1 个人项目

1.1 缺陷记录日志

缺陷记录日志

日期 **2019.03.18**

程序号 **ase_hyz_02**

日 期	编号	类型	引入阶段	排除阶段	修复时间	相关缺陷
2019.03.18	70	**数据**	**编码**	**编码**	3h	
	项目实现逻辑不合理					
2019.03.18	10	**注释**	**编码**	**编码**	5min	
	注释描述太少，不利于别人观看代码					
2019.03.18	40	**赋值**	**编码**	**运行**	1h	
	在部分条件下没有为全局变量重新赋值，导致了代码实现有歧义					
2019.03.18	50	**接口**	**引入**	**引入**	30min	
	函数返回两个参数，接收时用两个变量名直接接收，并不能实现对应的赋值					
2019.03.18	20	**语法**	**编码**	**编译**	2min	
	If/else 的使用要在语句后面没有加冒号，导致了编译直接报错					
2019.03.18	80	**函数**	**编码**	**编译**	10min	
	引入类的过程没有在 main 函数中实例化变量直接调用，显然是不可能实现的					

缺陷类型标准

类型编号	类型名称	描　述
10	文档	注释，消息
20	语法	拼写，标点符号，打字，指令格式
30	联编打包	变更管理，库，版本控制
40	赋值	说明，重名，作用域，限制
50	接口	过程调用和引用，输入/输出，用户格式
60	检查	出错信息，不合适的检查
70	数据	结构，内容
80	函数	逻辑，指针，循环，递归，计算，函数缺陷
90	系统	配置，记时，内存
100	环境	设计，编译，测试，其它支持系统问题

1.2 效能分析报告

2 两人合作（结对编程）

2.1 项目升级描述

项目首先从面向对象方面考虑，修改了项目的需求分析逻辑，实现了更有实际意义的四则运算，引入了 random 库，实现整数的生成。由用户输入四则运算和参数，系统给出正确答案升级成为用户选择四则运算，系统根据用户的选择生成一个两个一百以内的数组成的四则运算，由用户输入答案，根据用户的输入检测答案是否正确，如果答案输入错误，则给出正确答案。运算结束以后可以由用户自行选择是否继续进行四则运算，输入‘n’或者其他字符都将自动退出程序，只有选择‘y’时才会继续下一个运算。

2.2 结对经验及感受

结对合作最大的好处是考虑问题比较全面，代码最主要的是有可复用性，如果陷入一个人的固性思维，很难达到良好的用户体验度。而且两个人掌握的专业知识肯定大于一个人，在项目功能的实现上能够想出更多更好的办法实现。在代码的测试时也会测试的更全面，减少一些不必要的错误。

3 团队合作

3.1 创建 GitHub

3.2 代码编写规范

一. 代码编排

总体原则：代码整齐、整洁。便于阅读。

① 缩进　4 个空格的缩进（编辑器都可以完成此功能），不要使用 Tap，更不能混合使用 Tap 和空格。

② 每行最大长度 79，换行可以使用反斜杠，最好使用圆括号。换行点要在操作符的后边敲回车。

③ 类和 top-level 函数定义之间空两行；类中的方法定义之间空一行；函数内逻辑无关段落之间空一行；其他地方尽量不要再空行。

二 代码编排

① 模块内容的顺序：模块说明和 docstring—import—globals&constants—其他定义。其中 import 部分，又按标准、三方和自己编写顺序依次排放，之间空一行。

② 不要在一句 import 中多个库，比如 import os, sys 不推荐。

③ 如果采用 from XX import XX 引用库，可以省略‘module.’，都是可能出现命名冲突，这时就要采用 import XX。

三 空格的使用

总体原则，避免不必要的空格。

① 各种右括号前不要加空格。

② 逗号、冒号、分号前不要加空格。

③ 函数的左括号前不要加空格。如 Func(1)。

④ 序列的左括号前不要加空格。如 list[2]。

⑤ 操作符左右各加一个空格，不要为了对齐增加空格。

⑥ 函数默认参数使用的赋值符左右省略空格。

⑦ 不要将多句语句写在同一行，尽管使用‘；’允许。

⑧ if/for/while 语句中，即使执行语句只有一句，也必须另起一行。

四 注释

总体原则，错误的注释不如没有注释。所以当一段代码发生变化时，第一件事就是要修改注释！注释最好是完整的句子，首字母大写，句后要有结束符，结束符后跟两个空格，开始下一句。如果是短语，可以省略结束符。

① 块注释，一段代码前增加的注释。在‘#’后加一空格。段落之间以只有‘#’的行间隔。

② 行注释，一句代码后加注释。比如：x = x + 1 # Increment x 。但是这种方式少使用。

③ 避免无谓的注释。

五 命名规范

总体原则，新编代码必须按下面命名风格进行，现有库的编码尽量保持风格。

① 尽量单独使用小写字母‘l’，大写字母‘0’等容易混淆的字母。

② 模块命名尽量短小，使用全部小写的方式，可以使用下划线。

③ 包命名尽量短小，使用全部小写的方式，不可以使用下划线。

④ 类的命名使用 CapWords 的方式，模块内部使用的类采用_CapWords 的方式。

⑤ 异常命名使用 CapWords+Error 后缀的方式。

⑥ 全局变量尽量只在模块内有效，类似 C 语言中的 static。实现方法有两种，一是__all__机制;二是前缀一个下划线。

⑦ 函数命名使用全部小写的方式，可以使用下划线。

⑧ 常量命名使用全部大写的方式，可以使用下划线。

⑨ 类的属性（方法和变量）命名使用全部小写的方式，可以使用下划线。

⑩ 类的属性有 3 种作用域 public、non-public 和 subclass API，可以理解成 C++中的 public、private、protected，non-public 属性前，前缀一条下划线。

11 类的属性若与关键字名字冲突，后缀一下划线，尽量不要使用缩略等其他方式。

12 为避免与子类属性命名冲突，在类的一些属性前，前缀两条下划线。比如：类 Foo 中声明__a,访问时，只能通过 Foo._Foo__a，避免歧义。如果子类也叫 Foo，那就无能为力了。

13 类的方法第一个参数必须是 self，而静态方法第一个参数必须是 cls。

3.3 团队代码评审表

4 项目代码及实验心得

```
'''
作者：韩育珍
版本：第三周作业
新增功能 1：四则运算
新增功能 2：由系统自动生成四则运算
'''
import random

#判断用户输入的答案是否正确
def true_or_false(answer,userAnswer):

    if (userAnswer == answer):
        print('回答正确')
    else:
        print('回答错误,正确答案是:{},还需要多加练习'.format(answer))

#四则运算
def subtracting_asmd():

    subtracting = input("请选择四则运算: (add/sub/mul/div) ")
    userAnswer = 0
    answer = 0
    sub1 = random.randint(1, 100)  # 产生 1 到 10 的一个整数型随机数
    sub2 = random.randint(1, 100)
    if (subtracting == 'add'):
        userAnswer = int(input('{}+{}='.format(sub1, sub2)))
        answer = sub1 + sub2
        true_or_false(answer,userAnswer)
    elif (subtracting == 'sub'):
        userAnswer = int(input('{}-{}='.format(sub1, sub2)))
        answer = sub1 - sub2
        true_or_false(answer,userAnswer)
    elif (subtracting == 'mul'):
        userAnswer = int(input('{}*{}='.format(sub1, sub2)))
        answer = sub1 * sub2
        true_or_false(answer,userAnswer)
    elif (subtracting == 'div'):
        userAnswer = int(input('{}/{}='.format(sub1, sub2)))
        #被除数为 0 的情况
        if (sub2 == 0):
            print("被除数不能为 0, 请重新输入")
```

```
            answer = subtracting_asmd()
        else:
            answer = float(sub1 / sub2)
        true_or_false(answer,userAnswer)
    else:
        print('不支持此种四则运算输入格式')
        answer = subtracting_asmd()
```

```
#主函数
def main():
    is_operation = True
    while(is_operation):
        subtracting_asmd()
        yes_or_no = input('是否继续运算：（y/n）')
        if(yes_or_no == 'n'):
            is_operation = False
```

```
#主函数入口
if __name__ == '__main__':
    main()
```

实验心得

①编写代码最重要的就是要具有实用性。在初始代码中实现了由用户选择四则运算，并输入运算需要的两个参数，然后给出正确结果，显然这样的逻辑没有实际的作用，因此花费了一些时间改进了代码的功能，由原先的功能更改为用户选择四则运算后系统会自动生成一道相应的运算题目，由用户计算，并给出计算结果正确与否。可以由用户自行选择是否继续运算。这样就保证了代码的合理以及实用性。

②编写代码要具有一定的条理性以及便利性，代码的实现要具有可复用性。初始代码都是通过自己编写的一个简单的四则运算，对于变量等都尽量采用规则命名，因此基本上没有添加注释，但是这样每个人的编程习惯不同，不能理所当然的以为自己看的明白别人也看的明白，这显然不是一个计算机学生该有的思维模式，于是在新的代码中添加了比较合理的注释信息。

③代码的优化过程中要关注全局的代码，方便查找问题，修改 BUG。由于采用了较多的 if/else 判断语句，在函数的起始定义变量并复制之后对于大部分的语句都是能够合理运行的，但是在 else 语句的条件下就会产导致输出结果有歧义。为此在大部分条件下要用的的共同的方法重新定义了一个函数并调用，从而合理的解决了歧义问题。

④要准确的明白函数的返回和调用之间值的接收方式。错误的使用会造成代码修复的困难性。优化后的函数返回两个变量，接收时用两个参数接收，想达到对应赋值，但是变量接收和函数接收两个对应的参数是不相同的，变量接收函数的两个返回值，实际上是把两个值放在一个集合中传给了第一个变量，第二个变量接收到的是一个 0。