基于python2.7的某阅读刷题工具

概述：基于python2.7的经典阅读刷题工具，基于seleuimn库，通过实现页面模拟人工点击来实现自动做题功能。该程序仅为验证技术实现请勿用于商业目的。

实验平台：windows10（x64）FireFox（32）

目录：

0x00 简介

0x01环境配置

0x02使用说明

0x03程序接口

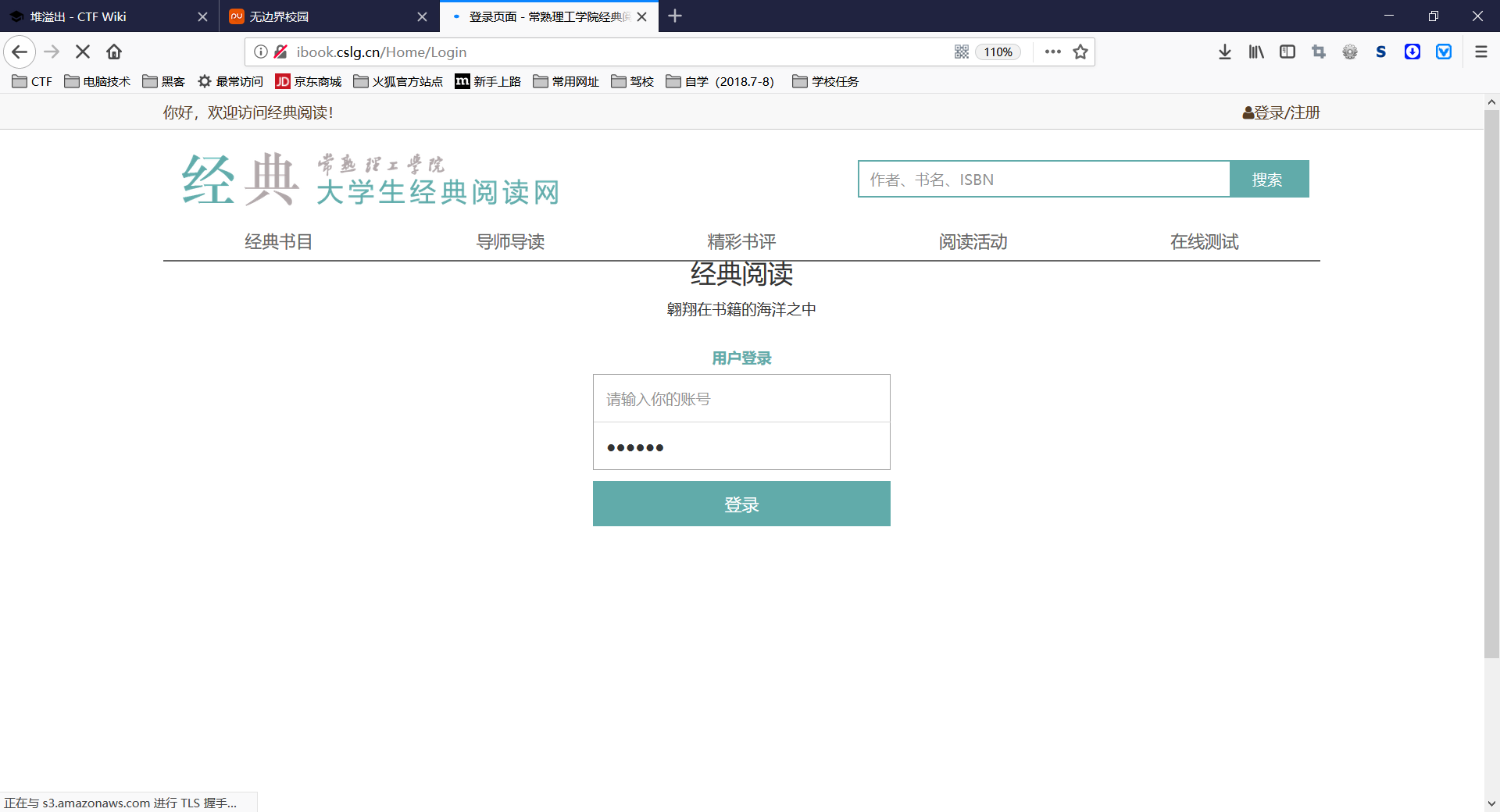
0x00 简介

题目地址：

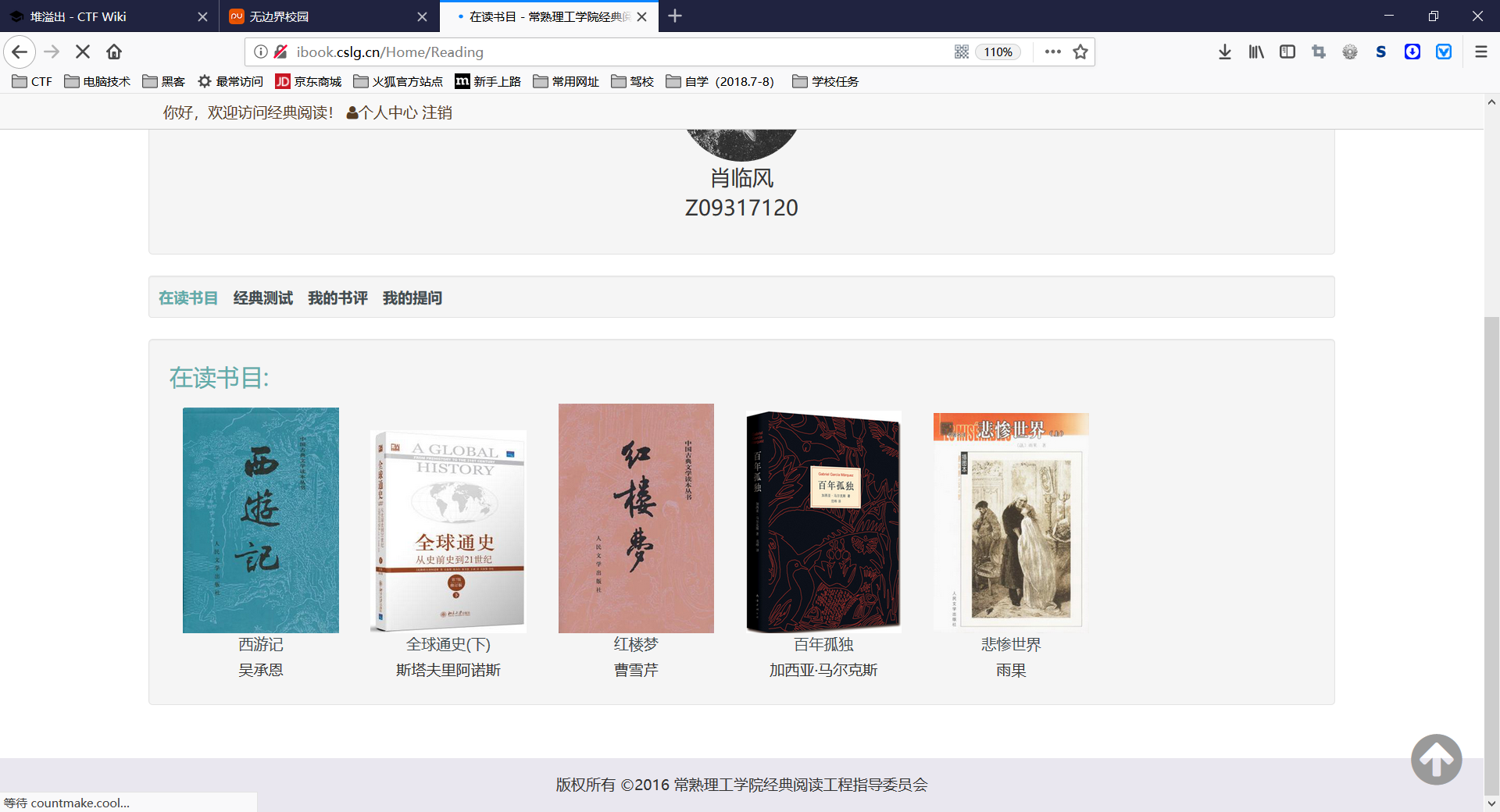
<http://ibook.cslg.cn/Exam>

网页结构分析：

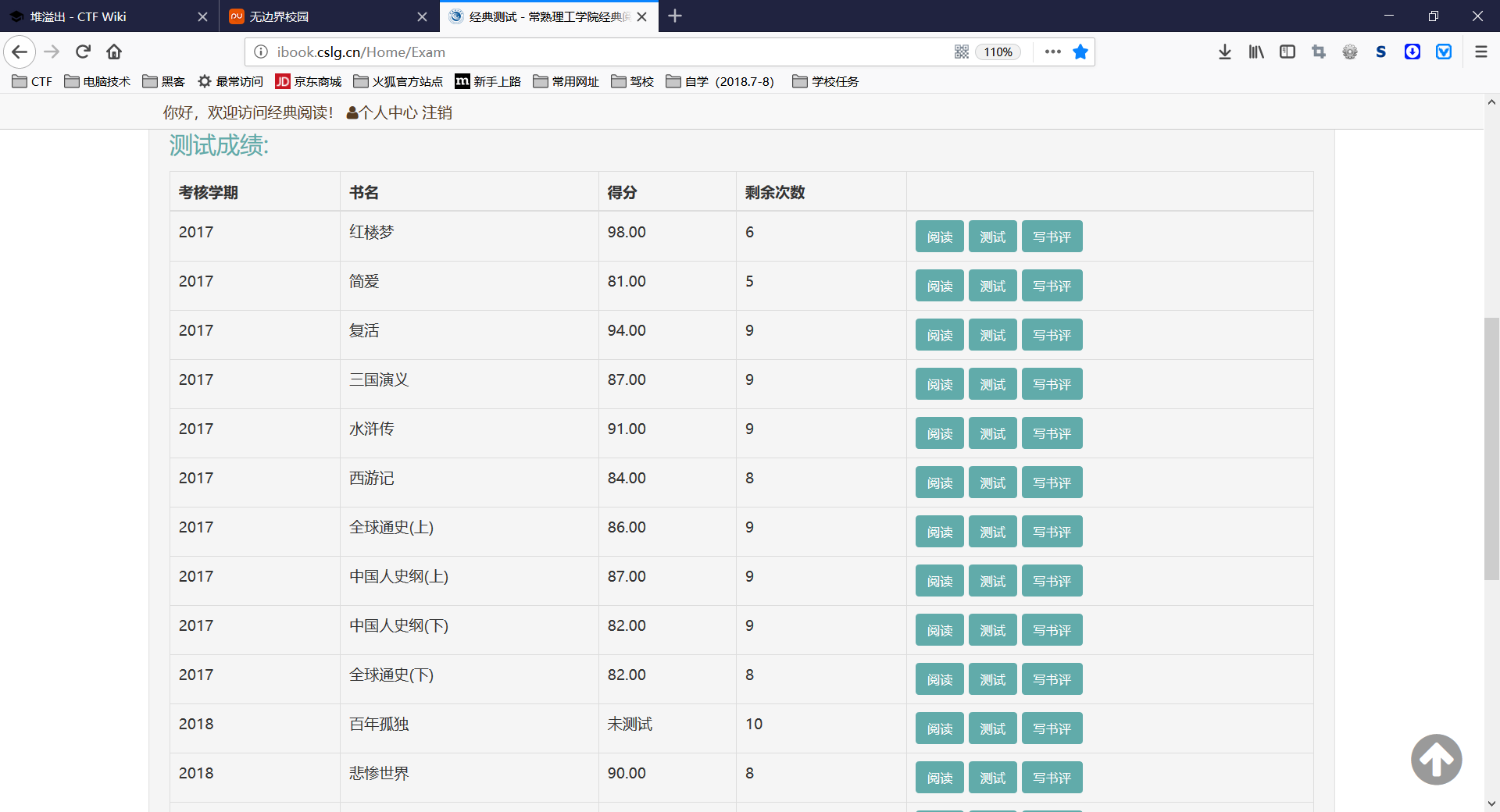
首先是登陆页面，用户输入账号和密码。



登陆之后会进入主页面，存在一个经典测试的URL

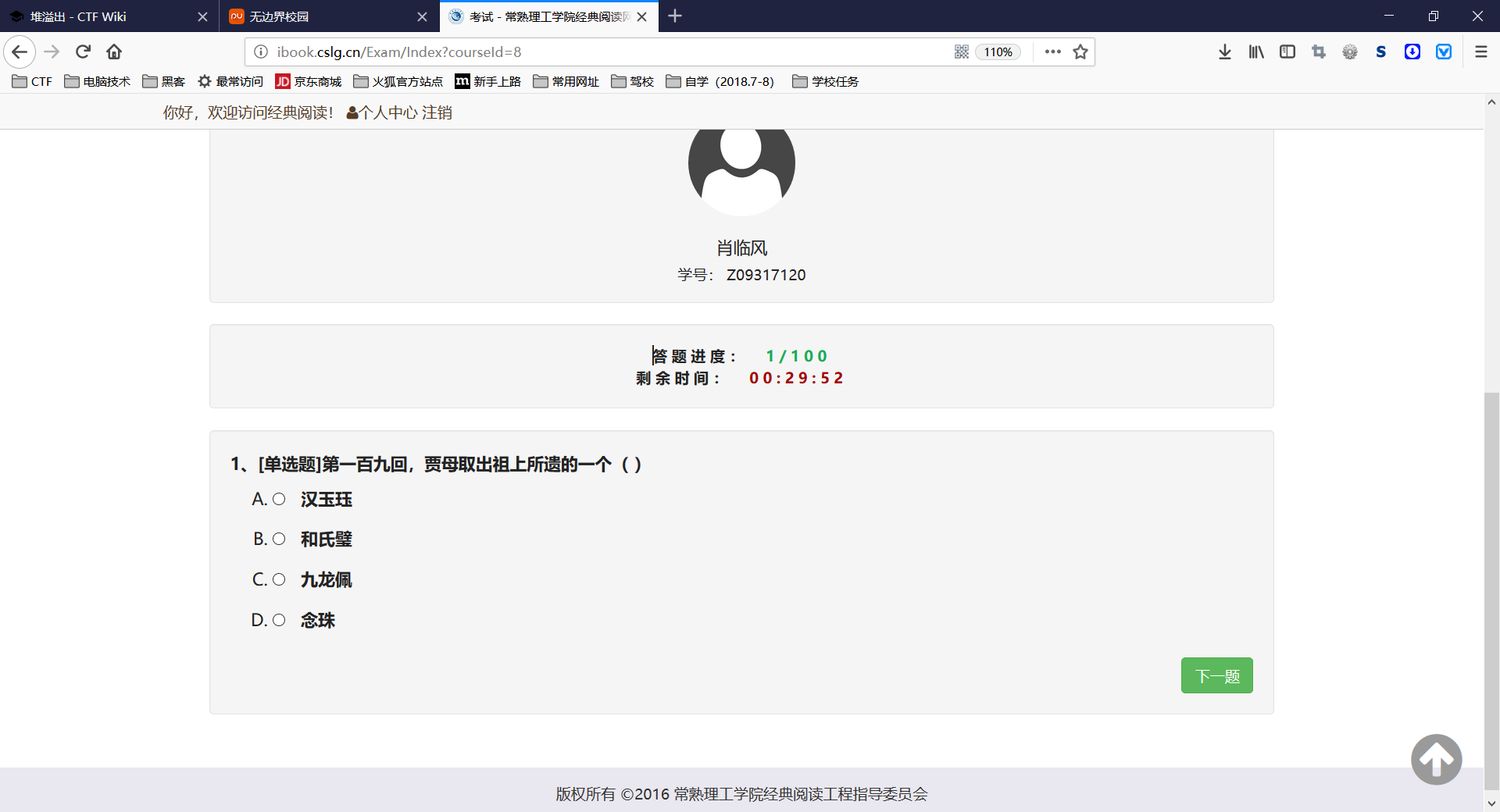


进入测试页面，每个阅读题目都存在一个“测试”选项，选择测试即可进入试卷。



进入测试界面，只有选择后才可以点击下一题。但是值得注意的是，做题过程中不会发送get请求，因此所有题目实际上已经存在于页面中，只是被隐藏了。所以题目都做完了就可以选择交卷。

另外如果不使用火狐浏览器，页面内的题目内容是被Base64加密过的，需要额外解密。



0x01环境配置

源码下载：https://github.com/migraine-sudo/ClassicKiller

配置文件会打包在github上

步骤<<

1.安装python2.7

2.安装pycharm

3.安装seleuimn+BeautifulSoup

4.安装FireFox和对应的驱动

5.运行程序

1.安装python2.7

在官网上下载python2.7,下载到本地安装即可，略

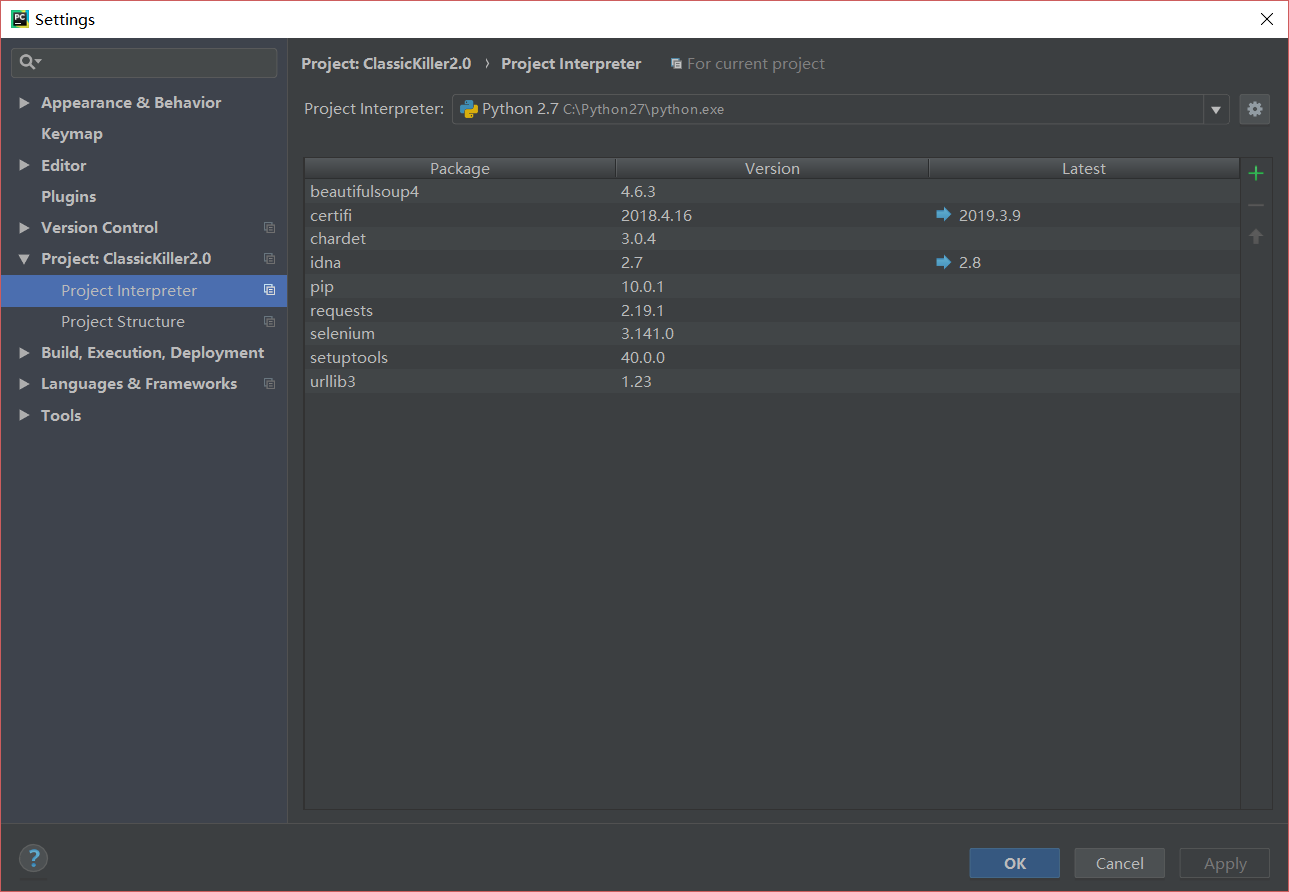
2.安装pycharm（seleuimn+BeautifulSoup）

略

1. 安装seleuimn+BeautifulSoup

在pycharm中Setting中找到Project Interpreter

选择python2.7，点击右边的绿色+号，添加seleuimn，BeatuifulSoup库（如果有遗漏，按照下图补充）



4.安装FireFox和对应的驱动

可以使用官网的安装器安装最新版的FireFox，但是出于稳定还是建议安装61.01版。

并且选择32位安装

#否则需要重新下载对应的驱动。

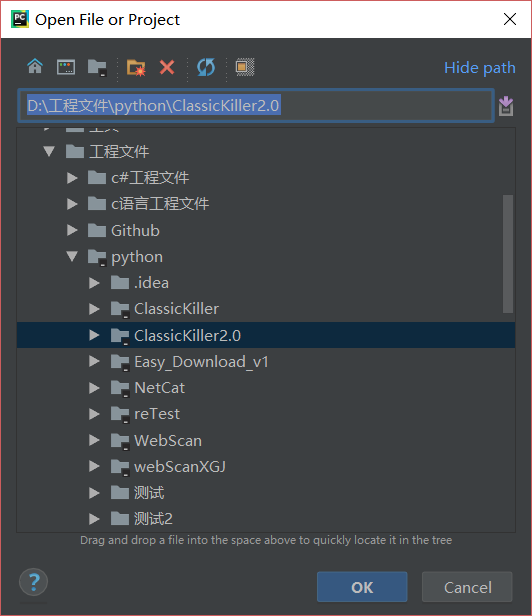


安装驱动

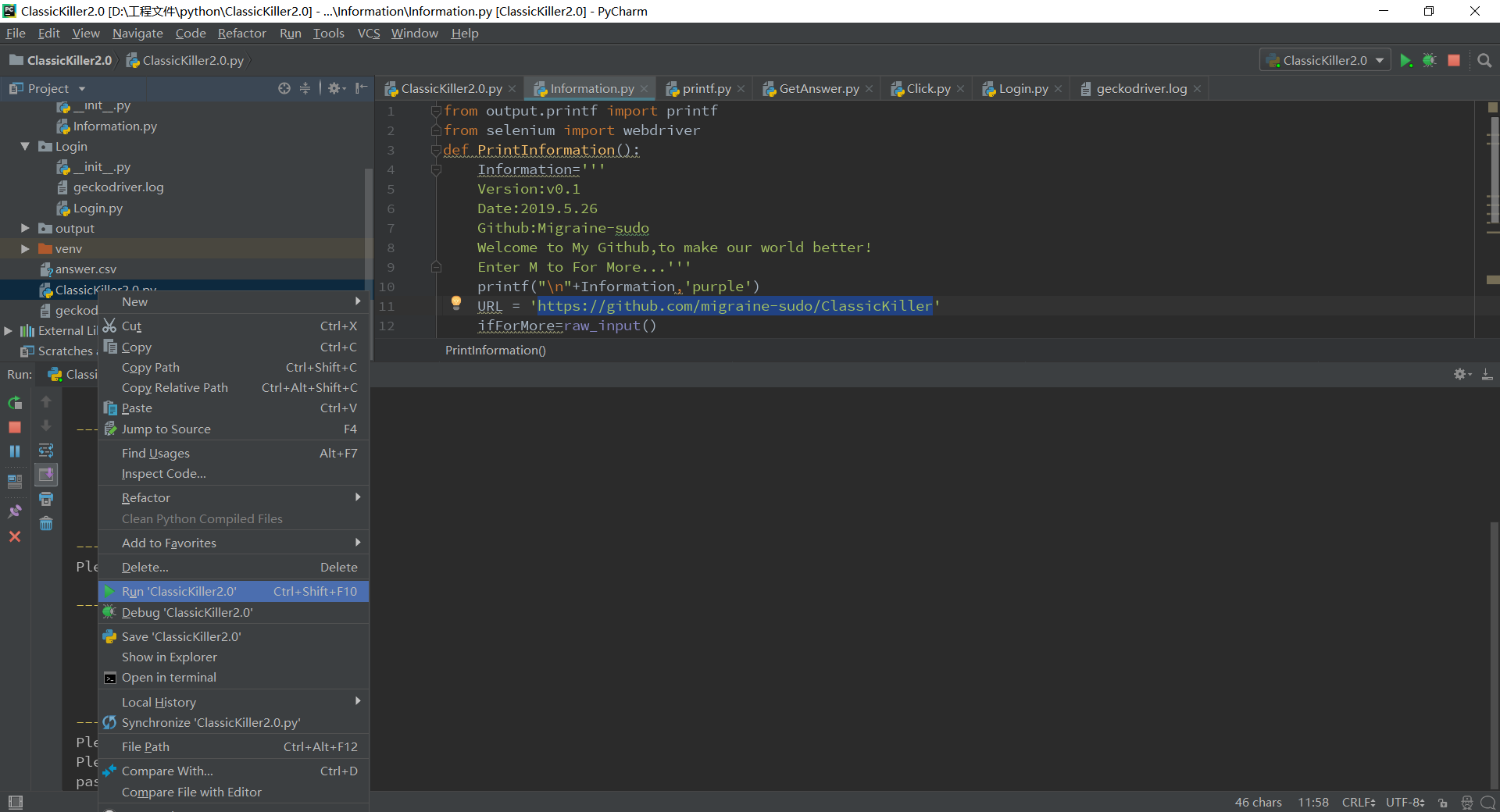
打开环境配置目录里的geckodriver-v0.24.0-win32，解压出来就是浏览器驱动，放入python27的根目录即可。#如果因为浏览器更新而无法使用，请到FireFox的github下载最新的驱动。

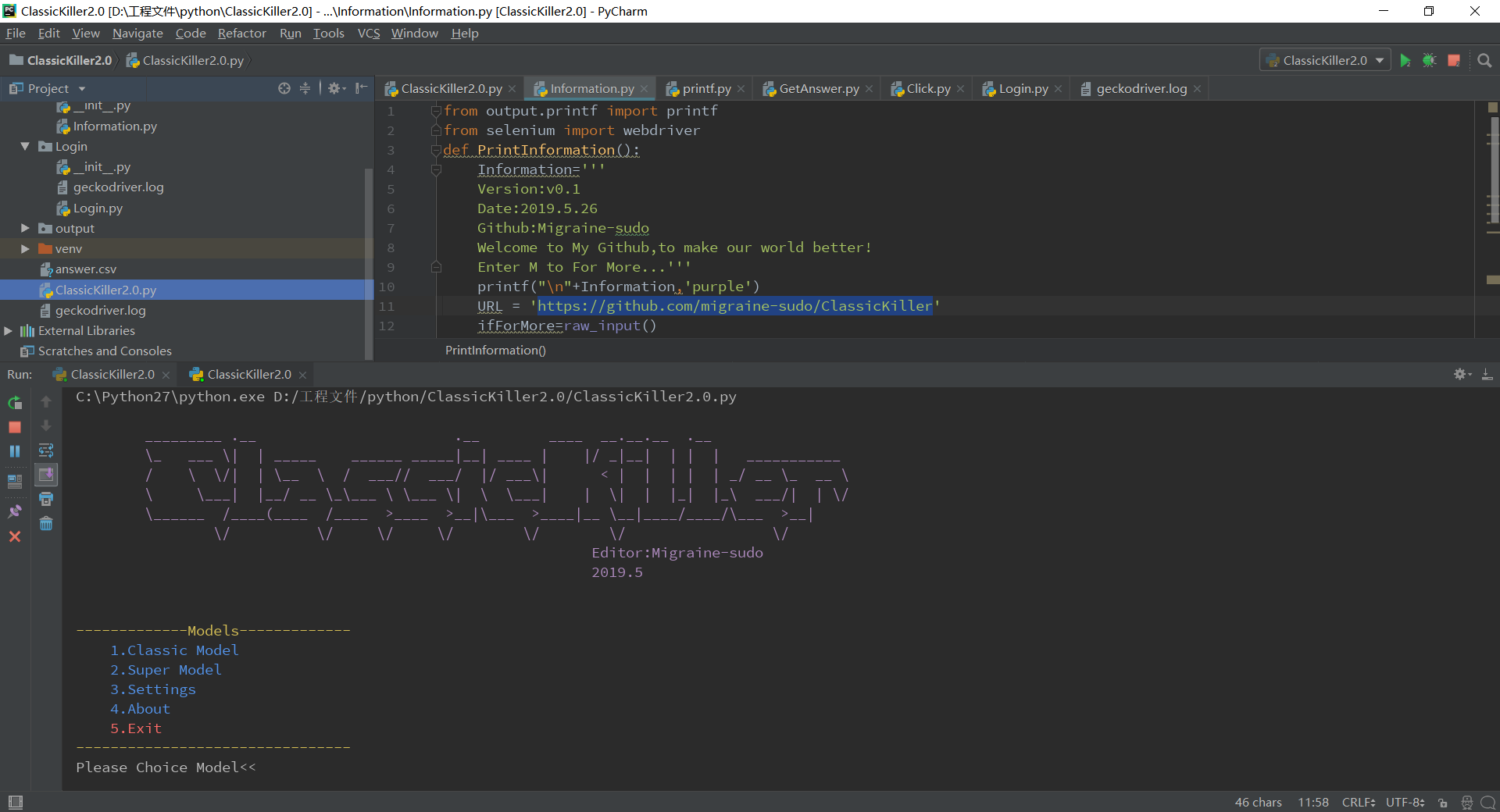
5.运行程序

PyCharm-Open在目录中找到解压好的Classic Killer2.0



在源码中找到ClassicKiller2.0.py文件，右击Run(设置Interpreter为python2.7)，运行成功，显示LOGO！

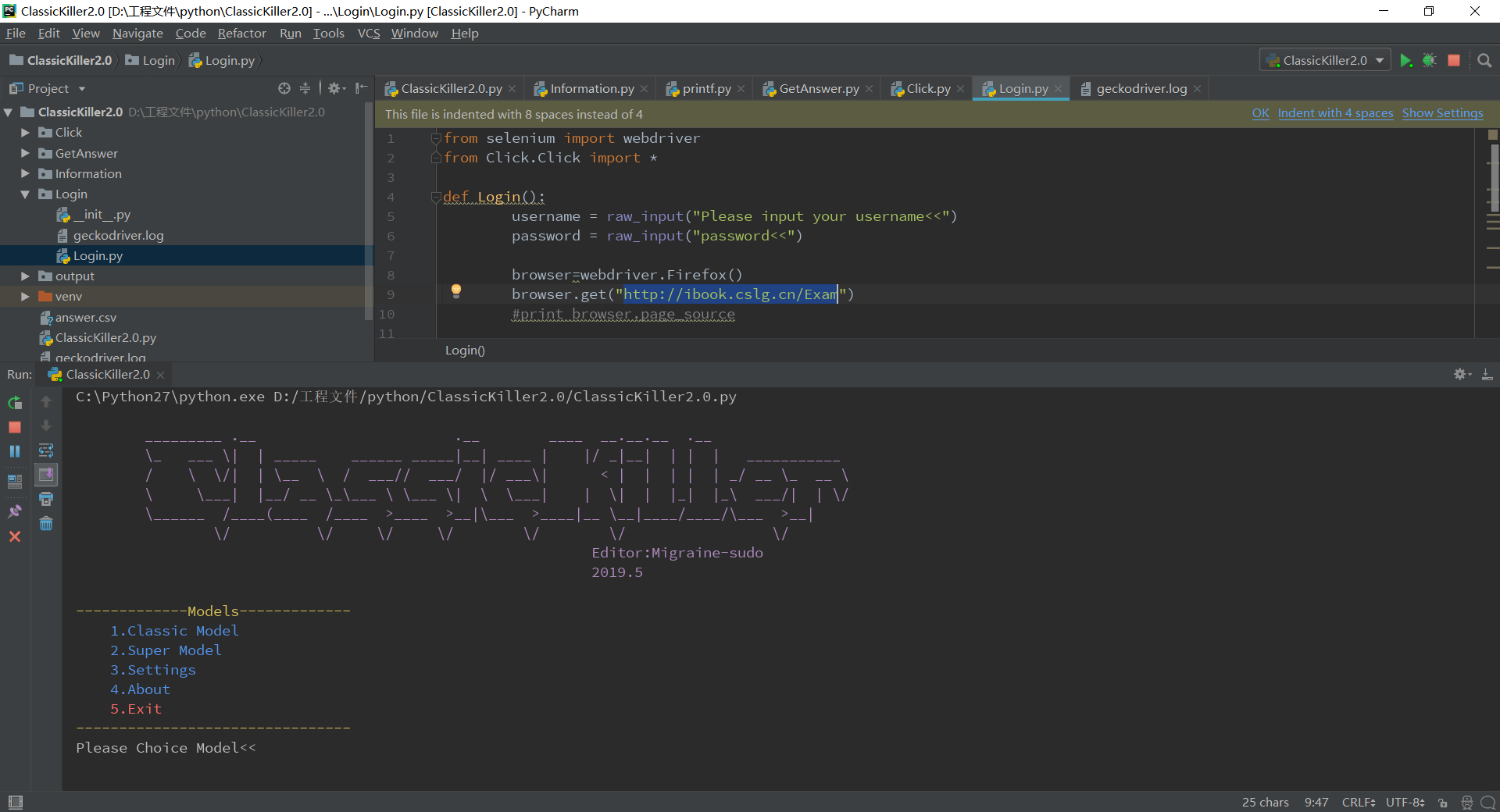




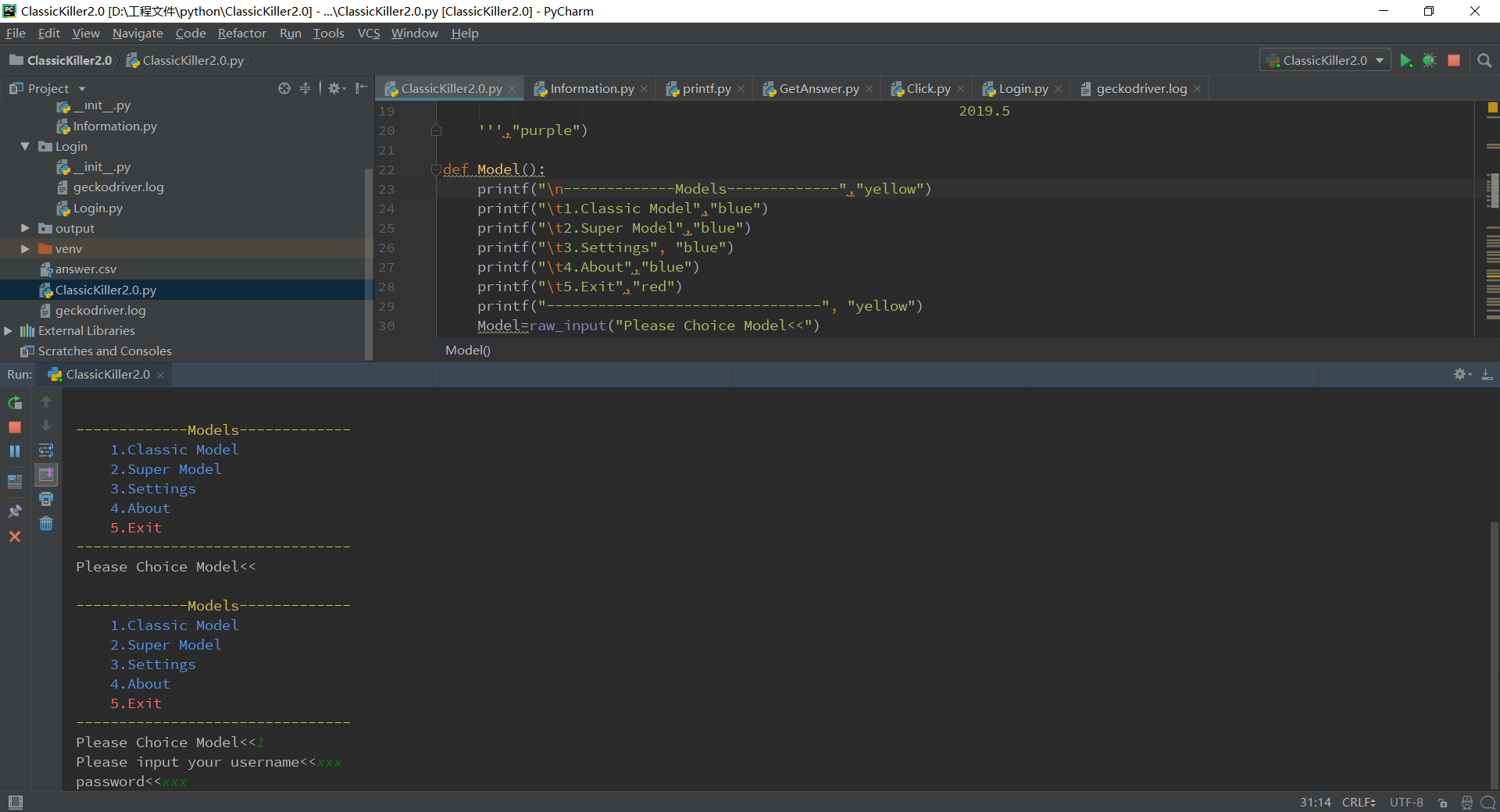
0x02使用说明

运行程序，主页面会打印出选项。只需要输入想要执行的序号，程序就会执行。

1. Classic Model 经典模式，程序会让你输入账号密码和选择的试卷（根据从上到下的排布，输入想要做的试卷序号即可）
2. Super Model 超级模式，全局刷题模式，一次性刷所有题目。#仍在测试阶段
3. Settings 设置
4. About 打印程序版本号，以及访问代码项目库
5. Exit 退出程序

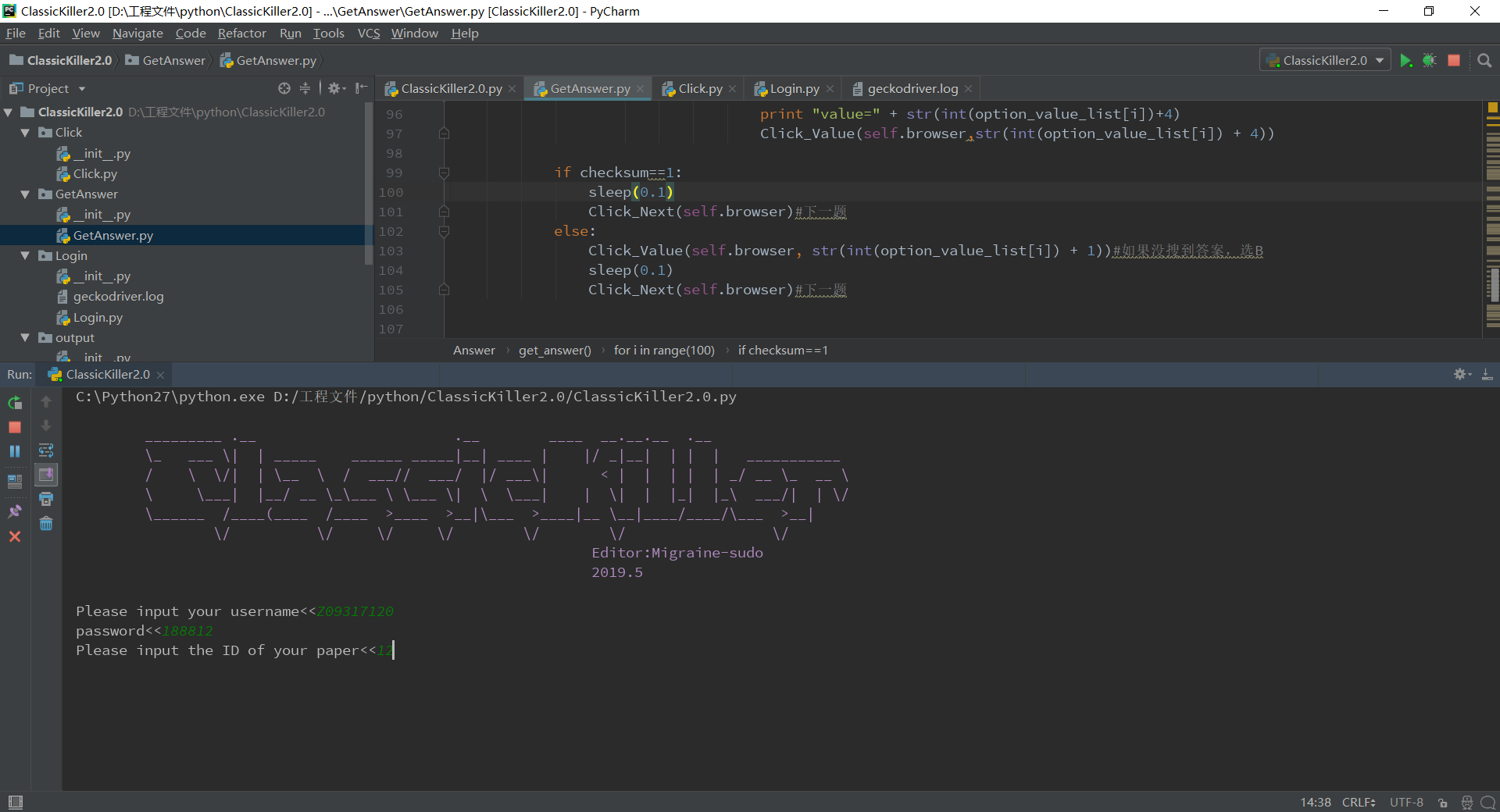


选择经典模式，程序要求输入账号和密码



程序使用FireFox打开题目网页。悲惨世界排在第12个，于是在程序要求输入paperid的位置输入12。





程序自动将答案选填完毕之后，只需要等到可以交卷的时候点提交即可。#这是整个程序唯一需要直接对浏览器操作的地方

交卷成功后，可以看到成功拿到了90.

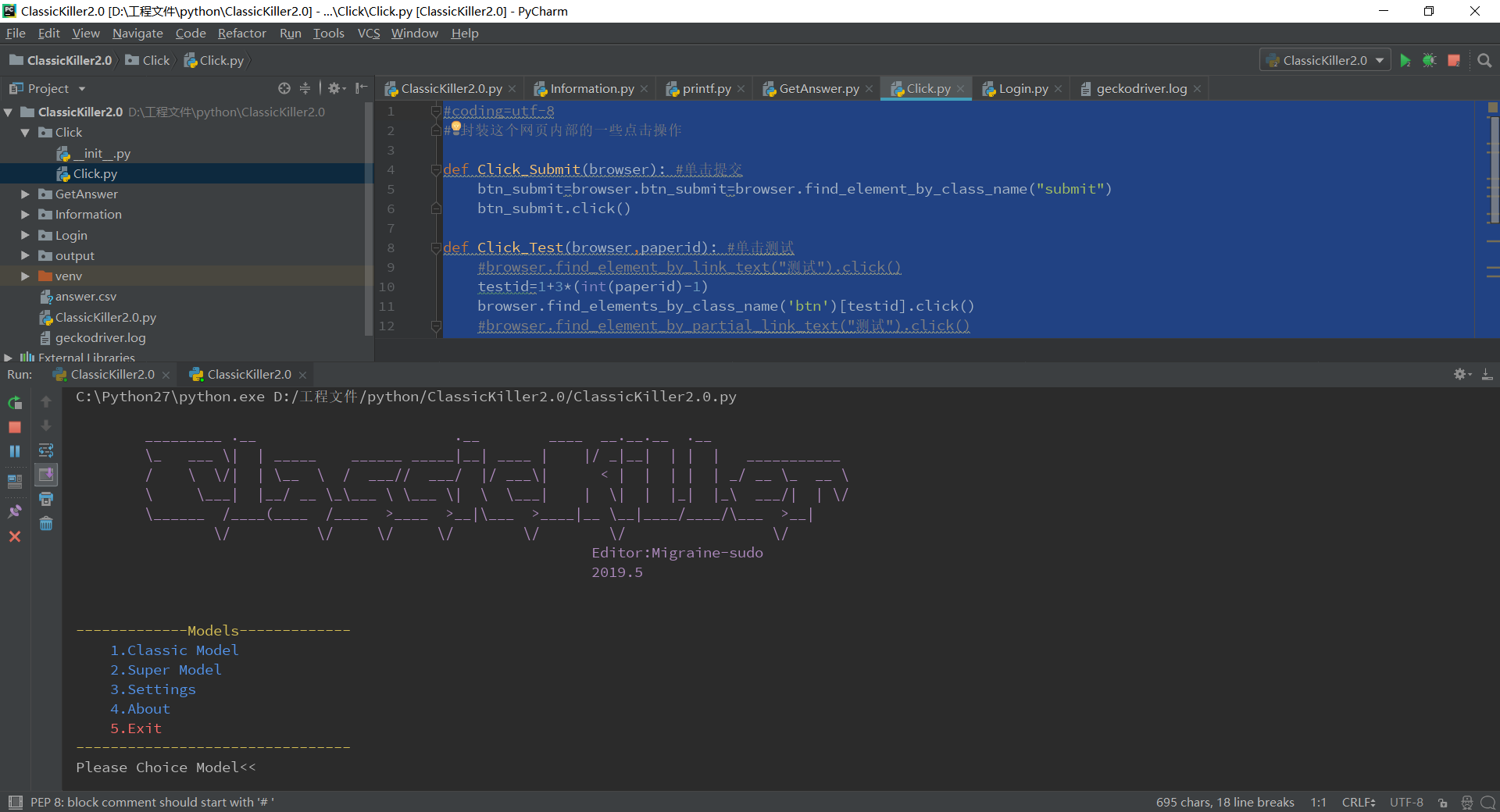


0x03程序接口

Click接口

程序封装了Click接口，将点击操作封装为方法。

实现了 ‘点击下一题’，‘点击选项’ 等功能的自动化



源码如下

#coding=utf-8

# 封装这个网页内部的一些点击操作

def Click\_Submit(browser): #单击提交

btn\_submit=browser.btn\_submit=browser.find\_element\_by\_class\_name("submit")

btn\_submit.click()

def Click\_Test(browser,paperid): #单击测试

#browser.find\_element\_by\_link\_text("测试").click()

testid=1+3\*(int(paperid)-1)

browser.find\_elements\_by\_class\_name('btn')[testid].click()

#browser.find\_element\_by\_partial\_link\_text("测试").click()

#browser.find\_element\_by\_class\_name("btn btn-defualt btn-sm")[1].click()

def Click\_Next(browser): #单击下一题

browser.find\_element\_by\_xpath("//\*[contains(text(),'下一题')]").click()

def Click\_Value(browser,value):

browser.find\_element("id",value).click() #因为源码中id和value值相同