学完就能掌握的Python asyncio 基础课

谢乾坤（青南）
2021-07-31

Agenda

- 术语
- 什么代码可以异步，什么代码不能异步
- asyncio 编程入门
- 总结

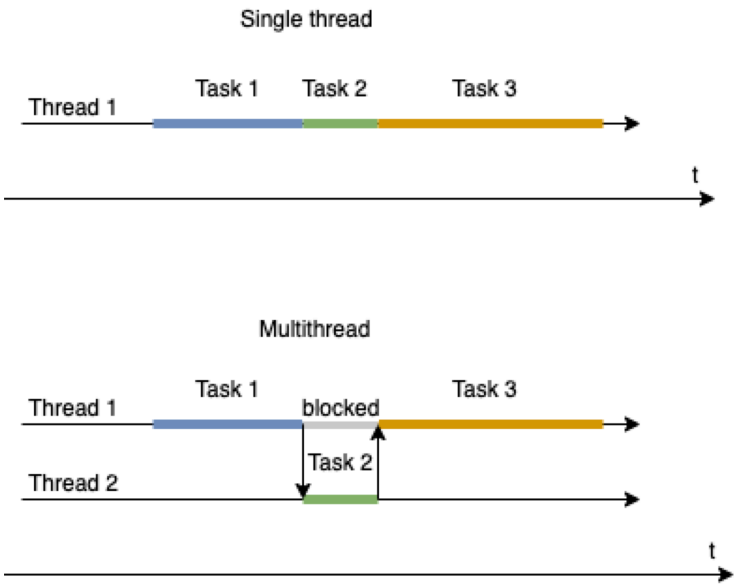
术语


self.file = None
self.fingerprints = set()
self.logdupes = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
self.file = open(os.path.join(path, 'requests.seen'), 'a+')
self.file.seek(0)
self.fingerprints.update(x.rstrip() for x in self.file)
@classmethod
def from_settings(cls, settings):
debug = settings.getbool('DUPEFILTER_DEBUG')
return cls(job_dir(settings), debug)
def request_seen(self, request):
fp = self.request_fingerprint(request)
if fp in self.fingerprints:
return True
self.fingerprints.add(fp)
if self.file:
self.file.write(fp + os.linesep)
def request_fingerprint(self, request):
return request_fingerprint(request)

同步与异步

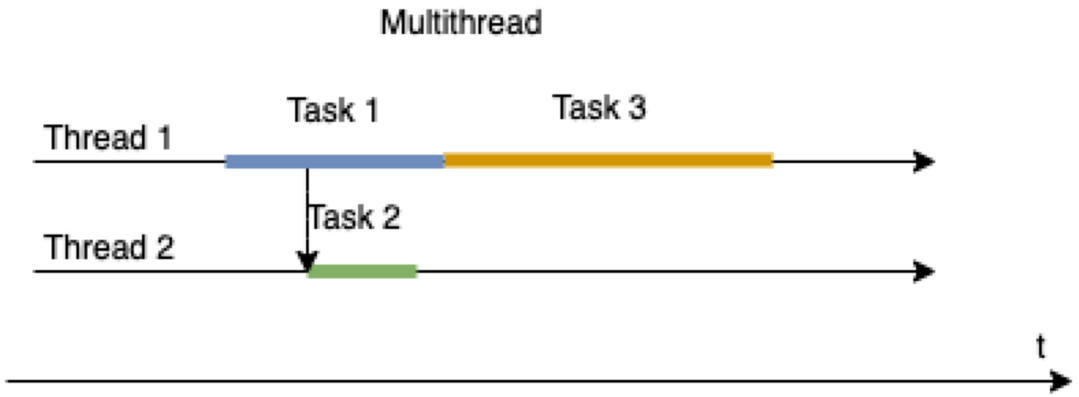
同步（Synchronous）

- 按顺序执行
- A->B->C



异步（Asynchronous）

- 执行 A，但不用等待 A 完成



并发与并行

并发（Concurrency）

- 一个人合理利用时间，做多件事情
- 逻辑上的同时发生
- 宏观上同时发送
- 微观上交替运行

并行（Parallelism）

- 多个人做多件事情
- 物理上的同时发生
- 宏观上同时发生
- 微观上同时发生

Python 的协程（Coroutine）

下面两个概念，都可以简称为协程，我们不做区分

- 协程函数：使用 async def 定义的函数
- 调用协程函数以后返回的对象

Coroutines

Python coroutines are *awaitables* and therefore can be awaited from other coroutines:

```
import asyncio

async def nested():
    return 42

async def main():
    # Nothing happens if we just call "nested()".
    # A coroutine object is created but not awaited,
    # so it *won't run at all*.
    nested()

    # Let's do it differently now and await it:
    print(await nested())  # will print "42".

asyncio.run(main())
```

Important: In this documentation the term "coroutine" can be used for two closely related concepts:

- a *coroutine function*: an `async def` function;
- a *coroutine object*: an object returned by calling a *coroutine function*.

* https://docs.python.org/3/library/asyncio-task.html

Python 的线程与协程的关键区别是什么

线程

- 资源占用高
- 开发者不能限制什么时候切换线程
- 能同时启动的线程少
- 确实有多个线程，但任何时间只有一个线程在做事情，其它线程阻塞

协程

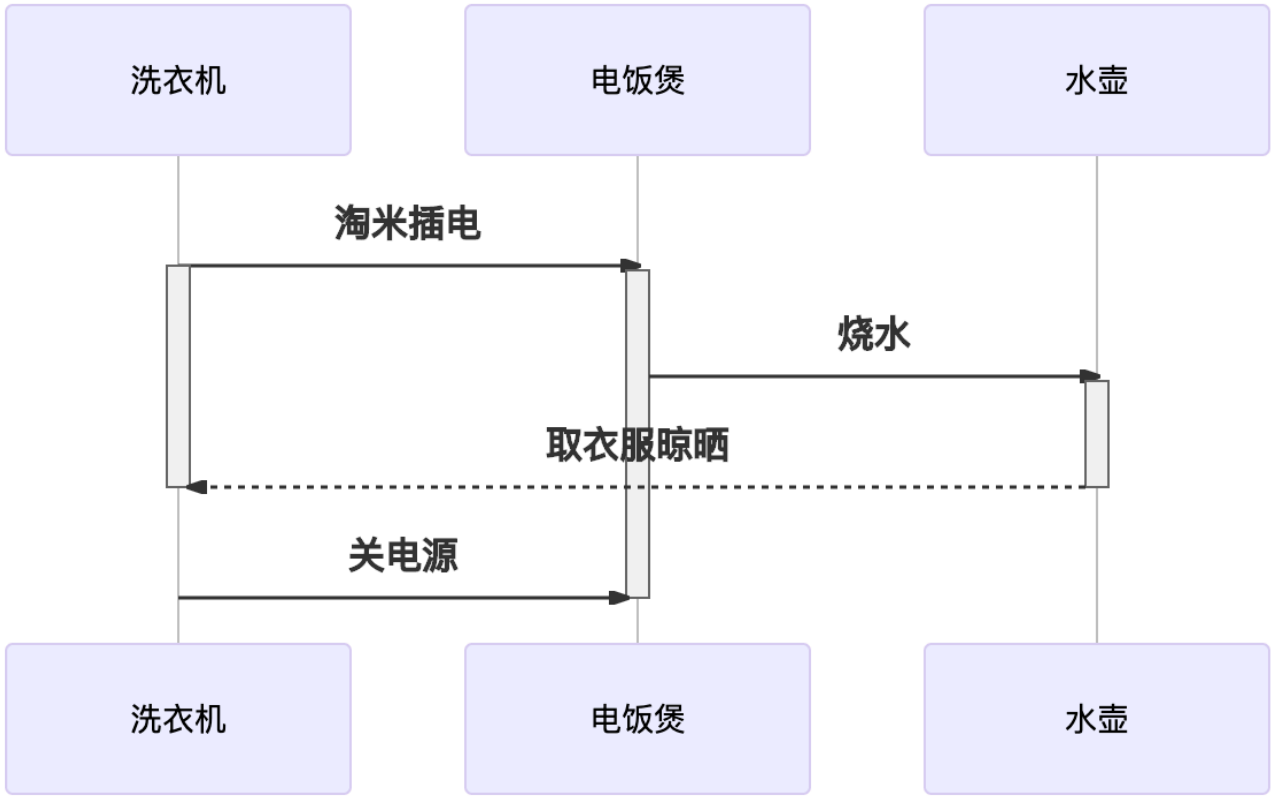
- 资源占用低
- 开发者可以指定切换的时机
- 能同时调度的协程多
- 总共只有一个线程在做事情，充分利用了IO 等待时间

我们今天要讲的，是使用asyncio 实现的基于协程的异步高并发程序

什么代码可以异步执行
什么代码不能异步执行

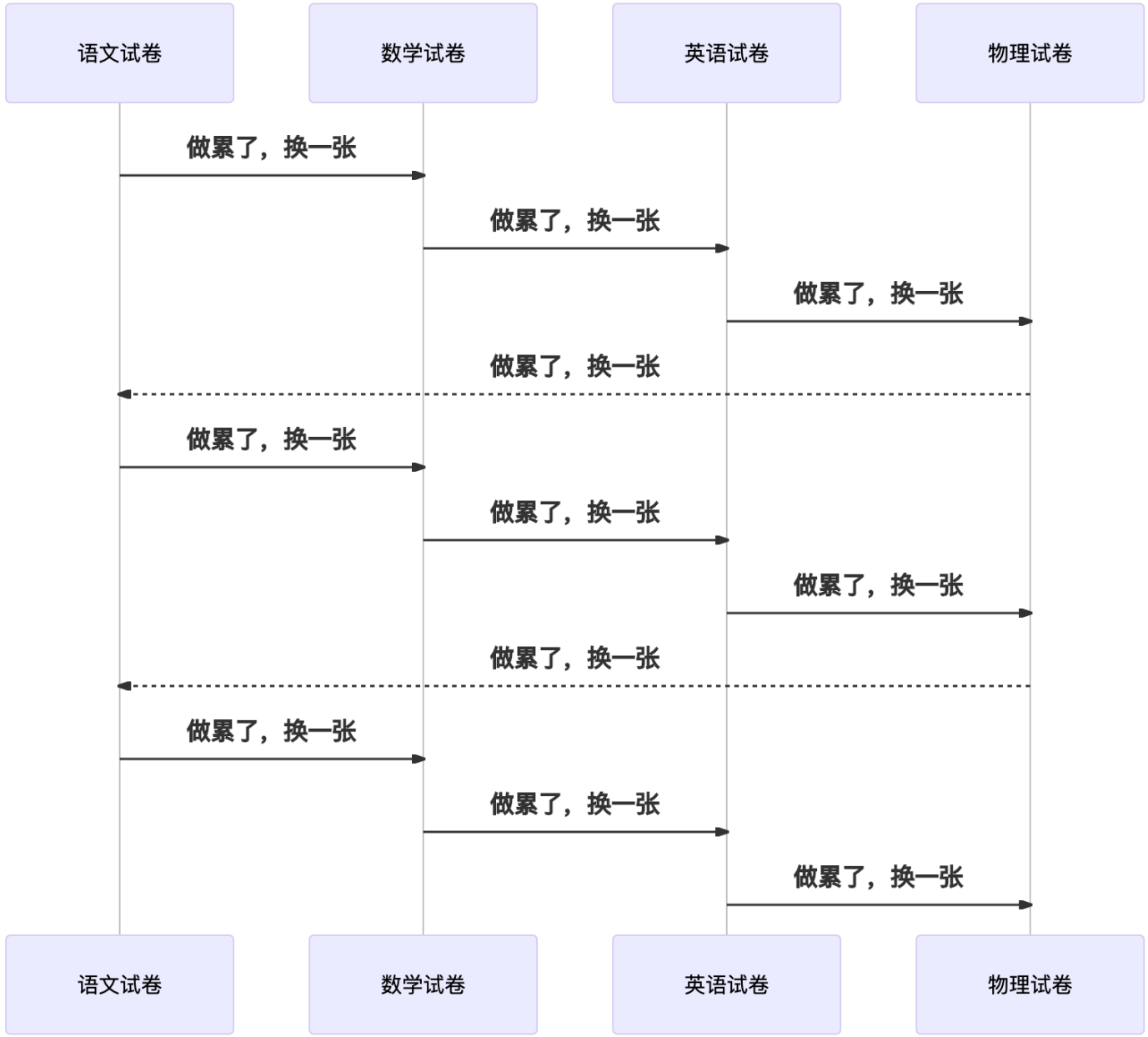
案例1：

洗衣机、电饭煲、水壶都是在人工启动以后，就能自动工作的设备。



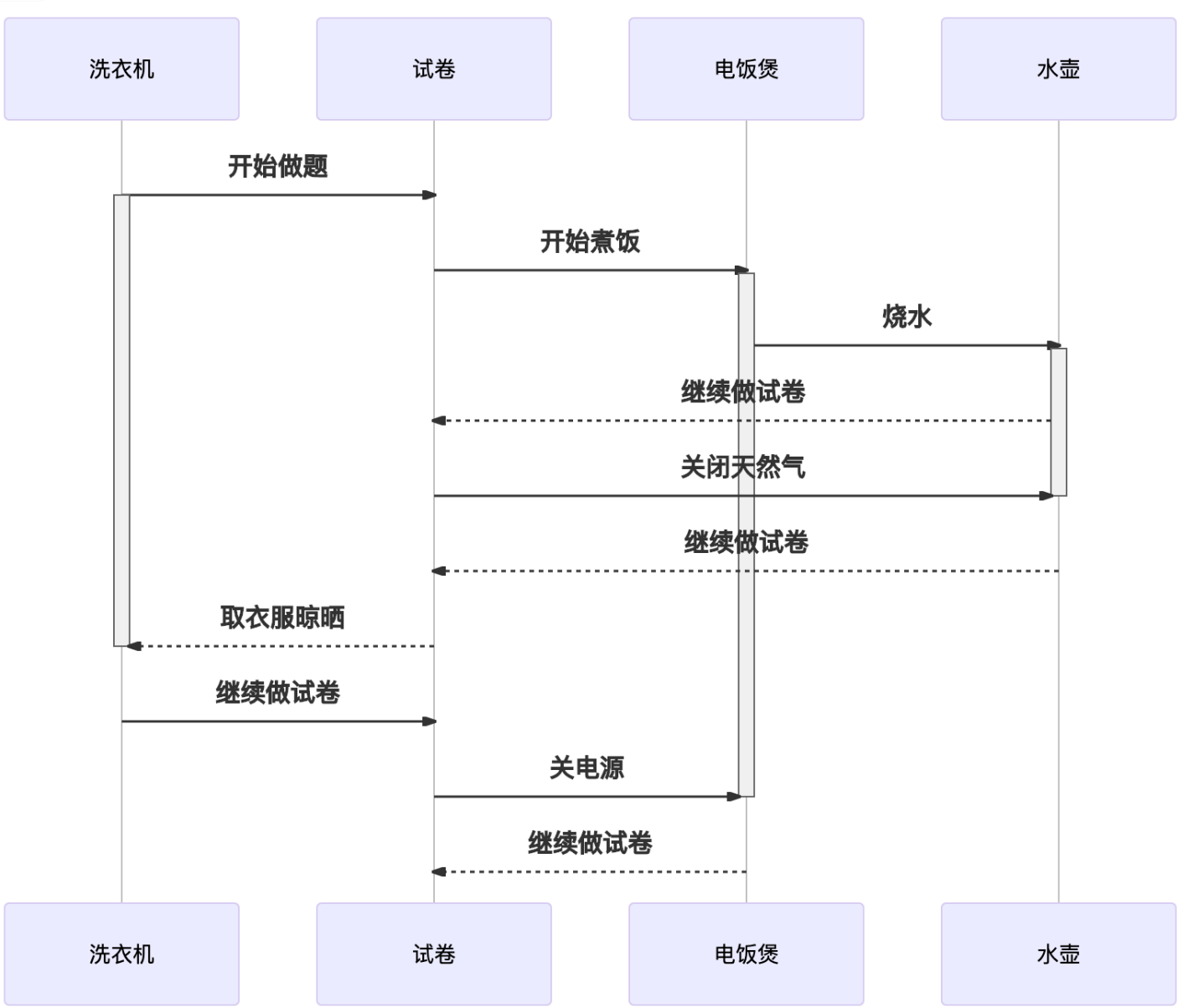
案例2：

每一张试卷都需要人主动去填写，试卷无法自己完成自己。



案例3：

混合执行两种不同类型的工作。



asyncio
编程入门

来看代码演示

总结

- 只有 IO 相关的代码，使用协程才有优势
- 使用 async def 定义协程
- 使用 await 等待协程返回并切换另一个协程
- 入口协程通过 asyncio.run 来调用
- 使用 asyncio.gather 让 asyncio 自动调度不同的协程
- 耗时的同步函数会阻塞协程

main.py

```
import asyncio
async def func():
  print('我是异步函数')

async def main():
  await func()
  tasks = [func2(), func3()]
  await
asyncio.gather(*tasks)


asyncio.run(main())
```

谢谢大家