1. unit test

web开发: 需求分析，设计，实现，测试: 写某一个函数: 可能这个函数写出来的原因总是那个，所以函数对应的逻辑总是相同的，这时写完了只是说明语法没有问题，但是需求是否实现还要看: 因为函数、参数列表、返回值都没有变，变的只是内部函数是怎么写的，所以函数对外界调用者的表现形式是一样的，所以测试也总是相同的逻辑，这个时候最好把测试的逻辑写成一段代码封装起来：单元测试

1. assert: 扮演先知的角色: 单元测试预测到某个位置应该有什么样的值
2. 其实assert就是要考虑所有的可能的但是不满足参数要求的各种情况, 并抛出AssertionError; 相当于在执行最终程序的时候之前埋雷
3. 试图函数中的登录: 从前端拿到传过来的用户名和密码信息: 是表单中的信息以POST方式传过来的**(所以装饰试图函数的路由中要写`methods=["POST"]`)**: `user_name = request.form.get("username")`
4. 测试web后端的代码, 万能的方法是利用类似爬虫的方法: 用request/urllib进行模拟浏览器向后端发送数据获取相应,后端响应之后会返回数据
5. 不过flask中提供了测试模拟前端的客户端: `client = app.test_client()`
6. 拿到返回值之后进行断言测试: flask测试中可以不用python定义好的`assert`, 二是直接调用封装好的方法: `self.assertIn(), self.assertEqual()` etc.
7. **其实单元测试要模拟所有的可能情况,**
8. 测试的重要代码:

```
import unittest

class LoginTest(unittest.TestCase):
    def setUp(self):
        # 开启testing模式
        app.testing = True
        # app中的test client端开启
        self.client = app.test_client()
```

1. 测试: 网络接口/试图函数; 数据库