多进程：

例：程序process\_thread\_test/process\_thread\_test1.py

import os  
from multiprocessing import Process  
  
  
def run\_process(name):  
 print('Run child process {0} ({1})'.format(name, os.getpid()))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print('Parent process {0}.'.format(os.getpid()))  
 child = Process(target=run\_process, args=('test',))  
 print('Child Process will start.')  
 child.start() # 启动进程  
 child.join() # 等待子进程执行完毕在  
 print('Child process end.')

进程间通信

例：程序process\_thread\_test/process\_thread\_test2.py

from multiprocessing import Process, Queue  
import os, time, random  
  
  
def write(queue):  
 print('Process to write: {0}'.format(os.getpid()))  
 for value in ['A', 'B', 'C']:  
 print('Put {0} to queue...'.format(value))  
 queue.put(value)  
 time.sleep(random.random())  
  
  
def read(queue):  
 print('Process to read: {0}'.format(os.getpid()))  
 while True:  
 value = queue.get(True)  
 print('Get {0} from queue.'.format(value))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 q = Queue()  
 process\_write = Process(target=write, args=(q,))  
 process\_read = Process(target=read, args=(q,))  
 process\_write.start()  
 process\_read.start()  
 process\_write.join()  
 process\_read.terminate()

多线程：

例：程序process\_thread\_test/process\_thread\_test3.py

import threading  
  
balance = 0  
lock = threading.Lock()  
  
  
def change(n):  
 global balance  
 balance += n  
 balance -= n  
  
  
def run\_thread(n):  
 for i in range(1000):  
 lock.acquire() # 加锁  
 try:  
 change(i)  
 finally:  
 lock.release() # 释放锁  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 t1 = threading.Thread(target=run\_thread, args=(5,))  
 t2 = threading.Thread(target=run\_thread, args=(8,))  
 t1.start()  
 t2.start()  
 t1.join()  
 t2.join()  
 print(balance) # 0

ThreadLocal对象

例：程序process\_thread\_test/process\_thread\_test4.py

import threading  
  
# 全局ThreadLocal对象  
# local\_school为全局对象  
# 但其属性为线程的局部对象  
# 可以在多线程中任意读写而互不干扰  
local\_school = threading.local()  
  
  
def get\_student():  
 # 获取当前线程关联的student  
 student = local\_school.student  
 print('Hello, {0} (in {1})'.format(student, threading.current\_thread().name))  
  
  
def process\_thread(name):  
 local\_school.student = name  
 get\_student()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 t1 = threading.Thread(target=process\_thread, args=('Alice',), name='Thread A')  
 t2 = threading.Thread(target=process\_thread, args=('Bob',), name='Thread B')  
 t1.start()  
 t2.start()  
 t1.join()  
 t2.join()

Python中由于全局解释器锁GIL的存在，多线程只能交替执行，即使有多个CPU，也只会用到一个CPU。

多线程不使用函数，而使用类对象来执行线程处理。

例：程序process\_thread\_test/process\_thread\_test5.py

import threading  
from time import sleep, ctime  
  
loops = [4, 2]  
  
  
class ThreadFunc:  
 def \_\_init\_\_(self, func, args, name=''):  
 self.name = name  
 self.func = func  
 self.args = args  
  
 def \_\_call\_\_(self, \*args, \*\*kwargs):  
 # 运算符重载  
 self.func(\*self.args)  
  
  
def loop(nloop, nsec):  
 print('start loop', nloop, 'at:', ctime())  
 sleep(nsec)  
 print('loop', nloop, 'done at:', ctime())  
  
  
def test():  
 print('starting at:', ctime())  
 threads = []  
 nloops = range(len(loops))  
  
 for i in nloops:  
 thread = threading.Thread(target=ThreadFunc(loop, (i, loops[i]), loop.\_\_name\_\_))  
 threads.append(thread)  
  
 for i in nloops:  
 threads[i].start()  
  
 for i in nloops:  
 threads[i].join()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 test()

输出为：

starting at: Sun May 20 16:50:15 2018

start loop 0 at: Sun May 20 16:50:15 2018

start loop 1 at: Sun May 20 16:50:15 2018

loop 1 done at: Sun May 20 16:50:17 2018

loop 0 done at: Sun May 20 16:50:19 2018

all DONE at: Sun May 20 16:50:19 2018

继承Thread，自定义多线程：

例：程序process\_thread\_test/process\_thread\_test6.py

import threading  
from time import sleep, ctime  
  
loops = [4, 2]  
  
  
class MyThread(threading.Thread):  
 def \_\_init\_\_(self, func, args, name=''):  
 super().\_\_init\_\_()  
 self.name = name  
 self.func = func  
 self.args = args  
  
 def run(self):  
 self.func(\*self.args)  
  
  
def loop(nloop, nsec):  
 print('start loop', nloop, 'at:', ctime())  
 sleep(nsec)  
 print('loop', nloop, 'done at:', ctime())  
  
  
def test():  
 print('starting at:', ctime())  
 threads = []  
 nloops = range(len(loops))  
  
 for i in nloops:  
 thread = MyThread(loop, (i, loops[i]), loop.\_\_name\_\_)  
 threads.append(thread)  
  
 for i in nloops:  
 threads[i].start()  
  
 for i in nloops:  
 threads[i].join()  
  
 print('all DONE at:', ctime())  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 test()

输出为：

starting at: Sun May 20 16:50:53 2018

start loop 0 at: Sun May 20 16:50:53 2018

start loop 1 at: Sun May 20 16:50:53 2018

loop 1 done at: Sun May 20 16:50:55 2018

loop 0 done at: Sun May 20 16:50:57 2018

all DONE at: Sun May 20 16:50:57 2018