多线程

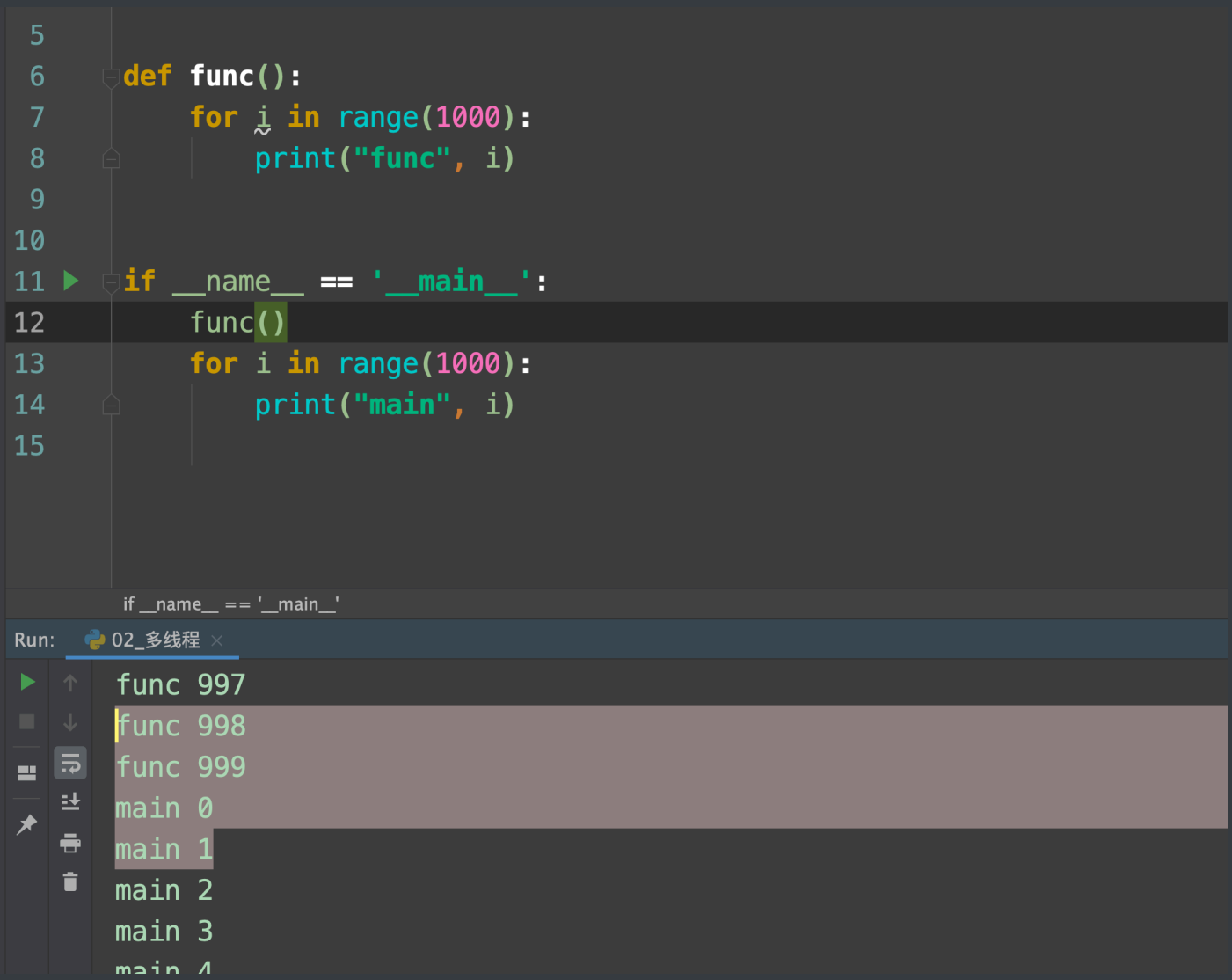
python中实现多线程非常简单. 我们要借助Thread类来完成.

先看单线程效果~

```
def func():
    for i in range(1000):
        print("func", i)


if __name__ == '__main__':
    func()
    for i in range(1000):
        print("main", i)

```



执行过程: 程序启动 --> 加载func() --> 执行main --> 调用func() --> func执行完毕, 继续执行main中的内容

整个过程是一条线跑下来的, 这就是单线程.

多线程:

```python
from threading import Thread


def func():
    for i in range(1000):
        print("func", i)


if __name__ == '__main__':
    t = Thread(target=func)
    t.start()
    for i in range(1000):
        print("main", i)
```

```python

from threading import Thread


def func():
    for i in range(1000):
        print("func", i)

if __name__ == '__main__':
    t = Thread(target=func)
    t.start()
    for i in range(1000):
        print("main", i)
```

```
Run: 02_多线程
main28
 12func 29
func 30
func
main 13
main 14
main 15 31

funcmain 32
func 33 16
main 17
```

程序效果: main和func交替执行(如果速度够快, 给我们的感觉就是一起执行)

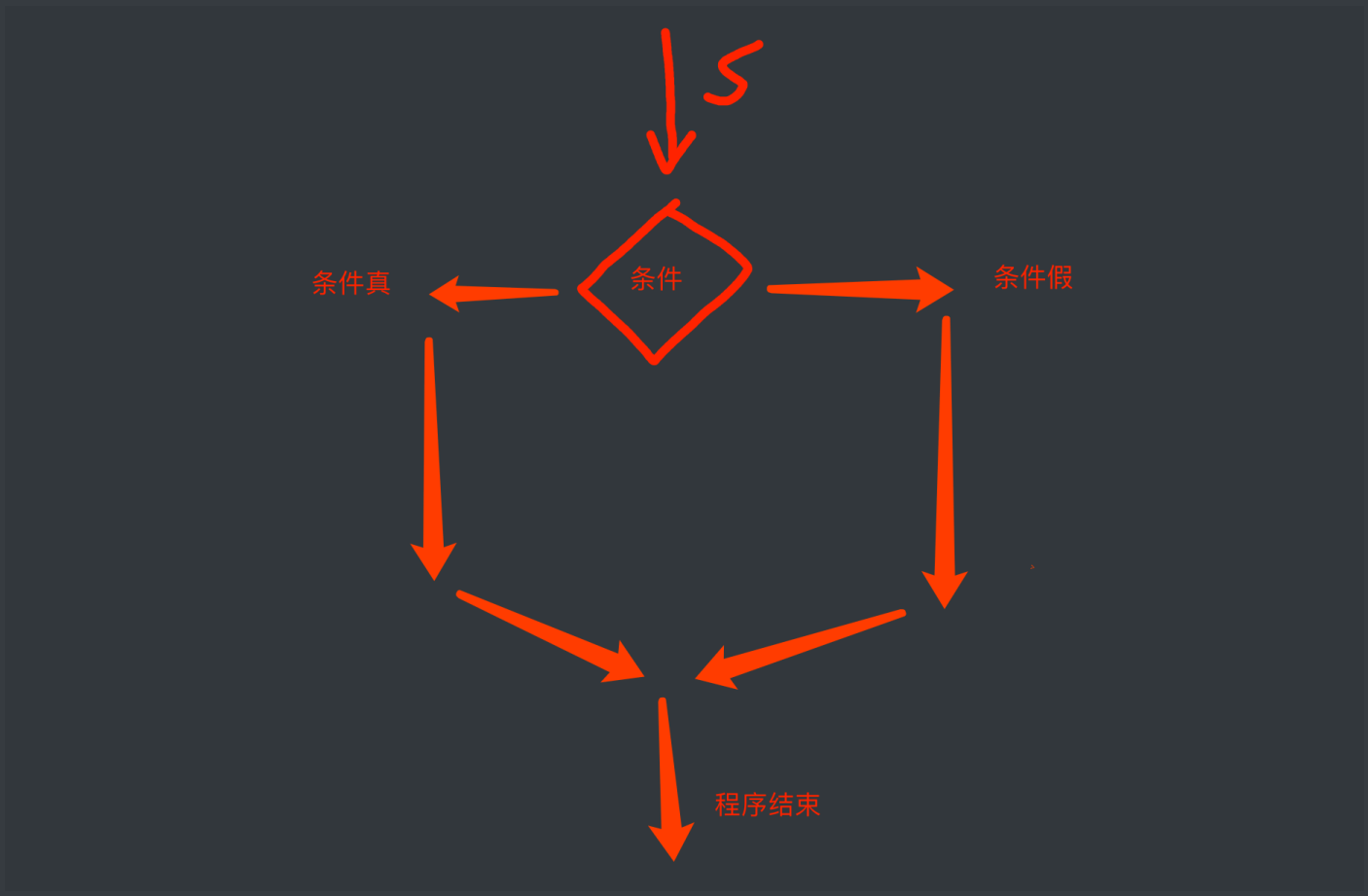
执行过程: 加载func() -> 执行main -> 创建子线程t -> 子线程t启动 -> 执行func中的内容 |-> 继续执行main

我们成功的让两件事同时发生了. 那么想一下, 如果我有1000个url准备去下载. 那么交给每个func单独去执行就好了啊. 主函数该干嘛还干嘛~

误区:

你说了.单线程是一条线跑下来的, 那我如果写个if是不是就是两条线了?

非也~, 我们先看图.



我们要注意一个细节. 不论程序真还是假. 它只能选择一条路走. 所以还是单线程. 并没有异步的效果

多线程的另一种写法

```
1 from threading import Thread
```

```
 2
 3
 4 class MyThread(Thread):
 5     def run(self):
 6         for i in range(1000):
 7             print("func", i)
 8
 9
10 if __name__ == '__main__':
11     t = MyThread()
12     t.start()
13     for i in range(1000):
14         print("main", i)
```

执行效果是一样的. 这里就不放图了.