Day03多任务

- 目的
  - 为了提高程序的执行效率
- 概念
  - 并发
    - 同时运行多个任务,但同一刻只有一个任务在执行,多个任务间交替轮换执行
  - 并行
    - 真正的同时执行,一般体现在多核CPU执行多任务时, 多个任务在同一时刻同时运行
  - 同步
    - 当一个任务执行结束后再执行另外一个任务,任务执行顺序 是线性的,称为同步
  - 异步
    - 多个任务同时执行但互不干扰.称为异步
- 步骤
  - 1. 导入相应包或模块
  - 2. 实现任务函数
  - 3.创建多任务对象
  - 4. 启动多任务对象
- 进程
  - 进程是一个程序在运行时的表现形式
  - 一个程序至少对应一个进程
  - 进程是资源分配最小单位
  - 有效的利用多核 CPU,但是资源开销比较大
  - 进程间不共享全局变量
  - 主进程会等待子进程结束之后再结束
  - 进程对象 = mulitprocessing.Process(target=任务函数名, args=(任务函数需要参数))
  - 进程对象.start()
  - 进程对象.join()
    - 将指定进程插入到当前进程之间执行
  - 进程对象.terminate()
    - 手动结束子进程
  - 进程对象.daemon
    - 设置守护进程
      - 必须在开启进程之前设置
  - mulitprocessing.current_proces()
    - 获取当前进程对象
    - .name属性
      - 获取进程名
  - os
    - getpid()
      - 获取当前进程编号
    - getppid()
      - 获取当前进程的父进程ID
- 线程
  - 最小的CPU调度执行单位
  - 线程依附在进程中的,一个程序在运行时,默认会有一个主进程,进程中默认会有一个主线程
  - 线程比较轻量 ,因为多个线程都是共享进程资源,所以资源开销比较小
  - 但是由于GIL存在,导致同一时刻,只能有一个线程执行,无法利用多核 ,程序执行效率会降低
  - 线程间间共享全局变量
  - 多线程在共享全局变量时,会产生资源竞争问题
    - 解决办法
      - 同步
      - 互斥锁
        - lock = Thread.Lock()
          - 加锁
            - lock.acquire()
          - 解锁
            - lock.release()
        - Lock 是一个函数,返回了一个锁类型的对象
      - 死锁
        - 死锁是一种在使用锁时不当产生逻辑错误
        - 导致了锁没有被正常释放,让其它线程在抢占锁时,一直处于等待状态.导致程序无法向下执行
  - 父线程会等待子线程结束后再结束
    - 设置守护线程
      - 线程对象 = threading.Thread(target=任务函数名,args=(任务函数参数), daemon= True)
      - 线程对象.daemon = True
      - 线程对象.setDaemon(True)
  - 线程对象 = threading.Thread(target=任务函数名,args=(任务函数参数))
  - 线程对象.start()
  - 线程对象.join()
    - 将指定的线程加入到当前线程之前执行
- 协程
  - 微线程
    - 是在同一个线程内,不断的去切换多个任务函数来实现的多任务方式
  - 安装第三模块
    - pip3 install gevent
  - 导包
    - import gevent
  - 创建协程对象
    - g = gevent.spawn(任务函数名, 任务函数参数....)
  - 启动协程对象
    - g.join()
    - gevent.joinall(列表)
      - 列表中的对象是协程对象
  - 转换规则
    - 当协程遇到耗时操作时,会自动进行任务的转换切换
    - 耗时操作
      - 大量运算
      - IO读写
      - time.sleep()
      - gevent.sleep()
    - 为了兼容所有的耗时操作.需要给协程打补丁
      - 猴子补丁
        - from gevent import money
        - money.patch_all()

XMind:ZEN | 试用模式