

Python 入门基础和 Python 视觉应用

23020007160 张绍延

2024 年 9 月 12 日

1 实验目的

掌握 Python 的基本语法和编程习惯，能够编写简单的程序。

了解图像处理的基本概念，能够使用 Python 进行图像读取、显示和保存。

通过实现一个简单的图像分类或识别项目，理解计算机视觉的基本流程。

2 介绍

2.1 优点

1. Python 具有清晰的语法结构，对初学者友好，易于上手。Python 在多个领域都有应用，如数据分析、网络开发、人工智能等，掌握基础后可以轻松转向其他领域。

2. Python 视觉应用可以将抽象的算法和理论转化为直观的图像结果，帮助理解

3. 命令行可以更加高效、方便、快捷地完成工作，直接对系统进行操作

3 练习内容

3.1 Python 学习例子 10 个

1. print() 打印的意思 `print("hello,world")`

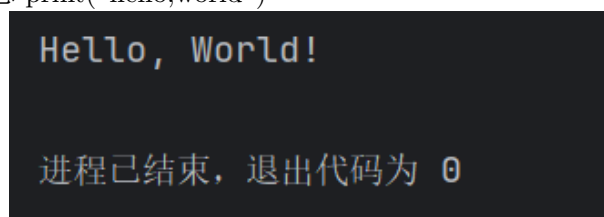
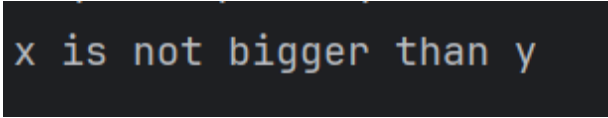


图 1: 用 print 来打印想输出的内容

2. 条件语句的使用

```
x=2 y=3
if x > y:
    print("x is bigger than y")
else:
    print("x is not bigger than y")
```

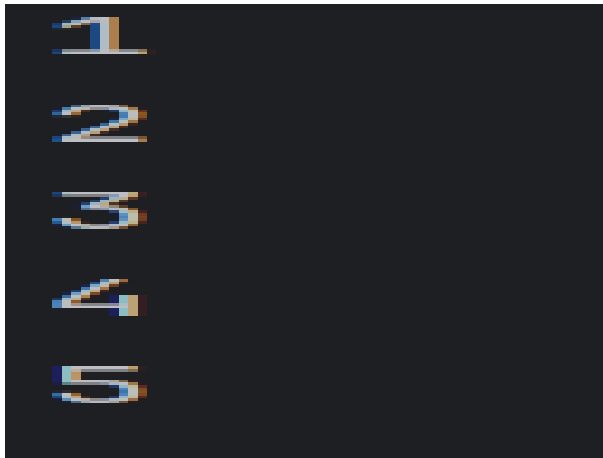


```
x is not bigger than y
```

图 2: 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)
```

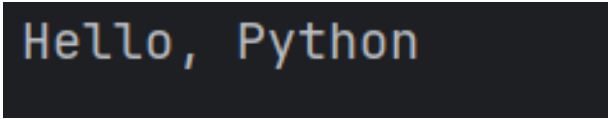


```
1  
2  
3  
4  
5
```

图 3: 循环语句的使用

4. 函数的定义和使用

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



```
Hello, Python
```

图 4: 函数的定义与使用

5. 列表的使用

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

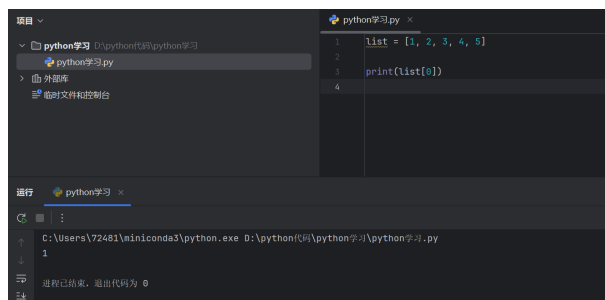


图 5: 列表的输出结果

6. 字典的使用:

```
person = {
    '名字': '张先生',
    '年龄': 30,
    '城市': '泰安',
    '职业': '计算机工程师'
}
```

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

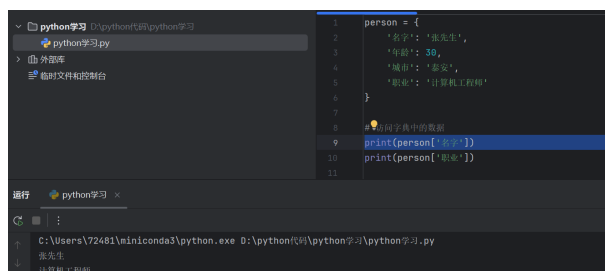


图 6: 字典的输出结果

7. 元组的使用:

```
yuanzu= (666, "work", 3.1415)
print(yuanzu[1])
```

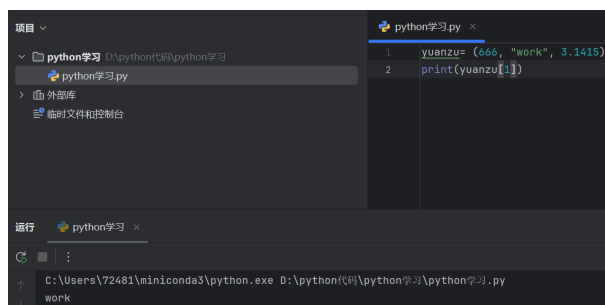


图 7: 元组的输出结果

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)
#int是标明输入的类型
#number=input()是向number里输入数据
```

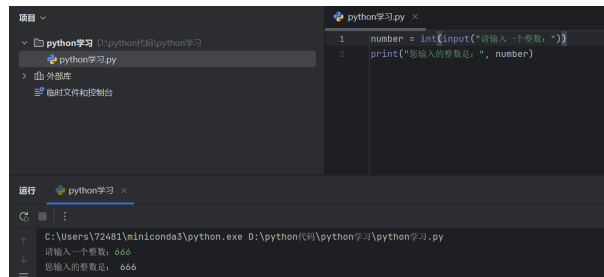


图 8: 程序和用户的交互

9. 类的使用:

```
class Car:
    def __init__(self, make, model, year):
        """初始化汽车属性"""
        self.make = make
        self.model = model
        self.year = year
        self.odometer_reading = 0 # 汽车里程表初始为0

    def describe_car(self):
        """返回汽车描述信息"""
        return f"{self.year} {self.make} {self.model}"

# 创建一个Car实例
mycar = Car('Toyota', 'Corolla', 2020)

# 打印汽车描述信息
print(mycar.describe_car())
```

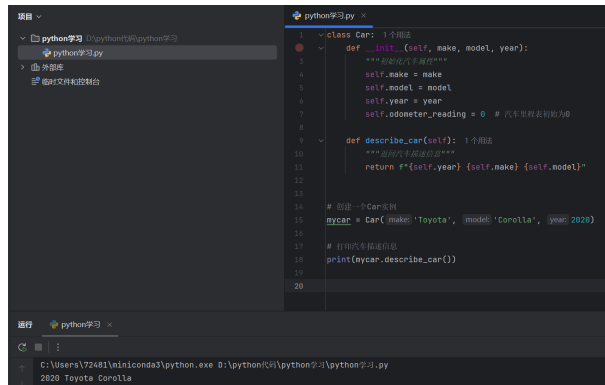


图 9: 类的使用

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)

```

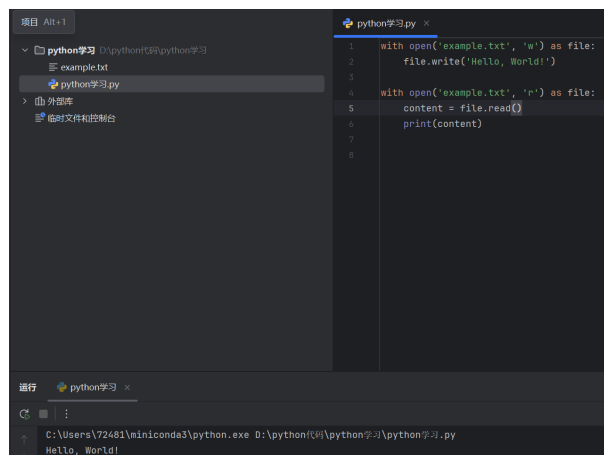


图 10: 文件的写入和读取

3.2 Python 视觉应用 5 个例子

1. 图像灰度变换

```

from PIL import Image

# 读取图像
image = Image.open('测试图片.png')

# 显示图像
image.show()

# 转换为灰度图像
gray_image = image.convert('L')

gray_image.show()

```



图 11: 灰度变换前



图 12: 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()

```

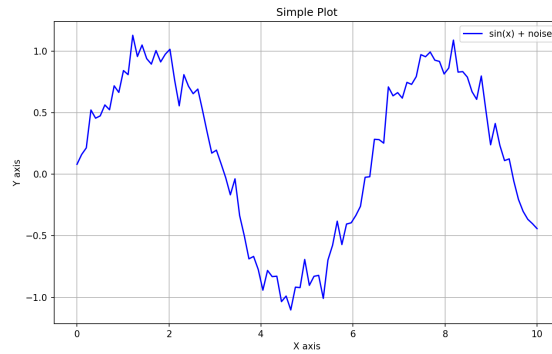


图 13: 绘制的折线图

3. 图像模糊

```

from PIL import Image
import numpy as np
import cv2
# 读取图像
image = Image.open('测试图片.png')
# 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, (5, 5))
# 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 14: 图像模糊之前如下:



图 15: 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('测试图片.png')
# 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 16: 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('测试图片.png') # 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()
```

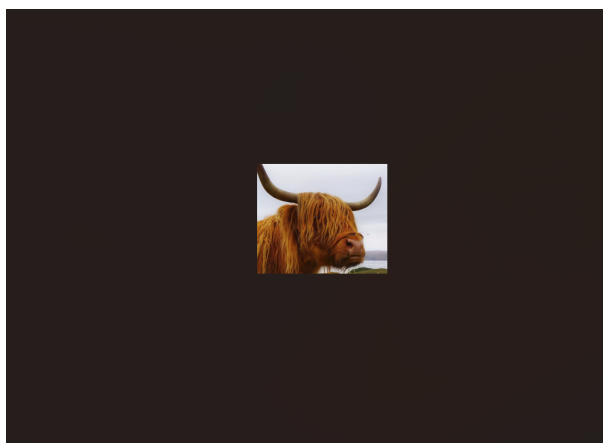
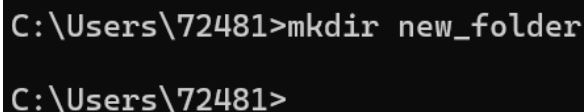


图 17: 缩略图绘制

3.3 命令行学习 5 个例子

1. 创建文件夹:

```
mkdir new_folder
```

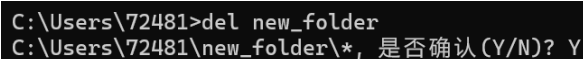


```
C:\Users\72481>mkdir new_folder  
C:\Users\72481>
```

图 18: 创建文件夹

2. 删除文件夹:

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



```
C:\Users\72481>del new_folder  
C:\Users\72481\new_folder\*, 是否确认(Y/N)? Y
```

图 19: 删除文件夹

3. 文本文件的创立与写入

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



```
C:\Users\72481\Desktop\命令行>echo "Hello, World!" > hello.txt
```

图 20: 文本文件的创立与写入

4.type example.txt 查看文件内容



```
C:\Users\72481\Desktop\命令行>type hello.txt  
"Hello, World!"
```

图 21: 列表的输出结果

5.wmic memorychip list brief 查看内存信息



```
C:\Users\72481>wmic memorychip list brief  
Capacity DeviceLocator MemoryType Name Tag TotalWidth  
8589934592 Controller0-ChannelA-DIMM0 0 物理内存 Physical Memory 0 64  
8589934592 Controller1-ChannelA-DIMM0 0 物理内存 Physical Memory 1 64
```

图 22: 内存信息

4 解题感悟

通过学习 Python, 我了解了一种更加简便快捷的语言, 它的语法简单易学。同时 Python 实用性很强, 可以用于 Web 开发、数据分析、人工智能、自动化脚本、游戏开发等多个领域, 我会深化对他的学习, 提高对 Python 的掌握程度。

通过 Python 的可视化, 我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。命令行则教会了我如何高效地与计算机系统互动, 方便快捷地完成工作。

今后我会用好这两大工具来处理数据和文件, 更快捷地完成工作。

github 路径您可以在这里查看项目的源代码: