

# pyautogui 学习笔记-掌握自动化操作工具——介绍Python库PyAutoGUI

---

## 快速了解

掌握自动化操作工具——介绍Python库PyAutoGUI

引言：

什么是PyAutoGUI？

PyAutoGUI的主要功能

使用PyAutoGUI实现自动化任务

PyAutoGUI的应用场景

结论：

## 代码教程

控制功能

自动防故障功能

停顿功能

鼠标操作

控制鼠标移动

控制鼠标点击

控制鼠标拖动

控制鼠标滚动

屏幕处理

获取屏幕截图

识别图像

键盘输入

键盘输入函数

键盘特殊按键

快捷键

提示信息框

提示框/警告框

选择框

密码输入

普通输入

实例

鼠标控制鼠标画一个正方形

获取鼠标的实时位置

获取鼠标位置与所在位置的颜色

爆笑，打地鼠的极限，PyAutoGUI的开始~

介绍

思路

具体代码如下：

总结

## 快速了解

## 掌握自动化操作工具——介绍Python库PyAutoGUI

### 引言：

在现代科技时代，自动化操作成为提高工作效率和减轻重复性工作负担的重要手段。而Python作为一门强大而灵活的编程语言，提供了丰富的库和工具来实现各种自动化任务。本文将重点介绍Python库PyAutoGUI，它是一个用于模拟鼠标和键盘操作的工具，可以帮助我们实现自动化操作。让我们一起来探索PyAutoGUI的强大功能和用法。

### 什么是PyAutoGUI?

- 简介：PyAutoGUI是一个Python库，它允许我们通过编程方式模拟鼠标和键盘的操作，实现自动化任务。
- 安装：通过pip命令可以轻松安装PyAutoGUI，确保你的Python环境已经正确配置。

### PyAutoGUI的主要功能

- 鼠标操作：PyAutoGUI可以模拟鼠标移动、点击、拖拽等操作，可以控制鼠标的位置和点击的坐标。
- 键盘操作：PyAutoGUI可以模拟键盘按键和组合键的操作，如按下和释放按键、输入文本等。

- 屏幕操作：PyAutoGUI可以获取屏幕的大小、截屏、查找指定图像的位置等。
- 延时控制：PyAutoGUI可以控制鼠标和键盘操作的延时，以确保操作的正确性和稳定性。

## 使用PyAutoGUI实现自动化任务

- 导入库和初始化：导入PyAutoGUI库，并进行初始化设置，如延时时间等。
- 模拟鼠标操作：使用PyAutoGUI的鼠标操作函数，模拟鼠标移动、点击、拖拽等操作。可以结合屏幕坐标和图像识别等技术来实现更复杂的操作。
- 模拟键盘操作：使用PyAutoGUI的键盘操作函数，模拟键盘按键和组合键的操作。可以实现自动化的文本输入、快捷键操作等。
- 屏幕操作和图像识别：使用PyAutoGUI获取屏幕信息，如屏幕大小、截屏和查找图像位置等。可以用来自动化识别和操作特定的图像场景。
- 错误处理和异常处理：在使用PyAutoGUI过程中，要注意异常处理和错误处理，以确保程序的健壮性和稳定性。

## PyAutoGUI的应用场景

- 自动化测试：PyAutoGUI可以模拟用户的操作，用于自动化测试各种软件和系统的功能。
- 数据采集和处理：PyAutoGUI可以帮助自动化从网页或应用程序中采集数据，并进行处理和分析。
- 软件演示和教程录制：PyAutoGUI可以用于自动化演示软件的功能和操作，录制教程视频等。
- 游戏辅助工具：PyAutoGUI可以用来制作游戏辅助工具，自动化完成一些重复性操作。

## 结论：

PyAutoGUI是一个方便且功能强大的Python库，用于实现自动化操作，无论是自动化测试、数据采集还是辅助工具开发，PyAutoGUI都是一个强大而实用的工具。通过熟练掌握PyAutoGUI，我们可以大幅提升工作效率，实现自动化操作的便利性和稳定性。让我们一起开始探索PyAutoGUI吧！

## 代码教程

### 控制功能

控制鼠标键盘使用的模块为：**pyautogui**，这个模块操作起鼠标键盘的时候，非常的迅速，而且如果该模块控制了鼠标后，程序比较难关闭，这时我们有两个方法专门针对以上的情况：

## 自动防故障功能

```
1 pyautogui.FAILSAFE = False
```

默认这项功能为True, 这项功能意味着：当鼠标的指针在屏幕的最坐上方，程序会报错；目的是为了  
防止程序无法停止；

## 停顿功能

```
1 pyautogui.PAUSE = 1
```

意味着所有pyautogui的指令都要暂停一秒；其他指令不会停顿；这样做，可以防止键盘鼠标操作  
太快；

## 鼠标操作

### 控制鼠标移动

#### 获得屏幕分辨率

```
1 # 返回所用显示器的分辨率；
2 # 输出: Size(width=1920, height=1080)
3 print(pyautogui.size())
4 width,height = pyautogui.size()
5 print(width,height)
6 # 1920 1080
```

#### 移动鼠标

Python |

```
1 pyautogui.moveTo(100,300,duration=1)
```

Python |

```
1 # 按方向移动，左右正负值对应右左，上下正负值对应下上
2 # moveRel(): 这是PyAutoGUI库中的一个函数，用于模拟相对于当前鼠标位置的移动操作。
3 # 第一个参数是左右移动像素值，第二个是上下，向右移动100px，向下移动500px，这个过程持续
  1 秒钟；
4 pyautogui.moveRel(100,500,duration=1)
```

## 获取鼠标位置

Python |

```
1 print(pyautogui.position()) # 得到当前鼠标位置；输出：Point(x=200, y=800)
```

## 控制鼠标点击

Python |

```
1 # 点击鼠标
2 pyautogui.click(10,10) # 鼠标点击指定位置，默认左键
3 pyautogui.click(10,10,button='left') # 单击左键
4 pyautogui.click(1000,300,button='right') # 单击右键
5 pyautogui.click(1000,300,button='middle') # 单击中间
```

Python |

```
1 # 双击鼠标
2 pyautogui.doubleClick(10,10) # 指定位置，双击左键
3 pyautogui.rightClick(10,10) # 指定位置，双击右键
4 pyautogui.middleClick(10,10) # 指定位置，双击中键
```

Python |

```
1 # 点击 & 释放
2 pyautogui.mouseDown() # 鼠标按下
3 pyautogui.mouseUp() # 鼠标释放
```

## 控制鼠标拖动

```
1 # 拖动到指定位置，相对于原点
2 pyautogui.dragTo(100,300,duration=1)
3
4 # 拖动到指定位置，相对于当前位置
5 pyautogui.dragRel(100,500,duration=4)
```

## 控制鼠标滚动

控制鼠标滚动的函数是scroll(), 传入一个整数的参数, 说明向上或向下滚动多少个单位; 单位根据操作系统不同而不同;

```
1 # 向上滚动300个单位;
2 pyautogui.scroll(300)
```

## 屏幕处理

### 获取屏幕截图

我们控制鼠标的操作, 不能盲目的进行, 所以我们需要监控屏幕上的内容, 从而决定要不要进行对应的操作, pyautogui 提供了一个方法screenshot(), 可以返回一个Pillow的image对象;

这里三个常用函数:

- `im = pyautogui.screenshot()`: 返回屏幕的截图, 是一个Pillow的image对象
- `im.getpixel((500, 500))`: 返回im对象上, (500, 500) 这一点像素的颜色, 是一个RGB元组
- `pyautogui.pixelMatchesColor(500,500,(12,120,400))`: 是一个对比函数, 对比的是屏幕上(500, 500) 这一点像素的颜色, 与所给的元素是否相同;

```
1 im = pyautogui.screenshot()
2 im.save('屏幕截图.png')
```

```

1  # screenshot()返回一个Pillow的image对象
2  im = pyautogui.screenshot()
3
4  # 输出 (500, 500) 像素点的颜色, 一个RGB元组
5  print(im.getpixel((500, 500)))
6  # (45, 42, 46)
7
8  # 判断颜色是否匹配
9  print(pyautogui.pixelMatchesColor(500, 500, (12, 120, 400)))

```

## 识别图像

首先, 我们需要先获得一个屏幕快照, 例如我们想要点赞, 我们就先把大拇指的图片保存下来;

然后使用函数: `locateOnScreen('zan.png')`,

- 如果可以找到图片, 则返回图片的位置, 如: `Box(left=25, top=703, width=22, height=22)`;
- 如果找不到图片, 则返回`None`;

如果, 屏幕上有多处图片可以匹配, 则需要使用`locateAllOnScreen('zan.png')`,

- 如果匹配到多个值, 则返回一个list, 参考如下:

```

1  # 图像识别 (一个)
2  btm = pyautogui.locateOnScreen('赞.png')
3  print(btm)
4  # Box(left=1200, top=350, width=50, height=50)
5
6  # 图像识别 (多个)
7  btm = pyautogui.locateAllOnScreen('赞.png')
8  print(list(btm))
9  # [Box(left=1200, top=350, width=50, height=50), Box(left=25, top=594, width=50, height=50)]

```

`pyautogui.center((left, top, width, height))` 返回指定位置的中心点; 这样, 我们就可以再配合鼠标操作点击找到图片的中心;

## 键盘输入

## 键盘输入函数

```
1 # 模拟按键按下;
2 pyautogui.keyDown('shift')
3 # 模拟按键释放;
4 pyautogui.keyUp('shift')
5 # 就是调用keyDown() & keyUp(),模拟一次按键
6 pyautogui.press('4')
7 # 第一参数是输入内容, 第二个参数是每个字符间的间隔时间;
8 pyautogui.typewrite('this',0.5)
9 # typewrite 还可以传入单字母的列表;
10 pyautogui.typewrite(['T','h','i','s'])
```

```
1 # 输出: $
2 pyautogui.keyDown('shift') # 按下shift
3 pyautogui.press('4')      # 按下 4
4 pyautogui.keyUp('shift')  # 释放 shift
```

```
1 # 缓慢的输出$:
2 pyautogui.typewrite('$$$$', 0.5)
```

## 键盘特殊按键

有时我们需要输入一些特殊的按键, 比如向左的箭头, 这些有相对应的键盘字符串表示, 例如:

```
1 # 输出: This
2 pyautogui.typewrite(['T','i','s','left','left','h',])
```

解释: 这里的left就是向左的箭头; 诸如此类的键盘字符串, 还有很多, 参考下表:

键盘字符串	说明
enter(或return 或 \n)	回车
esc	ESC键



shiftright, shiftright	左右SHIFT键
altleft, altright	左右ALT键
ctrlleft, ctrlright	左右CTRL键
tab (\t)	TAB键
backspace, delete	BACKSPACE 、 DELETE键
pageup, pagedown	PAGE UP 和 PAGE DOWN键
home, end	HOME 和 END键
up, down, left,right	箭头键
f1, f2, f3.... f12	F1.....F12键
volumemute, volumedown,volumeup	声音变大变小静音（有些键盘没有）
pause	PAUSE键， 暂停键
capslock	CAPS LOCK 键
numlock	NUM LOCK 键
scrolllock	SCROLLLOCK 键
insert	INSERT键
printscreen	PRINT SCREEN键
winleft, winright	Win键（windows）
command	command键（Mac OS X）
option	option（Mac OS X）

## 快捷键

如果我们需要模拟复制的快捷键 ctrl + c，如果用前面的方法，则代码为：

```
1 pyautogui.keyDown('ctrl')
2 pyautogui.keyDown('c')
3 pyautogui.keyUp('c')
4 pyautogui.keyUp('ctrl')
```

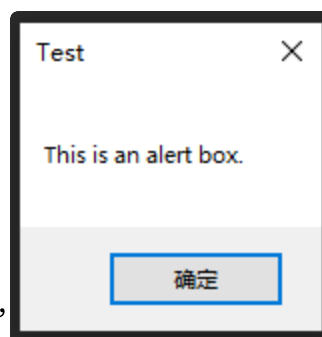
快捷键的按键与释放顺序非常关键，这时我们可以使用 `pyautogui.hotkey()`，这个函数可以接受多个参数，按传入顺序按下，再按照相反顺序释放。上述快捷键 `ctrl + c`，可以将代码变为：

```
1 pyautogui.hotkey('ctrl','c')
```

## 提示信息框

### 提示框/警告框

```
1 import pyautogui
2 a = pyautogui.alert(text='This is an alert box.', title='Test')
3 print(a)
```



输出如下图：点击确定，返回值为‘OK’

### 选择框

```
1 import pyautogui
2 a = pyautogui.confirm('选择一项', buttons=['A', 'B', 'C'])
3 print(a)
```

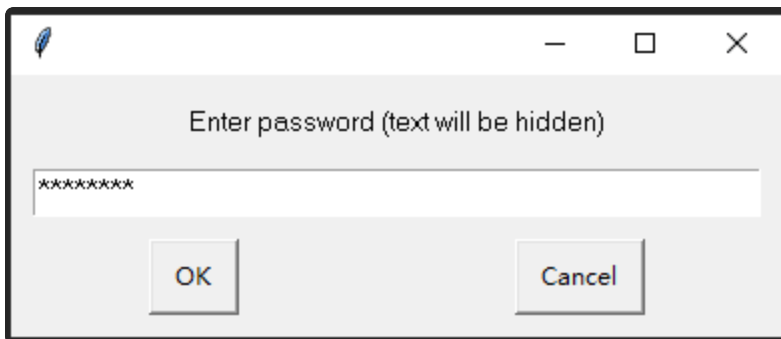


输出如下图：点击B选项，返回值为‘B’

## 密码输入

```
Python |
1  import pyautogui
2
3  a = pyautogui.password('Enter password (text will be hidden)')
4  print(a)
```

输出如下图：输入密码，显示为密文，点击OK，返回值为刚刚输入的值；



## 普通输入

```
Python |
1  import pyautogui
2
3  a = pyautogui.prompt('请输入一个数字：')
4  print(a)
```



输出如下图：显示为明文，点击OK，返回值为刚刚输入的值；

## 实例

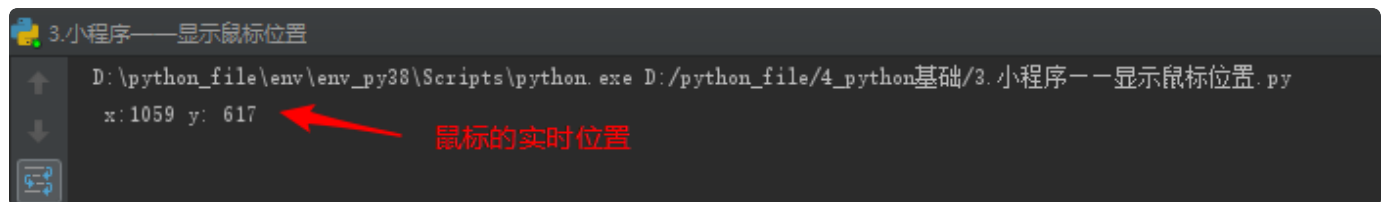
### 鼠标控制鼠标画一个正方形

```
1 for i in range(2): # 画正方形
2     pyautogui.moveTo(200,200,duration=1)
3     pyautogui.moveTo(200,400,duration=1)
4     pyautogui.moveTo(400,400,duration=0.5)
5     pyautogui.moveTo(400,200,duration=2)
```

### 获取鼠标的实时位置

```
1 import pyautogui
2 import time
3
4 try:
5     while True:
6         x,y = pyautogui.position()
7         posi = 'x:' + str(x).rjust(4) + ' y:' + str(y).rjust(4)
8         print('\r',posi,end='')
9         time.sleep(0.5)
10 except KeyboardInterrupt:
11     print('已退出!')
```

显示效果：



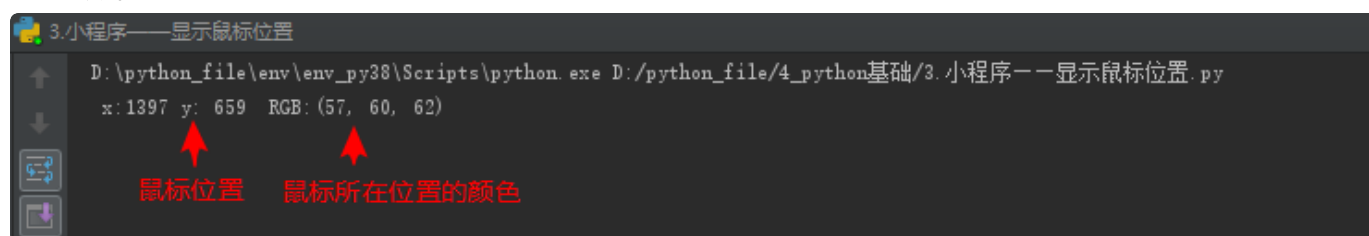
### 获取鼠标位置与所在位置的顏色

```

1  import pyautogui
2  import time
3
4  try:
5      while True:
6          x,y = pyautogui.position()
7          rgb = pyautogui.screenshot().getpixel((x,y))
8          posi = 'x:' + str(x).rjust(4) + ' y:' + str(y).rjust(4) + ' RGB:'
9          + str(rgb)
10         print('\r',posi,end='')
11         time.sleep(0.5)
12 except KeyboardInterrupt:
13     print('已退出! ')

```

显示效果:



## 爆笑，打地鼠的极限，PyAutoGUI的开始~

游戏地址: [http://www.4399.com/flash/178030\\_3.htm](http://www.4399.com/flash/178030_3.htm)

视频教程地址: <https://www.bilibili.com/video/BV1gm4y1x7QW/>

## 介绍

当提到自动化控制鼠标和键盘的Python库时，pyautogui是一个不可忽视的工具。它为用户提供了简单而强大的功能，使得编写自动化脚本变得轻而易举。在本文中，我将向大家介绍如何使用pyautogui编写一个简单而有趣的打地鼠脚本。

首先，让我们来了解一下pyautogui的基本概念。pyautogui是一个跨平台的库，允许我们模拟鼠标和键盘的操作，如移动鼠标、点击和拖动等。它可以用于各种自动化任务，包括测试、GUI交互和游戏脚本

等。


## 思路

我们首先需要安装pyautogui库。可以通过在命令行中运行以下命令来安装：

```
1 pip install pyautogui
```

安装完成后，我们可以开始编写我们的打地鼠脚本。

我们首先导入pyautogui库，并设置一些全局变量。设置FAILSAFE变量为True，使得程序可以根据鼠标位置来中断。设置PAUSE变量为0.05，以便每个自动化操作之间有一个短暂的延迟。这些设置可以帮助我们更好地控制自动化脚本的执行。

接下来，我们使用一个无限循环来不断寻找屏幕上的目标图像。使用pyautogui的 `locateAllOnScreen` 函数，我们可以查找与给定图像（在本例中为）匹配的所有位置。我们还可以指定一个置信度参数，以控制匹配的严格程度。

如果我们找到了目标图像的位置，我们可以遍历所有匹配的位置，并使用 `click` 函数点击每个位置的中心点。这将模拟鼠标点击操作，就像我们真正点击了那个位置一样。

在点击地鼠后，我们打印一条消息来确认我们发现了地鼠。然后，我们将鼠标移动到指定位置（在本例中为(300, 300)），以便进行下一次的寻找。

这就是整个打地鼠脚本的基本步骤。您可以根据自己的需求进行修改和扩展，例如添加计分或计时器等功能来增加游戏的乐趣。

## 具体代码如下：

```
Python |  
  
1  # 导入pyautogui库，用于自动化控制鼠标和键盘  
2  import pyautogui  
3  
4  # 开启安全模式，使根据鼠标位置来中断程序的功能生效  
5  pyautogui.FAILSAFE = True  
6  # 设置每个自动化操作的延迟时间，以降低操作速度  
7  pyautogui.PAUSE = 0.05  
8  
9  # 无限循环  
10 while True:  
11     # 在屏幕上查找所有与给定图像匹配的位置  
12     # 注意：要使用confidence参数需要安装三方库：opencv-python  
13     tars = list(pyautogui.locateAllOnScreen('img.png', confidence=0.8))  
14     # 如果存在匹配的位置  
15     if tars:  
16         # 遍历所有匹配的位置  
17         for i in tars:  
18             # 点击该位置的中心点  
19             pyautogui.click(pyautogui.center(i))  
20             # 打印提示信息  
21             print('发现地鼠了~')  
22             # 将鼠标移动到指定位置  
23             # 为什么要移动到指定的位置？因为防止前后地鼠出现位置一样，这样锤子挡住了地  
鼠，就会识别不到这个地鼠  
24             pyautogui.moveTo(300, 300)
```

## 总结

编写一个简单的打地鼠脚本只是pyautogui的众多可能用途之一。无论是自动化测试、GUI交互还是游戏脚本，pyautogui都可以为您提供简单而强大的工具。因此，如果您还没有尝试过pyautogui，我强烈建议您开始使用它，并发现它在自动化和自动化任务中的巨大潜力。