实例 1：生产者与消费者模式

生产者与消费者模式是多线程同步应用的经典案例，它通过一个固定大小的缓冲区解决了代表“生产者”和代表“消费者”的两个线程在实际运行时发生的强耦合问题——由于生产者的生产能力与消费者的消费能力互不匹配，导致双方必须互相阻塞等待处理。在生产者与消费者模式中，生产者与消费者彼此之间通过缓冲区进行通讯，示意过程如图 1 所示。

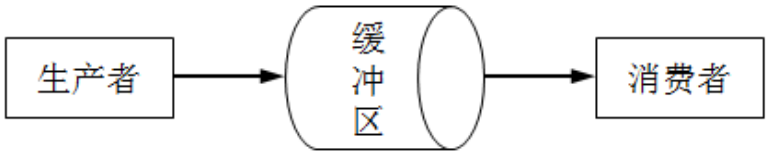

图1 线程同步示例

由图 1 可知，生产者在生产完数据之后直接将数据存储到队列中，无需等待消费者处理；消费者直接从队列中取出数据，无需再等待生产者生产，平衡了生产者与消费者的能力。

假设现在有一群生产者（Producer）和一群消费者（Consumer）通过一个市场来交互产品。生产者的“策略”是若市场上剩余的产品少于 1000 个则生产 100 个产品放到市场上；而消费者的“策略”是若市场上剩余产品的数量多于 100 个则消费 3 个产品。

现在请使用 Queue 类编写一个案例，模拟以上描述的生产者与消费者模式的场景。

实例目标

- 熟练地自定义线程
- 掌握 Queue 类的使用，会使用队列实现线程同步

实例分析

按照实例描述，我们可将生产者与消费者视为两个线程，它们的要求如下：

(1) 生产者线程：若队列数据长度小于 1000，则调用 put()方法向队列中写入数据。

(2) 消费者线程：若队列中数据的长度大于 100，则调用 get()方法从队列取出数据。

按照如上要求需要自定义两个线程类 Producer 和 Consumer。为了更真实地模拟生产者与消费者的场景，这里分别创建两个生产者和 5 个消费者。

代码实现

（1）导入 Queue 类，创建一个队列对象，具体代码如下。

```
from  queue import Queue
```

```
queue = Queue()                                    # 创建队列对象
```

（2）定义一个代表生产者的线程类 Producer。在 Producer 类中重写父类的 run()方法，该方法中首先设置初始的产品数量为 0，然后判断队列大小是否比 1000 大，若大于 1000 则不做任何操作，否则就开始向队列中添加数据，最后让线程休眠一秒钟降低其执行效率，具体代码如下。

```
import threading
import time
class Producer(threading.Thread):      # 代表生产者的线程
    def run(self):
        global queue
        count = 0                              # 初始的产品数量
        while True:
            for i in range(100):
                if queue.qsize() > 1000: # 若队列的大小比 1000 大，不做任何操作
                    pass
                else:
                    count += 1
                    message = '生成产品' + str(count)
                    queue.put(message)     # 把新生产的产品放到队列
                    print(message)
            time.sleep(1)
```

（3）定义一个代表消费者的线程类 Consumer。在 Consumer 类中重写父类的 run()方法，该方法中首先判断队列中数据的大小是否小于 100，若小于 100 则不做任何操作，否则就开始从队列中取出数据，然后让线程休眠一秒钟降低其执行效率，具体代码如下。

```
class Consumer(threading.Thread):      # 代表消费者的线程
    def run(self):
        global queue
        while True:
            for i in range(3):
                if queue.qsize() < 100:   # 若队列的大小比 100 小，不做任何操作
                    pass
                else:
```

```
                    # 从队列中销售产品
                    message = self.name + '消费了 ' + queue.get()
                    print(message)
            time.sleep(1)
```

（4）分别创建两个代表生产者的线程和 5 个代表消费者的线程，并向队列中添加 500 个产品，具体代码如下。

```
if __name__ == '__main__':
    for i in range(500):
        queue.put('初始产品' + str(i))    # 往队列中放入初始产品 500 个
    for i in range(2):
        producer = Producer()
        producer.start()
    for i in range(5):
        consumer = Consumer()
        consumer.start()
```

代码测试

运行程序，控制台不停地打印如下信息：

```
生成产品 95
生成产品 96
生成产品 97
生成产品 98
生成产品 99
生成产品 100
Thread-3 消费了 初始产品 15
Thread-3 消费了 初始产品 16
Thread-3 消费了 初始产品 17
Thread-4 消费了 初始产品 18
Thread-4 消费了 初始产品 19
Thread-4 消费了 初始产品 20
```

```
生成产品 101
生成产品 102
生成产品 103
生成产品 104
生成产品 105
```