
1 person = {  
2     '名字': '李华',  
3     '年龄': 20,  
4     '城市': '青岛',  
5     '职业': '学生'  
6 }  
7  
8 # 访问字典中的数据  
9 print(person['名字'])  
10 print(person['职业'])
```

```
测试 ×
```

```
D:\py项目及代码\项目\pythonProject\.venv\Scripts\python.exe D:\py项目及代码\pythonProject\测试.py  
李华  
学生
```

```
进程已结束，退出代码为 0
```

图 11: 字典的输出结果

7.元组的使用：

```
yuanzu= (123, "work", 258)  
print(yuanzu[1])
```

The screenshot shows a PyCharm interface. On the left is a project tree with a 'pythonProject' folder containing '.venv' and '测试.py'. The '测试.py' file is selected. The code editor window shows the following Python code:

```
yuanzu = (123, "work", 258)
print(yuanzu[1])
```

Below the code editor is a terminal window titled '测试' with the following output:

```
D:\py项目及代码\项目\pythonProject\.venv\Scripts\python.exe D:\py项
work

进程已结束，退出代码为 0
```

图 12: 元组的输出结果

8. 程序和用户的交互:

```
number = int(input("请输入一个整数: "))
print("您输入的整数是: ", number)
```

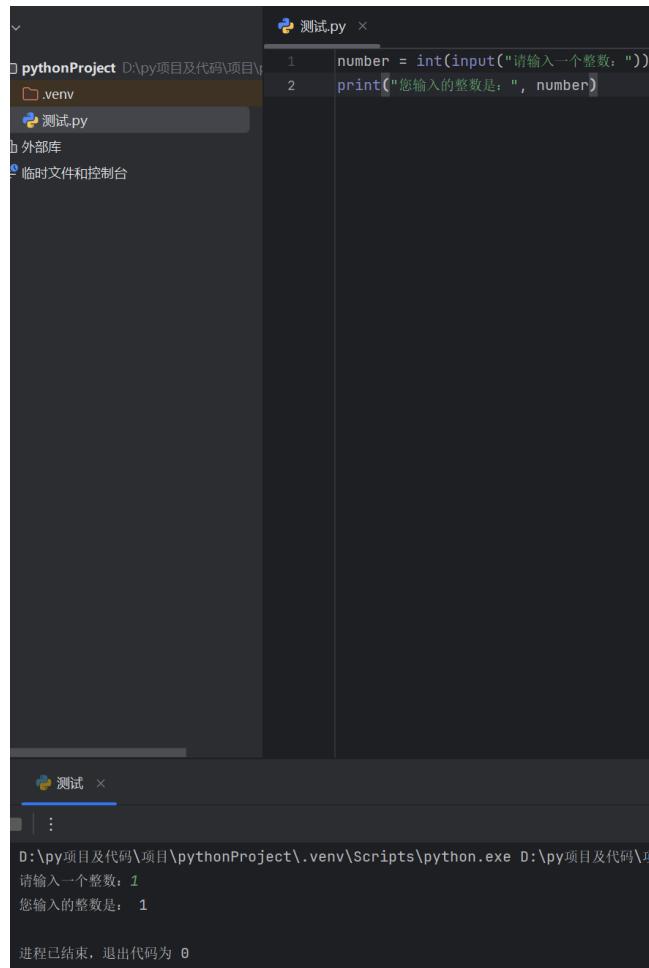


图 13: 程序和用户的交互

9. 类的使用:

```
class Class:
    def __init__(self, chinese, math, english):
        self.chinese = chinese
        self.math = math
        self.english = english

    def describe_class(self):
        return f"{self.chinese} {self.math} {self.english}"
myclass = Class('Yuwen', 'Shuxue', 'Yingyu')

print(myclass.describe_class())
```

The screenshot shows a Python code editor with a dark theme. A file named '测试.py' is open, containing the following code:

```
1 class Class: 1个用法
2     def __init__(self, chinese, math, english):
3         self.chinese = chinese
4         self.math = math
5         self.english = english
6
7     def describe_class(self): 1个用法
8         return f'{self.chinese} {self.math} {self.english}'
9
10 myclass = Class(chinese='Yiwen', math='Shuxue', english='Yingyu')
11 print(myclass.describe_class())
```

The code defines a class 'Class' with an __init__ method that initializes three attributes: chinese, math, and english. It also defines a describe_class method that returns a string in the format 'chinese math english'. An instance 'myclass' is created with values 'Yiwen', 'Shuxue', and 'Yingyu' respectively, and its describe_class method is called, outputting the result.

图 14: 类的使用

10. 文件的写入和读取

```
with open('example.txt', 'w') as file:
    file.write('Hello, World!')
with open('example.txt', 'r') as file:
    content = file.read()
print(content)
```

The screenshot shows a Python development environment. On the left is a file tree with a project named 'pythonProj' containing a '.venv' folder and a '测试.py' file. The '测试.py' file is open in the main editor area, displaying the following code:

```
1 with open('example.txt', 'w') as file:  
2     file.write('Hello, World!')  
3 with open('example.txt', 'r') as file:  
4     content = file.read()  
5 print(content)
```

Below the editor is a terminal window titled '测试'. It shows the command 'D:\py项目及代码\项目\pythonProject\.venv\Scripts\python.exe D:\py项目' followed by the output 'Hello, World!'. At the bottom of the terminal window, it says '进程已结束, 退出代码为 0'.

图 15: 文件的写入和读取

2.3 Python视觉应用样例5个

1. 图像灰度变换

```
from PIL import Image  
# 读取图像  
image = Image.open('ceshi.jpg')  
# 显示图像  
image.show()  
# 转换为灰度图像  
gray_image = image.convert('L')  
gray_image.show()
```



图 16: 灰度变换前



图 17: 灰度变换后

2.利用Matplotlib完成折线图的绘制

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100) # 生成从0到10的100个点
y = np.sin(x) + np.random.normal(0, 0.1, 100) # 生成正弦曲线并添加一些随机噪声
# 创建图形和轴
plt.figure(figsize=(10, 6)) # 设置图形的大小
plt.plot(x, y, label='sin(x) + noise', color='blue') # 绘制折线图
# 添加标题和标签
plt.title('Simple Plot')
plt.xlabel('X axis')
plt.ylabel('Y axis')
# 添加图例
plt.legend()
# 显示网格（可选）
plt.grid(True)
# 显示图形
plt.show()
```

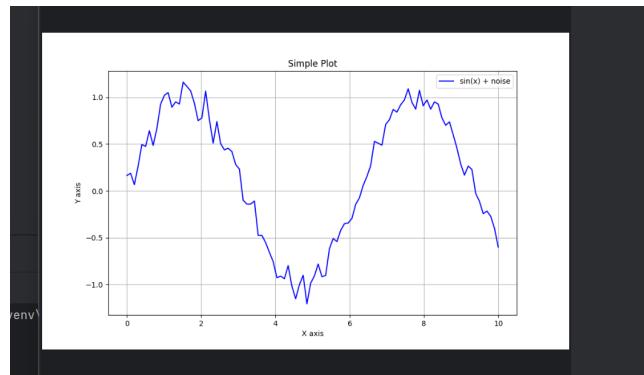


图 18: 绘制的折线图

3. 图像模糊

```
from PIL import Image
import numpy as np
import cv2
# 读取图像
image = Image.open('ceshi.jpg')
# 转换图像为numpy数组
image_np = np.array(image)
# 应用均值模糊
blurred_image_np = cv2.blur(image_np, (15, 15))
# 转换numpy数组回PIL图像
blurred_image = Image.fromarray(blurred_image_np)
# 显示原始图像和均值模糊后的图像
image.show()
blurred_image.show()
```



图 19: 图像模糊之前如下:



图 20: 图像模糊之后如下:

4. 图像缩放

```
import matplotlib.pyplot as plt
# 读取图像
image = plt.imread('ceshi.jpg')
# 设置缩放比例
scale_percent = 50
width = int(image.shape[1] * scale_percent / 100)
height = int(image.shape[0] * scale_percent / 100)
# 创建一个缩放后的图像
resized_image = image[0:height, 0:width]
# 显示缩放后的图像
plt.imshow(resized_image)
plt.axis('off') # 不显示坐标轴
plt.show()
```

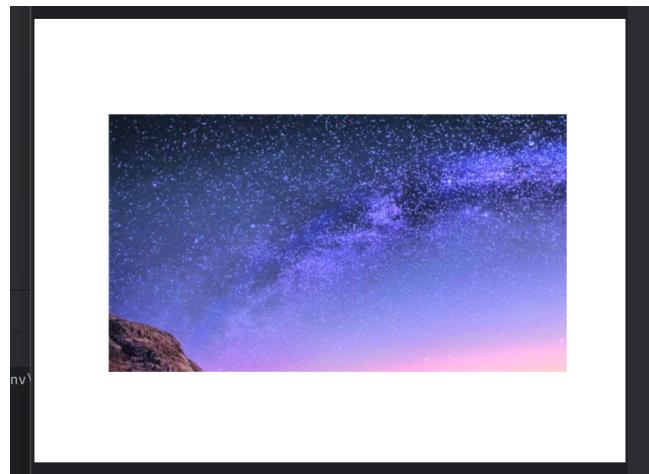


图 21: 缩放后的图片

5. 缩略图绘制

```
from PIL import Image
# 读取原始图像
original_image = Image.open('ceshi.jpg') # 替换为您的图像路径
# 设置缩略图的比例因子
thumbnail_ratio = 0.25 # 缩略图尺寸是原始尺寸的25%
# 创建缩略图
thumbnail = original_image.resize((int(original_image.width * thumbnail_ratio), int(original_image.height * thumbnail_ratio)))
# 显示原始图像和缩略图
original_image.show()
thumbnail.show()
```

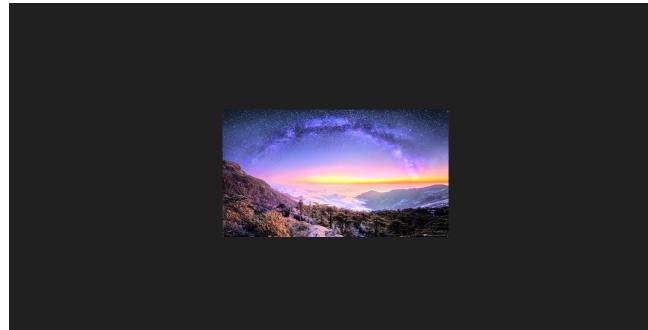


图 22: 缩略图绘制

3 解题感悟

Python是一种简便快捷的语言，非常容易入门。

Python的可视化能够将抽象的数据转化为直观的图像，这让信息变得更加易于理解。

通过命令行能够高效地与计算机系统互动，方便快捷地完成工作。

github路径您可以在此查看项目的源代码：

<https://github.com/L-c-hang/homework3>