**threading多线程**

threading用于提供线程相关的操作，线程是应用程序中工作的最小单元。

threading模块提供的类：

Thread, Lock, Rlock, Condition, Semaphore, Event, Timer, local。

# constant

threading.TIMEOUT\_MAX

The maximum value allowed for the timeout parameter of blocking functions (Lock.acquire(), RLock.acquire(), Condition.wait(), etc.). Specifying a timeout greater than this value will raise an OverflowError.

# functions

## threading.active\_count()

Return the number of Thread objects currently alive. The returned count is equal to the length of the list returned by enumerate().

## threading.current\_thread()

Return the current Thread object, corresponding to the caller’s thread of control. If the caller’s thread of control was not created through the threading module, a dummy thread object with limited functionality is returned.

## threading.get\_ident()

Return the ‘thread identifier’ of the current thread. This is a nonzero integer. Its value has no direct meaning; it is intended as a magic cookie to be used e.g. to index a dictionary of thread-specific data. Thread identifiers may be recycled when a thread exits and another thread is created.

## threading.enumerate()

Return a list of all Thread objects currently alive. The list includes daemonic threads, dummy thread objects created by current\_thread(), and the main thread. It excludes terminated threads and threads that have not yet been started.

## threading.main\_thread()

Return the main Thread object. In normal conditions, the main thread is the thread from which the Python interpreter was started.

## threading.settrace(func)

Set a trace function for all threads started from the threading module. The func will be passed to sys.settrace() for each thread, before its run() method is called.

## threading.setprofile(func)

Set a profile function for all threads started from the threading module. The func will be passed to sys.setprofile() for each thread, before its run() method is called.

threading.stack\_size([size])

Return the thread stack size used when creating new threads. The optional size argument specifies the stack size to be used for subsequently created threads, and must be 0 (use platform or configured default) or a positive integer value of at least 32,768 (32 KiB). If size is not specified, 0 is used. If changing the thread stack size is unsupported, a RuntimeError is raised. If the specified stack size is invalid, a ValueError is raised and the stack size is unmodified. 32 KiB is currently the minimum supported stack size value to guarantee sufficient stack space for the interpreter itself. Note that some platforms may have particular restrictions on values for the stack size, such as requiring a minimum stack size > 32 KiB or requiring allocation in multiples of the system memory page size - platform documentation should be referred to for more information (4 KiB pages are common; using multiples of 4096 for the stack size is the suggested approach in the absence of more specific information).

# classes

## Thread

threading.Thread(group=None, target=None, name=None, args=(), kwargs={}, \*, daemon=None)¶

如果是后台线程，主线程执行过程中，后台线程也在进行，主线程执行完毕后，后台线程不论成功与否，主线程和后台线程均停止

如果是前台线程，主线程执行过程中，前台线程也在进行，主线程执行完毕后，等待前台线程也执行完成后，程序停止

设置join之后，主线程等待子线程全部执行完成后或者子线程超时后，主线程才结束

join()阻塞当前上下文环境的线程，直到调用此方法的线程终止或到达指定的timeout，即使设置了setDeamon（True）主线程依然要等待子线程结束