Lab07-迭代器效率比较

1 基本信息

姓名	**李懋**
学号	**2213189**
实验题目	**面向对象编程**
完成时间	**2024.11.6**

2 实验目的

- **练习面向对象编程**

3 实验内容

3.1 问答题

3.1.1 简述面向对象的三大特性

- 封装
- 继承
- 多态

3.1.2 将以下函数改成类的方式并调用

```
def func(a):
        print(a)
```

```
def func(a):  1 usage
    print(a)

class MyClass:  1 usage
    def __init__(self,value):
        self.value=value

    def func(self):  1 usage
        print(self.value)

#调用函数
func("北风毫不留情")
#创造类的实例并调用方法
obj=MyClass("把叶子吹落")
obj.func()
```

```
北风毫不留情
把叶子吹落

Process finished with exit code 0
```

3.1.3 面向对象和函数的区别?

- **面向对象编程 和 函数式编程** 是**两种不同的编程范式**，它们在**设计思想、代码组织方式和解决问题的方法**上有显著的区别。
- **面向对象编程（OOP）**：
  - 核心思想：对象和类的抽象。
  - 代码组织：以类和对象为中心。
  - 状态管理：可变状态，通过方法操作。
  - 解决问题：适合复杂的业务逻辑和系统设计。
- **函数式编程（FP）**：
  - 核心思想：函数的纯粹性和不可变性。
  - 代码组织：以函数为中心。
  - 状态管理：不可变数据，无副作用。
  - 解决问题：适合数据处理和算法问题。

3.1.4 面向对象中的self指的是什么

- 在面向对象编程中，`self` 是一个特殊的参数，用于表示类的实例（对象）本身。
- 它是一个约定俗成的名称，通常作为类方法的第一个参数。
- 通过 `self`可以访问和操作对象的属性和其他方法。
- `self` 的作用
  1. **访问实例属性**：通过 `self`，方法可以访问和修改对象的属性。
  2. **调用其他方法**：通过 `self`，方法可以调用对象的其他方法。
  3. **区分实例变量和局部变量**： `self` 帮助区分实例变量和方法内部的局部变量。

3.1.5 以下代码体现面向对象的什么特点

```
class Person:  1 usage
    def __init__(self,name,age,gender):
        self.name=name
        self.age=age
        self.gender=gender

obj=Person( name: 'limao', age: 20, gender: 'Male')
print(obj.name)
print(obj.age)
print(obj.gender)
```

1. 封装
2. 抽象
3. 实例化

3.1.6 以下代码体现了面向对象的什么特点

```
class Message:
    def email(self):pass
    def msg(self):pass
    def wechat(self):pass
```

- Message类，包含了三个方法：email msg wechat, 但都为空
- 体现了
    1. 封装
    2. 抽象
    3. 多态

3.2 编程题

3.2.1 基于代码框架，实现用户的注册和登录

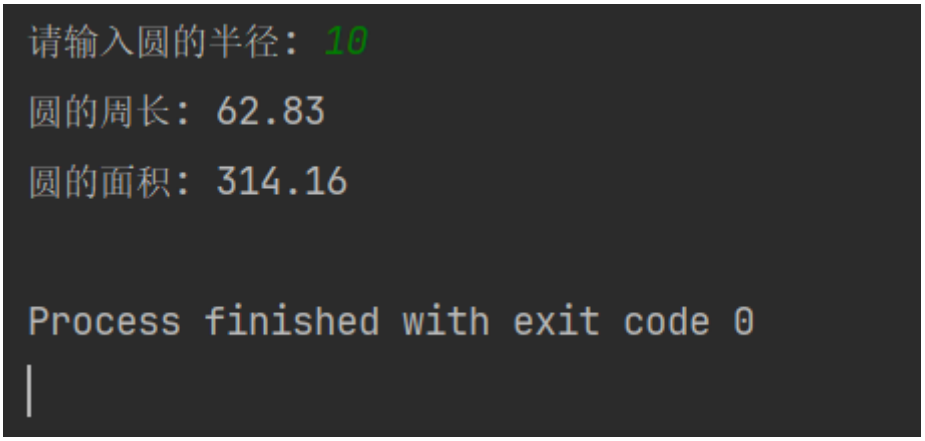
```
class User:
    def __init__(self, name, pwd):
        self.name = name
        self.pwd = pwd


class Account:
    def __init__(self):
        self.user_list = []  # 用户列表，数据格式：[User 对象, User 对象, User 对象]

    def login(self):
        """
        用户登录，用户输入用户名和密码并去 self.user_list 中检查用户是否合法
        """
        print("\n开始登录流程...")
        for attempt in range(3):
            name = input("请输入用户名: ")
            pwd = input("请输入密码: ")
            for user in self.user_list:
                if user.name == name and user.pwd == pwd:
                    print("登录成功！")
                    return
            print(f"用户名或密码错误，请重试。（剩余重试次数：{2 - attempt}）")
        print("登录失败，已达到最大重试次数。")

    def register(self):
        """
        用户注册，动态创建 User 对象，并添加到 self.user_list 中
        """
        print("\n开始注册流程...")
        name = input("请输入用户名: ")
        pwd = input("请输入密码: ")
        user = User(name, pwd)
        self.user_list.append(user)
        print("注册成功！")

    def run(self):
        """
        主程序，先进行 2 次用户注册注册两个不同的用户，再进行用户登录（3 次重试机会）
        """
        print("欢迎使用用户管理系统！")
        for _ in range(2):
            self.register()
        self.login()


if __name__ == "__main__":
    obj = Account()
    obj.run()
```

```
D:\Applications\编程软件\Python\python.exe D:\desktop\大三上-课程资料\3-Pyt
欢迎使用用户管理系统!

开始注册流程...
请输入用户名: ganchao
请输入密码: 1234
注册成功!

开始注册流程...
请输入用户名: fengzhuoyi
请输入密码: 12345
注册成功!

开始登录流程...
请输入用户名: ganchao
请输入密码: 123
用户名或密码错误，请重试。（剩余重试次数：2）
请输入用户名: ganchaoo
请输入密码: 1234
用户名或密码错误，请重试。（剩余重试次数：1）
请输入用户名: fengzhuoyi
请输入密码: 12345
登录成功!

Process finished with exit code 0
```

3.2.2 定义一个圆的类，其中有计算周长和面积的方法（圆的半径通过参数传递到构造方法）。运算结果保留两位。

```python
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def calculate_circumference(self):
        """
        计算圆的周长
        """
        circumference = 2 * math.pi * self.radius
        return round(circumference, 2)

    def calculate_area(self):
```

```
        """
        计算圆的面积
        """
        area = math.pi * (self.radius ** 2)
        return round(area, 2)

# 获取用户输入的半径
radius = float(input("请输入圆的半径: "))

# 创建 Circle 类的实例
circle = Circle(radius)

# 计算并输出周长和面积
print(f"圆的周长: {circle.calculate_circumference()}")
print(f"圆的面积: {circle.calculate_area()}")
```



4 实验总结

1. 通过本次实验初步熟悉了OOP编程，后续还需要多加练习，加强理解
2. 可以多实现课件上的例子