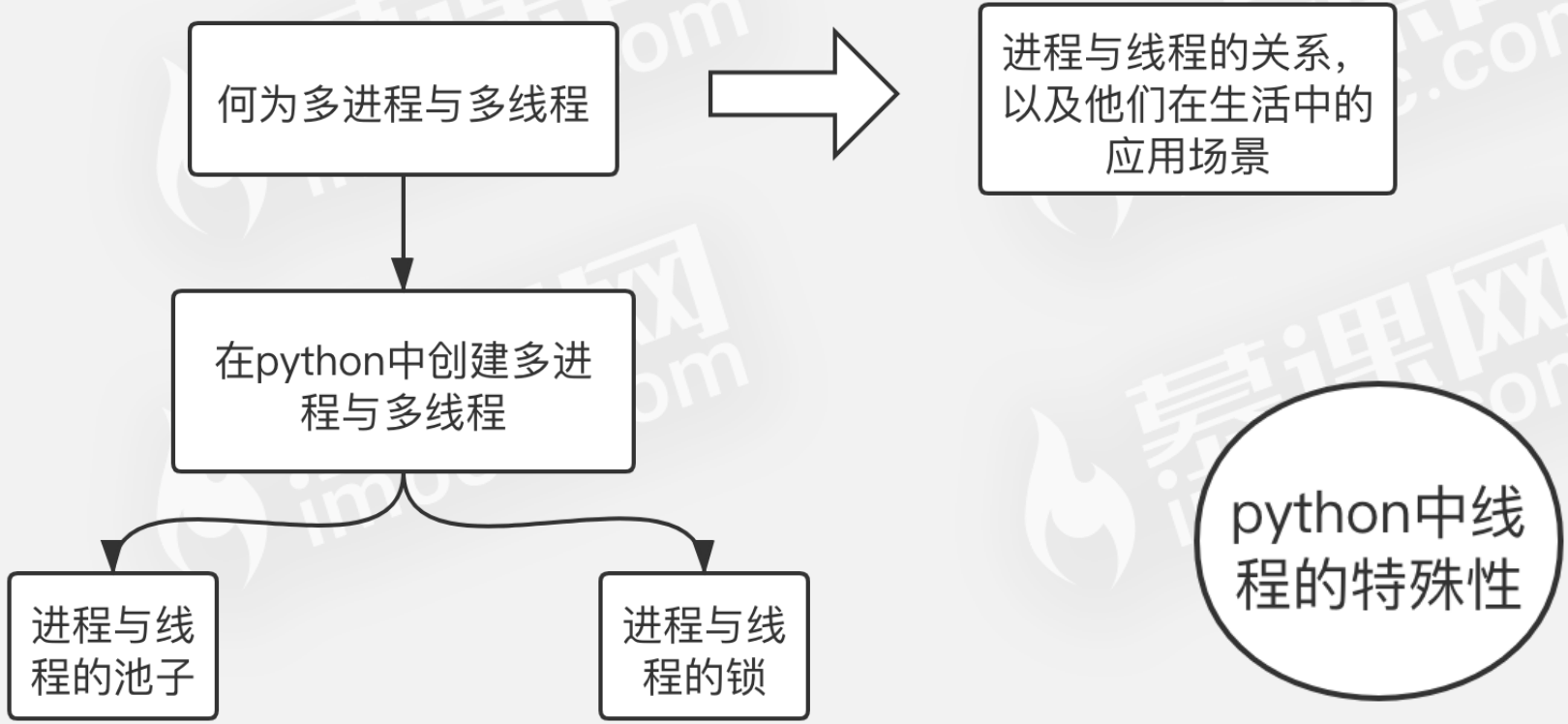
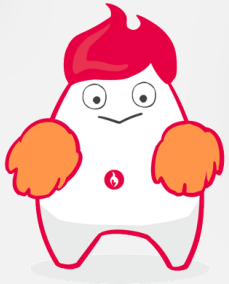
Python多线程编程

多进程与多线程


何为多进程与多线程
进程与线程的关系，以及他们在生活中的应用场景
在python中创建多进程与多线程
进程与线程的池子
进程与线程的锁
python中线程的特殊性

进程



本节课内容

- ◆什么是进程
- ◆进程在生活中的应用
- ◆进程的口粮
- ◆多进程
- ◆多进程的执行方式

什么是进程

- 进程就是程序执行的载体

进程在生活中的应用

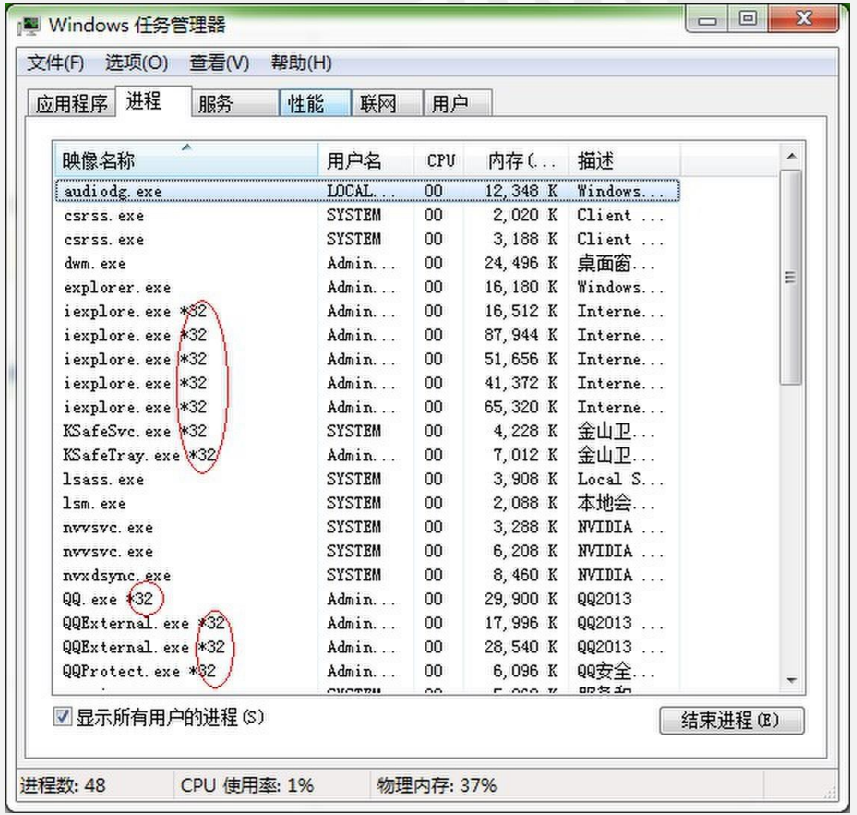

```
if __name__ == '__main__':
    asyncio.run(main())
```

进程在生活中的应用

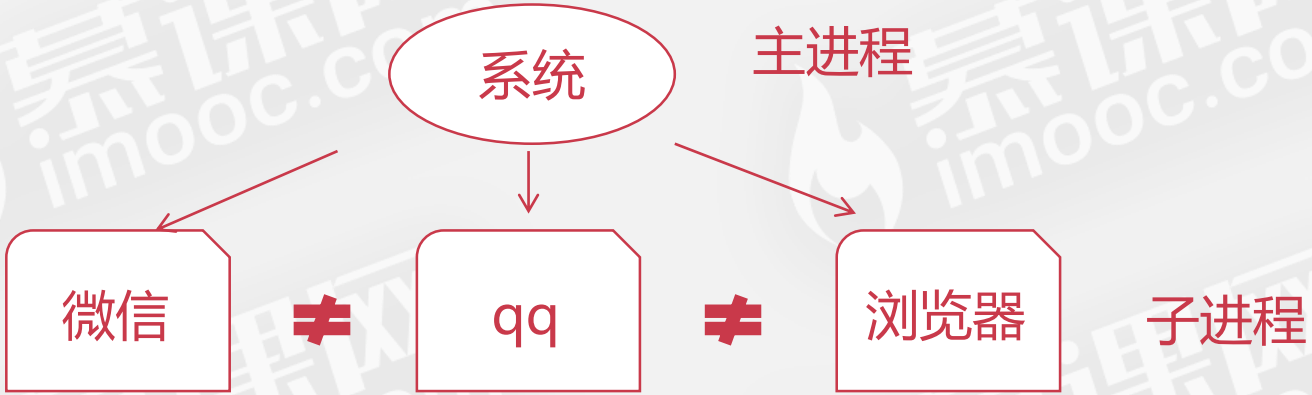
- 我们打开的每个软件 游戏，执行的每一个python脚本都是启动一个进程
- 软件（游戏，脚本） == 进程

进程的口粮

每一个进程像人一样需要吃饭，他的粮食就是：cpu和内存

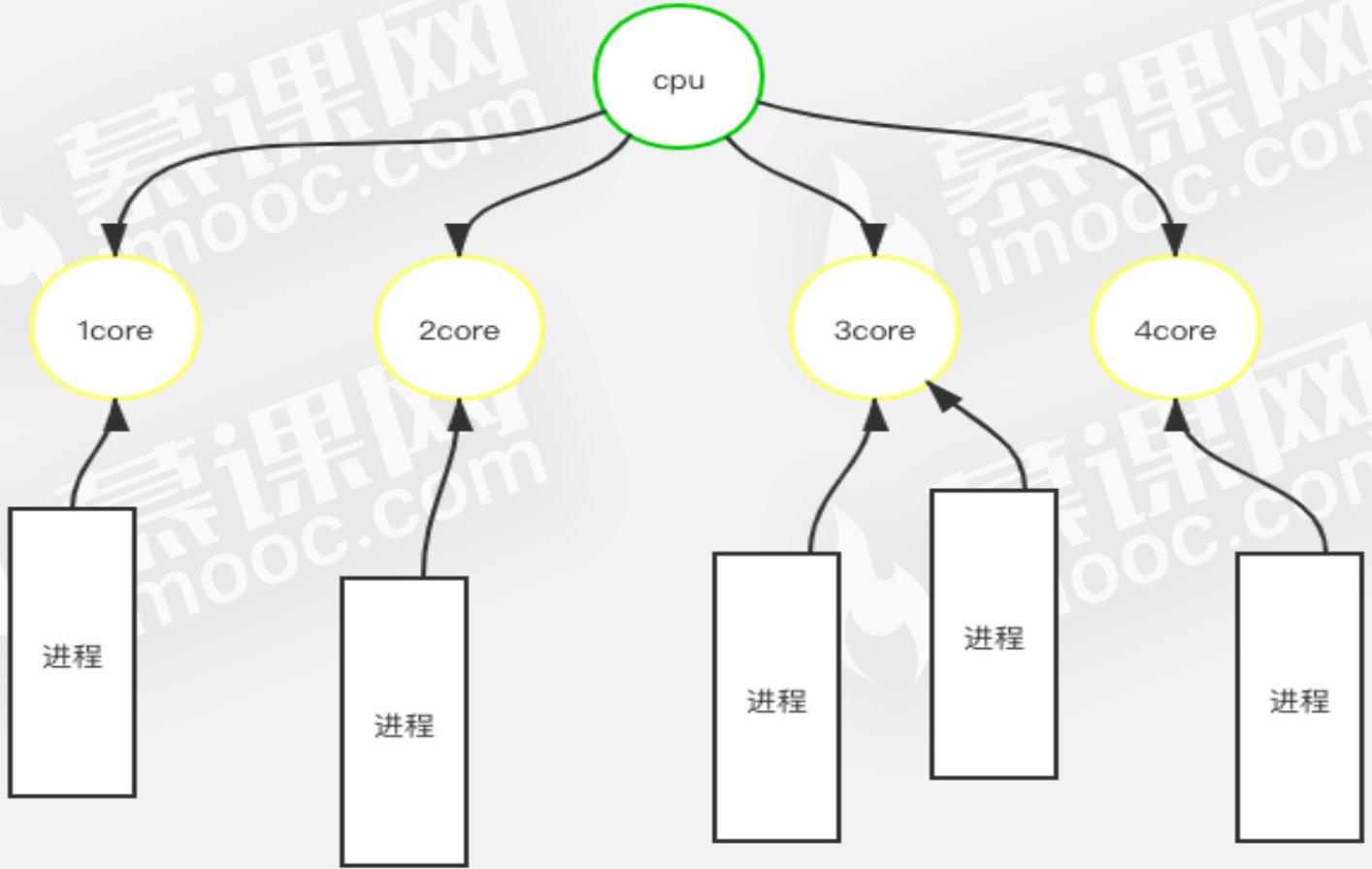


多进程

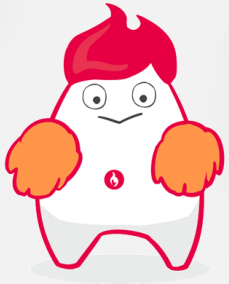


- 同时启动多个软件（进程），多个进程同时在执行程序，他们之间互不干扰，各自执行自己的业务逻辑

多进程与并行概念图



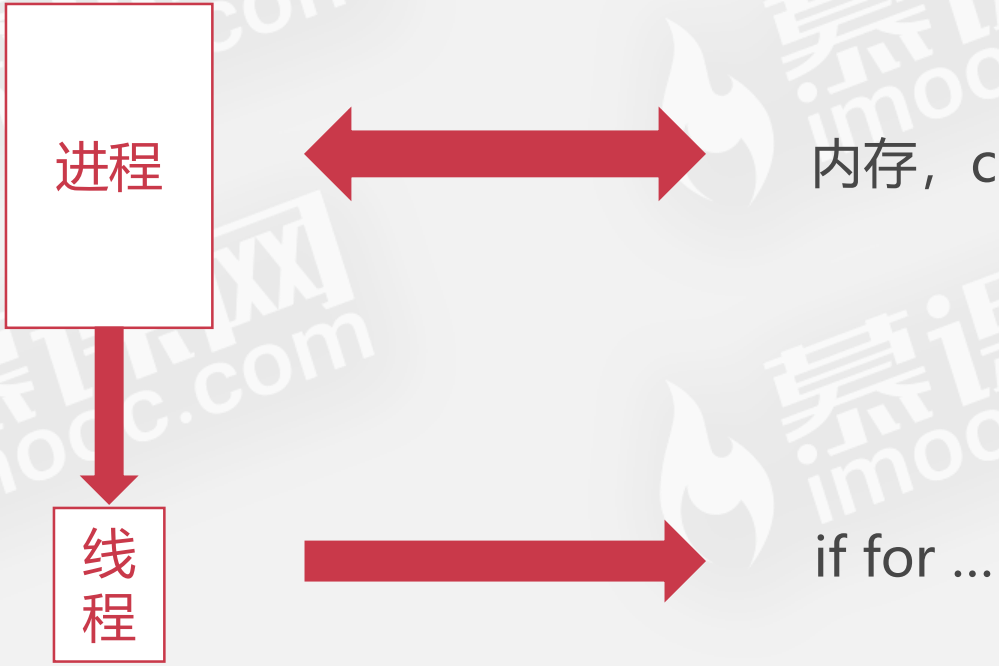
线程



本节课内容

- 什么是线程
- 线程与进程的关系
- 多线程
- 多线程的执行方式

什么是线程



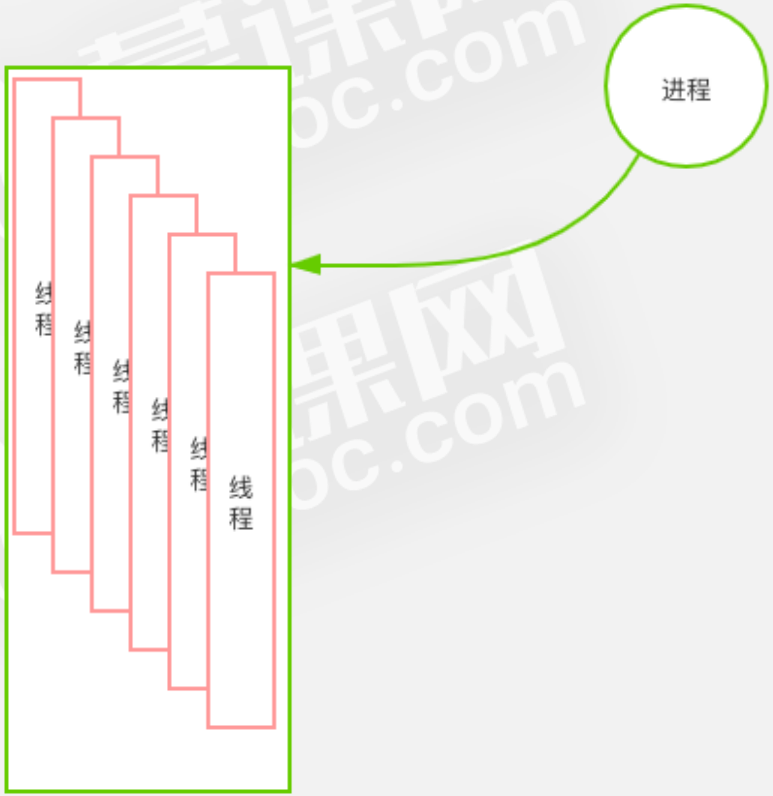
线程与进程的关系

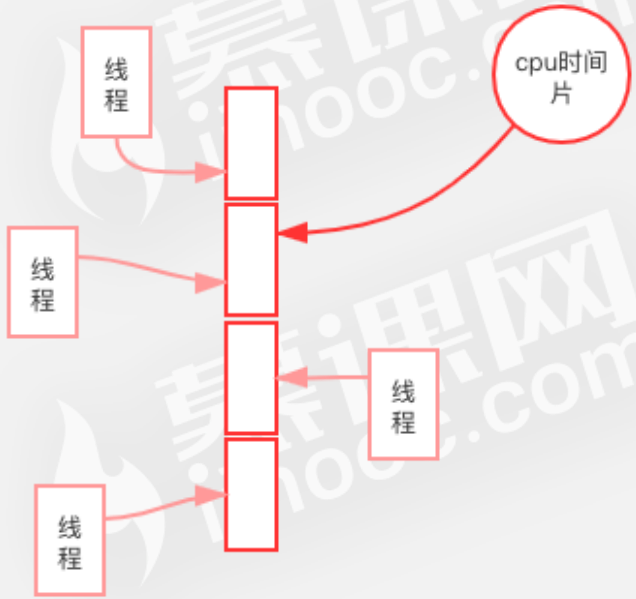
- 进程提供线程执行程序的前置要求，线程在重组的资源配备下，去执行程序

多线程

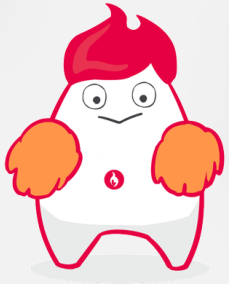

Kube
Kube
Sear
kube
clien
sf (18)
v1 - (
v1 - (
v1 - (
clien
clien
how
Sear
core
aint-and-toleration.html
开起一个浏览器进程后,从浏览器(主线程)中创建出多个线程来开启多个页面

多线程的执行方式





进程的创建



本节课内容

- 进程的创建模块—multiprocessing
- 进程的问题

进程的创建模块--multiprocessing

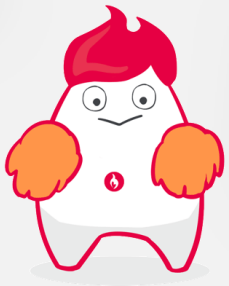
函数名	介绍	参数	返回值
Process	创建一个进程	target, args	进程对象

函数名	介绍	参数	返回值
start	执行进程	无	无
join	阻塞程序	无	无
kill	杀死进程	无	无
is_alive	进程是否存活	无	bool

多进程的问题

- 通过进程模块执行的函数无法获取返回值
- 多个进程同时修改文件可能会出现错误
- 进程数量太多可能会造成资源不足，甚至死机等情况

进程池与进程锁

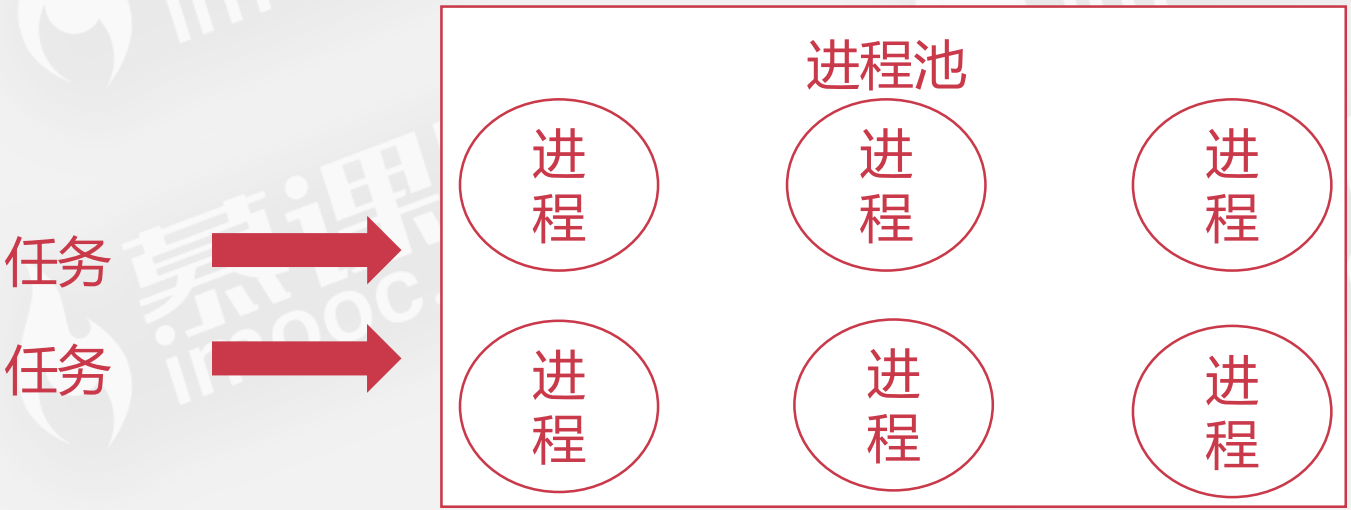


本节课内容

- 什么是进程池
- 进程池的创建
- 进程锁
- 进程锁的用法

什么是进程池

避免了创建与关闭的消耗

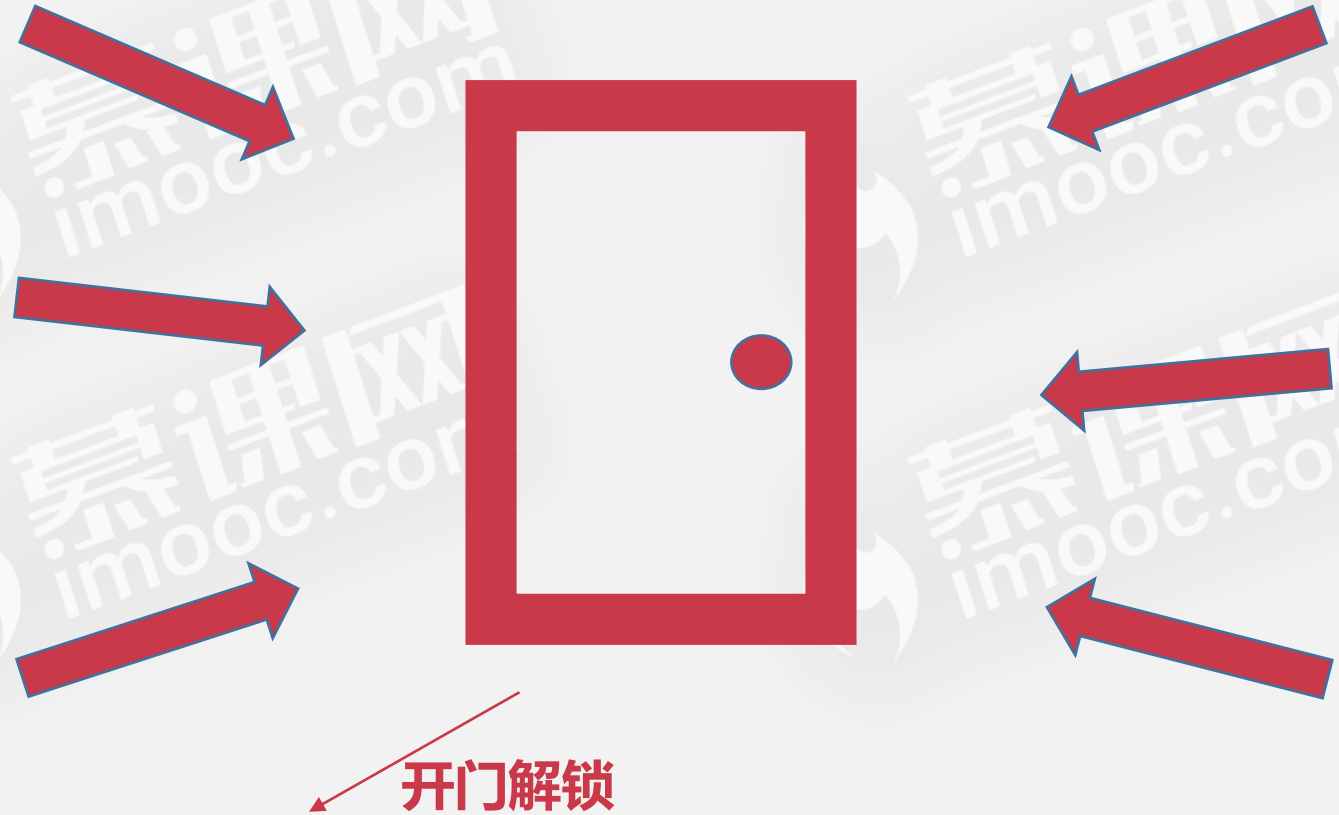


进程池的创建-multiprocessing

函数名	介绍	参数	返回值
Pool	进程池创建	Processcount	进程池对象

函数名	介绍	参数	返回值
apply_async	任务加入进程池(异步)	func，args	无
close	关闭进程池	无	无
join	等待进程池任务结束	无	无

进程锁

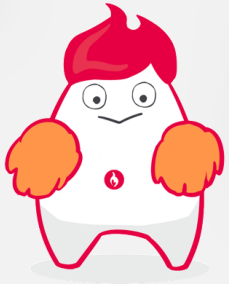


进程锁的加锁与解锁

- from multiprocessing import Manager
- manage = Manager()
- lock = manage.Lock()

函数名	介绍	参数	返回值
acquire	上锁	无	无
release	开锁（解锁）	无	无

进程的通信



本节课内容

- 什么是进程的通信
- 进程通信的方法

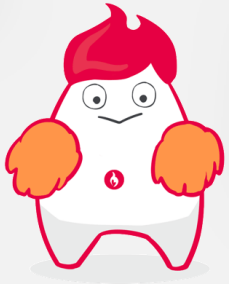
什么是进程的通信

队列的创建-multiprocessing

函数名	介绍	参数	返回值
Queue	队列的创建	mac_cout	队列对象

函数名	介绍	参数	返回值
put	信息放入队列	message	无
get	获取队列信息	无	str

线程的创建



本节课内容

- 线程的创建
- 线程的问题

线程的创建---threading

方法名	说明	举例
Thread	创建线程	Thread(target, args)

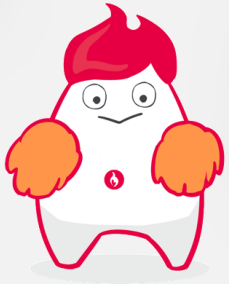
线程对象的方法

方法名	说明	用法
start	启动线程	start()
join	阻塞直到线程执行结束	join(timout=None)
getName	获取线程的名字	getName()
setName	设置线程的名字	setName(name)
is_alive	判断线程是否存活	is_alive()
setDaemon	守护线程	setDaemon(True)

线程的问题

- 通过线程执行的函数无法获取返回值
- 多个线程同时修改文件可能造成数据错错乱

线程池的创建



本节课内容

◆线程池的创建

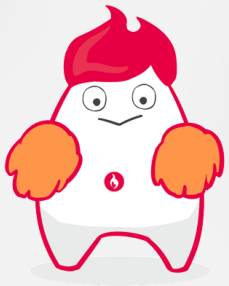
线程池---concurrent

方法名	说明	举例
futures.ThreadPoolExecutor	创建线程池	tpool = ThreadPoolExecutor(max_workers)

线程池---concurrent

方法名	说明	用法
submit	往线程池中加入任务	submit(target, args)
done	线程池中的某个线程是否完成了任务	done()
result	获取当前线程执行任务的结果	result()

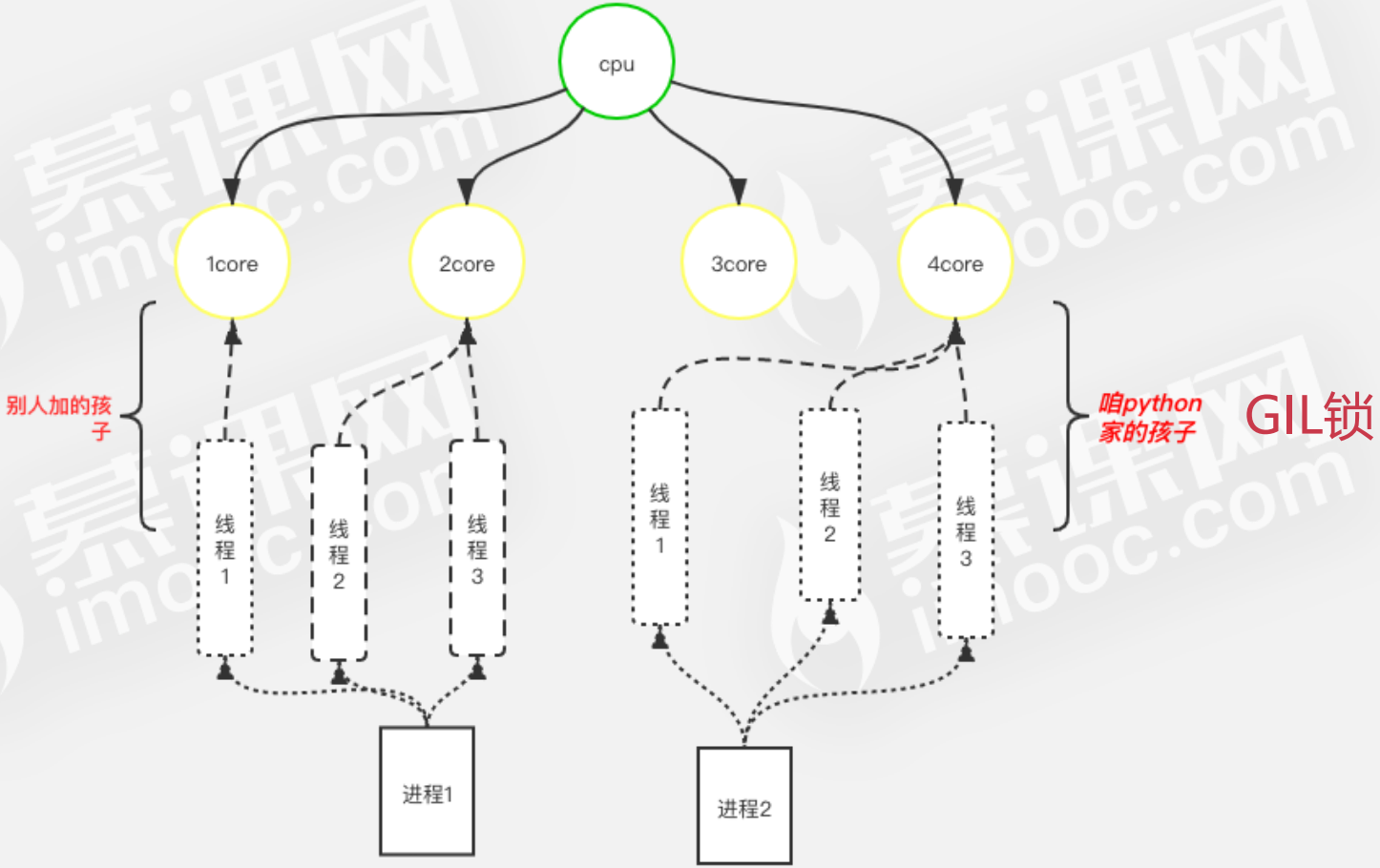
认识GIL全局锁



本节课内容

- 别家的线程与我家的线程
- Gil的作用

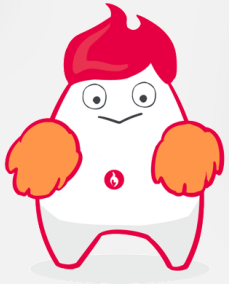
别家的线程与我家的线程



GIL的作用

- 单一cpu工作
- 线程安全
- pypy
- 多进程+多线程

初探异步



本节课内容

- ◆什么是异步与异步的好处
- ◆异步与多线程多进程
- ◆async，await 与 asyncio模块的
- ◆gevent 异步模块的使用

什么是异步与异步的好处

多进程 多线程?

异步，不阻塞

异步与多线程多进程

- 轻量级的线程　　协程
- 可以获取异步函数的返回值
- 主进程需要异步才行
- 更适合文件读写使用

async与await关键字

◆async 定义异步

```
async def test():
    return 'a'
```

◆await 执行异步

```
async def handle():
    result = await test()
```

主程序如何执行异步函数呢?

```
if __name__ == '__main__':
    ???
```

asyncio调用async函数

函数名	介绍	参数	返回值
gather	将异步函数批量执行	asyncfunc...	List 函数的返回结果
run	执行主异步函数	[task]	执行函数的返回结果

asyncio调用async函数

```
async def main():
    result = await asyncio.gather(
        a(),
        b()
    )
    print(result)
if __name__ == '__main__':
    asyncio.run(main())
```

gevent

- pip install gevent
- Microsoft Visual C++
- pip install wheel

gevent模块常用方法

函数名	介绍	参数	返回值
spawn	创建协程对象	Func，args	协程对象
joinall	批量处理协程对象	[spawnobj]	[spawnobj]

Gevent协程对象的方法

函数名	介绍	参数	返回值
get(value)	获取异步程序结果	无	函数的返回值
join	阻塞等待异步程序结束	无	无
kill	杀掉当前协程	无	无
dead	判断协程是否消亡	无	bool