实验报告

1 预习内容在Python中，函数是组织和重用代码的基本单元，通过def关键字定义。例如，def greet(name): return "Hello, " + name。函数可以接受参数，并通过return语句返回值。Python函数的参数有多种形式，包括位置参数、关键字参数、默认参数和可变参数（\*args 和 \*\*kwargs）。变量作用域决定了变量的可见性和生命周期，分为局部作用域和全局作用域。局部变量在函数内部定义，只在函数内部有效；全局变量在函数外部定义，可以在整个模块中访问。通过global关键字，可以在函数内部修改全局变量。此外，nonlocal关键字允许在嵌套函数中修改外层非全局变量。这些概念和机制使得Python函数具有很高的灵活性和可用性，有助于代码的模块化和组织。

2 程序代码

（1）def buy\_chickens(A, B, C, X):  
 solutions = []  
 for cock in range(X + 1):  
 for hen in range(X + 1 - cock):  
 chicks = X - cock - hen  
 if chicks % C == 0 and (cock \* A + hen \* B + chicks // C) == X:  
 solutions.append((cock, hen, chicks))  
 return solutions  
  
# 测试函数  
A = 1  
B = 2  
C = 3  
X = 100  
solutions = buy\_chickens(A, B, C, X)  
for solution in solutions:  
 print(f"公鸡: {solution[0]} 只, 母鸡: {solution[1]} 只, 小鸡: {solution[2]} 只 , 共 {A\*solution[0] + B\*solution[1] + solution[2]/C} 元")

（2）def generate\_fibonacci(n):  
 if n <= 0:  
 return []  
 elif n == 1:  
 return [1]  
 elif n == 2:  
 return [1, 1]  
  
 fibonacci\_sequence = [1, 1]  
 for i in range(2, n):  
 next\_value = fibonacci\_sequence[-1] + fibonacci\_sequence[-2]  
 fibonacci\_sequence.append(next\_value)  
  
 return fibonacci\_sequence  
  
  
n = 10  
fibonacci\_sequence = generate\_fibonacci(n)  
print(f"前 {n} 项斐波那契数列: {fibonacci\_sequence}")

（3）def max\_area(height):  
 left, right = 0, len(height) - 1  
 max\_area = 0  
  
 while left < right:  
 width = right - left  
 current\_height = min(height[left], height[right])  
 current\_area = width \* current\_height  
 max\_area = max(max\_area, current\_area)  
 if height[left] < height[right]:  
 left += 1  
 else:  
 right -= 1  
  
 return max\_area  
  
  
height = [1, 8, 6, 2, 5, 4, 8, 3, 7]  
result = max\_area(height)  
print(f"最大容积是: {result}")

3 成果展示

公鸡: 0 只, 母鸡: 40 只, 小鸡: 60 只 , 共 100.0 元

公鸡: 5 只, 母鸡: 38 只, 小鸡: 57 只 , 共 100.0 元

公鸡: 10 只, 母鸡: 36 只, 小鸡: 54 只 , 共 100.0 元

公鸡: 15 只, 母鸡: 34 只, 小鸡: 51 只 , 共 100.0 元

公鸡: 20 只, 母鸡: 32 只, 小鸡: 48 只 , 共 100.0 元

公鸡: 25 只, 母鸡: 30 只, 小鸡: 45 只 , 共 100.0 元

公鸡: 30 只, 母鸡: 28 只, 小鸡: 42 只 , 共 100.0 元

公鸡: 35 只, 母鸡: 26 只, 小鸡: 39 只 , 共 100.0 元

公鸡: 40 只, 母鸡: 24 只, 小鸡: 36 只 , 共 100.0 元

公鸡: 45 只, 母鸡: 22 只, 小鸡: 33 只 , 共 100.0 元

公鸡: 50 只, 母鸡: 20 只, 小鸡: 30 只 , 共 100.0 元

公鸡: 55 只, 母鸡: 18 只, 小鸡: 27 只 , 共 100.0 元

公鸡: 60 只, 母鸡: 16 只, 小鸡: 24 只 , 共 100.0 元

公鸡: 65 只, 母鸡: 14 只, 小鸡: 21 只 , 共 100.0 元

公鸡: 70 只, 母鸡: 12 只, 小鸡: 18 只 , 共 100.0 元

公鸡: 75 只, 母鸡: 10 只, 小鸡: 15 只 , 共 100.0 元

公鸡: 80 只, 母鸡: 8 只, 小鸡: 12 只 , 共 100.0 元

公鸡: 85 只, 母鸡: 6 只, 小鸡: 9 只 , 共 100.0 元

公鸡: 90 只, 母鸡: 4 只, 小鸡: 6 只 , 共 100.0 元

公鸡: 95 只, 母鸡: 2 只, 小鸡: 3 只 , 共 100.0 元

公鸡: 100 只, 母鸡: 0 只, 小鸡: 0 只 , 共 100.0 元

前 10 项斐波那契数列: [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

最大容积是: 49

4 心得体会

通过"买鸡问题"的实现，我掌握了如何使用嵌套循环和条件判断来解决实际问题。这个程序不仅训练了我的逻辑思维能力，还让我体会到在实际编程中，代码的可读性和优化的重要性。通过多次调试和改进，我逐步优化了代码，使其运行更加高效。

在编写斐波那契数列生成函数时，我学会了如何处理递归和循环之间的选择。通过分析斐波那契数列的递归和非递归实现，我深刻理解了不同算法的时间复杂度和空间复杂度的差异。在实际编程中，我更倾向于使用非递归的迭代方法，以避免递归带来的栈溢出问题。

通过"最大容积问题"的实现，我了解了双指针算法的基本原理和应用场景。这种算法的实现不仅提高了程序的执行效率，还让我体会到在实际编程中，选择合适的数据结构和算法的重要性。在实际应用中，面对不同的问题，灵活选择和使用适当的算法是解决问题的关键。

总体而言，本次课程设计让我在理论知识与实际编程能力方面都有了显著提升。我学会了如何在编写复杂程序时，合理划分功能模块，并通过调试和测试，不断改进和优化代码。同时，我也认识到在编程学习中，持续不断地练习和总结是非常重要的。未来的学习和工作中，我将继续深入探索Python的更多高级特性和应用，不断提高自己的编程能力和解决实际问题的能力。

# 封面格式

出于保留格式的需求，相应封面在后续页。

|  |  |
| --- | --- |
| **成 绩** |  |

****

**中 国 矿 业 大 学**

**China University of Mining and Technology**

**20 - 20 学年**

**《Python编程实践》**

**课 程 设 计 报 告**

院 系： 信息与控制工程学院

班 级：

学 号：

日 期：

姓 名：

签 名：

注：若要求交打印稿，（1）请在姓名处手写签名 ，（2）并删除本行文字

|  |  |
| --- | --- |
| **成 绩** |  |

****

**中 国 矿 业 大 学**

**China University of Mining and Technology**

**20 - 20 学年**

**《Python编程实践》**

**第 次实验报告**

院 系： 信息与控制工程学院

班 级：

学 号：

日 期：

姓 名：

签 名：

注：若要求交打印稿，（1）请在姓名处手写签名 ，（2）并删除本行文字