Python中多进程中全局变量的问题

我是Python编程的新手（使用v.3.8.8），有一个非常基本的问题，就是在一个使用进程的多线程程序中访问全局变量。例如，我有以下的简单代码。

```
from multiprocessing import Process

global_var = 777

def workerThread():
    global global_var

    my_file = open("out_file",'w')
    print(global_var, file=my_file)
    my_file.close()

    return


if __name__ == '__main__':
    global_var = 999

    if (True):
        # use multiprocessing, but launch only one thread
        procs = Process(target=workerThread, args=())
        procs.start()
        procs.join()
    else:
        # standard function call, so these two paths should be equivalent
        workerThread()
```

我可以用两种代码路径执行它。 "True "在多线程模式下运行（但只有一个线程），"False "直接调用该函数，所以这两种情况基本上是等同的。

因此，我认为两者的行为是一样的，都会输出 "999"，因为在这两种情况下，workerThread()被调用*after*，全局变量被设置为999（我把输出管到 "out_file"，因为多线程时stdout不打印）。

但是由于某些原因，多线程方法输出的是原始值777，而直接调用函数输出的是999。这对我来说没有任何意义。为什么它们会不同呢？ 我怎样才能解决这个问题？

我最初想到在 `if __name__ == '__main__':` 后面加一个 `global global_var` ，以确保它是在设置全局变量（而不是同名的局部变量），但这没有意义，因为 "if" 语句不是一个单独的函数，而是 "main" 函数的一部分。事实上，当我尝试这样做时，它给了我一个语法错误 `"SyntaxError: name 'global_var' is assigned to before global declaration"` 。所以这显然不是正确的答案。

所以我不确定发生了什么，以及如何正确地做这件事。这似乎是一件非常简单的事情，应该很容易做到，但我完全被卡住了，四处寻找却没有找到答案。有什么建议吗?

最后，在最后，我不想使用在这个文件中声明的全局变量，而是在另一个文件中声明的全局变量，比如说 "my_globals.py"，这样我就可以在程序运行时从多个文件访问同一个全局变量。所以我首先尝试在workerThread中以 `my_globals.global_var` 的方式访问它，但也没有成功，这导致我进行了简化，直到我得到这段代码。

而在这种情况下，语句 `global my_globals.global_var` 是无效的（语法错误），所以我不确定如何保证workerThread函数会使用另一个文件中的全局变量。还是因为它是 my_globals 的一个属性而自动被认为是全局变量？试图弄清楚C语言中是否有类似 `extern` 的东西...

提前感谢您对这些新手问题的帮助。我只是对Python不是很在行。

--Miguel

解决方案 | *1*

当操作系统使用 `spawn` 来创建新的进程时，比如 Windows，它会创建一个新的地址空间，并启动一个新的 Python 解释器，从第一行代码开始执行（解释）你的源程序。这意味着全局范围内的任何可执行语句都将被执行。然而，在这个新过程中，变量 `__name__` 将不再是 `__main__` 。这就是为什么你把创建新进程的代码放在以 `if __name__ == '__main__':` 开头的块控制的代码中，因为如果你不这样做，程序将进入一个无限的、递归的循环，试图创建新进程。

让我们再仔细看一下你的代码。你的主进程在创建新进程之前就执行了 `global_var = 999` 。但是正如我刚才所描述的，当新进程开始执行时，*it starts running in a new address space that inherits nothing from the main process and all the global definitions in the source file are*

executed before your function `workerThread` *gets invoked.*的一个全局定义是 `global_var = 777` 。另一个对 `global_var` 的赋值没有被执行，因为 `__name__` 不再是 '__main__' 。这就是为什么你看到了你所看到的结果。

你需要把赋值 `global_var = 999` 移到 `if` 语句的外面。

```
global_var = 999
if __name__ == '__main__':
    ... # etc.
```

但是请记住：你的子进程是在这个全局的一个副本上工作的。如果 `workerThread` 修改了作为子进程运行的全局，这个变化将不会反映在主进程的副本中。

参考: How to terminate a thread from outside or use input in multiprocessing in python