# 编程技巧说明

目录

[编程技巧说明 1](#_Toc53611238)

[1.使用from导入 2](#_Toc53611239)

[2.面向对象 2](#_Toc53611240)

[继承 4](#_Toc53611241)

[使用私有属性 4](#_Toc53611242)

[3.有关函数 4](#_Toc53611243)

[封装一些常用函数 4](#_Toc53611244)

[使用controlState() 5](#_Toc53611245)

[4.减少全局变量 6](#_Toc53611246)

[5.关于Debug 6](#_Toc53611247)

[补充: 使用git 6](#_Toc53611248)

## 1.使用from导入

首先，我使用了PyGame这个第三方模块

为了方便，我使用了from pygame.locals import \*的方法导入，接下来handleEvent就会简洁许多。

看我的handleEvent:(部分)

 for event in pygame.event.get():  
     if event.type == pygame.QUIT or (event.type == KEYDOWN and event.key == K\_ESCAPE):  
         judgment = easygui.boolbox("是否退出游戏?", "注意", ["确定退出", "算了，再玩会儿"])  
         if judgment:  
             pygame.quit()  
             sys.exit()  
         else:  
            ...

如果不是使用from导入，这里将会多写好多pygame.locals, 不仅麻烦，可读性也差。

 for event in pygame.event.get():  
     if event.type == pygame.QUIT or (event.type == pygame.locals.KEYDOWN and event.key == pygame.locals.K\_ESCAPE):  
         judgment = easygui.boolbox("是否退出游戏?", "注意", ["确定退出", "算了，再玩会儿"])  
         if judgment:  
             pygame.quit()  
             sys.exit()  
         else:  
            ...

## 2.面向对象

我将每一个圆点视为一个对象，这样就会很方便，随时可以调用相关数据：

 class CharacterObject(object):  
     """所有人物的父类"""  
 ​  
     def \_\_init\_\_(self, x, y, width, height, colour):  
         self.attributeDict = {"People": {"Health": 0, "Symptoms": 1, "Asymptomatic": 2, "Death": 5},  
                               "Doctor": {"Health": 3, "Symptoms": 4}}  
         self.x = x  # x坐标  
         self.y = y  # y坐标  
         self.width = width  # 宽  
         self.height = height  # 高  
         self.life = 300  # 基础生命值  
         self.immunity = random.randint(50, 100)  # 免疫力  
         self.defensive = self.life + self.immunity  # 总防御值  
         self.can\_infectOther = True  # 能否感染他人  
         self.infectedPeoples = 0  # 被self感染的人数  
         self.infectedTime = None  # 被感染时的时间  
         self.will\_go\_hospital = False  # 即将去医院  
         self.in\_hospital = False  # 在医院  
         self.life\_decline\_lastTime = time.time()  # 上次减少生命值的时间  
         self.rehabilitate\_lastTime = time.time()  # 上次自我修复时的时间  
         self.attribute = self.attributeDict["People"]["Health"]  # 属性  
         self.colourDict = {"green": 0, "yellow": 1, "red": 2, "dh": 3, "ds": 4, "death": 5}  # 储存不同颜色  
         self.colour = self.colourDict[colour]  # 确定颜色  
         # 通过colour确定attribute以及image\_path  
         if colour == 0:  
             self.attribute = self.attributeDict["People"]["Health"]  
         if colour == 1:  
             self.attribute = self.attributeDict["People"]["Asymptomatic"]  
         if colour == 2:  
             self.attribute = self.attributeDict["People"]["Symptoms"]  
         if colour == 3:  
             self.attribute = self.attributeDict["Doctor"]["Health"]  
         if colour == 4:  
             self.attribute = self.attributeDict["Doctor"]["Symptoms"]  
 ​  
         self.image\_path = "..\\data\\images\\" + str(self.colour) + ".png"  # 图片路径  
         self.load\_image = pygame.image.load(self.image\_path)  # 预加载图片  
 ​  
     def hit(self, component):  # 判断碰撞  
         c = component  
         return c.x > self.x - c.width and c.x < self.x + self.width and \  
                c.y > self.y - c.height and c.y < self.y + self.height  
 ​  
     def paint(self):  # 绘制self  
         canvas.blit(self.load\_image, (self.x, self.y))  
 ​  
     def reset\_pos(self):  # 重置坐标  
         if self.x <= 70: self.x = random.randint(90, bgWidth - 50 - 20 - 20)  
         if self.x >= bgWidth - 50 - 20: self.x = random.randint(90, bgWidth - 50 - 20 - 20)  
         if self.y <= 20: self.y = random.randint(50, bgHeight - 50)  
         if self.y >= bgHeight - 20: self.y = random.randint(50, bgHeight - 50)  
 ​  
     def CanInfectOther\_qm(self):  # qm: question mark  
         if self.infectedPeoples == GameVar.RO:  
             self.can\_infectOther = False  
 ​  
     def reloadImage(self):  # 重载图片  
         self.image\_path = self.image\_path = "..\\data\\images\\" + str(self.colour) + ".png"  
         self.load\_image = pygame.image.load(self.image\_path)  
 ​  
 .............更多请阅读main.py

### 继承

之前是CharcterObject的部分代码，源码下面还有Doctors Peoples两个类都是继承CharcterObject的。

### 使用私有属性

因为有时候类里的函数只需要自身调用，无序外部调用，因此可以在名称前加上一条下划线\_来表示这是私有属性。

 # 举例  
 class Ball:  
     def \_\_init\_\_(self):  
         self.ball = "football"  
       
     def \_play(self):  
         self.ball = "playball"  
 # 外部无法调用\_play()函数

## 3.有关函数

我认为，能封装成函数/类的就绝不零零散散地写得乱七八糟。

比如，我将运行时需要用到的函数根据不同用途封装成了generate componentPaint componentMove showData ChangingProperties 等函数。

### 封装一些常用函数

比如在我的代码里有许多的isActionTime , 这就是个常用函数， 还有之前提到的handleEvent也属于常用函数。

isActionTime是这样的：

 def isActionTime(lastTime, interval):  
     currentTime = time.time()  
     return currentTime - lastTime >= interval

一看就懂，是判断时间的。

在接下来的componentPaint 之类的函数里会经常用到这个isActionTime.

### 使用controlState()

以前，我经常会遇到一个问题：游戏的状态切换很困难，有时候还在Start时就开始运行了Running改运行的东西。我的解决办法是：使用一个controlState来控制状态，请看部分源码：

 def controlState():  
     if GameVar.state == GameVar.STATES["START"]:  
         key = starter()  
         if type(key) == str:  
             GameVar.state = GameVar.STATES[key]  
             print(key)  
         handleEvent()  
 ​  
     if GameVar.state == GameVar.STATES["LOGIN"]:  
         key = login()  
         if type(key) == str:  
             GameVar.state = GameVar.STATES[key]  
             print(key)  
         handleEvent()  
 ​  
     if GameVar.state == GameVar.STATES["RUNNING"]:  
         handleEvent()  
         componentPaint()  
         componentMove()  
         ChangingProperties()  
         showData()  
     if GameVar.state == GameVar.STATES["PAUSE"]:  
         componentPaint()

通过部分代码， 可以看出这样的好处：

* 简介、明了、可读性高
* 逻辑清晰、不会感到混乱

**也就是说，这样的效率很高。**

## 4.减少全局变量

虽然Python不存在内存的问题，但是还是少用全局变量为妙。

为了减少全局变量，我使用了一个类GameVar (*Var*是*Variable*的简称)来存放变量，这样就只有少数全局变量(bgWidth bgHeight one\_day等).

另外，这是一个比较另类的类，因为它不需要实例化：

 class GameVar:  
     Name = Value

GameVar差不多都是这种格式， 存放了peoples patients等十分重要的变量。

需要用到GameVar里面的变量时， 直接GameVar.xxxx即可。

## 5.关于Debug

调试一直都是一件让人头疼的事。我来介绍一下我主要使用的调试方法：

首先， 可以使用一个变量Demo, 当Demo为“\_\_main\_\_”时， 运行主程序。当Demo为“demo”时，运行你要调试的函数。

还可以使用PyCharm自带的Debug调试。

## 补充: 使用git

Git 是一个开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。

Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

Git 与常用的版本控制工具 CVS, Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持。

--------来源：[菜鸟教程：Git](https://www.runoob.com/git/git-tutorial.html)

Git可以很好的管理你的代码， 可以恢复任何一个commit的版本， 还有不同的分支。

 git init   
 git add -A  
 git commit -m "注释"  
 git remote add origin [网址]  
 git push -u origin master

Git这里我不在赘述， 因为毕竟与代码关系不大。

**我的编程技巧就这些了，感谢您的阅读！**