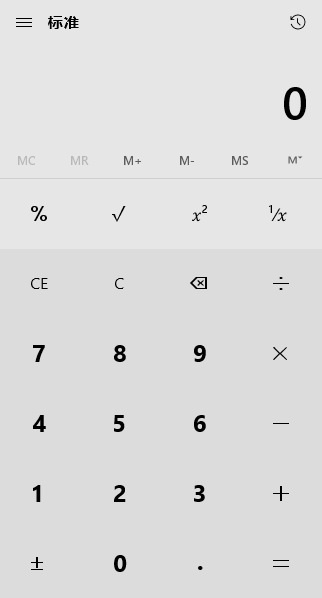
### **8.3 定位函数**

可以定位截图在屏幕上的坐标位置。比如，你需要在计算器里输入：

如果你不知道按钮的位置，就不能用moveTo()定位和click()点击。而且每次计算器的位置可能会变化，这时即使有来坐标也不好用了。但是如果你有要点击按钮的截图，比如数字7：IMG_257

你可以调用pyautogui.locateOnScreen('calc7key.png')函数来获得7的屏幕坐标。返回的是一个元组(top, left, width, height)。这个元组可以用pyautogui.center()函数来获取截图屏幕的中心坐标。如果截图没找到，pyautogui.locateOnScreen()函数返回None：

In [ ]:

**import** **pyautogui**

button7location = pyautogui.locateOnScreen('pyautogui/calc7key.png')

button7location

Out[ ]:

(1226, 546, 29, 28)

In [ ]:

button7x, button7y = pyautogui.center(button7location)

button7x, button7y

Out[ ]:

(1240, 560)

In [ ]:

pyautogui.click(button7x, button7y)

locateCenterOnScreen()等价于上面的前两布操作，直接获得截屏屏幕中心坐标：

In [ ]:

**import** **pyautogui**

x, y = pyautogui.locateCenterOnScreen('pyautogui/calc7key.png')

pyautogui.click(x, y)

在1920×1080的屏幕上，定位函数需要1~2秒时间。对视频游戏（LOL、DOTA）来说就太慢了，但是上班干活还是绰绰有余。

还是几个定位函数。都是从左上角原点开始向右向下搜索截图位置：

* locateOnScreen(image, grayscale=False)：返回找到的第一个截图Image对象在屏幕上的坐标(left, top, width, height)，如果没找到返回None
* locateCenterOnScreen(image, grayscale=False)：返回找到的第一个截图Image对象在屏幕上的中心坐标(x, y)，如果没找到返回None
* locateAllOnScreen(image, grayscale=False)：返回找到的所有相同截图Image对象在屏幕上的坐标(left, top, width, height)的生成器
* locate(needleImage, haystackImage, grayscale=False)：返回找到的第一个截图Image对象在haystackImage里面的坐标(left, top, width, height)，如果没找到返回None
* locateAll(needleImage, haystackImage, grayscale=False)：返回找到的所有相同截图Image对象在haystackImage里面的坐标(left, top, width, height)的生成器

两个locateAll\*函数都可以用for循环和list()输出：

In [ ]:

**for** pos **in** pyautogui.locateAllOnScreen('pyautogui/calc7key.png'):

print(pos)

(1227, 546, 29, 28)

In [ ]:

list(pyautogui.locateAllOnScreen('pyautogui/calc7key.png'))

Out[ ]:

[(1227, 546, 29, 28)]

#### **8.3.1 灰度值匹配**

可以把grayscale参数设置为True来加速定位（大约提升30%），默认为False。这种去色（desaturate）方法可以加速定位，但是也可能导致假阳性（false-positive）匹配：

In [ ]:

**import** **pyautogui**button7location = pyautogui.locateOnScreen('pyautogui/calc7key.png', grayscale=**True**)button7location

Out[ ]:

(1227, 546, 29, 28)

#### **8.3.2 像素匹配**

要获取截屏某个位置的RGB像素值，可以用Image对象的getpixel()方法：

In [ ]:

**import** **pyautogui**im = pyautogui.screenshot()im.getpixel((100, 200))

Out[ ]:

(255, 255, 255)

也可以用PyAutoGUI的pixel()函数，是之前调用的包装：

In [ ]:

pyautogui.pixel(100, 200)

Out[ ]:

(255, 255, 255)

如果你只是要检验一下指定位置的像素值，可以用pixelMatchesColor()函数，把X、Y和RGB元组值穿入即可：

In [ ]:

pyautogui.pixelMatchesColor(100, 200, (255, 255, 255))

Out[ ]:

True

In [ ]:

pyautogui.pixelMatchesColor(100, 200, (255, 255, 245))

Out[ ]:

False

tolerance参数可以指定红、绿、蓝3种颜色误差范围：

In [ ]:

pyautogui.pixelMatchesColor(100, 200, (255, 255, 245), tolerance=10)

Out[ ]:

True

In [ ]:

pyautogui.pixelMatchesColor(100, 200, (248, 250, 245), tolerance=10)

Out[ ]:

True

In [ ]:

pyautogui.pixelMatchesColor(100, 200, (205, 255, 245), tolerance=10)

Out[ ]:

False