Ball的注释

```
import pygame
from data.config import g#注意路径问题，以及这里只import了config里面的常量g
from components import tools#引入tools
class Ball(pygame.sprite.Sprite):#Ball是一个pygame.sprite.Sprite的子类
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)#sprite对象都要初始化，都是这一句
        self.image =
pygame.transform.scale(pygame.image.load("./images/ball.png","The
ball").convert(),(120,120))
        #这句话可以分为两部分，一部分是pygame.transform.scale(image,size)，而其中的
image就是pygame.image.load(path, notes).convert()
        #pygame.image.load(path, notes)可以从path路径加载一个图片，notes是注释。而
后面加.convert()是pygame的一个方法，可以便于pygame渲染，有兴趣了解原理的可以自行学习
        #pygame.transform.scale(image, size)可以把image缩放成size，比较好理解
        #所以这句话总的意思就是读入一个图片，然后把它缩放成120x120
        self.rect = self.image.get_rect()#获得这个球对象的rect，因为pygame是基本
在操作rect的
        self.image.set_colorkey((255,255,255))#按(255,255,255)抠图
        self.windowsInfo = pygame.display.Info()#这里上课也没讲，是获取你的窗口的
宽高信息，返回一个二元组
        self.rect.center =
(self.windowsInfo.current_w/2,self.windowsInfo.current_h/2)#设置球的初始位置
        self.speed = 0#设置球的初始速度

    def limit(self, SCREEN_WIDTH, SCREEN_HEIGHT):#这是限制函数，主要用于限制球在
窗口内运动
        if self.rect.left <= 0 :#左
            self.rect.left = 0
        if self.rect.right > SCREEN_WIDTH:#右
            self.rect.right = SCREEN_WIDTH
        if self.rect.top <= 0 :#上
            self.rect.top = 0
        if self.rect.bottom > SCREEN_HEIGHT:#下，同时反弹
            self.rect.bottom = SCREEN_HEIGHT
            self.speed = (self.speed*-2)/3
    def onGround(self,SCREEN_HEIGHT):#判断球是否在地上
        if self.rect.bottom >SCREEN_HEIGHT:
            return True
        else:
            return False
    def move(self, posx, posy):#移动球到posx ,posy
        self.rect.center = (posx, posy)
        self.speed = 0
    def update(self):#更新球，这里这个函数被主函数里的sprites.update()调用了
        self.limit(self.windowsInfo.current_w, self.windowsInfo.current_h)#
时时刻刻限制球
        if((abs(self.speed)<0.2)&(self.onGround(self.windowsInfo.current_h-
1))):#这里是为了防止一个小bug，球会在地板上一直小幅度上下移动，可以把这个注释掉，把下面的
else去掉然后看下效果
            self.speed = 0
        else:
            self.rect = self.rect.move([0, self.speed])#球自然下落
```

```
            self.speed += g#模拟重力加速度

```

这里我只细讲一下上课没讲清楚的类和属性和方法，关于Ball.py的注释在上面打上了。

关于类：

- 类就是字面意思，一个类代表着一类东西，这类东西可以有他们自己的操作（也就是我们所说的方法，函数等等），或者自己的属性（长宽高重等等）。
- 在python中，类的定义语法如下

```
class myClass():#创建一个名字叫myClass的类，括号内可以有参数，表示是它的父类是谁
    def __init__(self[,a,b,c,……]):#这个函数是定义类一般会有的方法，每次你创建一个新的对象的时候都会调用一次，也就是初始化。其中参数可以自己定，有几个需要的就写几个。
        #your code
```

这里参数的含义可能还是不太好理解，我举个例子。

现在你定义了一个Animal类：

```
class Animal():
    def __init__(self):
        pass
```

现在它里面什么都没有，你创建一个它的对象

```
myAnimal = Animal()
```

不会有任何反应，它也只是一个空对象。

我现在可以给它加上一些方法，比如所有的动物都会吃。

```
class Animal():
    def __init__(self):
        pass
    def eat(self):
        print("Now animal is eating.")
```

这里有一点需要注意，所有类的内置方法第一个参数都必须是self，表示这个对象它自己，但是在调用的时候不需要指明。比如我现在调用。

```
myAnimal.eat()
```

会输出Now animal is eating.

那么现在，animal有名字了，这是它的属性，我们可以给它加上：

```
class Animal():
    def __init__(self, name):
        self.name = name #想一想，这里前面和后面的name是同一个东西吗？它们的值相等吗？
    def eat(self):
        print("Now " +self.name+" is eating.")
```

这时我再创建一个Animal对象，就必须加上它的名字了，比如：

```
myAnimal = Animal("Cat")
```

这时再运行

```
myAnimal.eat()
```

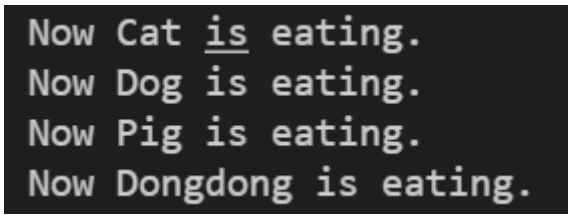
会输出"Now Cat is eating."

同样，我们可以创建很多很多个不同的对象，比如

```
Cat = Animal("Cat")
Dog = Animal("Dog")
Pig = Animal("Pig")
Dongdong = Animal("Dongdong")
```

我们分别调用他们的eat()方法：

```
Cat.eat()
Dog.eat()
Pig.eat()
Dongdong.eat()
```



这就是类和对象。同一个类下的对象拥有所有类的属性和方法，但是他们同时又是相互独立的。

之前讲到列表可以放其它类型，我们也可以放对象进去。

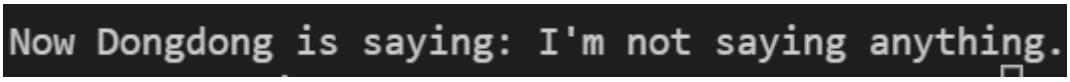
```
Cat = Animal("Cat")
Dog = Animal("Dog")
Pig = Animal("Pig")
Dongdong = Animal("Dongdong")
animalList = [Cat, Dog, Pig, Dongdong]
for animal in animalList:
    animal.eat()
```

这样就可以方便调用。甚至，我们还可以直接创建多个对象，有兴趣的同学可以自己尝试用循环创建一个有多个不同对象的列表。

再稍微提一下父类和子类，这里包含了继承和多态，不多讲。

我现在有了一个新类Human，它属于Animal，那么我们可以：

```
class Animal():
    def __init__(self, name):
        self.name = name #想一想，这里前面和后面的name是同一个东西吗？它们的值相等吗？
    def eat(self):
        print("Now " +self.name+" is eating.")
class Human(Animal):#这里表示Human类是Animal的子类，所有Human对象都拥有Animal的所有属性和方法
    def __init__(self, name):
        Animal.__init__(self, name)
    def speak(self, sentence):
        print("Now "+self.name+" is saying: "+sentence)
Dongdong = Human("Dongdong")
Dongdong.speak("I'm not saying anything.")
```



这里有一个需要注意的点，虽然继承了父类，但是父类的属性并没有初始化，因此要在Human的__init__方法里手动初始化Animal的name。

在类内部调用对象的属性啊方法啊之类的，使用self.xxx