编程规范v1.0.1

为了使代码具有较高的可读性，简单来说就是让其他人更好地理解代码，我们有必要说明一下编程规范

一、命名

1.变量命名

变量命名要**简单，直白，表达准确**

对于循环用变量，不做具体要求，只用一个字母即可。不过要注意变量的用意，不要混乱。

代码1-1

```
for i in range(10):

    for j in range(4):

        print("helloworld",end='\t')

    print()
```

对于用于赋值，计算，管理的变量，要根据所表达的数据，适当进行**压缩，生成变量名称，并适当作注释。**

比如我们要设置一个变量为输入的学号，我们可以以下面这种格式命名

代码1-2

```
StuNum = int(input()) #学号
```

学生号码英文全称为 student number，则我们要把**前后两个单词取几个（至少三个）字母并将每个单词所取字母的第一个字母大写**

这几个字母不一定是每个单词的前几个字母，只要能够具有可识别性即可。

2.函数命名

当我们对于某一常用的算法，我们有必要打包，叫做定义为一个函数

函数的命名规则与变量相同，但要**在两个单词之间加上下划线“_”，并在前一行进行注释并以括号的形式标明需要引用的变量的名称以及含义**，如、

代码1-3

```
#信息录入（学生姓名，性别，学号，电话号）
def Infor_Write(StuName,Sex,StuNum,StuCall):
    BODY
```

3.注释

一些简短注释，直接在所要注释的代码行后注释即可，如代码1-2；

对于一些较长的代码以及自定义函数，除长代码要适当换行以外（后面会讲到），要将注释放到其所注释的代码行之前，如代码1-3

对于短代码需要很长的一段注释，需要将注释从代码的后一行写起并适当换行

二、代码格式

1.缩进

对于python而言，做主要的表达层次关系的格式就是缩进，python是**严格缩进**的，所以，如果算法和语法本身没有问题而不能运行，请首先检查缩进是否严格。

代码2-1 错误的缩进格式

```
for i in range(10):

  for j in range(4):

        print("helloworld",end='\t')

      print()
```

要保证每相邻两个层级之间的缩进为四个空格，即一个Tab（键盘上）

2.适当插入空行

一个文件里的代码可能会有上百行，将许多算法堆在一起不便于其他人读取代码内容

适当的空行对代码没有影响，但可以保持整个文件内代码的美观

3.长代码换行

当一行代码过长（如果是必要的话），全部写在一行并不方便其他人读取。为此，长代码应该**在适当的地方进行换行**

这里所讲的换行，不是在某一函数后换行，而是在函数后括号内换行。虽然前者语法上并不会出现错误，但是代码的层次关系表达不如后者。以下为两种换行方法：

代码2-2 括号内换行方式

```
print("此曲只应天上有，人间能得几回闻",
       end = '\t')
```

代码2-3 函数后换行方式

代码中的“\”为IDE自动生成

```
print\
("此曲只应天上有，人间能得几回闻", end = '\t')
```

4.空格

对于空格，在任何编程语言中，有些地方是要求必须加入空格，有的地方是不必要加入空格的。所以对于后者，我一般都是不加空格，而PyCharm不会自动补入的，而Visual Studio以及其他的IDE或Code Editor会自动补入的。这里建议适当加入

代码2-4 非必要空格

```
a=1
b = 2
c=a+b
d = b - a
print(c,end='\n')
print(d, end = '\t')
range[1,2,3]
s=[1,24,657,980,76]
```

代码2-5 必要空格（条件以及循环函数）

```
while i > 0:
    if i < 9:
        for j in range(3):
            print('hello')
        i+=1
    if i = 9:
        break
```

因为python的格式相对简单而且严格缩进，所以对于格式上的硬性要求较少，下面以上次日历的作业为例，放一个规范的程序

```
#获取所要输出的月历的年月
def get_year_month():
    year = eval(input('请输入年份：'))
    month = eval(input('请输入月份：'))
    return year, month #输出年月的值

#计算本月第一天是星期几（年份，月份）
def date_of_first_day(year,month):
    day = 0
    #这个数决定每星期的开头 我试了下任何自然数都可以 起决定性的是与7相除的余数
    #这里就相当于在每月1日之前加上的天数
    #加上之后每月1日的起点会向后推 因此 后面的星期也会相应提前
    #于是不同的数字每星期第一天也不一样
    #0 星期一 1 星期日 2 星期六 ... 6 星期二 7(0) 星期一 ...
    y0 = year-(14-month)//12
    x = y0 + y0 // 4 - y0 // 100 + y0 // 400
    m0 = month + 12 * ((14-month) // 12) - 2
    date = (day + x + (31 * m0) // 12) % 7
    #根据蔡勒公式计算星期几
    return date

#判断该年是否为闰年（年份）
def is_leap_year(year):
```

```
    isLeapYear = (year % 4 == 0)
    isLeapYear = isLeapYear and (year % 100 != 0)
    isLeapYear = isLeapYear or (year % 400 == 0)
    return isLeapYear

#确定该月的天数（年份）
def days_of_month(year,month):
    days = 0
    if month in [1,3,5,7,8,10,12]:
        days = 31
    elif month in [4,6,9,11]:
        days = 30
    else:
        if is_leap_year(year):
            days = 29
        else:
            days = 28
    return days

#打印日历（年，月，星期，天数）
def print_calendar(year,month,date,days):
    print_head(year,month)
    print_body(date,days)

#print_calender()的子函数，输出表头（年，月）
def print_head(year,month):
    print('\t\t {}年{}月'.format(year,month))
    print('一\t二\t三\t四\t五\t六\t日')

#print_calender()的子函数，输出表体（星期，天数）
def print_body(date,days):
    count=date
    for i in range(date):
        print('\t',end="")
    for d in range(1,days+1):
        print(str(d)+'\t',end="")
        count=(count + 1) % 7
       #每七天一个循环，换行
        if count == 0:
            print()
    print()

#输出日历全体
def month_calendar():
    #对各个函数进行赋值
    year,month = get_year_month()
    date = date_of_first_day(year,month)
    days = days_of_month(year,month)
    print_calendar(year,month,date,days)

month_calendar()
```