

L12-4 | 重返地球

# 模块47: 寻找猴赛雷 & 模块48: debug大挑战



ANALYTICAL  
INTELLECTUALLY CURIOUS  
CREATIVE



## 项目内容

### 项目内容

添加角色

角色初始位置

键盘操作&角色移动

鼠标操作&切换造型

综合应用

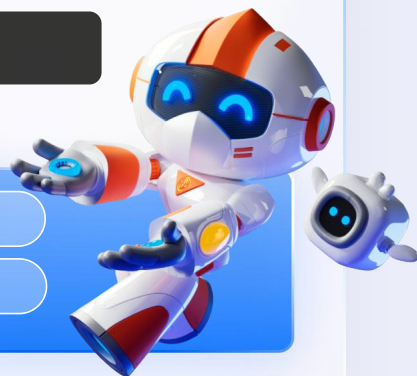
### 单词卡

play:播放

stop:停止

debug:调试

button:纽扣





## 添加角色



添加角色需要两个步骤：第一步创建角色，第二步绘制角色。

### 知识复习

**Actor()**命令可以用于创建角色。放置不同图片的名称，就可以创建不同样貌的角色。



格式：角色变量 = **Actor**('角色图片的名称')

绘制角色的命令，通常需要编写在**draw()**函数里。



格式：角色名.**draw()**



代码示例

```
import pgzrun
```

```
WIDTH = 800
```

```
HEIGHT = 600
```

-----> 设置窗口大小

```
hsl = Actor('箱子') -> 创建hsl角色
```

```
def draw(): -----> draw()函数：用于在窗口中绘制图像的函数
```

```
    screen.clear()
```

```
    screen.blit('仓库', [0, 0])
```

```
    hsl.draw() -----> 在窗口中绘制hsl角色
```

```
pgzrun.go()
```



## 角色初始位置

### 概念详解

- 创建好角色后，使用**角色.x**和**角色.y**的数值，可以给角色**设置一个初始位置**。
- 如果不设置角色的位置，它就会默认出现在**窗口的左上角**。

### 代码示例

```
hsl = Actor('箱子') --> 创建hsl角色
```

```
hsl.x = 500  
hsl.y = 0 -----> 设置角色初始位置
```

```
def draw(): -----> 绘制图像的函数  
    screen.clear()  
    screen.blit('仓库', [0, 0])  
    hsl.draw()
```



## 键盘操作&角色移动

### 知识复习

- `on_key_down()`函数是与**键盘操作**相关的函数，按下按键时执行。
- 增大或减小角色的**x坐标值**可以让角色**左右移动**；
- 增大或减小角色的**y坐标值**可以让角色**上下移动**。

### 代码示例

```
def on_key_down(key): --> 按键按下时执行，参数key  
                           中存放了按键的名称  
    if key == keys.LEFT: --> 如果按下的是左方向键  
        hsl.x -= 5 -----> 角色的x坐标减小，向左移动  
    if key == keys.RIGHT:  
        hsl.x += 5
```



## 鼠标操作&切换造型

### 概念详解

- 与**鼠标操作**相关的两个函数：
  - `on_mouse_down()`在鼠标每次**按下**时执行，
  - `on_mouse_up()`在鼠标每次**松开**时执行。
- 角色名.**image** = 图片名 的方式可以更换角色的造型。

### 代码示例

```
def on_mouse_down(): -----> 鼠标按下时执行
    music.play('yoyo') -----> 播放歌曲yoyo
    dajisi.image = '大祭司唱歌' ---> 切换dajisi的造型
    lengxuefeng.image = '冷雪丰回头'

def on_mouse_up(): -----> 鼠标松开时执行
    music.stop() -----> 停止播放声音
    dajisi.image = '大祭司'
    lengxuefeng.image = '冷雪丰'
```





## 综合应用



在pygame中，作用不同的代码放置的位置一般也不同。

### 知识详解

需要一直执行的代码 → update()函数里面

绘制背景或角色的代码 → draw()函数里面

需要按下键盘就执行的代码 → on\_key\_down()函数里面

需要按下鼠标就执行的代码 → on\_mouse\_down()函数里面

需要松开鼠标就执行的代码 → on\_mouse\_up()函数里面



检测碰撞的代码需要一直执行，  
所以要放在update()函数里。

### 代码示例

```
def update(): ---> 在程序运行时持续被调用
```

```
    if hsl.colliderect(button): -> 如果hsl碰到button
```

```
        button.image = '按下' ----> 切换button的造型
```



## 项目日志

