

**学生实验实习报告册**

|  |  |
| --- | --- |
| 学年学期： | 2023-2024 学年 🞏春■秋学期 |
| 课程名称： | Python程序设计与应用 |
| 学生学院： | 自动化学院 |
| 专业班级： | 08052102 |
| 学生学号： | 2021212981 |
| 学生姓名： | 王忠全 |
| 联系电话： | 1508669338 |

**重庆邮电大学教务处制**

|  |  |  |  |
| --- | --- | --- | --- |
| **实验测验名称** | **Python基本数据类型和组合数据类型实验** | | |
| **地点** | C318 | **时间** | 第七周9-12节 |
| **9-12指导教师** | 张 焱 | **成绩** |  |
| **一、实验目的**  1、了解程序的基本结构并绘制流程图；  2、掌握程序的分支结构；  3、运用if语句实现分支结构；  4、掌握程序的循环结构；  5、运用for语句和while语句实现循环结构；  6、掌握随机库的使用方法；  7、了解程序的异常处理方法。  8、掌握函数的定义和调用方法；  9、理解函数的参数传递过程以及变量的作用范围；  10、了解lambda函数；  11、掌握时间日期标准库的使用；  12、理解函数递归的定义和使用方法。 | | | |
| **二、实验测验所需仪器设备和软件**  Pychram图标 的图像结果  Pycharm 2022.2.3  环境：Python 3.8 | | | |

|  |
| --- |
| **三、实验题目**   1. 输入小明身高1.75，体重80.5kg。请根据BMI公式（体重除以身高的平方）帮小明计算他的BMI指数，计算规则为：  * 低于18.5：过轻 * 18.5-25：正常 * 25-28：过重 * 28-32：肥胖 * 高于32：严重肥胖  1. 输入两个整数，打印他们相除后的结果，若输入的不是整数或除数为0，进行异常处理。 2. 使用蒙特·卡罗方法计算圆周率近似值。 3. 使用枚举法验证6174猜想。 4. 模拟报数游戏（约瑟夫环问题） 5. 模拟轮盘抽奖游戏 6. 一年365天，如果好好学习时能力值比前一天提高1%，当放任时相比前一天下降1%，编程计算两种情况效果相差值。   8、凯撒加密算法原理与实现。  **接下来分别对每个题目进行分析：**   1. **【计算BMI指数】**   输入：身高tall与体重kilo  输出：身体的BMI指数和BMI等级“过轻、正常、过重、肥胖、严重肥胖”      图1 BMI指数示意图  **2、【输入整数相除】**  输入两个整数，打印他们相除后的结果，若输入的不是整数或除数为0，进行异常处理。定义函数，利用Python内置isinstance(x, int)函数判断输入的是否为整数，如果除数和被除数都是整数，则正常相除，如果不是则判断是除数还是被除数不是整数，进行异常处理输出判断。  **3、【蒙特·卡罗方法计算圆周率近似值】**  蒙特卡罗法方法近似圆周率，按照我的理解设计思路：就是计算机模拟在正方形内撒一把豆子，计算落在圆内的豆子数量，按照圆豆子/正方形豆子比例，即可近似出圆周率计算公式推导：  二者相比有：  即可近似得：    图2 蒙特·卡罗方法计算圆周率近似值示意图  **4、【使用枚举法验证6174猜想】**  任意一个四位数（四位数有大有小，不能完全相同相同），把它重新排列成从大到小的四位数，再排列成从小到大的四位数，然后用大的减去小的，所得的结果，再进行相同的排列、相减，最多7次，最后必将得到6174这个结果    图3 6174猜想示意图  枚举法顾名思义，就是要循环遍历不全相同的四位数，验证每个四位数的6174算法，之后我定义了6174验证算法，该函数就是在8步之内，将四位数按大到小排序，减去按按小到大排序，判断相减是否得到6174，如果没有得到就将相减的数作为新数再重复排序相减。如果超出8步且没有得到6174即视为猜想错误，立刻结束程序。如果枚举所有四位数都得到6174即验证猜想正确。  **5、【模拟报数游戏（约瑟夫环问题）】**    图4模拟报数游戏（约瑟夫环问题）示意图  如上述图所示，我自己模拟了一个报数游戏过程，8个小朋友做成一个圈，每个小朋友身上都有一个编号，每个小孩报数报到3就退出圈（不参与接下来的报数），经过四轮报数淘汰，剩下7号小朋友。  那接下来就需要通过编写代码，实现模拟实现报数游戏。我的思路是通过列表模拟约瑟夫环，定义一个指针，每次指针累加（报数）到报数的倍数时，就弹出列表这个数字，把列表长度作为每一轮。  输入：多少个小朋友参加报数游戏、报到多少退出圈子。  输出：最后留下的是多少号小朋友。  **6、【模拟轮盘抽奖游戏】**  模拟转盘抽奖游戏，输入抽奖人数，自定义抽奖中奖一等奖、二等奖、三等奖，概率分别为10%、20%、70%    图5 模拟转盘抽奖示意图  **7、【一年365天，如果好好学习时能力值比前一天提高1%，当放任时相比前一天下降1%，编程计算两种情况效果相差值】**  在这里我假设初始成绩为100分，每天努力提高1%，与每天放任下降1%后，进行对比，这里的主要思想就是迭代，每天都需要在前一天的基础上进行一个提高1%或者下降1%，通过Python编程就能实现这个过程。  每天努力一点点乘以365 的图像结果  图6 能力值计算示意图  **8、【凯撒加密算法原理与实现】**  凯撒密码最早由古罗马军事统帅盖乌斯·尤利乌斯·凯撒在军队中用来传递加密信息，故称凯撒密码。此为一种位移加密手段，只对26个（大小写）字母进行位移加密，规则相当简单，容易被破解。凯撒密码的替换方法是通过排列明文和密文字母表，密文字母表示通过将明文字母表向左或向右移动一个固定数目的位置。    图7 凯撒加密示意图 |

|  |
| --- |
| **四、实验结果及分析讨论（含原始数据记录等）**   1. **【计算BMI指数】**     图8 计算BMI流程图   1. # 2.1-肥胖指数BMI计算 2. # 输入小明身高1.75m，体重80.5kg.请根据BMI公式（体重除以身高的平方）帮小明计算他的BMI指数，计算规则为： 3. # 18.5<     过轻 4. # 18.5-25   正常 5. # 25-28     过重 6. # 28-32     肥胖 7. # >32       严重肥胖 9. **def** My\_BMI(tall, kilo): 10. BMI = kilo / pow(tall, 2) 11. **if** 0 <= BMI < 18.5: 12. **print**('BMI为{0:.2f}：过轻'.format(BMI)) 13. **elif** 18.5 <= BMI < 25: 14. **print**('BMI为{0:.2f}：正常'.format(BMI)) 15. **elif** 25 <= BMI < 28: 16. **print**('BMI为{0:.2f}：过重'.format(BMI)) 17. **elif** 28 <= BMI < 32: 18. **print**('BMI为{0:.2f}：肥胖'.format(BMI)) 19. **elif** 32 <= BMI < 50: 20. **print**('BMI为{0:.2f}：严重肥胖'.format(BMI)) 21. **else**: 22. **print**('输入错误'.format(BMI))  25. x = float(input("请输入身高：")) 26. y = float(input("请输入体重：")) 27. My\_BMI(x, y)     图9 计算BMI运行图  如上图所示，输入小明的身高1.75，体重80.5，经过我的BMI算法得到BMI指数为26.29，判断为过重。实验结果与预期相符合，算法正确。  **2、【输入整数相除】**    图10 整数相除流程图   1. # 输入两个整数，打印他们相除后的结果，若输入的不是整数或除数为0，进行异常处理 3. **def** My\_Divi(x, y): 4. **if** isinstance(x, int) **and** isinstance(y, int) **and** y != 0:    # 如果被除数x和除数y都是整数 5. num = x / y 6. **print**(num) 7. **elif** **not** isinstance(x, int): 8. **print**("被除数{0}不是整数".format(x)) 9. **elif** **not** isinstance(y, int): 10. **print**("除数{0}不是整数".format(y)) 11. **elif** y == 0: 12. **print**("除数为0啦") 13. **else**: 14. **print**("错误")  17. num1 = eval(input("请输入被除数：")) 18. num2 = eval(input("请输入除数：")) 19. My\_Divi(num1, num2)     图9 计算BMI运行图  输入被除数32与除数4得到8，输出正常，接下来尝试输出不是整数。    上图，输入被除数23.5，不是整数，则抛出异常。    上图，输入除数3.5，不是整数，则抛出异常。    上图，输入除数0，抛出异常。经过测试，实验结果与预期相符合，算法正确。  **3、【蒙特·卡罗方法计算圆周率近似值】**    图11蒙特·卡罗方法计算圆周率近似值流程图   1. # 使用蒙特.卡洛法计算圆周率近似值 2. **import** random 3. **from** tqdm **import** tqdm    # 进度条  6. **def** MY\_Pi(N):    # 一共在正方形扔N次数 7. count = 0    # 统计落在圆内的次数 8. **for** i **in** tqdm(range(N)):    # 一共抛N次 9. x = random.uniform(-1, 1)    # 随机浮点数 10. y = random.uniform(-1, 1) 11. **if** pow(x, 2) + pow(y, 2) <= 1:    # 在圆内 12. count += 1    # 落在圆内，计数器加1 13. pi = 4 \* count/N 14. **print**("运用蒙特卡洛法，执行{0}次，近似pi={1}".format(N, pi))  17. k = int(input("请输入执行次数:")) 18. MY\_Pi(k)     输入执行10000次蒙特卡洛算法后，近似圆周率为3.138    输入执行100000次蒙特卡洛算法后，近似圆周率为3.14148，由运行结果可知，当抛豆子的次数越多的时候，越逼近圆周率，经过测试，实验结果与预期相符合，算法正确。  **4、【使用枚举法验证6174猜想】**      图12 验证6174猜想流程图   1. # 使用枚举法验证6174猜想：6174猜想 ，1955年，卡普耶卡(D.R.Kaprekar)研究了对四位数的一种变换： 2. # 任给出四位数k0,用它的四个数字由大到小重新排列成一个四位数m,再减去它的反序数rev(m),得出数k1=m-rev(m), 3. # 然后，继续对k1重复上述变换，得数k2.如此进行下去，卡普耶卡发现，无论k0是多大的四位数， 只要四个数字不全相同，最多进行7次上述变换，就会出现四位数6174 4. **def** My\_sort(n):    # 把数字从大到小排序 5. s = str(n)         # s是n的字符串 6. s\_bu = '0' \* (4 - len(s)) + s   # 把不足四位的数补零 7. s\_sort = ''.join(sorted(s\_bu, reverse=True))   # 返回排序后的数字 8. **return** s\_sort  11. **def** My\_6174(n): 12. step = 1 13. old\_n = n 14. **print**('验证：', old\_n) 15. **while** step <= 8:  # 一直循环 16. n = int(n)  # 把n改成整数 17. m = int(My\_sort(n))  # 把n从大到小排序 18. m\_re = int(str(m)[::-1])  # 把m变成字符串后反序再变成整数 19. k = m - m\_re  # 新的差 20. **print**("{0} - {1} = {2}".format(m, m\_re, k)) 21. **if** k == 6174:  # 如果是6174则跳出循环 22. **break** 23. **else**: 24. n = k  # 把差值作为新的数字 25. step += 1  # 步数加一 26. **if** k == 6174 **and** step <= 8: 27. **print**('{0}进行了{1}次算法得到了6174'.format(old\_n, step)) 28. **return** True 29. **else**: 30. **print**('验证6174失败') 31. **return** False  34. Flag = True 35. i = 999 37. **while** Flag **and** i < 9999:    # 如果正确就会一直验证 38. i += 1 39. **if** len(set(str(i))) == 1:    # 如果所有数字都相同，则跳过 40. **continue** 41. **else**:    # 否则就进行验证 42. Flag = My\_6174(i) 44. **if** Flag == True: 45. **print**('在四位数中验证6174没有失败！！猜想正确！') 46. **elif** Flag == False: 47. **print**('在四位数中验证6174有失败！！猜想错误！')     点击运行后，程序自动枚举不全相同的四位数，进行6174算法验证，从1000至9998遍历猜想，并且输出算法几次得到6174。    验证到9998时，程序结束，且输出在四位数中验证6174没有失败，全部成功，猜想正确，实验结果与预期相符合，算法正确。  **5、【模拟报数游戏（约瑟夫环问题）】**  通过前期自己绘图，模拟这个报数游戏，8个小朋友报数3后退出，最后剩7号小朋友。      图13模拟报数游戏（约瑟夫环问题）流程图   1. # 模拟报数游戏（约瑟夫环问题） 2. # 有n个人围成一圈，从1开始按顺序编号，从第一个人开始从1到k（假设k=3）报数， 3. # 报到k的人退出圈子；然后圈子缩小，从下一个人继续游戏，问最后留下的是原来的第几号。 5. **def** My\_baoshu(n, k): 6. t = [1]    # 生成原始列表--圈圈 7. **for** i **in** range(1, n): 8. t.append(i + 1) 9. count = 0    # 定义计数指针 10. **while** len(t) != 1: 11. t\_new = t[:] 12. **print**(list(t\_new)) 13. **for** j **in** range(0, len(t\_new)):    # 循环新列表 14. count += 1 15. **if** count % k == 0:    # 当报到k时弹出 16. t.remove(t\_new[j])    # 在原来的列表里除去点到的数 17. **print**('留下的是原来的{0}号'.format(list(t)))  20. n = int(input("请输入多少个人参与：")) 21. k = int(input("请输入报到多少退出圈子：")) 22. My\_baoshu(n, k)     由图可知，输入8个小朋友参与报数，围成一个圈圈，报到3后退出这个圈，第一轮剩余【1，2，4，5，7】第二轮剩余【2，4，7，8】第三轮剩余【4，7】，最后只剩下【7】号小朋友，这与我们最开始模拟的圈图一致，实验结果与预期相符合，算法正确。  **6、【模拟轮盘抽奖游戏】**    图14模拟轮盘抽奖游戏流程图   1. # 模拟转盘抽奖游戏 2. # 一等奖、二等奖、三等奖 3. # 一等奖【0--0.1】 4. # 二等奖【0.1--0.3】 5. # 三等奖【0.3--1】 7. **import** random 8. **from** tqdm **import** tqdm  11. **def** My\_gift(n): 12. count = [0, 0, 0] 13. **for** i **in** tqdm(range(n)): 14. k = random.uniform(0, 1) 15. **if** 0 <= k < 0.1: 16. count[0] += 1 17. **elif** 0.1 <= k < 0.3: 18. count[1] += 1 19. **elif** 0.3 <= k < 1: 20. count[2] += 1 21. **else**: 22. **print**("错误") 23. **print**("一等奖个数{0}二等奖个数{1}三等奖个数{2}".format(count[0], count[1], count[2]))  26. x = int(input("请输入抽奖多少次：")) 27. My\_gift(x)     输入模拟抽奖1000次，输出得到一等奖106人、二等奖198人、三等奖696人，实验结果与预期相符合，算法正确。  **7、【一年365天，如果好好学习时能力值比前一天提高1%，当放任时相比前一天下降1%，编程计算两种情况效果相差值】**    图14能力值问题流程图   1. # 7、一年365天，如果好好学习时能力值比前一天提高1%，当放任时相比前一天下降1%，编程计算两种情况效果相差值。 3. **def** My\_Study(day): 4. num\_pro = 100 5. num\_des = 100 6. **for** i **in** range(day): 7. num\_pro = num\_pro \* 1.01 8. num\_des = num\_des \* 0.99 9. diff = (num\_pro - num\_des) / num\_pro \* 100 10. **print**("{0}天后，好好学习成绩：{1:.2f}；放任后成绩{2:.2f};好好学习成绩与放任成绩相比：{3:.2f}%".format(day, num\_pro, num\_des, diff))  13. x = int(input('请输入多少天：')) 14. My\_Study(x)     输入365天，原先100分的基础，每天提升1%，最后成绩3778.34分，但是每天放任1%，最后只有2.55分，相比差了99.93%，实验结果与预期相符合，算法正确。  **8、【凯撒加密算法原理与实现】**    凯撒密码最早由古罗马军事统帅盖乌斯·尤利乌斯·凯撒在军队中用来传递加密信息，故称凯撒密码。此为一种位移加密手段，只对26个（大小写）字母进行位移加密，规则相当简单，容易被破解。凯撒密码的替换方法是通过排列明文和密文字母表，密文字母表示通过将明文字母表向左或向右移动一个固定数目的位置。    图15凯撒加密流程图   1. # 凯撒加密算法原理与实现。 2. **import** string  5. **def** My\_kaisa(s, k): 6. low = string.ascii\_lowercase    # 小写字母 7. up = string.ascii\_uppercase     # 大写字母 8. before = string.ascii\_letters   # 变换前的字母表 9. after = low[k:] + low[:k] + up[k:] + up[:k] 10. table = ''.maketrans(before, after) 11. new\_s = s.translate(table) 12. **print**("凯撒加密后前的字符串：{0}-->凯撒加密后的字符串{1}".format(s, new\_s))  15. m = input("请输入代加密字符：") 16. n = int(input("请输入凯撒加密移动数：")) 17. My\_kaisa(m, n)     如图所示，输入【Hello My Name is Akaxi】经过凯撒加密（移动三个字母单位）后，输出【Khoor Pb Qdph lv Dndal】，实验结果与预期相符合，算法正确。 |

|  |
| --- |
| **五、实验测验总结与体会**  实验一：学会了定义函数的运用，并且在函数中运行if判断，计算输出的身体BMI值，函数的传参，函数的使用。  实验二：学会了eval函数isinstance函数判断整数与除法的合法性，在自定义函数中学会了函数调用判断。  实验三：学会了使用蒙特卡洛法计算圆周率，学会了重要随机库random的使用，利用循环结构加上判断条件，把抽象的数学转化为代码，实现了圆周率的近似，科学的魅力，代码世界的魅力无穷。  实验四：学会了函数相互调用，6174思想与枚举遍历思想，数字大小排序，将复杂的任务划分为部分的化整为零思想。  实验五：学会了解决约瑟夫环的思想，通过画图模拟，将抽象的事情一步一步利用Python代码实现，利用强大的列表+循环解决。  实验六：学会了随机库random的使用，结合循环判断条件实现了转盘抽奖的设计。  实验七：学会了递归循环的思想，每天都是前一天的基础上进行增加/减少，也告诉我们学习积累的重要性，坚持努力收获满满。  实验六：学会了string库的使用，对小写字母、大写字母的处理，以及列表的切片处理，maketrans映射表的使用。 |
| **六、评阅意见**  评阅人签字：  评阅日期： |