Python 编程基础与 HTTP 接口测试

阿奎　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书主要包含四大部分内容，第一部分概括介绍了 Python 语言流行的原因，以及测试人员如何学习接口测试与自动化测试；第二部分为 Python 编程基础；第三部分是 HTTP 协议；第四部分是 HTTP 接口测试实战。前三部分让读者可以从零开始，逐步掌握 Python 语言，具备基本的 Python 编程基础。通过练习的方式了解 HTTP 协议中常用的概念和机制。最后，第四部分将两者相结合，使读者学会用 Python 进行 HTTP 接口测试，主要采用 Python 的单元测试方法进行 HTTP 接口测试工作。通过一步一步的练习，读者可掌握 Unittest 单元测试框架的使用，掌握测试数据的外部化（到文件和到 Excel 等），掌握 HTTP 接口的发现和探测。

本书亦学亦练，学练结合，每个章节分为多个主题小节。小节的前半部分会对本小节的知识点主题进行详细的介绍，后半部分会启动一个练习，让读者边阅读边练习，在练习中检验学习的成果。

本书适合所有对 Python 语言和测试感兴趣的程序员、编程者、测试人员，也适合高校计算机专业学生补充学习、扩充视野。

图书在版编目（CIP）数据

Python 编程基础与 HTTP 接口测试 / 阿奎编著. —北京：电子工业出版社，2018.1
ISBN 978-7-121-32995-1

Ⅰ. ①P… Ⅱ. ①阿… Ⅲ. ①软件工具－程序设计②计算机网络－通信协议 Ⅳ. ①TP311.56②TN915.04

中国版本图书馆 CIP 数据核字（2017）第 264345 号

策划编辑： 张瑞喜
责任编辑： 张瑞喜
印　　刷： 中国电影出版社印刷厂
装　　订： 中国电影出版社印刷厂
出版发行： 电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本： 710×1000　1/16　印张：14.5　字数：252 千字
版　　次： 2018 年 1 月第 1 版
印　　次： 2018 年 1 月第 1 次印刷
定　　价： 45.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：zhangruixi@phei.com.cn。

序 1

身处软件开发这个颇具颠覆性的行业，又恰逢智能技术的爆发期，很多认知都在被刷新着。和阿奎相识源于组织的敏捷转型合作，初几次见面印象是此人技术问得很细，聊了很多关于代码重构和自动化测试的观点，这可能也是敏捷圈子里最能够试探彼此是否为圈内人士的方法。那个时候关于“测试已死”的观点还是颇为流行的，当然意思是传统的人肉测试最终都会被自动化测试所取代，而开发人员通过测试驱动开发这样的技术实践自然就能生成自动化测试集，那专职测试还留着作甚？

前两周圈子里的人工智能专家给我转来一篇论文，快速扫描后大意是一组科研人员正通过语义和图形分析的技术对算法类论文进行直接算法生成，也就是说给这程序一个算法描述，它就还你一个真正的算法实现。当然现实中给一个高保真网页图片，还你一个真实网站的在线服务已经存在了。稍加思索你会得出一个具有讽刺意味的推论：是不是“开发已死”？具备一定智能的程序已经可以自己产生相应的程序了。而测试却好像并没有死，反而这两年从探索性测试到针对新交互渠道的感知测试还越来越火了。

看到这本书的时候，很高兴当年和阿奎讨论的一个观点现在仍然是对的，那就是“测试人员必须写代码”。不管是现代的分布式互联网应用，还是即将到来的物联网设备，其测试的难度都提升了很多，甚至可以说如果不采用自动化的程序是不可能进行有效测试的。想像一下把应用商城刷积分的人肉模式复制到未来一平米可能有上万个物联网设备接入的场景下，你可能需要一个 Alpha Go 来完成这样的“人肉”测试。有幸的是我认识了几位把测试人员写代码真正贯彻和执行下去的管理者，阿奎就是其中一位。

Python 作为编程语言自不用多讲，曾经在测试领域因为其灵活性大受欢迎，积累了不少有用的测试库，即使现在很多项目上我们还会日常使用。而大数据

时代的来到更催生了这门语言的学习热潮，不少业务人员开始学习 Python 作为数据分析的编程语言。时下的大数据分析不同于之前既定规则的数据处理，更多是在大数据中去挖掘和探索，某种意义上我们又何尝不可以说是在进行“数据测试”呢？所以学习 Python 应该是符合时代背景高投入产出比的事情。

从互联网到移动互联网，再到物联网，很多技术都会被改变，最有可能持续发光发热的“组件”可能就是我们的应用通信协议 HTTP 了。针对 HTTP 通信的接口测试也是最常见的，经常也是最繁琐的。学习如何进行有效的自动化是避免自己被机器人取代的必备技能。

最后，很感谢阿奎把自己的学习心得和方法也融汇到了此书中。刻意练习是技术学习的不二窍门，如何坚持确是相当有难度的一件事情。和广大读者一样，阿奎并非身处一个纯粹的技术公司，日常有很多的管理会议和业务需求，所以他的学习经验分享更值得大家借鉴和试验。

肖然
精益敏捷专家
ThoughtWorks 咨询与设计总监

序 2

转眼，阿奎工作已经 13 个年头了，先后做过程序员、软件工程师、团队经理，现在是一名系统分析师和内部敏捷教练，积极地参与组织中的敏捷实践和推广工作。

在担任团队经理期间，阿奎先后带过 C/C++、Cobol 和 Java 为开发语言的开发团队，多年来一直保持对技术研发工作的热爱。最有意思的是，多年以前他还在一个以主机 Cobol 为开发语言的部门的时候，竟然会去参加以 Java 为语言栈的编程大赛。

接触最多的是 2013 年调到新技术实验室的时候，那个时候互联网金融风头正盛，敏捷开发方法作为一种新的软件开发方法，正在被各个大型企业所认可和尝试。那是他接触敏捷的缘起，甚至可以说也是这本书的缘起。

阿奎自 2013 年开始从事敏捷实践和推广工作，他将自己定位为一名技术性管理者，在从事团队管理的同时，一直保持着对各种新技术的关注，在工作中对于如何提升团队的工作效率也多有思考。众所周知，敏捷开发转变的核心在于人的转变，而在人的转变过程中，人员能力的提升和对新的开发模式的适应又是尤为重要的。

随着敏捷、精益、DevOps 等新的软件开发理念的盛行，软件测试的从业人员也正在经历工作环境和工作要求的改变。这种改变需要软件测试从业者成为真正的 IT 人，具备计算机基础理论知识和网络知识，掌握一门编程语言，熟悉一个操作系统和一个数据库。

环顾业内，很多软件测试从业者都正在经历从“点点点”工程师到自动化测试工程师的转变，这个转变是痛苦的，也是迷茫的。阿奎就是看到了这样一个需要，利用一年的业余时间，写了这本书，其中的每一个练习都是他精心设计的，并且都自己进行了实战和解答。这本书是他专门针对没有编程基础的软

件测试工程师如何快速、高效地学会编程这一课题的一次实践和有益的尝试。

书中提到的“闯关式学习”实际上就是“刻意练习”的学习思想，在编程语言学习上的实地应用，并且经过他的一些实验，取得了非常好的学习效果。

特别祝贺阿奎，能够将自己长期工作、学习的收获和心得，通过一本书的形式进行一次总结，特别是这样一本帮助大家学习编程的书。

薛勇
主任工程师
中国银行软件中心

前言

每个人都应该花 1 年时间学习编程。

——史蒂夫•乔布斯

我一直以来有一个观点，对于希望从事自动化测试工作或者希望掌握自动化测试技能的人士，掌握一门计算机语言是绕不过去的一个坎。就是这个坎，让很多从事手工测试的工程师对于自动化测试工作望而却步，也让很多人不由自主地退而求其次，转向去学习一些自动化测试的工具，来回避语言学习的困难。

看到身边很多的测试工程师对待编程语言的学习畏之如虎，谈之无力，让我产生了**“为希望转型成为自动化测试工程师的软件测试从业者提供一本靠谱的编程语言学习指导书”**的想法。这也是促成这本书初稿形成的原始动力。

随着信息技术的发展，计算机办公技能成为一名现代白领工作者的必备技能，我们日常工作中有很多的时间都是在与计算机进行交互。通过对计算机语言的学习，可以为你打开一扇与计算机进行深层次互动交流的大门。同时，在编程语言的学习中，你会学到一种新的思考方式并找到另一种看待问题、解决问题的视角。

本书的内容包含两部分：Python 编程基础和基于 Python 的 HTTP 接口测试。

在 Python 编程基础部分，主要关注 Python 语言的基础知识的学习和掌握，这一部分对于每一名希望快速掌握一门计算机语言的学习者都是适用的。

基于 Python 的 HTTP 接口测试部分，重点面向希望转型为自动化测试工程师的软件测试从业者，以 HTTP 接口测试为应用场景，来学习和掌握 Python 的相关知识点和使用技巧。

本书以**闯关式学习方法**为指导进行编写，让读者通过一个一个小的关卡的

刻意练习，在不知不觉中掌握“Python 编程基础”和“HTTP 接口测试”的相关技能，完成从手工测试工程师到自动化测试工程师的转变。

所谓**闯关式学习方法，通俗地讲，就是通过不断的刻意练习，打通一个一个的练习关卡来进行自我提升和学习的方法。**闯关式学习方法首先要求有一名有经验的指导者，针对学习目标刻意设计的一个又一个的练习关卡，即挑战问题，读者通过一定的前期知识的学习和练习后，在给定的时间内对第一个关卡的问题发起挑战，即闯关。如果给定的时间内不能有效地完成挑战问题，就需要反复的刻意练习，直至轻松完成挑战问题为止，即闯关成功。此时，才可以进入下一个练习关卡。详细的关于如何使用本书进行闯关式学习的内容，请读者阅读本书的“如何阅读和使用本书”部分。

我一直认为，学习软件开发、测试技能，和学习骑自行车、游泳一样，是一项技能的修炼，而非仅仅是知识或者概念的了解和掌握。要习得一项技能需要刻意地练习。

编程作为一门技艺，是可以习得的，习得是有方法的！这本书为愿意学习的读者提供了习得编程技艺的方法，就是“闯关式学习”。

但是,“没有任何有效的学习和精进的过程是惬意的和不需要付出努力的。”

所以，**这是一本针对 Python 编程基础和 HTTP 接口测试技能，进行刻意练习的学习指南，这不是一本可以靠在沙发上阅读的消遣书。**

“学习之路挖坑容易挖井难”，祝大家利用这本精心打造的“闯关之书”，挖出属于自己的“Python 之井”。

阿奎（于洪奎）

如何阅读和使用本书

相信很多读者都有过自学软件测试或者其他技能的经历，期间坚持与放弃的挣扎，学习方向的迷茫，学习资料的收集，学习内容的选择……甘苦自知。

作家格拉德威尔在《异类》一书中指出：“人们眼中的天才之所以卓越非凡，并非天资超人一等，而是付出了持续不断的努力。1 万小时的**锤炼**是任何人从平凡变成超凡的必要条件。”他将此称为“一万小时定律”。要成为某个领域的专家，需要 10000 小时，按比例计算就是：如果每天工作八个小时，一周工作五天，那么成为一个领域的专家至少需要五年。

我并不是想用这段引用来告诉大家，成为自动化测试工程师需要五年的时间，而是希望大家看到其中“锤炼”一词的深意——成就超凡技艺的过程，实际上是一个“锤炼”的过程，也就是“刻意练习”的过程。

既然叫做刻意练习，有别于随意的练习。随意的练习并不能带来有效的学习和精进，并且会浪费时间，打击练习者的信心。没有任何有效的学习和精进的过程是惬意的，不需要付出努力的，要做到刻意练习有如下四个要点：

- 有目的的练习。刻意练习一定是针对某一个既定的目的进行练习，即每一个练习都是有具体目标的。
- 精神高度集中的练习。刻意练习一定是需要练习者精神高度集中，全身心投入的，只有这样才能带来学习上的领悟和突破。
- 反复的练习。在以上两点的基础上，重复的练习是技艺提升不可缺少的一环，没有什么技巧性的技艺是一次就可以完全掌握和彻底领悟的，编程尤其如此。只有通过不断的重复的练习才能将编程中用到的一些基本的概念和模式，变成自己的肌肉记忆和大脑反射。
- 获得有效反馈的练习。反复练习中，还需要有效的反馈来确定练习是否取得了进展，这种反馈一方面可以促进练习者更好的坚持练习，另一方

面也可以告诉练习者，练习到什么程度就够了。

以上也是闯关式学习方法的理论依据和本书设计编写的初衷。

本书采用“闯关式学习方法”编写。阅读的过程中，需要读者边读边练习，每一章包含多个小节，每个小节都是一个小的关卡。

再次重申：**这是一本针对 Python 编程基础和接口测试技能进行刻意练习的学习指南，这不是一本可以靠在沙发上阅读的消遣书。**

读者在使用这本书的时候，应该是在电脑旁边的，并且电脑是联网的，这样你才能通过刻意练习，成功挑战书中一个一个的小节关卡，体会成功闯关后的喜悦和掌握一个一个技能要点后发自内心的欢喜。

小节关卡

本书的每一个小节一般包含：学习目标、学习资源、知识准备、挑战问题、难点提示、知识总结、拓展问题七个段落。

学习目标：描述本小节的学习目标，即通过本小节的学习掌握的知识要点。

学习资源：当今社会已经是一个信息资源极其丰富的社会，我们不再缺乏信息，而是缺乏优质的，经过整理的高质量信息，本小节会提供一些与学习目标相关的高质量网络学习资源的链接，方便读者进行扩展阅读和深入学习。

知识准备：本小节主要就完成学习目标，需要学习和掌握的知识和概念进行阐释。本小节的阐释不求全面，主要以常用的知识和概念以及技能的阐释和演示为主。更全面详细的信息，读者可以通过学习资源小节中提供的链接进行深入的学习。

挑战问题：本小节要解决的问题。注意只有解决问题才算通过本小节的学习，如果上一个小节的问题没有解决，不建议继续学习，因为，后面小节的解题和闯关过程中一般会用到前面小节的知识点和技能。同时，也是最重要的，小节问题的解决过程中可以借助互联网或者其他资料，但是，编码过程中，必须保证完全自主完成，也就是，在开始解题时候，应该自己完全手工完成，不能直接拷贝粘贴，也不要对着屏幕或者书本看一个单词敲一个单词。软件自动化测试的学习从归类上属于技能类学习，而技能类学习的关键就是刻意练习。所以，闯过关卡不是目的，重要的是切实掌握闯过关卡用到的知识点和技能。

难点提示：在进行小节闯关的过程中可能会碰到一些陷阱或者难于解决的

问题，会在这一部分给予一些提示。

知识总结：将本小节的知识点进行简单列举，需要读者根据知识点中的提示，翻看提供学习材料，对知识点进行深入的学习和必要的记忆、理解。

拓展问题：有的关卡的题目会有一些变化的问法或者多个解答方法，会在这一部分给出说明和提示。

使用指南

（1）学习顺序：建议首先通读一遍本小节的所有内容，再通过翻阅提供的学习资料将知识点中提到的内容记忆和理解，最后再进行本小节问题的解决，即实施闯关，闯关成功后，再将知识总结小节的内容做一个回顾，然后再开始下一个小节的学习。

何为“闯关成功”？即在给定时间内独立完成挑战题目。在完成挑战题目的基础上，有两个要求：在给定时间内独立地完成。

如果完成挑战题目的时间过长或者过程中有一边看资料一边写代码的情况，这都不能算作闯关成功。此时，建议读者删除代码重新进行闯关。

（2）刻意练习：

- 不要拷贝粘贴。
- 不照抄照搬。
- 十分钟原则。
- 举一反三，主动拓展。

备注：

“十分钟原则”：每一个小节的练习问题需要在**十分钟**内解决，一般前面几个小节，读者都不会有问题，但在后面的时候，便很难在这个时间里完成。而所谓的**“十分钟原则”**就是，如果你没有在**十分钟**的时间里完成小节的练习，就将代码删除，重新写一遍，是在编程学习中对“刻意练习”的贯彻。

“举一反三，主动拓展”又称为“不以闯关为目的的闯关”。有的同学练习过程中对于闯关很有兴趣，又有一点编程基础，一上来就一口气闯到了第五关，甚至第九关，这样的闯关是没有意义的。这些关卡的设计过程中，笔者针对每一个关卡都做了两次以上的练习，就是至少删除过一次代码。目的就是更好地

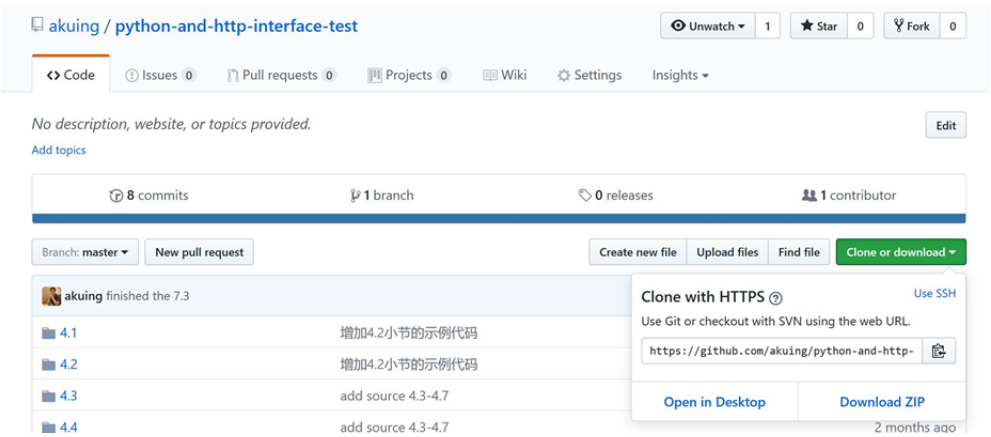
体会学习目标和知识点，更好地形成所谓“肌肉记忆（muscle memory）”。经过思考，笔者还在多个章节设计了拓展环节，就是为了更好地掌握相关知识和技能。“不以闯关为目的的闯关”，才是这些关卡设计的本意。

（3）找到组织：

学习过程中会遇到很多的问题，如果有老师或者同行者，一起互相激励，遇到问题也可以互相请教，在交流中收获更多。读者可以关注本书的公众号：IT 匠艺教研室。公众号中有微信交流群，可以扫描加入，随时进行交流和学习。

为方便读者自学，笔者将本书各个小节的练习代码上传到了 GitHub，读者可以自行下载参考。

地址为：https://github.com/akuing/python-and-http-interface-test。点击右侧的绿色按钮“clone or download”，选择 Download ZIP 即可以下载整个参考代码和示例的压缩包。



本书编写的目的是通过练习，让读者将技能真正掌握到自己手中。所以，不要为了闯关而直接下载运行参考示例代码，这对你的学习不会有任何实际的帮助，只会让你浪费掉一个作者苦心设计的用于检验学习进展的关卡。

所以，建议仅仅在自己已经完成了关卡的要求之后，再参考这些示例代码，与其互相取长补短，巩固所学。

再次强调，这不是一本只是用来“看”的书，这是一本练习的路线图，如

果学习过程中有任何困难，可以发邮件或者加入公众号：IT 匠艺教研室，进入公众号中的学习交流群询问，我非常愿意与学习者交流学习的问题，以帮助更多的读者闯过学习第一门语言的大山。

学习纪律

实施闯关的过程要遵守如下学习纪律：

- 按部就班不跳关。
- 自主学习不拷贝。
- 刻意练习不照抄。
- 新词新知要记牢。

以上学习纪律，前三条就不多说了，对于“新词新知要记牢”进行一下特别的阐述。学习 Python 的过程中会出现很多新的知识点和新的词汇，其中不少是英文的。这对于英文基础不好的学习者会是一个很大的障碍。作者在十几年的工作中也曾看到很多学习者都是因为英语这一关没有过，导致编程语言的学习进步缓慢。

翻开这本书时，每一位读者的英语水平已经是一个客观事实，并不会因为读到了本书而有什么改变，并且，即使学习完这本书，学习者的英语水平仍然不会有什么改变。作者这里希望针对编程语言的学习和英语水平之间的关系做一个阐述。

同样的学习条件，英语基础好的，学习编程语言的时候会更加容易一些。

新知识一般有两种方式被大脑接纳。第一种是挂接或者建立在已有的知识体系上；第二种就是在大脑里重新打桩，建立记忆根。这两种模式下，对于同样的知识点的掌握，所要付出的努力是完全不同的，而这种不同主要来自不同个体大脑中已有的知识体系的差异。

比如，对于已经有一定英语基础的同学，在进行 MySQL 的学习时就会很容易记住 create、delete、update、select 这些 SQL 的关键字以及他们的用途；而对于一个英语基础薄弱的同学，如果他之前并不认识这四个单词，甚至只认识 26 个字母（不认识 26 个字母的读者，本文帮不到你太多，建议不要再浪费自己的时间看下去了，这里没有任何偏见和轻视，仅仅是真诚的提醒！），要记住这四个关键字，将不是一件容易的事情。为什么会这样，其实很简单。对于熟悉这四个单词的人，由于已经拥有了对单词自身含义的理解，他看到这四个单词的时候看到的是四个整体，他所要记忆的只是四个元素，至于每一个元素

的拼写、意义区分已经是他已有的知识体系里的内容；而对于不认识这四个单词的人，他要记忆的是 24 个元素，并且每六个元素被作为了一组，拥有一个固定的含义和拼写顺序。虽然从知识点来说两个人都是要掌握 MySQL 的四个 SQL 关键字，知识点是相同的，但是对于两个不同的个体，这个知识点的信息量是完全不同的。

这就是为什么“同样的学习条件，英语基础好的，学习编程语言的时候会更加容易一些。”的原因。

英语基础差的学习者，也是可以学会编程语言的，只是需要付出更多的努力，并掌握一定的方法。

对于英语基础薄弱的学习者，我有如下两点建议：

（1）从意识上认识清楚，自己就是要比英语好的同学付出更多的努力才能学好编程语言。因为，这个已经是既定的客观事实，并不会因为我们无视它，不去面对它而消失。并且，现实中很多英语基础并不好的同学的确通过自己的努力学会了一门编程语言，我们也没有必要非要要求自己一定要学好英语之后，才去学习编程。

（2）从方法上区别对待，对于编程语言学习过程中遇到的自己不认识的英语单词，按照英语的学习方法进行学习和记忆。具体来说，对于英语基础薄弱的学习者，我建议准备一个英语单词本，遇到自己不认识的英语单词，查字典并进行整理、记忆和默写。

总之，编程是一门技艺，不同的学习者会有不同的学习道路，而“刻意练习”是掌握这门技艺的不二法门。我真心的希望可以通过这本小书，让更多人，特别是那些希望转型成为自动化测试工程师的软件测试从业者，克服对编程语言的恐惧，真正的掌握这门即有用又好用的编程语言。

在线学习与疑难解答

本书特别为广大读者精心准备了专有的在线服务与交流平台。用微信扫描右侧二维码，关注“悦读力”，即可加入本书读者圈，获得关于 Python 编程及自动化测试相关的更多学习资源，并有机会向阿奎老师提问、得到在线学习指导。

读 者 圈

感谢

写书之前，我真的没有想到写一本书要付出这么多的精力。真的要特别的感谢在这本书的写作过程中，帮助和支持我的人。

有太多的人需要感谢，那么就让我按照时间的先后一一谢过吧！

本书的编写完全是出于对我日常工作中的所听所见的思考，我的身边有很多非常喜欢学习新技术的同事，他们有的从事开发、有的从事测试，有的甚至是管理人员——我为什么要用“甚至”——？在这本书还是一个电子的试读版本的时候，我就组织了一个利用业余时间的学习小组（测试融合训练营），主要是希望看看这样一本书是否真的能够帮到大家，没想到一经提议，大家就非常积极地参与，非常感谢我的同事白雪、史丽珍、关晓康、毛雪涛、张培、刘会琴、于国双(ygs)、农倩倩、胡江海，他们是电子版初稿的第一批阅读用户，在这一过程中大家给了很多有益的反馈。

同步的还有一个来自互联网的学习小组（用 Python 做 HTTP 接口测试学习班），主要是觉得同事们给建议的时候都会比较“委婉”，希望有一个来自互联网的测试和反馈。感谢杨艳艳、高园园、CJoy、王芳、汤濮瑜、海底小鱼的热情参与和学习过程中的积极反馈。特别是高园园在参与之后，写了一篇非常值得称赞的学习感想。这篇文章，我在征得她本人同意之后，附在了本书的附录中供大家参考。

在电子版的书经过验证之后，我就有了将其变成纸质书的想法，机缘巧合的是，在参加一次敏捷技术交流的时候，遇到了张瑞喜老师，在张老师的鼓励下，我才真正开始了这本书的编写过程，编写过程中，非常感谢张老师对一个初次写书的作者的耐心和辅导，特别是这本书的题目《Python 编程基础与 HTTP 接口测试》就是张老师取的，本书的电子版在“百度阅读”上的名字叫做《用 Python 做 HTTP 接口测试》。

这本书的第一个审校的人是我的太太悠悠女士，感谢她指出我书中原来的 28 处没有原因的“所以”。其实，我知道更应该感谢的，是她背后默默的支持，长时间的写作，占用了很多本应该陪她的时间。

目　录

读 者 圈

第一部分　初识与初心

第二部分　认识 Python

第三部分　初识 HTTP

第四部分　实践 HTTP 接口测试

第一部分 初识与初心

第一章 Python 正流行

1.1 语言排行榜与技术雷达

Python 语言的流行是有目共睹的，从语言排行榜和技术雷达中就可以得到实际的认证。

TIOBE 编程语言社区排行榜

TIOBE 公司成立于 2000 年 10 月 1 日，由瑞士的 Synspace 公司和一些独立的投资人创建。TIOBE 是"The Importance Of Being Earnest"的缩写，该公司主要关注于软件质量的评估。TIOBE 程序设计语言指数是由该公司推出并进行维护的，这个指数将程序设计语言以排名列表的形式提供出来，并且每个月更新一次，用来表示程序设计语言的流行度。

——互动百科，http://www.baike.com/wiki/开发语言排名

TIOBE 编程语言社区排行榜是编程语言流行趋势的一个重要指标，每月都会更新，这份排行榜排名基于互联网上有经验的程序员、课程和第三方厂商的数量，是一个反映编程语言热门程度的编程语言流行趋势指标。如图 1.1 所示。

Mar 2017	Mar 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.384%	-4.14%
2	2		C	7.742%	-6.86%
3	3		C++	5.184%	-1.54%
4	4		C#	4.409%	+0.14%
5	5		Python	3.919%	-0.34%
6	7	^	Visual Basic .NET	3.174%	+0.61%
7	6	v	PHP	3.009%	+0.24%
8	8		JavaScript	2.667%	+0.33%
9	11	^	Delphi/Object Pascal	2.544%	+0.54%
10	14	^^	Swift	2.268%	+0.68%

图 1.1 2017 年 3 月的 TIOBE 排行榜

注释：Mar 2017：2017 年 3 月，Mar 2016：2016 年 3 月，Change：变化，Programming Language：编程语言，Ratings：占比，Change：变化

通过图 1.1（2017 年 3 月）的 TIOBE 排行榜我们可以看到，Python 在 2016 年 3 月至 2017 年 3 月间一直处在流行度排行榜第五的位置，前面只有 Java、C、C++三个老牌语言和微软官方推出的 C#语言。

而在 2017 年 6 月的 TIOBE 排行榜中，Python 已经超过 C#上升到第四位。如图 1.2 所示。

（以上数据来自 TIOBE 官方网站：https://tiobe.com/tiobe-index/）

Jun 2017	Jun 2016	Change	Programming Language	Ratings	Change
1	1		Java	14.493%	-6.30%
2	2		C	6.848%	-5.53%
3	3		C++	5.723%	-0.48%
4	4		Python	4.333%	+0.43%
5	5		C#	3.530%	-0.26%
6	9	^	Visual Basic .NET	3.111%	+0.76%
7	7		JavaScript	3.025%	+0.44%
8	6	v	PHP	2.774%	-0.45%
9	8	v	Perl	2.309%	-0.09%
10	12	^	Assembly language	2.252%	+0.13%

图 1.2 2017 年 6 月的 TIOBE 排行榜

注释：Jun 2017：2017 年 6 月，Jun 2016：2016 年 6 月，Change：变化，Programming Language：编程语言，Ratings：占比，Change：变化

编程语言受欢迎程度排行榜

PYPL 是 PopularitY of Programming Language 的缩写，即编程语言受欢迎程度。编程语言受欢迎程度排行榜 PYPL 是通过分析 Google 上搜索语言教程（tutorials）的频率创建的，基于的分析假设是：语言的教程被搜索的次数越多，则语言越受欢迎。分析的原始数据来自 Google 趋势（Google Trends）。如图 1.3 所示。

“如果您相信集体行为的智慧，编程语言受欢迎程度排行榜可以帮助您决定要学习的语言，或者在新的软件项目中使用哪种语言。”

（以上内容来自编程语言受欢迎程度网站：http://pypl.github.io/PYPL.html）

Worldwide, Jul 2017 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	22.7 %	-1.2 %
2		Python	16.1 %	+3.8 %
3		PHP	9.3 %	-0.9 %
4		C#	8.2 %	-0.6 %
5		Javascript	7.9 %	+0.5 %
6		C++	6.8 %	-0.3 %
7		C	6.6 %	-0.1 %
8		Objective-C	3.7 %	-1.0 %
9		R	3.6 %	+0.4 %
10		Swift	2.8 %	-0.2 %

图 1.3 2017 年 6 月的 PYPL 排行榜

注释：Rank：排名，Change：变化，Language：语言，Share：占比，Trend：趋势

通过图 1.3 中 2017 年 6 月的编程语言受欢迎程度排行榜我们可以看到，Python 语言排第二位，仅排在 Java 语言之后。

技术雷达

技术雷达是 ThoughtWorks 公司以雷达图的独特形式记录其技术顾问委员会关于对行业有重大影响的技术趋势的讨论结果。雷达图分为四个象限和四个环，四个象限分别是：技术、平台、工具以及语言和框架；四个环分别是：采用、试验、评估以及暂缓。

ThoughtWorks 公司在每年都会出品两期技术雷达，这是一份关于技术趋势的报告，相比一些我们能在市面上见到的其他各种技术行情和预测报告，它更加具体，更具可操作性，因为它不仅涉及到新技术大趋势，比如云平台和大数据，更有细致到类库和工具的推介和评论，从而更容易落地。

具体信息可以到 http://insights.thoughtworkers.org/tech-radar/查看。

在其 2017 年 3 月的技术雷达《Technology Rada Vol.16》中专门对 Python 的流行进行了说明。

“作为一门易用的通用编程语言，Python 在数学和科学编程领域具有坚实的基础。这使得它一直以来都为草根阶层的学术研究社区所采用。”

并且，对于 Python 社区中一直以来关于学习 Python2 还是 Python3 的争论，在本期的技术雷达中，也给出了基于实际使用经验的观点。

“根据我们在机器学习和 Web 应用开发这样的领域中使用 Python3 的经验显示，语言本身以及大多数支持库都已经成熟到可以采用的程度。”

“如果你在使用 Python 做开发，我们强烈鼓励你使用 Python3。”

在学习 Python2 还是 Python3 的困惑方面，作者一直是 Python3 的坚定拥护者。相对于 Python2，Python3 更加规范，更易于学习和使用，如果实际使用过程中遇到了需要依赖 Python2 的第三方库情况（现在很多第三方库都支持 Python3 了），完全可以自行进行一些修复和升级的工作。

本书后面会用到一个来自 Github 的定制博客系统，该系统就是原作者基于 Python2 开发的，我 fork 之后，进行了基于 Python3 的修复和升级，改动代码量实际上很少。在修复和升级之后，我向原作者提交了 pull requests，很快原作者就接受了。

所以，我想告诉大家的是：

Python 的流行是有目共睹的；

选择 **Python** 作为一门新的学习语言进行学习，绝对是明智的选择；

要学习 **Python**，请选择 **Python3**。

1.2 Python 之禅

在交互式解释器中输入 import this 就会显示 Tim Peters 的 The Zen of Python，即 Python 之禅。

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
```

```
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Python 之禅，体现了 Python 这门语言的设计哲学，其中的很多观点对于日常的编程也是很有指导意义的。

```
The Zen of Python, by Tim Peters
Python 之禅，蒂姆·彼得斯
Beautiful is better than ugly.
优美胜过丑陋。
Explicit is better than implicit.
显示胜过隐式。
Simple is better than complex.
简单胜过复杂。
Complex is better than complicated.
复杂胜过繁复。
Flat is better than nested.
串行胜过嵌套。
```

Sparse is better than dense.

稀疏胜过稠密。

Readability counts.

可读性很重要。

Special cases aren't special enough to break the rules. Although practicality beats purity.

虽然理想很丰满，现实很骨感，但是所谓特例并不足以打破上面的这些规则。

Errors should never pass silently. Unless explicitly silenced.

所有错误都不应该被直接忽略，除非能够被精确的捕获之后。（其中一个典型的例子就是，不建议用 Exception:pass 来直接忽略所有异常。）

In the face of ambiguity, refuse the temptation to guess. There should be one-- and preferably only one --obvious way to do it. Although that way may not be obvious at first unless you're Dutch.

当面对不明确的情况时，要拒绝去猜测的诱惑。应该有一种，最好是唯一一种，显而易见的解决方案。尽管起初，那种解决方案可能并不是那么显而易见，因为你不是 Python 之父（这里的 Dutch 是指 Python 之父 Guido Van Rossum，他是荷兰人。）

Now is better than never. Although never is often better than *right* now.

现在行动胜过永不开始。尽管，永不开始经常好过冲动的开始。

If the implementation is hard to explain, it's a bad idea. If the implementation is easy to explain, it may be a good idea.

如果你的实现难于向别人解释，这往往不是个好主意。如果你的实现很容易向别人解释，这可能是个好主意。

Namespaces are one honking great idea -- let's do more of those!

命名空间是一个令人激动的伟大想法，让我们将它发扬光大。

以上 Python 之禅的翻译参考了网络上的翻译资料，有一定修改，具体链接参见附录参考资料。

1.3 无所不能的 Python

Python 语言的流行得益于其应用的广泛性。作为一门在各大语言排行榜中稳居前五的语言，Python 在当前主要的计算机应用相关领域都有广泛的应用。

在系统管理方面，有 Fabric，Ansible，Saltstack 等流行的配置管理工具；在系统开发工具方面有大名鼎鼎的版本管理工具 Mercurial；在 Web 开发方面有 Django, Flask 等 Web 框架，著名的 Youtube、Dropbox，以及国内的 Douban 都是基于 Python 构建的。

在最近大热的人工智能、大数据、云计算方面，Python 更是应用广泛，代表性的有深度学习框架 TensorFlow*，数据处理方面的 Pandas，特别是已经基本上成为私有云搭建不二选择的 OpenStack，这些框架和软件应用都是使用 Python 语言开发的。

在科学计算领域，一直以来都是 Matlab、Pascal 和 Fortran 等语言的天下，现在，Python 语言在科学计算领域的应用也非常广泛，用于数学计算的基础库 SciPy、NumPy，以及机器学习方面的 Scikit-learn 都是很好的代表。

Python 在影视制作方面也有广泛的应用，最著名的 3D 内容创建应用程序 Maya，就支持通过 Python 样式的脚本进行动画编程。

在软件测试领域，著名的功能自动化测试框架 RobotFramework 也是由 Nokia Siemens Networks 的开发者用 Python 编写的。

读 者 圈

第二章 接口测试的崛起

2.1 接口测试简介

在测试领域，一直存在很多测试的概念，如单元测试、集成测试、功能测试、性能测试、安全测试，等等，接口测试就是这众多的测试概念中比较重要的一个。

这里首先要介绍一下**接口**的概念。

接口，两个不同系统（或子程序）交接并通过它彼此作用的部分。

——《新华字典》

在计算机中，接口是计算机系统中两个独立的部件进行信息交换的共享边界。而接口测试，英文为 Interface Testing，是针对系统间或者系统的组件间的接口的一种测试，意在测试特定接口在给定输入下的行为与预期行为之间的符合性。

要对接口进行测试，首先要了解接口的协议和接口的定义。

接口协议（Interface protocol）指的是需要通过接口进行信息交换的通信双方之间需要遵从的通信方式和要求。

协议是个很复杂的概念，这里就不展开了，随着互联网的发展，因系统之间、系统的组件之间集成的需要，出现了很多类型的接口通信协议。其中最常用的是 HTTP 协议，因此基于 HTTP 协议的接口测试也逐渐成为接口测试的主要应用场景。

HTTP（HyperText Transfer Protocol），通常被翻译成超文本传输协议，HTTP 本身已经包含了协议（Protocol）的意思，只是在中文里大家习惯上会将 HTTP 所代表的协议也称为 HTTP 协议。关于 HTTP 协议的详细内容，在本书第三部分的第七章会有详细的阐述。

接口定义是对接口的功能，调用的前提条件，调用的方法，以及接口返回内容的描述，是接口测试案例编写的基础。

本书针对 HTTP 协议接口的定义给出了接口描述的“八个问题”，称之为“接口八问”。

这八个问题分别是：

- 接口的请求地址是什么？
- 接口的功能描述是什么？
- 请求接口是 GET 还是 POST？
- 接口需要在登录情况下才有用吗？
- 接口有上送数据吗？上送的数据是什么？
- 接口返回的状态码是多少？
- 接口返回报文体的格式和编码是什么？
- 接口返回的内容是什么？

针对一个给定的 HTTP 接口，读者要回答以上的八个问题并非易事，需要具备一定的技能。同样，在此基础上，能够利用 Python 完成针对这个给定的 HTTP 接口的自动化测试案例的编写，就更需要掌握包括 Python 语言、HTTP 协议以及 Requests 库等一系列知识和技能。通过对本书后续章节的学习，读者会一一掌握上述的知识和技能。

2.2 Ajax 接口与 Web 动静分离

当前，越来越多的应用产品都是采用 B/S 结构设计的。所谓 B/S 结构，简单来说，就是每个使用者都通过浏览器（如：Chrome、Firefox、Safari 和 IE 等）来访问和使用应用产品提供的服务，不需要用户安装单独的客户端。与 B/S 相对应的是 C/S 结构，C/S 结构要求应用产品有单独的客户端供用户使用，比如 PC 机上使用的 QQ 就是一个典型的 C/S 结构的应用产品，每个用户都要安装 QQ 客户端，然后，才能使用 QQ 后台服务提供的通信和交互功能。

B/S 结构的应用架构经历了长时间的发展，从最开始的 CGI 后台编码方式，到后来的 ASP、JSP 等混合编码方式，以及当前流行的动静分离 Web 架构，B/S 结构的应用产品架构随着互联网应用的发展也在逐步的发展和演变。

当前大多数新建的 B/S 结构的应用产品，如产品和服务类网站等，都会采用动静分离的 Web 架构，有的也叫前后台分离。所谓动静分离，是指动态内容的实现逻辑和静态内容的展示逻辑的分离。具体来说，就是实现前端界面展

示的 HTML、CSS 和 JavaScript 代码与实现后台业务逻辑的后台代码（如，Java 等）的分离和独立部署。

要实施一个动静分离的 Web 架构产品，要涉及到前后台两部分独立部署的代码在运行过程中的交互，当前用的比较多的前后台交互方式是采用 Ajax 接口方式。

Ajax 是 Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）的简称，简单来说，Ajax 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术，通过 Ajax 可以创造出动态性极强的 web 界面。

Ajax 是多种技术的联合使用，其中主要包括两种技术：

（1） 通过浏览器中的 XMLHttpRequest 对象，驱动浏览器和 Web 后台服务器实现异步的数据交换。

（2） 通过 JavaScript 的 DOM 操作功能，实现浏览器中已经加载的网页的动态更新，而不需要重新加载整个页面。

我们通常说的 Ajax 接口一般指的是通过 XMLHttpRequest 对象实现的驱动浏览器和 Web 后台服务器实现异步的数据交换接口，而该接口实际上就是一个普通的 HTTP 协议接口，所以，通过 XMLHttpRequest 对象发送的 HTTP 请求，与通过浏览器地址输入一个 URL 地址后，发送到 Web 服务端的 HTTP 请求，并没有什么本质的区别。

2.3 Restful 接口

REST 是一种互联网软件架构原则，即 Representational State Transfer 的缩写，由 Roy Thomas Fielding 在其 2000 年的一篇论文（Architectural Styles and the Design of Network-based Software Architectures）中提出。

The Representational State Transfer (REST) style is an abstraction of the architectural elements within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements. It encompasses the fundamental constraints upon components, connectors, and data that define the basis of the Web architecture, and thus the essence of its behavior as a network-based application.

表征状态转移（REST）风格是分布式超媒体系统中的架构元素的抽象。REST 忽略了组件实现和协议语法的细节，以便聚焦于组件的角色，组件间交互的约束以及对重要数据元素的解释。组件、连接器和数据是定义 Web 架构的基础，REST 涵盖了对这三要素的基本限制，从而，也涵盖了三要素作为基于网络的应用程序的行为本质。

以上内容引用、翻译自论文“5.2 REST Architectural Elements”小节，地址为：https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm。

通俗地讲，REST 风格的架构就是一种将要操作的业务数据作为资源，分配一个固定的 URL 地址，然后，通过 HTTP 的四个请求方式：POST、DELETE、PUT、GET，分别对应业务数据的增、删、改、查四种操作。

比如，我们有一个客户信息，给定一个 URL 为 http://www.example.com/customer，那么，可以通过对这个 URL 分别发送四个请求方式对应四个针对某一条业务数据的操作：

- POST 请求 /customer 创建一个新的客户。
- DELETE 请求 /customer/111 删除 id 为 111 的客户。
- PUT 请求 /customer/111 更新 id 为 111 的客户的信息。
- GET 请求 /customer/111 查询 id 为 111 的客户的信息。

虽然从原理上来说，REST 架构风格是无关乎通信协议的，但是在实际使用过程中，绝大部分 REST 架构组件之间的通信接口，也就是所谓的 Restful 接口，都是采用的 HTTP 作为其通信接口协议的。所以，一个 Restful 接口的请求，实际上很多情况下与一个普通的 HTTP 请求并无本质的区别。

通过对 Ajax 和 Restful 的简单了解，我们可以看到，随着互联网的发展，前后台部件之间以及网络应用组件之间的新的结构风格和通信机制层出不穷，而 HTTP 协议在这些新的架构风格和通信机制中扮演了非常重要的基础通信协议的角色。

对于一名测试人员来说，掌握接口测试，掌握 HTTP 协议，进而掌握 HTTP 协议下的接口测试已经变得日益重要起来。

读 者 圈

第三章 测试工程师的自动化测试转型

3.1 “点点点”测试工程师的困惑

很多从事手工测试的软件测试工程师都有一个困惑，到底应不应该学习自动化测试，手工测试会不会被淘汰？

针对这个问题，我在一个论坛上曾经有过一个非常简短的问答！

问题：手工测试真的要被淘汰了吗？

回答：不会，淘汰的不是方法，是人。仅仅会手工测试的人会比较危险。

当前有一种论调，即所有的手工测试都应该被自动化所取代，手工测试俨然成了落后的代名词。这是一种非常极端的认知。手工测试作为一种测试方法，是具有其特殊的优势应用场景的，并不是所有的测试都可以被自动化，也并不是所有的测试自动化后都会带来效益和效率的提升。

作为软件开发和交付的企业来说，进行自动化测试方面的建设和提升，最看重的是产生的效果，也就是在自动化测试上的投入要能够产出更大的收益。一味地将所有的手工测试都变成自动化测试，显然是盲目的。

作为一名软件测试工程师，在思考自己职业发展的时候，关注的不应该是方法，而应该是人。软件测试工程师作为产品交付团队的一员，其重要的职责是与团队一起高效地保证产品的交付质量。当前，自动化测试技术爆发式的流行是毋庸置疑的，其深层次的原因在于，自动化测试的确可以为开发团队的产品质量提升带来切实的帮助。那么，作为软件测试工程师，作为产品交付团队的一员，为了更大地发挥自身的价值，掌握自动化测试的相关技能，应该成为其提升自身职业素养和追求职业发展的必然选择。

综上，我要强调的是，作为一名软件测试工程师，如果希望在软件测试领域有所发展，不必纠结于手工测试会不会被淘汰，应该毫不犹豫地去学习自动化测试的相关技能，提升自身的职业素养。

3.2 自动化测试到底要学什么

很多从事了很长一段时间手工测试的从业者提起要学习自动化测试都感觉比较茫然，感觉有太多的知识、工具和技能要学习，的确要完全掌握自动化测试技能有很多的内容要学习，不过对于初学者来说，最基本和基础的内容逃不出下面的四加二：

- “四”是：计算机基础、计算机网络、一个操作系统（Linux）、一个数据库（MySQL）。
- “二”是：英语基础和一门编程语言（Python）。

计算机基础

计算机基础是一名软件测试工程师的基本功，其中包含了对计算机的最基本的认识和理解。这方面并不需要太多的描述，如果读者希望比较系统地学习计算机基础知识，或者检验一下自己是否真正掌握了计算机基础的相关内容，建议读者去找一份《一级计算机基础及 MS Office 应用考试》的真题，自己限定时间做一下，基本上就能知道自己的水平了。

计算机网络

计算机网络方面的知识非常繁杂，并非必须掌握所有的内容。但是，对于其中的网络基本知识如 IP、端口、域名、网络协议、网关、代理、局域网和广域网等基本概念还是需要理解的，要达到能够区分清楚这些基本概念，并用自己的话表达出这些基本概念的含义，在实际上网的过程中找到对应的实际场景。

操作系统——Linux

软件测试需要熟悉一个操作系统，为什么推荐 Linux，是因为现在很多互联网公司都是使用 Linux 部署产品。测试工程师会一点 Linux 就可以自己查看日志、甚至自己部署，绝对是大大的加分项。对 Linux 的学习只要熟悉 Linux 系统 Shell 的基本操作即可，比如创建、拷贝、删除文件和目录、查看文本文件、运行程序等。

在此，我推荐一个资料叫做《Linux 一页通》，读者可以通过微信，搜索并关注公众号：IT 匠艺教研室，回复“一页通”，你就会得到一个高清版的 Linux 命令脑图，将上面的命令一一了解并练习一下，基本上你的 Linux 就掌握的差不多了。

数据库——MySQL

软件测试工程师要熟悉一个数据库，建议学习一下 MySQL。计算机一级里面会让用 Access，这个用于入门可以，但是作为测试工程师，熟悉这个目前在互联网公司中普遍采用的开源数据库，对你后续的测试工作将大有裨益。

MySQL 数据库包含的内容也很多，作为测试工程师只要能够通过终端进行表的增、删、改、查就足够了，有余力可以再看看建库建表，至于安装、权限管理、备份运维等等，测试工程师一般用不上。

对于知识点的学习，我们应该有一个开放好学的心态，如果工作中觉得自己掌握某一项技能对于自己和团队的工作会有帮助，就应该尝试去学习。**艺不压身**的古语，对于所有从事技术工作的人士是通用的。

编程语言——Python

虽然业界有 LAMP 的说法，这里的 L 是指 Linux，A 是指 Apache，M 是指 MySQL，这几个前面都提到了，P 则指的是 PHP 语言，我这里还是要推荐 Python。Python 和 PHP 一样是一门脚本语言，但是，Python 对测试工程师来说更加实用，很多测试框架都是用 Python 编写的，在实际工作中也更有可能用到。

英语基础

无需质疑，英语是世界上最广泛使用的语言之一，几乎所有软件开发语言都以英文单词为载体。那么作为软件测试人员，到底需不需要掌握英语呢？答案是肯定的。任何时候都不能放弃学习英语，甚至应该花更多的时间和精力去学好英语，以便在 IT 行业有更好发展前景。

掌握英语，你可以翻阅 IT 领域内先进的原文文献，获取更多的知识、经验。本书中引用了一些英文资料，我都将原文保留了下来，英文的下面是我翻译的中文译文。希望读者在阅读的过程中能够互相参照，在 Python 编程基础

和 HTTP 接口测试技能提升的同时，也在英语学习上有所收益。

3.3 摆脱“点点点”从哪里开始

上一小节说的“四加二”是测试工程师的基本功，作为一名“点点点”工程师，希望进行自动化测试的学习应该从哪里开始呢？

2017 年初，针对整理的十几项自动化测试需要的技能，我做了一个调查问卷如下表所示：

领　域	主　题	子　项
理论知识	自动化测试理论	自动化测试的意义与局限
		测试金字塔
		测试四象限
	网络基础	基本概念：网络协议、IP 地址、端口、子网、网关、代理
		HTTP/HTTPS、Restful、FTP
		XML、JSON、Ajax
通用操作技能	Linux 操作系统	命令行操作
		VI 使用
	MySQL 数据库	SQL 操作
		通过客户端查看和维护数据
	正则表达式	
专用测试的工具	Selenium、WebDriver	
	Firefox、Chrome 浏览器查看看 Web 网络报文	
	Robotframework	
	Cucumber	
	Appium	
	QTP	
	Postman	
	Watri	
语言	Python	
	Javascript	
	Html/CSS	
	Shell	
	Ruby	
	VBScript	

请参与调查的测试工程师回答如下问题：

“您认为从手工测试到自动化测试最迫切需要学习的是以下哪三项？”

得到的结果如下表所示。

内　容	获得投票数占比
Python	23%
自动化测试理论	15%
网络协议	11%
RobotFramework/Selenium	9%
LoadRunner 和 Html/CSS	各自占比 8%

从这个调查结果可以看出，Python、自动化测试理论、网络协议获得的投票占比合计超过 49%。针对这个结果我又进行了一些调研后认为，**学习和掌握用 Python 进行 HTTP 接口自动化测试**，可以作为从手工测试工程师到自动化测试工程师转型的切入点。

以“学习和掌握用 Python 进行 HTTP 接口自动化测试”作为转型切入点，有如下的优点：

（1） 可以掌握 Python 编程语言，为后续从事更高级的测试开发方面的工作打下坚实的基础。

（2） 可以了解 HTTP 协议，HTTP 协议的流行和被广泛应用，已经不需要再论证。

（3） 为日后学习自动化测试框架打下坚实的基础。掌握直接使用 Python 编程语言进行自动化测试，对于今后使用其他自动化测试框架时，深入理解自动化测试框架的运行机制，快速掌握自动化测试框架的使用会有很大的帮助。

读 者 圈

第二部分 认识 Python

第四章 我来了

4.1 第一声问候

学习目标

安装 Python 语言环境和编辑器集成环境，编写第一个 Python 程序，并运行成功。

学习资源

http://www.w3cschool.cn/python3/python3-tutorial.html
http://www.runoob.com/python/python-install.html
http://www.runoob.com/python/python-ide.html
http://www.runoob.com/python3/python3-basic-syntax.html

知识准备

这里我要向大家推荐一本 Python 的入门书籍《简明 Python 教程》（《A Byte of Python》）。这是一本由 Swaroop C H 所创作，采用 **知识共享 署名-相同方式共享 国际 4.0 协议（CC BY-SA Intl. 4.0）**进行授权的自由图书。该书写成后，先后成为哈佛大学、加利福尼亚大学戴维斯分校等多所大学的 Python 编程课程参考教材。最新中文版由漠伦翻译和维护。

https://bop.molun.net

这本书非常适合初学者作为 Python 入门的起步教材。就像书中前言所说的：如果你对电脑知识的了解仅限于如何保存文本文件的话，那这本书就是为你准备的。

在进行 Python 基础部分的练习过程中，作者推荐《简明 Python 教程》，

作为该部分的主要参考资料。

请完成《简明 Python 教程》中“安装”和“第一步”两个小节的学习。

挑战问题

安装 Python3。

安装 PyCharm 社区版，注意，是社区版，Free Community。

编写一个 Python 程序，helloworld.py，运行程序可以打印出“Hello World！”如图 4.1 所示。

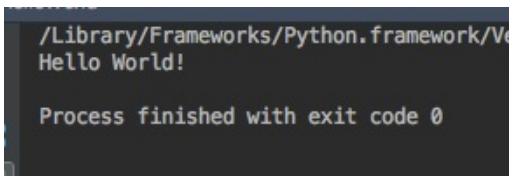


图 4.1 Hello world!

注意：完成本“挑战问题”，请在下载安装包并完成安装后开始计时，十分钟内将问题解决。切记，解决“挑战问题”过程中要闭卷完成。

如果读者之前完全没有接触过 Python，请先阅读下面的知识点，再进行“挑战问题”的解决，有疑问可以到上面推荐的两个网站中进行学习。

难点提示

请第一次接触 Python，并且没有其他编程语言基础的读者认真阅读以下内容。

Windows 用户，请根据操作系统的实际情况选择 32/64 位 executable installer 的安装包进行下载。如图 4.2 所示。

MacOS 用户，安装 Python3 之后，切记不要卸载本机自带的 Python，否则系统的其他功能可能会受到影响。同时，在 PyCharm 的安装过程中，会出现**选择工作**

Version	Operating System
Gzipped source tarball	Source release
XZ compressed source tarball	Source release
Mac OS X 64-bit/32-bit installer	Mac OS X
Windows help file	Windows
Windows x86-64 embeddable zip file	Windows
Windows x86-64 executable installer	Windows
Windows x86-64 web-based installer	Windows
Windows x86 embeddable zip file	Windows
Windows x86 executable installer	Windows
Windows x86 web-based installer	Windows

图 4.2 Python Package

区对话框，如下图所示，如果读者从未安装过 PyCharm，请按照图中所示进行选择即可，PyCharm 会为读者新建一个工作区。如图 4.3 所示。

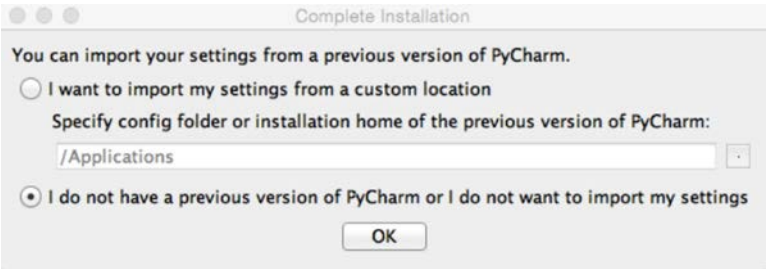


图 4.3 Complete Installation

知识点

Python3

Python 是一种脚本语言，所谓脚本语言就是源代码可以直接运行的语言，如 Javascript、Shell、VBScript、Ruby 等都是脚本语言。而像 C 语言，在编写完源文件后，需要经过编译将人可以阅读的源代码文件转成机器语言文件（即通常所说的可执行文件，比如经常说的 Windows 下扩展名为 exe 的文件），然后才能够直接在计算机上运行，这种类型的语言被称为编译性语言，如 Pascal、C++等都是编译性语言。还有一种情况如 Java 语言，是一种解释性语言，其源文件经过编译后生成的文件不能直接在计算机上运行，需要通过 Java 虚拟机进行加载和运行。按照编译、运行的方式不同，计算机语言可以分为脚本语言、解释性语言、编译性语言。

Python 官网地址：https://www.python.org/。

建议下载最新文档版本 3.6.0。

PyCharm

PyCharm 是由 JetBrains 打造的一款 Python IDE，包含一整套可以提高开发者工作效率的工具，比如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制。此外，该 IDE 提供了一些高级功能，用于支持 Django 框架下的专业 Web 开发。

PyCharm 是一个很好的 Python 集成开发工具，作者强烈推荐，后面的课程都会以此作为开发工具。

PyCharm 有免费的社区版，完全够用了。

地址如下：http://www.jetbrains.com/pycharm/

建议下载最新版本。

语言基础知识要点：

（1） Python 的源代码文件一般用.py 作为扩展名。

（2） Python 中每一行最后的分号不是必须的，可以省略，主要通过换行来识别语句的结束。

（3） Python 对大小写敏感，对缩进也敏感。

（4） print()函数在 Python 中是用来进行文字输出的。

4.2 小青，你几岁了？

学习目标

学习通过字符终端进行简单地输入或者输出，以及字符串拼接。本小节是后续学习和完成挑战问题的基础。

学习资源

http://www.runoob.com/python3/python3-inputoutput.html

http://www.runoob.com/python3/python3-basic-syntax.html

http://www.runoob.com/python3/python3-string.html

http://www.runoob.com/python3/python3-data-type.html

知识准备

完成《简明 Python 教程》中“基础”小节的学习，继续阅读下面知识准备中的内容后，再进行挑战问题的解答。

Python 交互模式界面

Python 的交互模式是一个命令行界面，也叫文本交互界面。安装后可以通过菜单打开。如图 4.4 所示。

图 4.4 Python Interactive!

或者通过 DOS 命令行界面输入：

```
python
```

会有“>>>”作为提示符的字符界面出现，如图 4.5 所示。

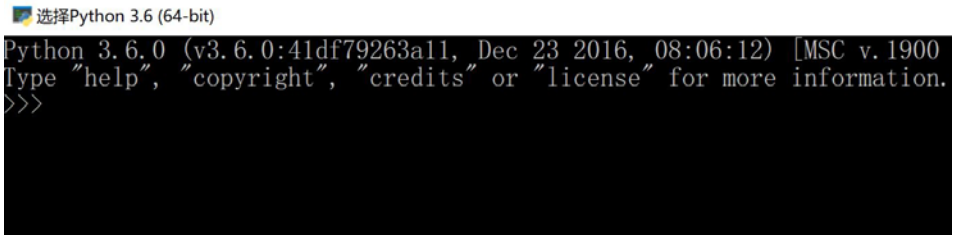


图 4.5 Python Interactive Terminal

在 Python 的交互模式中输入 Python 语句，回车使之返回一个结果。比如，在交互界面打印 Hello World！可以这样做：

```
>>> print("Hello world!!")
Hello world!!
>>>
```

注意：DOS 命令行界面和 Python 的交互界面是有区别的，主要的辨别方法就是看是否有“>>>”作为提示符，有“>>>”作为提示符的是 Python 交互界面，只能接收和识别 Python 语句。可以通过输入 quit()命令退出，或者使用 Ctrl+Z 快捷键退出。没有“>>>”的是 DOS 界面，可以运行 DOS 命令。特别提醒：pip install 命令只能在 DOS 命令行界面运行，在 Python 交互界面是不可以运行的。进入 DOS 界面的方法，Window 系统可以通过 Win 键+R，会出

现一个输入框，输入 cmd 后回车即可。

变量

Python 中可以定义一个**变量**用于保存数字、字符串或者其他数据信息。变量就是一个标识符，也可以理解为变量就是一个容器的名字，容器可以存放数字、字符串或者其他数据信息，而**变量的名字必须是数字、字母和下画线的组合，并且只能以字母或者下画线开头**。

```
>>> str = 'hello'
>>> print(str)
hello
>>> print('hello')
hello
```

上面的语句就定义了一个变量，变量的名字为 str，并且将字符串‘hello’赋值给了变量 str，等号(=)用于给变量赋值。然后，这个变量就代表字符串‘hello’，print(str)就会得到和 print('hello')一样的结果。

从终端读入字符串

通过上一个小节的学习，读者已经了解了 print()函数的简单用法，这一小节我们练习从终端读入字符串。在 Python 中提供了一个 input()内置函数用于从标准输入终端（默认情况下也就是键盘）读入一行文本：

```
result  = input(argument)
```

input 函数可以接受一个字符串参数(argument)用于作为输入的提示信息，而函数的返回值(result)就是从终端输入的字符串。

下面我们打开 Python 的交互模式，读取一个字符串并打印出来。

```
>>>
>>> str = input("请输入一个字符串：")
请输入一个字符串：闯关式学习，需要刻意练习。
>>> print(str)
闯关式学习，需要刻意练习。
>>>
```

从终端读入字符串就是这么简单。

字符串及其简单操作

字符串是 Python 的六大标准数据类型之一。字符串是用单引号(')或者双引号(")括起来的一串可见的字符的序列。对于不可见的字符，即一些特殊字符，一般用反斜杠(\)转义。

比如，回车符为"\n"，TAB 符为"\t"等。

```
>>> string = 'hello\tworld!!'
>>> print(string)
hello   world!!
>>>
```

上面定义了一个字符串赋值给了 string 变量，hello 和 world 之间有一个 TAB 符。

字符串可以通过加号（+）连接成为一个新的字符串，通过星号（*）可以复制字符串为多份并连接成为一个新的字符串。

```
>>> string='hello' + 'world'
>>> print(string)
helloworld
>>> string='hello'*3
>>> print(string)
hellohellohello
>>>
```

小练习：

请读者回答 Python 的六大标准数据类型都有哪些？

挑战问题

编写一个 Python 程序，howold.py，运行程序，程序会首先询问“你叫什么名字？”，得到答案后，会继续询问“【名字】你几岁了？”，得到年龄后会打印出“【名字】【年龄】。”

如图 4.6 所示。

```
C:\Python\Python36\python.exe C:/MyWork/myGitHub/
你叫什么名字？小青
小青你几岁了？1000
小青1000岁了。

Process finished with exit code 0
```

图 4.6 How old are you

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后重做一次。

难点提示

语言基础知识要点

（1） input 在 Python 中是用来接收终端文字输入的。特别提醒：Python3 中 input 和 Python2 中的是不同的，有兴趣的可以自行学习一下。

（2） 变量存储在内存中的值。这就意味着在创建变量时会在内存中开辟一个空间。

（3） Python 中的变量赋值不需要类型声明。

（4） 等号（=）用来给变量赋值。

（5） Python 有六个标准的数据类型

- Numbers（数字）。
- String（字符串）。
- List（列表）。
- Tuple（元组）。
- Set（集合）。
- Dictionary（字典）。

（6） 字符串拼接用“+”号。

拓展

用两种方法打印出最后的那句话“【名字】【年龄】。”

提示阅读，print()函数的格式化输出。

4.3 我会做加法

学习目标

学习算术运算符、赋值运算符和数字操作。

学习资源

http://www.runoob.com/python3/python3-basic-syntax.html
http://www.runoob.com/python3/python3-basic-operators.html
https://www.w3cschool.cn/python3/python3-basic-operators.html
http://www.runoob.com/python3/python3-number.html

知识准备

关于运算符的解释，《简明 Python 教程》一书中“运算符与表达式”小节讲解的并不是特别清晰，建议参考初级教程：（http://www.runoob.com/python3/python3-basic-operators.html）或者 w3cschool（https://www.w3cschool.cn/python3/python3-basic-operators.html）的 Python3 教程中的“运算符”小节。

为了完成本小节的挑战问题，主要学习其“运算符”小节中的**算术运算符**和**赋值运算符**就可以了。

所谓算术运算符就是：加(+)、减(−)、乘(*)、除(/)、模(%)、幂(**)六个运算符，在 Python 里面还增加了一个整除(//)。

赋值运算符，就是在以上七个运算符的基础上，增加了直接赋值(=)的运算符，共有八个：+=、−=、*=、/=、%=、**=、//=、=。

挑战问题

编写一个 Python 程序，computer.py，运行程序，提示“请输入第一个数字：”，输入第一个数字并回车后，提示“请输入第二个数字：”，输入第二个数字后，打印出“两个数字的和为：[两个数字之和]”。

如图 4.7 所示。

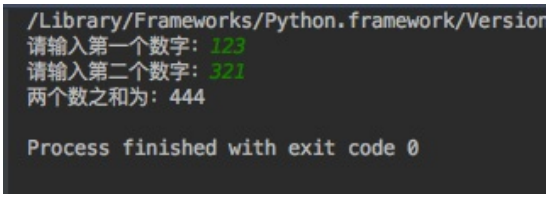


图 4.7 Computer

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后重做一次。

难点提示

input()函数的返回值默认是字符串，不能直接用于计算。

当遇到下面的错误提示信息，请仔细分辨报错的原因：

ValueError: invalid literal for int() with base 10: 'qqww'

说明你在将包含非数字的字符串’qqww’转换成整数，所以运行报错了。

TypeError: unsupported operand type(s) for +: 'int' and 'str'

说明你在将数字型的变量和字符串相加，出现类型错误，所以运行报错了。

TypeError: not all arguments converted during string formatting

在进行字符串格式化的时候，经常是打印的时候，有部分传入的参数没有被格式化，如果是 print 的时候出现，一般是 print()中的占位符的个数与输入的参数不匹配。

如下面这个语句：

```
>>> print("两个数之和为：%s"  % (1,2))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: not all arguments converted during string formatting
>>>
```

在 print()语句中，字符串"两个数之和为：%s"中只有一个占位符(%s)，而后面却试图将一个包含两个元素的元组作为参数传递给 print()函数，所以出现了 TypeError 的错误。

知识点

语言基础知识要点：

（1） Python 有七个算术运算符，要记住它们。

（2） 这七个算术运算符和“=”又组成七个赋值运算符，加上“=”本身，组成八个赋值运算符，要记住他们。

（3） Python 支持四种不同的数值类型：整型（int），浮点数（float），长整型（long），复数（complex），前两种是必须会用的。

（4） input 得到的返回值默认是字符串。

（5） int()函数是将括号里的值转换成整型，str()函数是将括号里的值转换成字符串。

（6） Python 支持四种不同的数字类型：

- int（有符号整型）。
- long（长整型[也可以代表八进制和十六进制]）。
- float（浮点型）。
- complex（复数）。

拓展

请读者在上面的挑战问题完成后，尝试在输入第一个数字的时候，输入“abc”，看看会发生什么？

4.4 这是奇数还是偶数？

学习目标

学习比较运算符和 if……else……语句。

学习资源

http://www.runoob.com/python3/python3-basic-operators.html
http://www.runoob.com/python3/python3-conditional-statements.html
http://www.runoob.com/python3/python3-number.html

知识准备

关于 if……else……语句，请仔细阅读《简明 Python 教程》中“控制流”小节中“if 语句”部分，其中猜数字的游戏很好玩，后续小节里，我们会尝试将它改造一下，增加其趣味性。

建议参考教程：（http://www.runoob.com/python3/python3-basic-operators.html）或者 w3cschool（https://www.w3cschool.cn/python3/python3-basic-operators.html）的 Python3 教程中的“运算符”小节。

为了完成本小节的挑战问题，主要学习其“运算符”小节中的**比较运算符**，熟悉六个比较运算符：其中包括：

等于、不等、大于、小于、大于等于、小于等于。

挑战问题

编写一个 Python 程序，oddoreven.py，运行程序，提示“请输入一个数字：”，输入数字并回车后，如果这个数字是偶数就打印出“您输入的数字为：偶数”，如果这个数字是奇数就打印出“您输入的数字为：奇数”如图 4.8 所示。

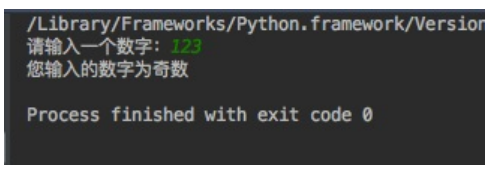


图 4.8 Odd or Even

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后重做一次。

知识点

语言基础知识要点：

（1） Python 有七个比较运算符。

（2） if……else……语句的冒号别忘了。

（3） 特别提醒：Python 对缩进敏感，Python 的代码块不使用大括号（{}）而是用缩进来界定代码块。相同的代码块必须包含相同的缩进空白数量，这个必须严格执行，否则，会得到如下错误提示：IndentationError: unexpected indent。

拓展

尝试输入不是数字的 abc 试试程序的反应，并想一下应该如何解决这个问题。

4.5 我们三个谁最大

学习目标

学习逻辑运算符和 if……elif……else……语句。

学习资源

http://www.runoob.com/python3/python3-basic-operators.html
http://www.runoob.com/python3/python3-conditional-statements.html
http://www.runoob.com/python3/python3-number.html

知识准备

逻辑运算符

建议参考教程（http://www.runoob.com/python3/python3-basic-operators.html）或者 w3cschool（https://www.w3cschool.cn/python3/python3-basic-operators.html）的 Python3 教程中的“运算符”小节。

为了完成本小节的挑战问题，主要学习其“运算符”小节中的**逻辑运算符**。

逻辑运算符只有三个：与（and）、或（or）、非（not），但是，这三个运算符和小括号（（））结合起来，变化莫测，需要细细体会。

较好的应用逻辑运算符需要对逻辑代数，也就是布尔代数有基本的了解和掌握。

尝试回答一下，a=91，b=33，c=35，d=190 的情况下，如下的表达式是真还是假？

((a==90) or (a==95)) and (b>=30 and b <=33) and ((c==31) or (c>=34)) and ((d==195) or (d==190))

可以自己写个小程序测试一下自己的答案是否正确。

如果读者可以较好地回答上面的问题，就可以继续进行后续的学习或者进行问题挑战了，如果对上面的问题回答错误，或者没有什么把握，建议读者加强一下**逻辑代数基础知识**的复习。

逻辑运算对于软件开发或者测试开发工作来说，是非常重要的一个基本技能，读者务必加以重视。可以通过阅读附录中的“逻辑运算基础”小节，并完成对应的练习题加以提高。

字符串操作

关于字符串的操作，我推荐菜鸟教程（http://www.runoob.com/python3/python3-string.html）“字符串”小节的内容。

Python 不支持单独的字符类型，而字符串可以看作字符组成的数组或者叫做列表，支持通过下标的方式直接访问字符串中的子串或者字符，甚至支持加法和乘法。

```
>>> str = "Hello world!"
>>> print(str)
Hello world!
>>> print(str[6:11])
world
>>> print(str[6])
w
>>> print(str + str)
Hello world!Hello world!
>>> print(str * 3)
Hello world!Hello world!Hello world!
```

这里说一个比较有意思的用法，字符串翻转，代码如下，有兴趣的可以去查一下关于字符串下标的更高级的用法。

```
>>> print(str[::-1])
!dlrow olleH
>>>
```

菜鸟教程“字符串”最大的好处是整理了关于字符串的 40 个内建函数，这里只对几个常用的演示如下，读者有兴趣可以进一步研究，不过一般不需要一一记住，重要的是知道有这样的一个功能，用到的时候知道到哪里查用法就可以了。

```
>>> str = "Hello world!"
>>> print(str)
Hello world!
#isdigit()函数，用于判断字符串是否是数字。
>>> print(str.isdigit())
False
#split()函数用于将字符串分割为字符串列表，默认以空格为分隔符。
>>> print(str.split())
['Hello', 'world!']
>>> str1,str2=str.split()
>>> print(str1)
Hello
>>> print(str2)
world!
>>>
```

挑战问题

编写一个 Python 程序，getmaxnum.py，运行程序，提示“请输入三个数字，数字之间用空格隔开：”，输入三个数字并回车后，打印出“您输入的最大数字为：[三个数字中最大的数字]”如图 4.9 所示。

图 4.9 Get Max Number

程序中要用三个变量（num1,num2,num3）保存输入的三个数字。

通过如下伪代码描述的逻辑编写代码 if【此处有一个条件表达式】输出第一个数字，即打印：您输入的最大的数字是[num1] elif【此处有一个条件表达式】输出第二个数字，即打印：您输入的最大的数字是[num2] else 输出第三个数字，即打印：您输入的最大的数字是[num3]。

至少通过如下测试案例：

1 2 3 最大值为 3；

3 2 1 最大值为 3；

2 2 1 最大值为 2。

小练习：

请读者根据等价类、边界值、异常场景推定的测试案例设计技术，再想至少 5 个测试案例。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后重做一次。

难点提示

对于三个逻辑运算符和比较运算符的优先级问题，其实不用特别在意，实际工作和应用过程中，建议通过小括号进行显式的优先级表达。

注意：要根据上面的伪代码写程序，并且至少通过题目中提示的三个测试案例才算通过。

知识点

语言基础知识要点

（1） Python 有三个逻辑运算符。

（2） if……elif……else……语句的冒号别忘了。

（3） 字符串的操作很多，本小节中就用到了 split()，建议仔细阅读 http://www.runoob.com/python3/python3-string.html，特别是关于字符串的 40 个内建函数，记住其功能，以方便使用时查找。

（4） 复习：int()，str()，分别可以将输入的参数转换为整型和字符串。

（5） list 数据类型的相关函数和方法的应用。http://www.runoob.com/python3/python3-list.html。

拓展

如果输入 10 甚至 20 个数字，我们的程序应该如何写？ 想想如果不用 if....else 有没有其他的解法？

4.6 FizzBuzz

学习目标

复习逻辑运算符和 if……elif……else……语句，学习 For/while 循环语句。

学习资源

http://www.runoob.com/python3/python3-conditional-statements.html
http://www.runoob.com/python3/python3-number.html
http://www.runoob.com/python3/python3-loop.html

知识准备

在 4.4 这是奇数还是偶数一节，已经完成了《简明 Python 教程》中“控制流”小节“if 语句”的学习，这一小节的挑战题目需要完成后续的“while 语句”、“for 循环”、“break 语句”和“continue 语句”的学习。

特别提到的是在“while 语句”部分，又提到了上面猜数字的小程序，这一次程序的名字叫做 while.py。

为了增加程序的趣味性，读者可以在完成《简明 Python 教程》中“控制流”小节的阅读之后，对该小节中的 while.py 程序进行一下微小的改动。

小练习：

对《简明 Python 教程》中“while 语句”部分的 while.py 程序进行升级，改名为 guessnNum.py，将原来固定的被猜测的数字 23，变成一个 1 到 100 之间的随机数，每次程序运行的时候都不同。

给大家一点提示：

介绍一个产生整数随机数的函数，来自于 random 库的 randint(num1, num2)。该函数会产生一个从 num1 到 num2 之间（包含 num1 和 num2 在内）的整数，用法如下：

```
>>> from random import randint
>>> number = randint(1,100)
>>> print(number)
12
>>> number = randint(1,100)
>>> print(number)
53
>>>
```

上面的“from random import randint”的意思是从 random 模块中引入 randint 函数。引入模块中的方法的详细内容，我们在后续的章节里会学到，这里大家只要知道这个用法，学会原样拷贝应用即可。

请大家根据上面的提示，改完对 while.py 的升级，编写一个 guessNum.py 程序，将其中的提示也改成中文，运行的效果如图 4.10 所示。

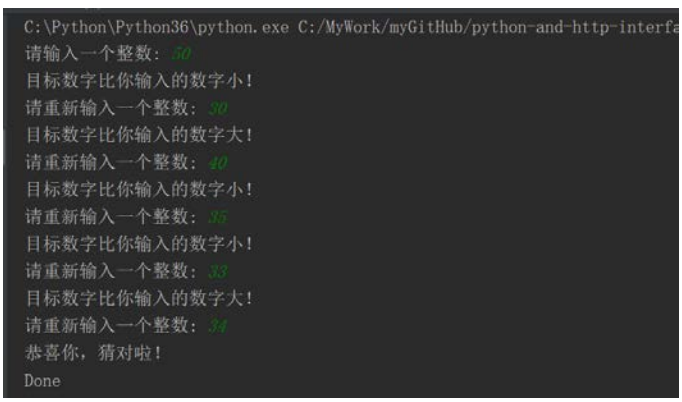


图 4.10 Guess Number

挑战问题

编程界有一个经典的编程练习题目，输出 0 到 100 的数字，如果数字是 3 的倍数输出 Fizz，5 的倍数输出 Buzz，同时是 3 和 5 的倍数的输出 FizzBuzz。

下面稍作改造，编写一个 Python 程序 FizzBuzz.py，运行程序，提示“请输入一个小于 100 的数字：”，程序遍历从 1 到输入的数字之间的所有数字，并判断该数字如果能被 3 整除，打印“Fizz”，该数字如果能被 5 整除，则打印“Buzz”；如果同时能被 3 和 5 整除，则打印“FizzBuzz”；其他情况打印原数字。运行如图 4.11 所示。

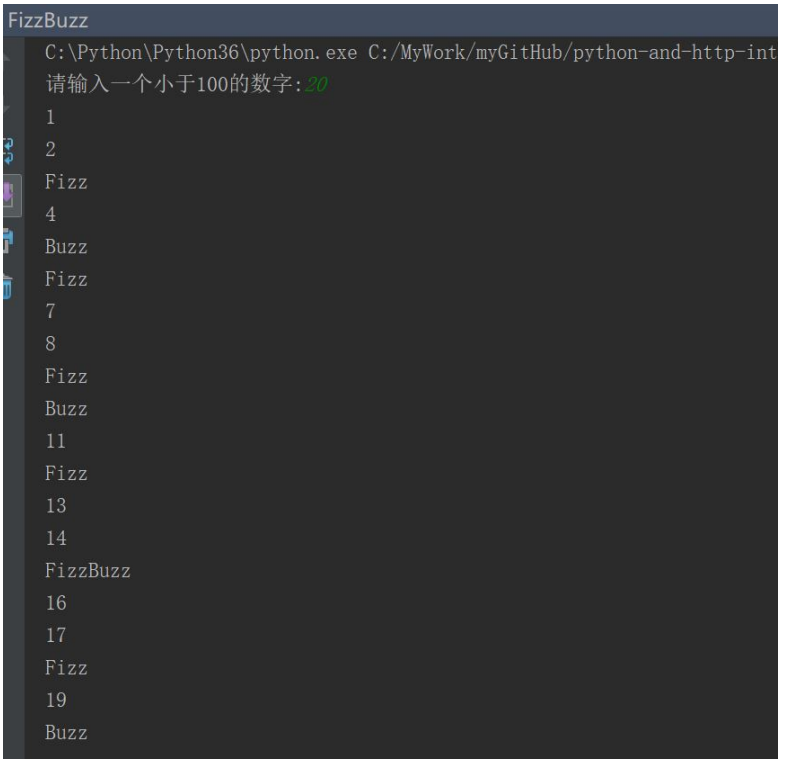


图 4.11 Fizz Buzz

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后重做一次。

难点提示

理解并掌握 for 循环和 while 循环的使用很重要。

知识点

语言基础知识要点：

（1） Python 中的循环语句有 for 和 while。

（2） for 循环可以遍历任何序列类变量，如一个列表或者一个字符串，甚至一个元组。

（3） 可以用内置 range()函数产生一个序列，range 函数最多可以有三个参数，分别是:起始数，结束数，步长。

（4） continue，break 和 pass 之间的区别是什么？

（5） 算术运算符中计算余数的模运算符%，被整除意味着模为零。

拓展

使用 for 和 while 两种方法解决这个问题。在使用 for 解决的时候再考虑使用 range 和不使用 range 两种写法。

你一共可以使用三种写法来完成这个题目。

4.7 建造星星塔

学习目标

学习 for 循环及嵌套循环的知识和 print 语句的格式。

学习资源

http://www.runoob.com/python3/python3-number.html

http://www.runoob.com/python3/python3-loop.html

http://www.runoob.com/python3/python3-inputoutput.html

知识准备

内建函数与 print

Python 解释器有一些自带的内建函数，截止 Python3.6，有 68 个内建函数，参考：

https://docs.python.org/3/library/functions.html?highlight=print#print。

其中的 int(),str(),range()等，相信大家都已经用过很多次了，这里向大家介绍一下 print()函数。

print()函数的声明：print(*objects, sep=' ', end='', file=sys.stdout, flush=False)

功能：打印传入的对象参数中的内容到字节流文件，以 sep 参数中的值作为分隔符，默认为空格；以 end 参数中的值为结束符，默认为换行符。

使用说明：sep，end，file 和 flush 四个参数必须以关键字参数的方式给出。所有非关键字参数都会被转换为字符串，并被写入到流文件中。

file 参数必须是一个有 write(string)方法的对象，并且 print()函数不能用于向文件对象写入二进制数据，如果要写二进制数据到文件中应该使用文件类的 write()方法。

通过上面的关于 print()函数的介绍我们可以知道，如果我们希望一个字符串，又不希望换行，可以在打印的时候将 print()函数的 end 参数赋值为空字符。

```
print("Hello",end="")
print("World",end="")
print("!")
```

想想上面三行语句的运行结果与下面的三行语句有什么不同？

```
print("Hello")
print("World")
print("!")
```

下边这一条语句呢？

```
print("Hello","World","!",sep="-")
```

仔细观察三条代码的运行结果，体会一下 print()函数的用法。

嵌套 for 循环

循环无论是 for 和 while，都是可以嵌套使用的，同时，读者一定要理解，计算机的执行过程是一条一条顺序执行的。

举个例子：

```
for i in range(1,5):
    for j in range(1,8):
        print("*",end="")
```

```
        print("")
    print("Done")
```

想一下，上面的代码会打印一个几行几列的由字符“*”组成的矩阵？

（1） 在执行了第一个外层的 for 循环后，i 的值为 1；

（2） 进入第二个 for 循环，j 的值为 1；

（3） 执行第三条语句，打印一个字符“*”，此时，第二个 for 循环并没有结束；

（4） 计算机回到第二条语句进行循环执行，此时 j 的值会变为 2，注意：此时 i 的值是不变的，仍然为 1；

（5） 接着循环执行到第三条语句，打印第二个字符“*”；

（6） 以此类推，直到执行第二条语句的时候 j 的值为 8，结束第二个 for 循环的执行；

（7） 执行第四条语句打印一个换行符，此时第一个外层的 for 循环的 i 值仍然为 1；

（8） 计算机回到第一条语句继续进行循环执行，执行后，i 的值为 2；

（9） 重复上面第二步的过程，直到第一条语句的 i 值为 5，彻底结束第一个外层 for 循环；

（10） 执行最后一条语句打印一行：Done。

所以，上面的程序打印的是一个四行七列的星星矩阵。

尝试运行上面的代码，体会两层 for 循环的运行过程。

挑战问题

这又是一个经典的编程问题。编写一个 Python 程序，trig.py，运行程序。提示“请输入塔高”，输入数字并回车后，打印出如图 4.11 所示的等腰三角星星塔，**每次只允许打印一个字符**。

图 4.11 Trig

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点。

（1） 嵌套 for 循环的使用。

（2） 在 for 循环中，理解边界的控制条件。

（3） print 语句中 end 的使用，以及如何控制 print 换行。

拓展

该题有两种解法，一种是两层嵌套 for 循环，一种是只用一层 for 循环。

读 者 圈

第五章 我长大了

5.1 函数是枝叶

学习目标

学习自定义函数和使用自定义的函数。

学习资源

http://www.runoob.com/python3/python3-number.html
http://www.runoob.com/python3/python3-loop.html
http://www.runoob.com/python3/python3-function.html

知识准备

函数的定义和参数

首先学习《简明 Python 教程》中的“函数”小节。
学会如何定义一个函数并使用它。
学习函数的参数，包括默认参数值、关键字参数、可变参数三个概念。
学习全局变量和局部变量的使用。

string 的 format 方法

字符串类型的 format()内建函数，用于对字符串进行格式化填充，并返回处理后的字符串。

先看如下的示例：

```
>>> str = '一个篮子有{}个鸡蛋，{}个篮子有{}个鸡蛋'
>>> str.format(3,4,3*4)
'一个篮子有 3 个鸡蛋，4 个篮子有 12 个鸡蛋'
```

```
>>> print(str)
一个篮子有{}个鸡蛋，{}个篮子有{}个鸡蛋
>>> print(str.format(3,2,3*2))
一个篮子有 3 个鸡蛋，2 个篮子有 6 个鸡蛋
```

我们看到，要使用字符串的 format()内建函数进行格式化填充，首先要求在字符串中将要填充的位置用{}进行占位。大括号的中间也可以填写数字，用于指定该位置填入的内容来自传入 format()函数的第几个参数，序号从 0 开始，默认不填写数字的情况下，大括号的位置与 format()函数中参数的位置是一一对应的。

仔细观察如下语句的不同，体会大括号占位符中数字的含义：

```
>>> str = '一个篮子有{}个鸡蛋，{}个篮子有{}个鸡蛋'
>>> print(str.format(3,2,3*2))
一个篮子有 3 个鸡蛋，2 个篮子有 6 个鸡蛋
>>> str = '一个篮子有{1}个鸡蛋，{0}个篮子有{2}个鸡蛋'
>>> print(str.format(3,2,3*2))
一个篮子有 2 个鸡蛋，3 个篮子有 6 个鸡蛋
>>>
```

format()函数还可以接受关键字参数或者字典作为参数，这样在前面的大括号里就可以通过关键字实现引用。下面的例子改编自：https://docs.python.org/3.6/library/string.html#formatspec

```
>>> '坐标: {latitude}, {longitude}'.format(latitude='37.24N', longitude='-115.81W')
'坐标: 37.24N, -115.81W'
```

以上是通过关键字参数的方式进行字符串填充。

```
>>> coord = {'latitude': '37.24N', 'longitude': '-115.81W'}
>>> '坐标: {latitude}, {longitude}'.format(**coord)
'坐标: 37.24N, -115.81W'
```

以上是通过字典参数的方式进行字符串填充的。

更多的字符串填充和美化的内容可以参阅，http://www.runoob.com/python3/python3-inputoutput.html

列表和字典

通过 4.2 小节的学习，我们知道了 Python 中有六个标准的数据类型：

- Numbers（数字）。
- String（字符串）。
- List（列表）。
- Tuple（元组）。
- Set（集合）。
- Dictionary（字典）。

我们已经认识了数字和字符串，并且也已经使用这两种数据类型解决了很多“挑战问题”，本小节我们重点学习一下列表和字典两种数据类型。

列表

列表是一系列的数据元素的顺序集合，每一个都有一个给定的下标可以被访问到，下标从零开始，类似于很多其他的语言中的数组。

列表的定义

列表元素必须使用方括号（[]）括起来，并且在元素之间使用逗号分隔。

```
listExample = ['Akui',"Yu",1979,"a_kui@163.com"]
```

列表中的元素并不需要同一种数据类型，甚至列表中的元素还可以有列表，被称为嵌套列表。

列表的访问

- 列表中元素的访问通过方括号中放入下标来访问，下标从 0 开始。
- 访问多个元素的时候可以使用冒号(：)来分隔起始下标和结束下标，没有起始下标和结束下标只有一个冒号(：)的时候返回列表中所有的元素。
- 可以放入第二个冒号(：)，第二个冒号(：)后的数字表示步长，步长也可以是负数，表示从后向前。

请仔细阅读下面的代码和运行结果，体会上面的三条访问规则。

```
>>> listExample = ['Akui',"Yu",1979,"a_kui@163.com"]
>>> listExample[2]
1979
```

```
>>> listExample[1:3]
['Yu', 1979]
>>> listExample[1:]
['Yu', 1979, 'a_kui@163.com']
>>> listExample[:]
['Akui', 'Yu', 1979, 'a_kui@163.com']
>>> listExample[::2]
['Akui', 1979]
>>> listExample[::-1]
['a_kui@163.com', 1979, 'Yu', 'Akui']
>>>
```

列表的操作

列表有如下四种基本操作：+、*、in 和 not in。

- 加号(+)，用于拼接列表。
- 星号(*)，用于重复列表中的内容。
- in，用于判断元素是否在列表中，在列表中返回 True，否则返回 False。
- not in，用于判断元素是否不在列表中，在列表中返回 False，否则返回 True。

继续上面的代码，假设已经有一个 listExample=['Akui',"Yu",1979, "a_kui@163.com"]，仔细阅读下面的代码和运行结果，体会上面的四条规则。

```
>>> listExample2=listExample+[2017]
>>> listExample2[:]
['Akui', 'Yu', 1979, 'a_kui@163.com', 2017]
>>> listExample2=listExample+[2017]*3
>>> listExample2[:]
['Akui', 'Yu', 1979, 'a_kui@163.com', 2017, 2017, 2017]
>>> 1979 in listExample2
True
>>> 2000 in listExample2
False
>>> 2000 not in listExample2
```

```
True
>>>
```

列表数据类型有 4 个函数和 11 个方法，请读者参考以下链接加以学习，不必每一个都熟记其具体用法，但是，要了解和熟记其作用。http://www.runoob.com/python3/python3-list.html

阅读之后尝试回答如下问题：

- 如何替换列表中一个元素的内容？
- 如何删除列表中的一个元素？
- 如何增加一个元素到列表的指定位置？

字典

字典是一系列以键值对的方式存储的数据元素的集合，每一个键和值之间是一一对应的，键必须是唯一的，值则不必唯一，通常通过键元素来访问值元素，一般不通过下标的方式访问。

字典的定义

字典元素必须使用大括弧（{}）括起来，键元素和值元素之间用冒号(:)分隔，并且在每个键值元素对之间使用逗号分隔。

```
dictExample = {'name':'Akui','age':18,'email':"a_kui@163.com"}
```

字典中的元素并不需要同一种数据类型，字典的键元素必须是唯一的。

字典的访问

字典的访问通常是把相应的键元素放入方括弧（[]）来获取对应的值元素。

```
>>> dictExample = {'name':'Akui','age':18,'email':"a_kui@163.com"}
>>> print(dictExample['name'])
Akui
>>> print(dictExample['abcd'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'abcd'
>>>
```

仔细阅读以上代码，体会字典的访问。注意，如果用字典里不存在的键来访问字典，会得到 KeyError 错误。

字典的操作

字典有如下两种基本操作，in 和 not in 主要用于对键元素的判断。

- in，用于判断键元素是否在字典中，在列表中返回 True，否则返回 False。
- not in，用于判断键元素是否不在字典中，在列表中返回 False，否则返回 True。

仔细阅读下面的代码和运行结果，体会上面的两条规则。

```
>>> dictExample = {'name':'Akui','age':18,'email':"a_kui@163.com"}
>>> 'name' in dictExample
True
>>> 12222 in dictExample
False
>>> 12222 not in dictExample
True
```

字典有 3 个函数和 12 个方法，请读者参考以下链接加以学习，不必每一个都熟记其具体用法，但是，要了解和熟记其作用。http://www.runoob.com/python3/python3-dictionary.html

阅读之后尝试回答如下问题：

- 如何替换字典中一个元素的内容？
- 如何删除字典中的一个元素？
- 如何增加一个元素到字典中？

函数指针

函数指针可以简单地理解为一个指向函数的变量，也就是说函数是可以赋值给一个变量的，此时候这个变量就代表这个函数。

```
>>> def foo():
...     print("foo")
...
>>>
```

```
>>> function = foo
>>> function()
foo
>>>
```

上面的代码中定义了一个 foo()函数，之后 foo()函数被赋值给了 function 变量，最后一行代码中的 function()语句相当于在调用 foo()函数。函数的指针不仅仅可以保存在变量中，列表（list）可以用于存放函数指针。

```
def foo1():
    print("this is foo1!")
def foo2():
    print("this is foo2!")
listFun = [foo1,foo2]
for function in listFun:
    function()
```

当然，字典也同样可以用于存放函数指针，并且有的时候，这样的用法可以取得意想不到的效果。

挑战问题

编写一个 Python 程序 calculator.py，运行程序，提示“选择运算符”，输入“1/2/3/4”之一并回车后，继续输入要进行运算的两个数字后回车，打印出该运算结果，如图 5.1 和图 5.2 所示。

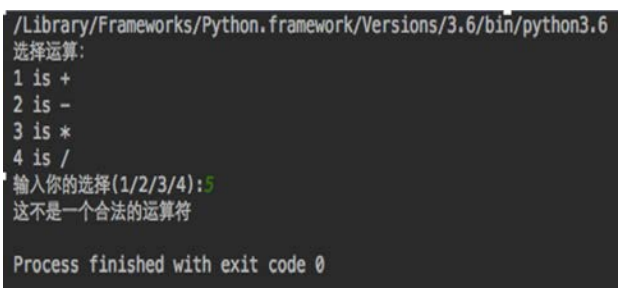


图 5.1 选择运算

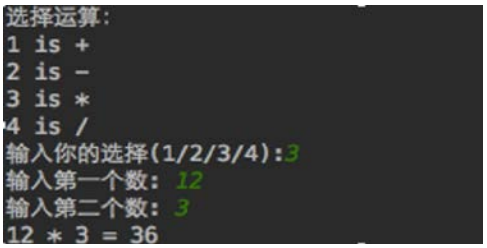


图 5.2 选择运算运行结果

问题约束

编写的程序要满足以下条件：

程序中要用三个全局变量（operator,num1,num2）代表运算类型和参加运算的两个数字 num1、num2。

定义四个函数：add()，minus()，multiply()，divide()，分别作为运算加、减、乘、除的函数。要求这些函数接收两个数字作为输入，返回一段包含对应运算等式的字符串作为结果。

如：add()函数，接受 num1 和 num2 作为输入参数，**返回如下的字符串**作为结果：

```
num1 + num2 = 【两个数的和】
```

比如：add(3,4)，函数输出的结果字符串为：

```
"3 + 4 = 7"
```

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

- 函数可以理解为一段可以重复使用代码块。
- 定义函数的规则如下：

– 函数代码块以 def 关键词开头，后接函数标识符名称和圆括号()。

– 任何传入参数和自变量必须放在圆括号中间，圆括号之间可以用于定

义参数。

- 函数的第一行语句可以选择性地使用文档字符串，用于存放函数说明。
- 函数内容以冒号起始，并且缩进。
- 函数以 return [表达式] 结束函数，选择性地返回一个值给调用方。不带表达式的 return 相当于返回 None。

- 在 Python 中，所有参数（变量）都是按引用传递。如果你在函数里修改了参数，那么在调用这个函数的函数里，原始的参数也被改变了。这一点很重要。
- 定义在函数内部的变量拥有一个局部作用域称为局部变量，定义在函数外的拥有全局作用域，称为全局变量。
- 判断错误输入并给于提示，避免程序在运行时出现异常。

拓展

尝试使用字典来存放四个函数的指针以简化代码。
回答前，可以阅读如下链接中的内容。

```
http://www.runoob.com/python3/python3-dictionary.html
```

5.2 模块是枝干

学习目标

学习模块的概念和自定义模块。

学习资源

http://www.runoob.com/python3/python3-module.html
https://pythonguidecn.readthedocs.io/zh/latest/writing/structure.html#id8

知识准备

学习《简明 Python 教程》中的“模块”小节，这个章节里几乎包含了作为初学者应该了解的关于模块的一切知识。

尝试回答如下问题：

1. import module 和 from module import * 的区别是什么？

2. 当模块中的功能被调用的时候，模块中的__name__变量的内容是什么（）？

A. "main"，B. "__name__"，C. 模块的名字，D. "__模块的名字__"

挑战问题

改写“5.1 小节”：编写一个 Python 程序 calcByModule.py，运行程序，提示“选择运算符”，输入“1/2/3/4”之一并回车后，继续输入要进行运算的两个数字后回车，打印出该运算结果。见上文 5.1 小节图 5.2 所示。

程序中要用到一个名字为 calculator 的自定义模块，它封装了加减乘除的函数，这四个函数接收两个数字作为输入，返回一段包含对应运算等式的字符串作为结果。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

以下内容改编自《简明 Python 教程》。

（1） 被引用（import）的模块，应该与引用程序放置在同一目录下，或者放置在 sys.path 所列出的目录下。

（2） 如同函数是程序中可重复使用部分一样，模块也是一种复用程序代码的方式。

（3） 包是用以组织模块的另一种层次结构。

（4） Python 所附带的标准库是由包和模块组成的。

5.3 面向对象是另一种看待世界的视角

学习目标

学习类和对象的使用。

学习资源

http://www.runoob.com/python3/python3-number.html
http://www.runoob.com/python3/python3-loop.html
http://www.runoob.com/python3/python3-class.html

知识准备

虽然有很多关于类和对象的解释，但是，我认为最简单直接、浅显易懂的理解是：类可以看作一系列函数定义和变量声明的集合，类中定义的函数称为方法，类中定义的变量称为成员。

一个自定义的类可理解为一个由开发人员自己定义的新的数据类型。而对象，就可以理解为是用这个复杂的数据类型声明出来的变量。

学习《简明 Python 教程》中的“面向对象编程”小节，这个章节里几乎包含了作为初学者应该知道的关于面向对象的一切知识。

尝试回答如下问题：

1. 从概念上来说一下类和对象的区别是什么？举个例子？

2. 狗是（ ）藏獒是（ ），我家的藏獒小花是（ ）。答案从下面的选项中选择（注意：这是一个多项选择题）

A. 类，B. 子类，C. 实例，D. 对象

3. 类方法与普通函数只有一种特定的区别——前者必须有一个额外的参数，这个参数的名字是（ ）。

4. **init** 方法会在（ ）时立即运行。

5. 请再列出 3 个类似于__init__的类的专有方面的名字：（ ）、（ ）、（ ）。

6. （ ）是共享的（Shared）—它们可以被属于该类的所有实例访问。（ ）由类的每一个独立的对象或实例所拥有，每个对象都拥有自己的一个副本，且不会共享。（注意：这是一个单项选择题）

A. 对象变量，B. 私有变量，C. 类变量，D. 公有变量

挑战问题

使用类的方法改写“5.1 小节”：编写一个 Python 程序 CalculatorByCls.py，

运行程序，提示“选择运算符”，输入“1/2/3/4”之一并回车后，继续输入要进行运算的两个数字后回车，打印出该运算结果，见 5.1 小节图示。

程序中要定义一个类，类的名字为 MyCalculator，该类要封装加、减、乘、除四个方法。这四个方法接收两个数字作为输入，返回一段包含相应的运算等式的字符串作为结果。

在 CalculatorCls.py 程序中，实例化类 MyCalculator，得到一个对象变量 calc，利用 calc 对象进行加减乘除的运算和对应的运算等式的字符串的获取。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

（1） 类可以看作一系列函数定义和变量声明的集合，类中定义的函数称为方法，类中定义的变量称为成员。

（2） 类就像一个自定义的数据类型，其使用过程一般首先声明一个对象变量，然后通过这个对象变量来完成相应的功能。

（3） 每个对象都是一个独立的内存实体，同一个类可以实例化出多个对象。

（4） 子类会继承父类的方法和成员。

（5） 类的私有方法和成员，都以两个下画线开头，不能被继承，也不能被类外部的程序访问。

第六章 我想和你谈谈

6.1 终端带来即时交互

学习目标

进一步熟悉终端输入和输出的即时交互编程方法，复习面向对象编程中的内容。

学习资源

http://www.runoob.com/python3/python3-string.html
http://www.runoob.com/python3/python3-loop.html
http://www.runoob.com/python3/python3-class.html

知识准备

本小节中并没有需要特别学习的新内容，主要是复习一直以来被用到的input()和 print()函数。读者可以参考学习资源中的内容及《简明 Python 教程》中的“输入与输出”小节中的“用户输入内容”部分。该部分关于判断一段输入是否是回文的练习，是一个典型的终端即时交互的例子。

挑战问题

编写一个程序 MoveStar.py，该程序会在字符终端 **1-24 之间**的位置**随机**打印出一个星号(*)，并提示"请输入移动星号的指令（L/l or R/r）："，如果用户输入 L 并回车，星号就会向左移动一个字符的位置，并被重新输出；如果用户输入 R 并回车，星号就会向右移动一个字符的位置，并被重新输出。程序会循环提醒用户输入指令。直到用户输入“EXIT”，程序才会退出。如图 6.1 所示。

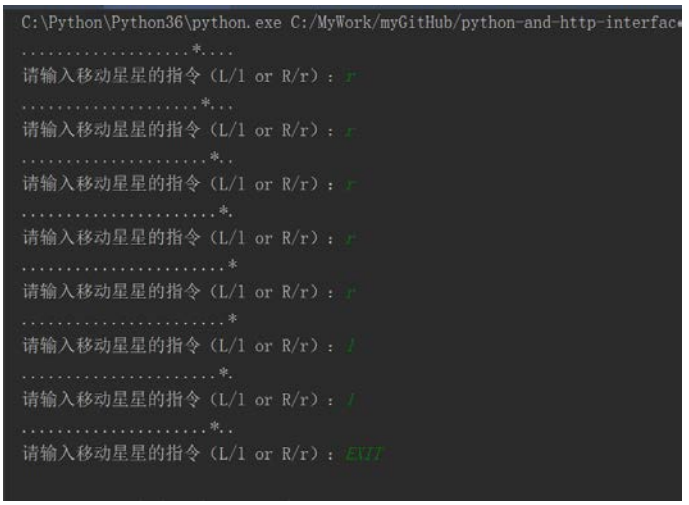


图 6.1 Move Star

要求采用面向对象的方法编写该程序，在实现的过程中至少定义一个类。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

难点提示

注意星号(*)不能移动出边界，注意测试和检查星号(*)在边界位置的情况，也就是星号(*)移动到第一个字符的位置和最后一个字符位置的情况。

星号(*)移动到第一个字符的位置后，将不能再向左移动；星号(*)移动到最后一个字符的位置后，将不能再向右移动。

随机数的产生，在“4.6 FizzBuzz”小节提到。

如有需要，建议复习“4.5 我们三个谁最大”小节中的**字符串操作**相关的内容。

知识点

语言基础知识要点

拓展

如果星号(*)是在一个 10*10 的空间中移动，并增加向上和向下的指令呢？

6.2 文件适用于批量交互

学习目标

学习文件的读写，以及复习字符串处理。

学习资源

http://www.runoob.com/python3/python3-inputoutput.html
http://www.runoob.com/python3/python3-file-methods.html

知识准备

文件的读写最重要的三个步骤：打开文件、读写文件、关闭文件。首先建议学习一下《简明 Python 教程》中的“输入与输出”小节中的“文件”部分，这个例子很巧妙地将文件的写和读有机地组合成为一个整体，将相关的常用方法进行了完整的展示。

打开文件

打开文件最简单，也最为常用，直接调用 open(filename, mode)方法，filename 参数用于传递文件的路径名，mode 参数用于指示文件打开的方式。

该方法常用的使用方式如下：

```
fileForRead = open("input.txt","r")
```

上面的代码是以读的方式打开文件，open()函数会返回一个值，务必把这个值保存到一个变量中，这个值一般被称为文件的操作句柄或者文件对象，后续关于文件的所有操作都是通过这个变量进行的。

```
fileForWrite = open("output.txt","w")
```

上面的代码是以写的方式打开文件，注意其模式和以读的方式打开文件的区别。

详细的关于 open()函数的使用说明可以参见 http://www.runoob.com/

python3/python3-inputoutput.html 的“读和写文件”部分。

读写文件

文件分为文本文件和二进制文件，这里只介绍文本文件。

文本文件的读，重点介绍两个方法：file.readline()和 file.readlines()。

file.readline()会读取文件的一行内容，并返回一个包含这一行内容的字符串。

file.readlines()会读取文件的所有行，并返回一个列表数据类型，列表中的每一个元素都是一个字符串对应文件中的一行内容。

这里要提示的是，**文件以文本形式**打开后，文件对象可以作为一个可遍历的类似列表数据类型的变量，通过 for 语句进行遍历访问。

```
for line in inputFile:
    print(line)
```

file.write(string)方法会将参数 string 中包含的字符串，写入到 file 文件对象对应的文件中，该文件一定要是用可写的方式打开的。

注意：file.write()方法并不会向 print()方法一样自动换行，所以如果你写入字符串后希望字符串独立成为一行，需要在字符串的后面自行增加回车换行符(\)。

```
fileForWrite.write(line+"\n")
```

关于二进制文件内容，建议读者可以自行进行一些学习和练习，进而挑战扩展小节中的问题。

关闭文件

文件操作完毕后，务必要调用 close()方法，将文件进行关闭。

```
fileForRead.close()
fileForWrite.close()
```


挑战问题

从一个文件 input.txt 读入每一行一个数字，将其翻译为 FizzBuzz 后，对

应地写入另一个文件 output.txt 中。从数字到 FizzBuzz 的转换规则同 4.6 FizzBuzz 小节：如果数字能被 3 整除，则在 output.txt 文件的对应行写入“Fizz”；如果数字能被 5 整除，则在 output.txt 文件的对应行写入“Buzz”；如果数字同时能被 3 和 5 整除，则在 output.txt 文件的对应行写入“FizzBuzz”；其他情况在 output.txt 文件的对应行写入原数字。

比如 input.txt 的内容如下所示：

```
2
5
 233
 345
 544
 693
459
8
6
4
92
```

文件中每一行都有一个数字，但是，数字的前面或者后面可能有空格。

经过处理后生成的 output.txt 内容如下所示：

```
2
Buzz
233
FizzBuzz
544
Fizz
Fizz
8
Fizz
4
92
```

要求采用面向对象的方法编写该程序，在实现的过程中至少定义一个类。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要

闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

难点提示

考虑将 FizzBuzz 的运算封装到一个类中，比如定义一个 FizzBuzz 类，并给类定义一个方法 getFizzBuzz(self,number)，该方法接受一个前后可能包含空格的数字字符串作为输入；返回一个经过转换的 FizzBuzz 字符串作为输出。

知识点

语言基础知识要点

文件的打开、读写和关闭用到的函数：

- open。
- readline。
- readlines。
- write。
- close。

拓展

从 github 上的 expansion.data 文件读取一个 **8 个字节**的二进制**无符号长长整型（unsigned long long）**数据。该数字是作者本人的微信号，作者热情地欢迎你加入学习群进行交流学习。

提示：需要用到二进制文件的读写和 struct 模块，推荐大家提前阅读 Pythonclub 的这篇文章。

```
http://www.pythonclub.org/python-files/binary
```

6.3 处理异常不要崩溃

学习目标

学习异常处理的使用，避免程序因为对异常场景或者异常数据的考虑不周

而崩溃。

学习资源

http://www.runoob.com/python3/python3-errors-execptions.html

知识准备

阅读《简明 Python 教程》中的“异常”小节，区分错误和异常，并且学会如何处理异常和抛出异常。

来自 Python 之禅的忠告

前面的“2.1 Python 之禅”小节中提到：

Errors should never pass silently. Unless explicitly silenced. 所有错误都不应该被直接忽略，除非能够被精确的捕获之后（其中一个典型的例子就是，不建议用 except:pass 来直接忽略所有异常）。

在处理异常的时候，建议读者尽量精确地处理异常情况，而不是简单地使用 except Exception 来粗暴处理。

常见的异常

以下内容改编自 Python 官方文档，附带了官方文档的英文原文，读者可以比较阅读，提升一下英语阅读的能力：

exception IndexError 索引错误异常：

Raised when a sequence subscript is out of range.

当访问序列或者列表的元素时，超出索引范围，即出现越界访问的时候，会报这个异常。

```
>>> list = [1,2,3]
>>> list[0]
1
>>> list[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
**IndexError: list index out of range**
>>>
```

exception KeyError，键错误异常：

Raised when a mapping (dictionary) key is not found in the set of existing keys.

当被访问的字典中不存在要访问的键元素的时候，会报这个异常。

```
>>> dictExample = {'name':'Akui','age':18,'email':"a_kui@163.com"}
>>> dictExample['abcdefg']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
**KeyError: 'abcdefg'**
>>>
```

exception NameError，名字错误异常：

Raised when a local or global name is not found.

当本地或者全局变量不存在的时候，会报这个异常。

```
>>> name = var
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
**NameError: name 'var' is not defined**
>>>
```

exception SyntaxError，语法错误异常：

Raised when the parser encounters a syntax error.

当解释器发现一个语法错误的时候，会报这个异常。

```
>>> '2' x 2
  File "<stdin>", line 1
    '2' x 2
        ^
**SyntaxError: invalid syntax**
```

exception TypeError，类型错误异常：

Raised when an operation or function is applied to an object of inappropriate type.

当操作或者函数应用的对象类型不正确的时候，会报这个异常。

比如下面的代码，字符串‘2’和数字 2 之间进行加法操作的时候，由于两个数据类型之间不能进行加法操作，就会报出 TypeError，即类型错误异常。

```
>>> '2'+2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not int
>>>
```

exception ValueError，值错误异常：

Raised when a built-in operation or function receives an argument that has the right type but an inappropriate value, and the situation is not described by a more precise exception such as IndexError.

当内建的操作或者函数的参数类型是正确的，但是参数中的内容不恰当，并且这种情况又不是通过一个其他的异常类型（比如，IndexError）能够更精确地描述的时候，会报这个异常。

```
>>> number = int("abc")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
**ValueError: invalid literal for int() with base 10: 'abc'**
>>>
```

exception ZeroDivisionError，零除异常：

Raised when the second argument of a division or modulo operation is zero.

当除法或者模运算的第二个参数为零的时候，会报这个异常。

```
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>>
```

挑战问题

挑战问题同 6.2 小节，但是，要增加一种容错能力。即，当输入文件 input.txt 中某一行的内容中包含了不能转换为数字的字符时，则在 output.txt 文件的对应行写入“Exception”。其他要求与“6.2 文件适用于批量交互”小节的挑战问题要求相同。

比如 input.txt 的内容如下所示：

```
2
5
 233
 345
 544
 akui
459
8
6
4
92
```

文件中每一行都有一个数字，但是，数字的前面或者后面可能有空格，也有可能包含不能转换为数字的字符串。

经过处理后生成的 output.txt 的内容应该如下所示：

```
2
Buzz
233
FizzBuzz
544
Exception
Fizz
8
Fizz
4
92
```

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

异常处理常用的句式为：

- try ... except。
- try ... except .... finally。
- 可以有多个 except。

拓展

仔细阅读下面的代码，推测运行后的输出是什么，并写下来：

```
def testTry1():
    try:
        return 0
    except(Exception):
        return 1
    finally:
        return 2

def testTry2():
    try:
        raise Exception
        return 0
    except(Exception):
        return 1
    finally:
        return 2

def testTry3():
    try:
```

```
        "2"+2
        return 0
    except(Exception):
        return 1
    finally:
        return 2


def testTry11():
    try:
        return 0
    except(Exception):
        return 1

def testTry21():
    try:
        raise Exception
        return 0
    except(Exception):
        return 1

def testTry31():
    try:
        "2"+2
        return 0
    except(Exception):
        return 1

if(__name__ == "__main__"):
    print(testTry1())
    print(testTry2())
    print(testTry3())
    print(testTry11())
    print(testTry21())
    print(testTry31())
```

运行上面的程序，将自己写下来的输出与真实的运行结果对比一下，看看是否一致？如果不一致，想一下为什么？重新阅读学习资源中的链接和《简明 Python 教程》中的“异常”小节！

读 者 圈

第三部分 初识 HTTP

本部分的学习中，你需要用到以下资料：

http://cn.python-requests.org/zh_CN/latest/

这是 requests 库的官方中文资料，并且是最新版本的。

软件技术的发展非常快，特别是 Python 语言的发展更是如此，在遇到问题进行资料查找时候，一定要养成查询最新资料的习惯，以免过时的解决方案和信息对自己产生误导。

具体来说，对于 Python 语言，建议只看三年以内的资料。

另外，本部分有的章节的学习和练习，需要你在进行挑战问题解决的过程中随时访问互联网。

第七章 相识前的准备

7.1 JSON 格式的通信录

学习目标

学习 JSON 数据格式。

学习资源

http://www.w3school.com.cn/json/index.asp
http://www.runoob.com/json/json-tutorial.html
http://www.runoob.com/python3/python3-json.html

知识准备

JSON 入门

一个典型的 JSON 的例子

```
{
“公司”:“IT 匠艺教研室”
"员工": [
{ "姓氏":"赵" , "名字":"江" },
{ "姓氏":"钱" , "名字":"河" },
{ "姓氏":"孙" , "名字":"湖" },
{ "姓氏":"李" , "名字":"海" }
]
}
```

JSON 是什么?

JSON：JavaScript 对象表示法（JavaScript Object Notation）。

- JSON 是存储和交换文本信息的语法，类似 XML。
- JSON 比 XML 更小、更快，更易解析。
- JSON 是轻量级的文本数据交换格式。
- JSON 独立于编程语言。
- JSON 具有自我描述性，更易理解。
- JSON 是纯文本。
- JSON 具有层级结构（值中存在值）。

JSON 语法规则

JSON 语法是 JavaScript 对象表示法语法的子集。

- 数据在“名称/值”对中。
- 数据由逗号分隔。
- 花括号保存对象。
- 方括号保存数组。

关于 JSON，建议读者学习 w3school 的在线教程，教程对 JSON 的基础知识有较为细致的介绍,这里就不赘述了。

http://www.w3school.com.cn/json/index.asp

Python 与 JSON

通过对 JSON 在线教程的学习，我们发现，JSON 的定义与 Python 的字典非常相像。我们在“5.1 函数是枝叶”小节学习过字典的定义。

但我们要分清 JSON 和字典之间的区别。

- JSON 严格意义上是一个由大括弧括起来的，通过键值对的方式将数据组织起来的数据表示格式；而 Python 的字典是 Python 语言的一种内建支持的数据类型，就像数字类型和列表类型一样。
- JSON 本质上是由一种固有的格式来进行数据表达的字符串，也就是说，JSON 数据就是一种固定格式的字符串，而 Python 的字典是 Python 语言的一种数据类型。

但是，在 JSON 和字典数据类型之间可以进行非常方便的数据转换。

json.dumps()函数：

json.dumps(obj, *, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, cls=None, indent=None, separators=None, default=None,

sort_keys=False, **kw)

Serialize obj to a JSON formatted str using this conversion table.

序列化对象到一个 JSON 格式的字符串，转换表如图 7.1 所示。

Python	JSON
dict	object
list, tuple	array
str	string
int, float, int- & float-derived Enums	number
True	true
False	false
None	null

图 7.1 Python and JSON Table

json.dumps()函数有多个参数：

- indent 参数用于表示美化格式输出时的缩进占位个数；
- ensure_ascii 参数默认为 True，这样 utf-8 格式的非 ASCII 编码内容会被编译成 ASCII 编码输出，要想得到字符的真实表示，需要将这个参数设置为 False。
- 其他参数还有：skipkeys，separators，sort_keys 等。

请仔细阅读下面的示例代码和输出：

```
>>> dictExample = {'name':'Akui','age':18,'email':"a_kui@163.com"}
>>> print(dictExample)
{'name': 'Akui', 'age': 18, 'email': 'a_kui@163.com'}
>>> import json
>>> print(json.dumps(dictExample))
{"name": "Akui", "age": 18, "email": "a_kui@163.com"}
>>> print(json.dumps(dictExample,indent=4))
{
    "name": "Akui",
    "age": 18,
    "email": "a_kui@163.com"
```

```
}


>>> dictExample = {'name':'阿奎','age':18,'email':"a_kui@163.com"}
>>> print(dictExample)
{'name': '阿奎', 'age': 18, 'email': 'a_kui@163.com'}
>>> print(json.dumps(dictExample))
{"name": "\u963f\u594e", "age": 18, "email": "a_kui@163.com"}
>>> print(json.dumps(dictExample,ensure_ascii=False))
{"name": "阿奎", "age": 18, "email": "a_kui@163.com"}
>>> print(json.dumps(dictExample,ensure_ascii=False,indent=4))
{
   "name": "阿奎",
   "age": 18,
   "email": "a_kui@163.com"
}
>>>
```

关于 json.loads()函数的用法，请读者自行参考学习中的资料。

挑战问题

编写一个 Python 程序 showjson.py，运行程序，提示“请输入您的姓名：”，回车后，提示，”请输入您的电话号码:”，回车后，用 JSON 格式打印出你输入的通信信息如下：{name:阿奎, phone:13901001234}

如图 7.2 所示。

```
C:\Python\Python36\python.exe C:/MyWork/myBook/用Python做HTTP
请输入你的姓名：阿奎
请输入你的电话：13901001234
{
    "name": "阿奎",
    "phone": "13901001234"
}

Process finished with exit code 0
```

图 7.2 Show JSON

姓名要支持中文，并且打印正常。格式要进行美化，不能只打印成这样：“{"name": "阿奎", "phone": "13901001234"}”

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言知识要点

- Python 用于进行 JSON 处理的包是 JSON 包，所以第一行要先 import 这个包。
- Python 语言的字典和 JSON 都是一种键值(key=>value)结构，也叫 Key—Value 结构。字典的键和值之间用冒号(:)分割，JSON 的键和值之间也用冒号(:)分割。
- json.dumps()函数用于对 Python 数据转换为一个 JSON 格式的字符串。

7.2 状态码的五个分类

学习目标

学习 HTTP 状态码的含义，理解状态码的五个分类。

学习资源

http://www.runoob.com/http/http-status-codes.html
http://www.runoob.com/python3/python3-dictionary.html
http://www.runoob.com/python3/python3-function.html

知识准备

掌握 HTTP 协议是进行 HTTP 接口测试的基本功，在进行 HTTP 协议学习之前，首先要了解一下 HTTP 的状态码。

HTTP 状态码，英文叫做 HTTP Status Code，是网页服务器返回给浏览器的用于表示响应状态的代码，状态码由三个十进制数字组成。所有状态码的第一个数字定义了状态码的类型，一共有五种响应状态类型。

五种响应状态分别是：

- 1XX 信息，服务器收到请求，需要请求者继续执行操作。
- 2XX 成功，操作被成功接收并处理。
- 3XX 重定向，需要进一步的操作以完成请求。
- 4XX 客户端错误，请求包含语法错误或无法完成请求。
- 5XX 服务器错误，服务器在处理请求的过程中发生了错误。

由此可见，状态码的取值范围从 100-599，实际使用中真正定义的状态码并没有那么多。

常用的状态码也就十几个，包括有：

- 200 - 请求成功。
- 301 - 资源（网页等）被永久转移到其他 URL。
- 404 - 请求的资源（网页等）不存在。
- 500 - 内部服务器错误。

还有一个很神奇的 302 状态码，我们在后面的章节中会专门提到。

挑战问题

编写一个 Python 程序 getstatustype.py，运行程序，打印出“请输入你要查询的状态码：[状态码]”，用户输入合法的 HTTP 状态码后，程序打印出该状态码所属的分类，以及分类描述。

如果输入的状态码不合法，也就是说输入的状态码不是 100-599 之间的数字，打印错误提示“输入的状态码不合法，合法的状态码为 100-599 之间的数字。”**程序不结束**，继续再次打印出“请输入你要查询的状态码：[状态码]”，并处理第二次的输入，直到接收到了合法的状态码，并打印出输入的状态码所属的分类，以及所属分类的分类描述，程序才结束。

如图 7.3 所示。

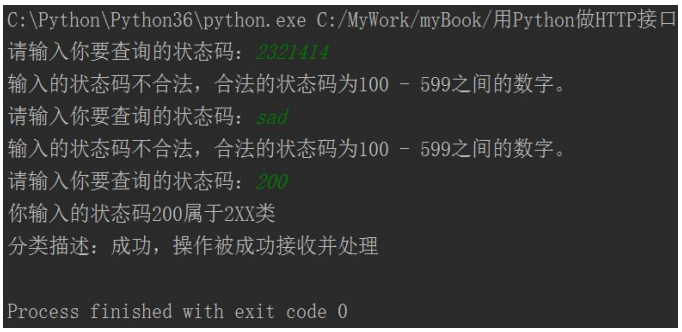


图 7.3 Get Status Type

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

- HTTP 状态码的五个分类以及四个常见的状态码。

拓展

完成上面的练习后，考虑增加在如下约束的情况下，对代码进行调整。首先检查代码是否满足下面的约束条件，如果没有满足约束条件，尝试对代码进行一下重写或者调整。

- 不使用 if 语句。
- 使用的 print 不能多于 3 个。
- 代码行数不要超过 18 行。
- 每行不超过 90 列。

提示：建议考虑用字典函数的方式对代码进行整合，如有需要可以重读一下“5.1 函数是枝叶”小节，特别是对其“扩展”部分的问题进行思考和练习。

7.3 HTTP 协议基础

学习目标

学习 Requests 模块安装，GET 请求发送，返回状态码解释，JSON 到字典，异常处理。

学习资源

http://www.runoob.com/http/http-tutorial.html
http://www.runoob.com/python3/python3-errors-execptions.html
http://cn.python-requests.org/zh_CN/latest/user/quickstart.html

知识准备

HTTP 协议简介

什么是“协议”。协议是指两个或两个以上实体为了开展某项活动，经过协商后双方达成的一致意见。——百度百科

我们这里说的协议，通常指的是“通信协议”。

通信协议是指双方实体完成通信或服务所必须遵循的规则和约定。——百度百科

简单来说，通信协议就是一种语言，是两个实体之间为了完成互相通信，共同约定的一种双方都懂，都遵守的语言。

HTTP 协议，就是浏览器和 Web 服务器网站之间为了完成互相通信而共同约定的一种语言。下面的链接是关于 HTTP 协议的介绍，建议认真阅读。

```
http://www.runoob.com/http/http-tutorial.html
```

要牢记和理解与 HTTP 协议相关的几个概念：

- 目前常用的协议是 HTTP1.1，在 HTTP1.1 协议里有八种请求方法，其中最常用的是 GET 和 POST。
- HTTP 协议中有请求和响应两种报文，两种报文又大致分为报文头和报文体。
- 报文头用于传递一些通用的信息或者指定某种行为。
- 报文体实际上就是请求或者响应中传递的数据内容。

在报文头中，需要记忆以下几个字段名：

- Cookies，请求报文中，用于存放 Cookies 内容。
- Content-Type，表示后面的文档属于什么 MIME 类型，Servlet 默认为 text/plain，但通常需要显式地指定为 text/html。
- Location，表示客户应当到哪里去提取文档。
- Set-Cookie，设置和页面关联的 Cookie。

备注：MIME 类型是什么？

MIME(Multipurpose Internet Mail Extensions)多用途互联网邮件扩展类型。——百度百科

通俗地说，MIME 就是为了在发送邮件的时候，为方便地打开不同的内容而制定的一个标准。比如你发送的是图片、声音，还是一个 html 的文件，都可以通过指定不同的 MIME 类型来告诉收件方的。这样收件方就可以正确的处理这些收到的数据了。HTTP 协议中借用了这个标准，用于在客户端和服务端间进行内容类型的声明。

发送的 HTTP 报文

认识 HTTP 协议最直观的方法就是阅读实际的 HTTP 报文。

我个人比较喜欢使用 curl 这个小工具，下面是用 curl 向 http://httpbin.org/ip 发送一个 GET 请求的 HTTP 协议会话过程。右尖括号“>”开头的是发送的数据，左尖括号“<”开头的是接收的数据。如图 7.4 所示。

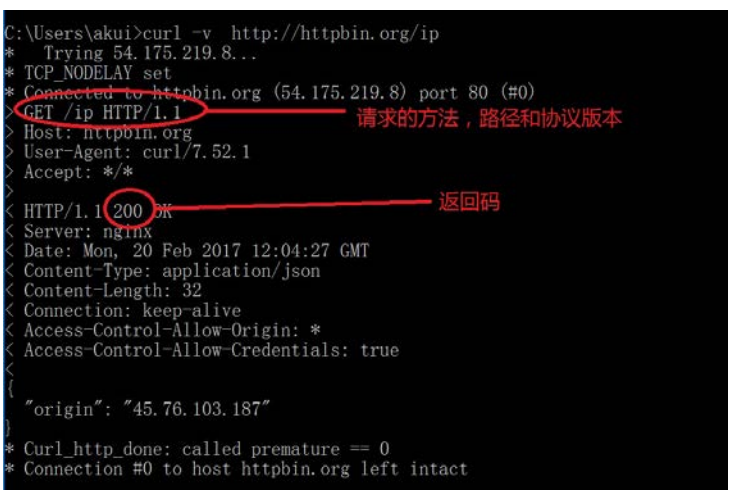


图 7.4 curl 输出解释

curl command line tool and library for transferring data with URLs。

curl 是一个用于通过 URLs 传递数据的命令行工具和库。

————来自 curl 官方网站：https://curl.haxx.se/

curl 是由瑞典黑客 Daniel Stenberg 创建和维护的，其个人网站为：https://daniel.haxx.se/。

curl 最常用的是通过命令行来给 Web 网站发送 HTTP 请求，并将 HTTP 报文内容在命令行界面输出。

可以通过以下两个地址下载 Windows 64 位版本的安装包。

http://www.paehl.com/open_source/downloads/curl_X64_ssl.7z

https://bintray.com/artifact/download/vszakats/generic/curl-7.54.1-win64-mingw.7z

这两个安装包均为绿色版本，解压缩后将 curl.exe 所在的目录加入 PATH，进入命令行模式即可使用。

curl 的官方下载链接：https://curl.haxx.se/download.html

curl 的详细介绍资料：https://www.gitbook.com/book/bagder/everything-curl/details

curl 的源代码 github 地址：https://github.com/curl/curl

Requests 库简介

Requests 是唯一的一个“非转基因的”Python HTTP 库，人类可以安全享用。Requests 允许你发送纯天然，植物饲养的 HTTP/1.1 请求，无需手工操作。

——Requests 官方网站：python-requests.org

Requests 库的确具有非常好的易用性和强大的功能，主要用于通过 Python 对 HTTP 接口进行操作。中文的官方网站请参见如下链接：

http://cn.python-requests.org/zh_CN/latest/user/quickstart.html

看完链接后，我们要明白如下内容要点：

- 发送 HTTP 请求，需要用到 Python 的第三方库，即 requests 库；
- 发送一个 GET 或者 POST 请求的方法，就是通过调用 requests.get()或者 requests.post()方法；

这两个方法有两个常用的参数，一个是 url，一个是 params，params 是可

选参数，比如如下代码中：

```
r = requests.get('https://github.com/timeline.json')
```

就是向地址为\'https://github.com/timeline.json \' 的 URL 发送一个 HTTP 协议的 GET 请求。

这里要注意，可以通过在 URL 后面增加问号(？)开头的字符链接给被请求的 URL 地址传递参数，问号(？)后面一般是一个 key=value 的字符串，如果后面需要连接多个 Key=Value 的参数，参数之间用与运算符(&)链接起来。比如：

```
r = requests.get("http://httpbin.org/get?name='akui'&email='a_kui@163.con'")
```

就是在向“http://httpbin.org/get”地址发送 GET 请求，并将 name=’akui’和 email=’a_kui@163.com’作为参数传递。

请读者在 PyCharm 中输入如下语句，观察一下输出的结果（注意电脑要联网）：

```
import requests
r = requests.get("http://httpbin.org/get?name='akui'&email='a_kui@163.con'")
print(r.text)
```

运行后，你将看到如下 JSON 格式的输出：

```
{
  "args": {
    "email": "'a_kui@163.con'",
    "name": "'akui'"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Connection": "close",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.13.0"
  },
  "origin": "61.148.242.253",
```

```
  "url": "http://httpbin.org/get?name='akui'&email='a_kui%40163.con'"
}
```

你看到的返回文本中有以下部分，

```
"args": {
   "email": "'a_kui@163.con'",
   "name": "'akui'"
 },
```

就是地址为“http://httpbin.org/get”的 HTTP 服务将 GET 请求中通过 url 传递的参数(name='akui' 和 email='a_kui@163.com')，以 JSON 格式返回了，而这些返回的内容都放在 requests.get()方法的返回值里。

说到返回值，无论是 requests.get()还是 requests.pos()方法，都会返回一个 response 对象。

Response 对象常用的属性和方法包括：

- Response.text：文本格式输出的返回报文内容。
- Response.json()：用 JSON 格式编码的返回报文内容，注意这个返回值是字典数据类型。
- Response.status_code：服务器响应的状态码，是一个数字类型。
- Response.headers：服务器响应头，数据类型为 requests.structures.CaseInsensitiveDict，本质上就是一个字典数据类型，不过是一个对大小写不敏感的字典数据类型。
- Response.Cookies：服务器响应的 cookies

可以尝试将上面代码 print(r.text)中的 text 改成 json()、status_code、headers，cookies 等，看一下输出的结果，感性认识一下 response 对象的这些属性和方法。

```
import requests
r = requests.get("http://httpbin.org/get?name='akui'&email='a_kui@163.con'")
print(r.text)
print(r.json())
print(r.status_code)
print(r.headers)
print(r.cookies)
```

Requests 库安装

要安装 requests 包，建议在联网的情况下，直接采用 pip 方式安装： pip install requests。如图 7.5 所示。

图 7.5 Install Requests

安装完成后，通过 Requests 发送 GET 请求到 http://httpbin.org/ip，检测是否能够成功的使用 Requests 库。

```
import requests
r = requests.get("http://httpbin.org/ip")
print(r.text)
```

通过 PyCharm 编辑运行上面的代码，会得到关于你的本机 IP 的信息。

```
{
 "origin": "61.148.242.253"
}
```

具体内容，你可以学习 Quickstart 的“发送请求”小节 http://cn.python-requests.org/zh_CN/latest/user/quickstart.html。

挑战问题

编写一个 Python 程序 getstatuscode.py，运行程序，打印出“请输入你要发送 HTTP 请求的 URL 链接：[http 链接的 url 地址]”；然后，程序向输入的链接发送 GET 请求，并打印返回的 status code；如果输入的 URL 不存在，则打印“你输入的 HTTP 请求地址：[http 链接的 url 地址]出现链接异常，请确认地址是否正确！”

如图 7.6～图 7.8 所示。

图 7.6 Get Status Code

图 7.7 Get Status Code

图 7.8 Get Status Code

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

难点提示

遇到网络问题（如：DNS 查询失败、拒绝连接等）时，Requests 会抛出一个 ConnectionError 异常，写代码的时候，写成 ConnectionError 如果不起作用，可能需要写成 requests.ConnectionError，想想为什么？

知识点

语言基础知识要点

（1） 任何时候调用 requests.get()或者 requests.post()方法，你都在做两件事情：

– 其一，你在构建一个 Request 对象，该对象将被发送到某个服务器请求或查询一些资源。

– 其二，一旦 requests 得到一个从服务器返回的响应就会产生一个 Response 对象。该响应对象包含服务器返回的所有信息，也包含你原来创建的 Request 对象。

（2） 本小节会用到 Python 的 except 处理，如有需要，请重新学习和体会 6.3 小节处理异常不要崩溃内容。

（3） json.loads()函数用于将一个 JSON 编码的字符串转换回一个 Python 数据结构。

（4） 字典结构在取用一个键对应值的时候，把相应的键字符串放入字典变量名后的方括弧内。详细内容请参考：http://www.runoob.com/python3/python3-dictionary.html

HTTP 协议基础知识要点

（1） HTTP 协议是 HyperText Transfer Protocol 的缩写，即超文本传输协议。

（2） HTTP 基于 TCP/IP 传输数据，默认端口号为 80 端口。

（3） GET 和 POST 是 HTTP 协议中最常用的两种向服务器端发送请求的方法。

（4） URI 和 URL 的区别：URI 是 uniform resource identifier 的缩写，即统一资源标识符；而 URL 是 uniform resource locator 的缩写，即统一资源定位符，可以简单地理解 URL 是一种特定的 URI 的具体实现。

（5）这里推荐一篇比较好的关于 HTTP 的文章：http://www.cnblogs.com/TankXiao/archive/2012/02/13/2342672.html，作者肖佳，英文名 Tank。

读 者 圈

第八章 交谈开始

8.1 我知道你是哪里人

学习目标

学习通过 GET 请求发送带参数的 HTTP 请求，复习 JSON 数据处理。

学习资源

http://cn.python-requests.org/zh_CN/latest/user/quickstart.html
http://cn.python-requests.org/zh_CN/latest/index.html
http://www.runoob.com/python3/python3-dictionary.html

知识准备

这个地址提供有 IP 地址的归属地信息的查询，
http://ip.taobao.com/service/getIpInfo.php
接口说明如下：

（1） 请求接口（GET）：/service/getIpInfo.php?ip=[ip 地址字串]。

（2） 响应信息：（JSON 格式的）国家、省（自治区或直辖市）、市（县）、运营商。

（3） 返回数据格式： {"code":0,"data":{"ip":"210.75.225.254","country": "e2d6fd", "area":"34e317", "region":"317eace02","city":"317eace02","county":"", "isp": "535fe1","country_id":"86","area_id":"100000","region_id":"110000","city_id": "110000", "county_id":"-1","isp_id":"100017"}}其中 code 的值的含义为，0：成功，1：失败。

以上内容引用自“淘宝 IP 地址库”网站，链接为：http://ip.taobao.com/instructions.php

挑战问题

编写一个 Python 程序 getipinfo.py，运行程序，打印出“请输入你要查询的 IP 地址：[本机的联网 IP 地址]”，输入 IP 地址并回车后，返回 IP 地址所在的国家（country）、地区（area）、省份（region）和城市（city）。

查询 IP 的地址信息，可以使用“淘宝 IP 地址库”网站的服务：http://ip.taobao.com/service/getIpInfo.php

如图 8.1 所示。

```
C:\Python\Python36\python.exe C:/MyWork/myBook/用Python做HTTP接口
请输入你要查询的IP地址：124.128.22.224

你输入的IP地址所在的国家：中国
你输入的IP地址所在的地区：华东
你输入的IP地址所在的省份：山东省
你输入的IP地址所在的城市：济南市

Process finished with exit code 0
```

图 8.1 Get IP Infomation

提示：不要短时间内大量发送这个请求，短时间发送大量请求会导致自己的 IP 被网站认为是恶意的，网站对该 IP 请求的响应时间会变长。

注意：以上问题，十分钟内就能够解决。不要一边看书或者查资料一边解决问题。如果第一次解决的过程中看书或者查资料了解决后，将编写的程序删除后重新做一次。

知识点

语言基础知识要点

（1） json.loads()函数用于加载一个 JSON 编码的字符串，将其进行转换后，返回一个 Python 的字典数据类型的值，具体可以参见 7.1 JSON 格式的通信录小节给出的提示和资料。

（2） 调用 requests.*()时返回的 Response 对象的 JSON 方法，当返回的报文体是 JSON 的时候使用，该方法直接返回对应的字典结构数据。

8.2 请查收我的 POST

学习目标

学习通过 POST 发送带数据的 HTTP 请求，复习 JSON 数据处理。

学习资源

http://cn.python-requests.org/zh_CN/latest/user/quickstart.html
http://cn.python-requests.org/zh_CN/latest/index.html
http://www.runoob.com/python3/python3-dictionary.html

知识准备

HTTPBIN.ORG

学习一个概念最好的方法就是实践。要学习 HTTP 协议，最好的方法就是与网站进行交互。这里推荐一个专门用于 HTTP 协议学习的网站：http://httpbin.org/

HTTPBIN 是一个覆盖所有 HTTP 场景的，专门用于测试 HTTP 代码库和深入了解、学习 HTTP 协议的网站。该网站的所有返回都采用 JSON 编码。

网站提供了很多有利于进行学习和研究的 URL 地址，这些地址读者都可以直接通过浏览器进行访问。

最简单的就是用于返回浏览器所在电脑的 IP 地址的：http://httpbin.org/ip。如图 8.2 所示。

图 8.2 httpbin 网站

通过浏览器访问 httpbin.org 网站上的/ip 位置，就会得到自己当前的 IP 地址信息，信息是以 JSON 格式编码的。

这里重点介绍一下 httpbin.org 网站上的/post，该 URL 会以 JSON 格式返回 HTTP 请求方通过 POST 请求发送给服务器的数据。如图 8.3 所示。

图 8.3 httpbin post in browser

我们发现通过浏览器直接访问 http://httpbin.org/post 的 URL 是不允许的，这是因为通过浏览器默认发送的是 GET 请求，而这个 URL 只支持 POST 请求。

用前文提到的 curl 命令，-d 选项就是通过 POST 请求向 URL 地址发送数据。

```
curl -d "name=akui&location=beijing" http://httpbin.org/post

{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "location": "beijing",
    "name": "akui"
  },
  "headers": {
    "Accept": "*/*",
    "Connection": "close",
    "Content-Length": "26",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "curl/7.54.1"
  },
  "json": null,
  "origin": "106.37.94.241",
```

```
  "url": "http://httpbin.org/post"
}
```

我们看到地址为 http://httpbin.org/post 的 HTTP 服务，在接收上送的数据("name=akui&location=beijing")之后，上送数据被作为“form”键的值元素，以 JSON 格式返回。

挑战问题

编写一个 Python 程序 sendpost.py，运行程序，打印出“请输入你的姓名：”；输入姓名并回车后，打印出“请输入你邮箱：”；输入邮箱并回车后，向网址 http://httpbin.org/post 发送如下定义的数据，data={"name":[输入的姓名],"email":[输入的邮箱]} 并将返回的状态码和 JSON 美化数据打印出来。

如图 8.4 所示。

```
C:\Python\Python36\python.exe C:/MyWork/myBook/用Python做HTTP接口测试/source/
请输入您的名字：阿奎
请输入您的邮箱：a_kui@163.com
返回的状态码： 200
返回的内容为：
{
    "args": {},
    "data": "",
    "files": {},
    "form": {
        "email": "a_kui@163.com",
        "name": "阿奎"
    },
    "headers": {
        "Accept": "*/*",
        "Accept-Encoding": "gzip, deflate",
        "Content-Length": "45",
        "Content-Type": "application/x-www-form-urlencoded",
        "Host": "httpbin.org",
        "User-Agent": "python-requests/2.13.0"
    },
    "json": null,
    "origin": "61.148.242.20",
    "url": "http://httpbin.org/post"
}
```

图 8.4 Send Post

注意输入的名字要支持中文，下面也要能够显示中文名字。返回的文本中数据应该出现在 form 节点。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

（1） Requests 库中 post 方法的 json、data、params 三个参数非常重要，请读者通过实践，理解三个参数使用时有什么不同。

（2） Requests 库中 post 或者 get 方法返回的 Response 对象中的 content，text 属性和 json()方法，分别会以不同的形式返回响应报文中的内容。

- Content 属性：会以字节的方式返回响应报文中的内容。
- Text 属性：会以 unicode 的方式返回响应报文中的内容。
- json()方法：会以 JSON 编码的方式返回响应报文的内容。如果响应报文体里没有合法的 JSON 内容，会出现异常（Exception），异常类型是 **ValueError**。

（3） Content-Type 的含义，特别是 application/x-www-form-urlencoded、multipart/form-data 和 application/json 这几种常用的 Content-Type，建议读者熟悉并牢记。

–application/json 用来告诉服务端消息主体是序列化后的 JSON 字符串。

–application/x-www-form-urlencoded 在使用表单提交数据的时候会用到。

- multipart/form-data 在使用表单上传文件的时候会用到。

（4） 这里推荐一篇比较好的关于 Content-Type 的含义的文章：https://imququ.com/post/four-ways-to-post-data-in-http.html，作者 Jerry Qu。

拓展

想办法让上送的数据分别出现在 **args 节点**和 **data 节点**。

8.3 厉害了，我的 302

学习目标

学习 redirect 概念，对 302 状态码有更深入的理解。

学习资源

http://httpbin.org
http://cn.python-requests.org/zh_CN/latest/user/quickstart.html#id9

知识准备

重定向（Redirect）

要想理解本小节的内容，首先需要理解 302 状态码和 redirect 的概念。

“3XX 响应结果表明浏览器需要执行某些特殊的处理以正确处理请求。”——引用自《图解 HTTP》P56

所有 3XX 类的状态码都是指示浏览器接到响应后要进行后续处理的，一般情况下这个所谓的特殊处理就是重新发送请求到服务器的另一个 URL 路径，实际上就是重定向（redirect）。

3XX 的状态码有 301，302，303，一般 302 使用的最多。当浏览器发送一个 POST 或者 GET 请求后，如果收到了来自服务器带有 302 返回码的响应，浏览器会根据响应报文头中的 Location 域中给出的地址重新发送 GET 请求。

比如，我们向 http://httpbin.org/redirect/3 的地址发送一个 GET 请求。

会得到如图 8.5 所示的访问结果。

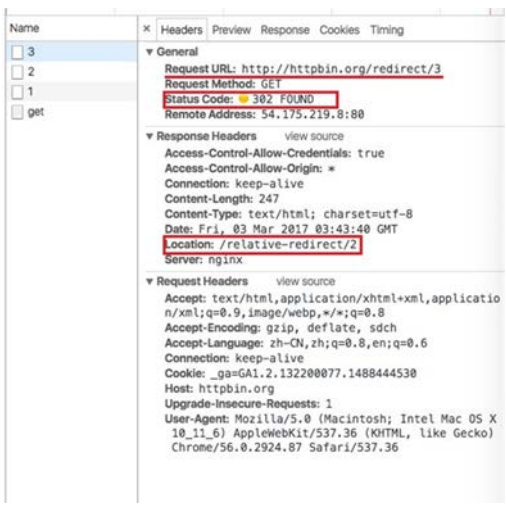


图 8.5 Redirect 跳转第一次

在图 8.5 的左侧，可以看出在浏览器和服务器之间一共有 3—>2—>1—>get 四次请求和相应的交互发生。

通过上图还可以看到：

- 请求的 URL，即 Request URL 为：http://httpbin.org/redirect/3。
- 响应码，即 Status Code 为：302。
- 响应报文头（Response Headers）中位置，即 Location 为：/relative-redirect/2。

说明浏览器发送了包含 URL 为 http://httpbin.org/redirect/3 的请求给域名为 httpbin.org 的 Web 服务器，服务器响应报文中给出的状态码为 302，Location 为“/relative-redirect/2”。浏览器收到这样的包含 302 状态码的响应报文，会向 Location 域中的地址重新发送 GET 请求。此时就出现了如图 8.6 所示的报文。

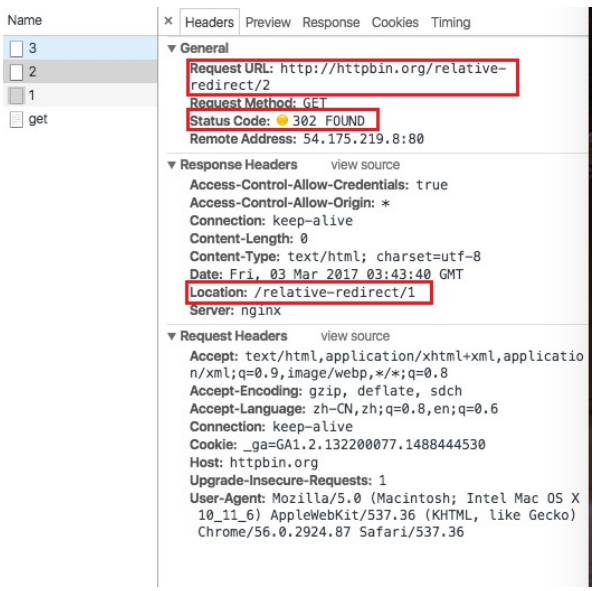


图 8.6 Redirect 跳转第二次

注意，这次交互的 Request URL、Status Code 和 Location 发生变化。如图 8.7 所示，是最后的 get 交互的报文内容。

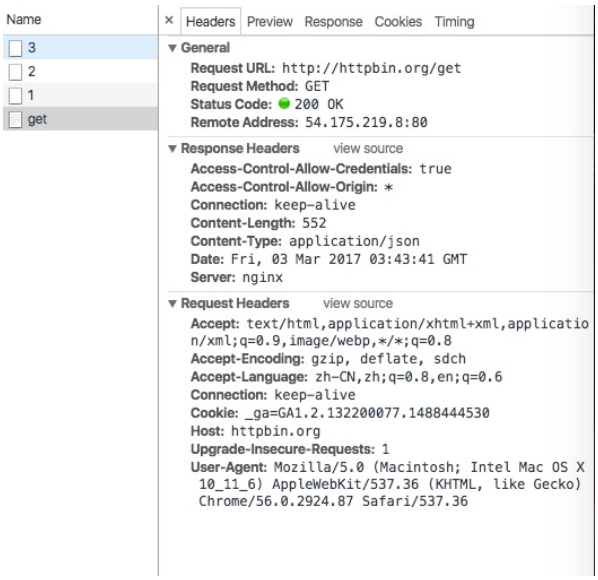


图 8.7 Redirect 跳转到 Get 请求

上面是四次交互中的三次的报文内容，分别是 3、2、get。请你写出 name 为 1 的倒数第二次交互的 Request URL、Status Code 和 Location 的内容。

Request URL：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

Status Code：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

Location：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

如果写不出来，请重新阅读一遍上面的内容，不建议直接进入下面小节的问题，因为那将对你真正的掌握 302 状态码以及对应的 HTTP 交互机制没有什么帮助。

Requests 库中的 Response 对象

要熟练使用 Requests 库，非常重要的一点就是要了解 Response 对象。

我们通过一个实验了解一下 Response 对象，注意，运行如下代码的时候要保持联网状态。

```
>>> import requests
>>> r = requests.get('http://github.com')  #向 github 网站，发送 GET 请求
>>> type(r)      #get()方法的返回值是一个 Response 类
<class 'requests.models.Response'>
```

```
>>> r
'<Response [200]>'
>>> type(r.url)   #Response 类的 url 属性是一个字符串
<class 'str'>
>>> r.url
'https://github.com/'
>>> type(r.status_code)  #Response 类的 status_code 属性是一个整数
<class 'int'>
>>> r.status_code
200
>>> type(r.headers)      #Response 类的 headers 属性是一个大小写不敏感的字典类型
<class 'requests.structures.CaseInsensitiveDict'>
>>> type(r.history)      #Response 类的 history 属性是一个列表类型
<class 'list'>
>>> str(r.history)       #history 属性的列表中有一个状态码为 301 的 Response 对象
'[<Response [301]>]'
>>> type(r.history[0])
<class 'requests.models.Response'>
```

通过上面的实验我们发现：

- Response 类的 headers 属性是一个对大小写不敏感的字典类型。(requests.structures.CaseInsensitiveDict)
- Response 类的 history 属性是一个列表类型，里面放的是 Response 对象。从 Requests.get()或者 Requests.post()方法发出请求后，到最后一次跳转之间的所有的重定向交互过程中的每一个响应信息，都有一个与之对应的 Response 对象存储在 history 属性中。

让我们验证一下：

```
>>> import requests
>>> resp = requests.get("http://httpbin.org/redirect/3")
>>> resp
<Response [200]>
>>> resp.history
[<Response [302]>, <Response [302]>, <Response [302]>]
```

我们可以看到三次跳转的 Response 对象都在 history 列表中。

挑战问题

编写一个 Python 程序 redirect.py，运行程序，打印出“请输入重定向跳转的次数（1-10 之间的整数）”，输入数字回车后，程序打印出通过 requests.get 方法向 http://httpbin.org/redirect/[输入的数字] 发送 GET 请求后得到状态码：200，并且获取这个请求中每一次跳转的 Location 并分别打印出来。如图 8.8 所示。

图 8.8 Redirect

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

（1） 理解 HTTP 交互中的 redirect 概念，以及 302 状态码。

（2） requests.get 或者 requests.post 方法返回的是一个 Response 对象。

（3） Response 对象的 headers 属性是一个对大小写不敏感的字典数据类型。

（4） Response 对象的 history 属性是一个包含了各次跳转的 Response 对象的 list 类型数据。

拓展

看到这里，读者是不是很想知道知识准备中的那些图是怎么得到的？如果有兴趣，建议自己学习一下，这对于今后进行面向 Web 站点的 HTTP 接口测试很有帮助。建议使用 Chrome 或者 FireFox 浏览器，上文中的图皆是在 Chrome 浏览器“开发者工具”中截取的，进入开发者工具界面的菜单见图 8.9 所示。

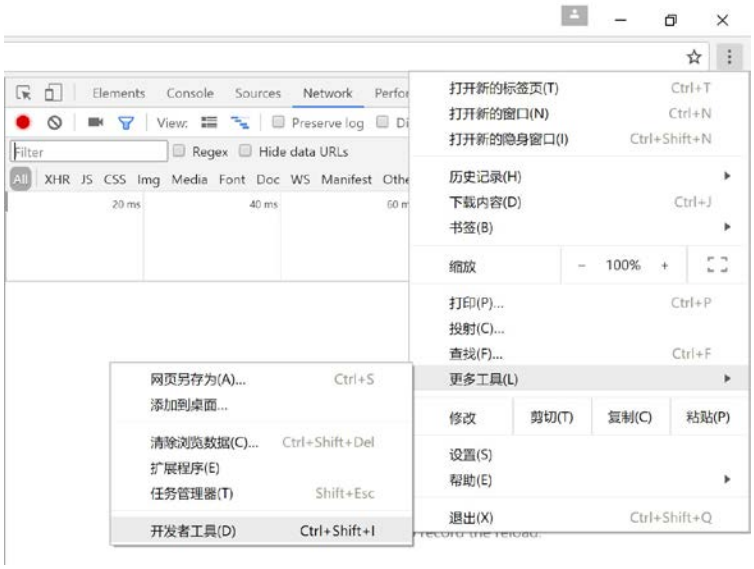


图 8.9 Chrome 开发者工具

8.4 把我藏在 Cookies 里

学习目标

学习 Cookies 的概念，巩固重定向 redirect。

学习资源

http://cn.python-requests.org/zh_CN/latest/user/quickstart.html
http://cn.python-requests.org/zh_CN/latest/index.html

http://cn.python-requests.org/zh_CN/latest/user/quickstart.html#cookie

知识准备

本节我们要学习 Cookie，有的时候用其复数形式 Cookies，意思相同。

Cookie（复数形态 Cookies），中文名称为“小型文本文件”或“小甜饼”，指某些网站为了辨别用户身份或者存储用户相关信息而储存在用户本地终端（Client Side）上的数据（通常经过加密），定义于 RFC2109。是网景公司的前雇员卢·蒙特利（英语：Lou Montulli）在 1993 年的发明。——整理改编自《必应网典》

Cookie 有以下特点：

- 保存在客户端，一般由浏览器负责存储在本地。
- 通常是加密存储的，不过由于存储在本地，很难保证数据不被非法访问，所以 Cookie 中不宜保存敏感信息，如密码等。
- 哪些信息需要作为 Cookie 保存在客户端本地，保存多长时间，一般是由服务器决定的，在 HTTP 协议中通过服务器返回的响应报文头中，有一个 Set-Cookie 域用来指示浏览器或者其他客户端，在本地保存 Cookie 信息。
- Cookie 保存在客户端本地的目的是为了下次访问网站的时候，可以直接调取，上送服务器。在 HTTP 协议中通过客户端发送给服务器的请求报文头中，有一个 cookies 域专门用于存放这个信息，以便客户端将 Cookie 信息发送给服务器。

首先介绍一下 httpbin.org 网站上关于 cookies 的三个网址以及功能：

http://httpbin.org/cookies

以 JSON 格式返回客户端上送到服务器的请求报文中 cookies 的内容。

```
curl -b "name=akui;location=beijing" http://httpbin.org/cookies
{
  "cookies": {
    "location": "beijing",
    "name": "akui"
  }
}
```

以上是通过 curl 向 http://httpbin.org/cookies 发送一个 HTTP 的 GET 请求，并且在请求中设定 cookies 为“name=akui;location=beijing”，注意两个 key=value 之间通过一个半角分号分隔。

curl 命令行工具的“-b”参数用于指示后面的字符串或者文件是需要随 HTTP 请求一起发送的 Cookie 信息。

curl 命令行工具的“-v”参数，用于以对话的方式将请求发送的 HTTP 报文和服务端的响应 HTTP 报文的详细信息都显示出来。

注意：右尖括弧(“>”)后面是发送到服务端的请求信息的说明，左尖括弧(“<”)后面是接收的服务端响应信息的说明。

```
curl -v -b "name=akui;location=beijing" http://httpbin.org/cookies
*   Trying 23.23.112.149...
* TCP_NODELAY set
* Connected to httpbin.org (23.23.112.149) port 80 (#0)
> GET /cookies HTTP/1.1
> Host: httpbin.org
> User-Agent: curl/7.54.1
> Accept: */*
> Cookie: name=akui;location=beijing
>
< HTTP/1.1 200 OK
< Connection: keep-alive
< Server: meinheld/0.6.1
< Date: Thu, 03 Aug 2017 07:12:49 GMT
< Content-Type: application/json
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: true
< X-Powered-By: Flask
< X-Processed-Time: 0.000571012496948
< Content-Length: 70
< Via: 1.1 vegur
<
{
  "cookies": {
    "location": "beijing",
```

```
    "name": "akui"
  }
}
* Connection #0 to host httpbin.org left intact
```

http://httpbin.org/cookies/set?name1=value1&name2=value2

返回 302 状态码，并且在返回报文头中，Location 指向/cookies。返回报文头中通过 Set-Cookie 域指定客户端在本地保存的 Cookie 信息，而指定要保存的信息就是请求报文中通过 URL 参数传递给服务器端的 name1=value1&name2=value2。想想浏览器或者网络客户端接收到这样的 302 状态码的响应报文会怎么做？

http://httpbin.org/cookies/delete?name1&name2 返回 302 状态码，并且在返回报文头中，Location 指向/cookies，返回报文头中通过 Set-Cookie 域分别指定 name1 和 name2 等于“空”，即指示客户端在本地保存的 Cookie 信息中删除 name1 和 name2。想想浏览器接收到这样的 302 状态码的响应报文会怎么做？

关于/set 和/delete 两个功能，请大家先了解和思考，详细内容将在下一个小节具体解释。

挑战问题

编写一个 Python 程序 showcookies.py，运行程序，打印出“请输入一组 cookie，格式为 key=value，直接回车表示输入结束：”，然后输入多组 key=value 格式的字符串作为上送的 Cookie 内容，最后一行回车后，程序将输入的多行内容作为一组 Cookie 上送 http://httpbin.org/cookies，并将返回的 JSON 格式的报文内容，以美化的 JSON 格式输出。

如图 8.10 所示。

图 8.10 Show Cookies

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

（1） Requests 库中 post、get 方法都可以跟一个 cookies 参数，本题目中就是要求将客户输入的键值通过该参数作为 Cookie 传递给 http://httpbin.org/cookies 这个 URL，并打印返回的结果。

（2） 传递给 cookies 的值是字典类型。

（3） 字符串的 split()内建函数可以根据特定的字符将字符串分隔成列表 list。

（4） 字符串的 strip()内建函数可以去掉字符串的空格。

（5） 字符串和列表数据结构都有 len()方法，分别返回字符串的长度和列表中元素的个数。

拓展

如果将上面题目中发送的 URL（http://httpbin.org/cookies）改成 http://httpbin.org/cookies/set?name=aaa，会发生什么？

观察如果“？”后跟的 cookies 的 key 值和通过界面输入相同的时候会出现什么情况？

如果将上面题目中发送的 URL（http://httpbin.org/cookies）改成 http://httpbin.org/cookies/delete?name=aaa，会发生什么？

观察如果“？”后跟的 cookies 的 key 值和通过界面输入相同的时候会出现什么情况？

尝试解释一下为什么？

8.5 让我们“保持通话”

学习目标

学习 session 的概念，巩固重定向以及 cookies 知识。

学习资源

http://cn.python-requests.org/zh_CN/latest/user/advanced.html#session-objects

知识准备

Session 简介

HTTP 是一个无状态的协议，所谓无状态的协议，就是第一次发送的请求与第二次发送的请求是独立的，第二个请求并不能继承上一个请求的处理状态。比如，你通过用户名和密码登录到一个网站上，很自然的希望在这个网站上进行其他的操作时，网站能够识别你是谁，并且记住你已经登录过了。可是，HTTP 协议本身并不能做到这一点。

上一节我们知道了 Cookies 的工作机制是用户识别和状态管理（比如保存用户的用户名、邮箱、地址等等），Web 网站为了管理用户的状态，通过 Web 浏览器把一些数据，作为 Cookies 信息临时写入用户的计算机内。当用户再次访问该 Web 网站时，会将之前保存的 Cookies（假如没有过期的话）取出来，发送给该 Web 网站。以此实现了用户识别和状态的管理。这种机制有两个缺点：

（1） 数据保存在客户端本地，安全性不高；

（2） 每次访问都要发送保存的 Cookies 数据，当网络访问量大的时候，会浪费网络带宽。

针对这两个问题，在 HTTP 协议之外，当我们进行 Web 网站的设计时，就需要引入 Session 的概念。

Session，中文一般翻译成“会话”，也是一种管理用户状态和信息的机制。与 Cookies 将数据保存在客户端本地不同的是，Session 的数据保存在服务器端，一般放在服务器的内存里。客户端和服务端通过一个 SessionID 来进行沟通，为了防止不同的客户之间出现冲突和重复，这个 SessionID 一般是一个较长的随机字符串（一般 32 或者 48 个字节）。

客户登录 Web 系统后，Web 系统会通过响应报文，返回给客户端一个 Key-Value 格式的 SessionID，当客户端第二次通过对 Web 系统发起请求的时候，只要带上这个 SessionID，Web 系统的服务器端就能够通过这个 SessionID 在内存中找到这个客户端对应的客户的信息和状态，比如用户名、地址、邮箱

甚至购物车里的商品，当前的支付状态等等。这样就实现了对客户的识别和对客户状态的管理。

通过对 Session 的介绍，我们可以看到，这种设计有效地解决了 Cookies 机制的两个问题，即通过数据保存在服务器端，解决了安全性问题；通过传递一个 SessionID，解决了浪费网络带宽的问题。

但是，Session 机制也有其缺点：比如对服务器内存的消耗，特别是并发用户量大的时候；另外，由于一般保存在服务器的内存里，如果 Web 服务器是多个节点的分布式系统，就需要进行一些特殊的设计和处理才能实现 Session 内容在多个节点之间的共享。

Cookie 补遗

继续 8.4 把我藏在 Cookies 里小节关于 http://httpbin.org/cookies/set 和 http://httpbin.org/cookies/delete 的话题。

这两个 URL 的设计很好地演示了 Cookie 的应用，下面详细的解释一下/set，而/delete 的部分由读者自行学习。

首先，我们看一下通过 Chrome 浏览器直接输入网络地址的交互过程。

如图 8.11 所示，在地址栏输入：

http://httpbin.org/cookies/ set?name1=value1&name2=value2

图 8.11 通过浏览器执行 set cookies

浏览器以 JSON 格式返回 Cookie 信息，注意，此时浏览器地址栏中的地址已经发生了变化。

到底发生了什么？让我们用 curl 来看一下。

下面的 curl 命令中：

- -c 参数指定一个用于临时存放和输出 Cookie 信息的文件，文件名可以

自己定义。

- -L 参数用于告诉 curl 见到 redirect 返回码的时候，要进行后续处理，也就是要继续进行新的地址的请求。
- -v 用于以对话的方式显示交互信息 > 表示发送， < 表示接受。

另外，http://httpbin.org/cookies/set?name1=value1&name2=value2 要用双引号引起来，因为 & 符号在 URL 中用于作为要传递给请求地址的 key=value 参数对的分隔符，而在命令行中也有特殊的含义的，所以一定要用双引号引起来，这样就会被作为一个完整的字符串传递给 curl 处理。

```
curl -c cookies.txt -Lv "http://httpbin.org/cookies/set?name1=value1&name2=value2"
*   Trying 23.23.134.171...
* TCP_NODELAY set
* Connected to httpbin.org (23.23.134.171) port 80 (#0)
> GET /cookies/set?name1=value1&name2=value2 HTTP/1.1
> Host: httpbin.org
> User-Agent: curl/7.54.1
> Accept: */*
>
**#接收到一个 302 返回码的响应报文**
< HTTP/1.1 302 FOUND
< Connection: keep-alive
< Server: meinheld/0.6.1
< Date: Thu, 03 Aug 2017 12:46:53 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 223
**#报文中指定了重定向(redirect)的地址为/cookies**
< Location: /cookies
* Added cookie name1="value1" for domain httpbin.org, path /, expire 0
**#响应报文中通过 Set-Cookie 指定客户端要保存 name1=value1 的 cookie 信息**
< Set-Cookie: name1=value1; Path=/
* Added cookie name2="value2" for domain httpbin.org, path /, expire 0
**#响应报文中通过 Set-Cookie 指定客户端要保存 name2=value2 的 cookie 信息**
< Set-Cookie: name2=value2; Path=/
< Access-Control-Allow-Origin: *
```

```
< Access-Control-Allow-Credentials: true
< X-Powered-By: Flask
< X-Processed-Time: 0.00105214118958
< Via: 1.1 vegur
<
* Ignoring the response-body
* Connection #0 to host httpbin.org left intact
* Issue another request to this URL: 'http://httpbin.org/cookies'
* Found bundle for host httpbin.org: 0x1a00d68ef90 [can pipeline]
* Re-using existing connection! (#0) with host httpbin.org
* Connected to httpbin.org (23.23.134.171) port 80 (#0)
**#根据响应报文中 Location 指定的位置，重新发送 GET 请求到/cookies**
> GET /cookies HTTP/1.1
> Host: httpbin.org
> User-Agent: curl/7.54.1
> Accept: */*
**#发送的报文中携带了上面的响应报文中要求客户端保存的 cookie 信息**
> Cookie: name1=value1; name2=value2
>
**接收到一个 200 返回码的响应报文**
< HTTP/1.1 200 OK
< Connection: keep-alive
< Server: meinheld/0.6.1
< Date: Thu, 03 Aug 2017 12:46:54 GMT
< Content-Type: application/json
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: true
< X-Powered-By: Flask
< X-Processed-Time: 0.00096607208252
< Content-Length: 69
< Via: 1.1 vegur
<
**#响应报文的报文体中以 JSON 格式返回了第二次请求中携带的 cookies 信息**
{
  "cookies": {
```

```
    "name1": "value1",
    "name2": "value2"
  }
}
* Connection #0 to host httpbin.org left intact
```

至此，/set 的完成处理过程读者已经有了了解。

读者可以自己动手，研究一下/delete 的详细交互过程，尝试回答 8.4 把我藏在 Cookies 里 小节，扩展部分提到的问题。

Session 与 Cookie 的对比

从功能上，Session 和 Cookie 都是用来临时存储来访者信息的 HTTP 协议，是一个无状态协议，仅仅凭借 HTTP 协议的内容，网站无法知道来自该用户的再一次访问之前已经做了用户名和密码的登录，并且已经通过验证，登录成功了。

这个时候，就用到了 Session 和 Cookie。

1. Cookie 的方案

当用户通过用户名、密码验证登录成功后，网站会返回给客户端，即浏览器，一个 Cookie 信息，里面有一个标志用户已经登录的特殊的值，比如："login=Yes;UserID=123456;",浏览器会将这个 Cookie 信息保存在用户的本地某个磁盘位置。当用户第二次访问该网站时，浏览器会带着这个“标志用户已经登录的特殊的值”，比如之前保存的字符串："login=Yes;UserID=123456;"，通过 HTTP 报文头的 Cookie 域，上送到网站的服务端，此时，服务端检测到上送的报文头中的 Cookie 域中包含了这个“标志用户已经登录的特殊的值”便知道，这个用户是已经登录过的用户，这样，就相当于在第二次访问的时候，知道了用户的当前状态。这样 Cookie 在这里就完成了用来临时存储来访者信息的一个功能。通过这种临时存储来访者信息的功能，完成了两次 HTTP 请求之间的状态转移。如图 8.12 所示。

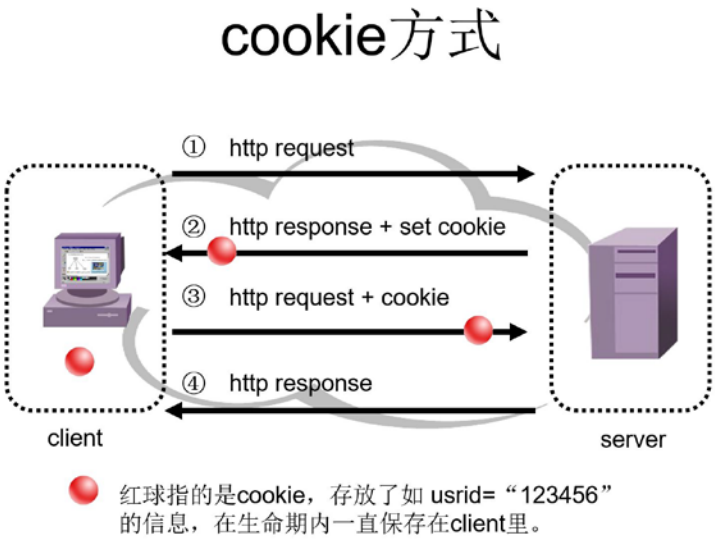


图 8.12 Cookie 方式

这里我们很容易想到一个问题，Cookie 是保存在用户本地的，信息的安全性很难保证，更不用说还可能存在假冒请求的情况出现。

2. Session 的方案

当用户通过用户名、密码验证登录成功后，网站会返回给客户端浏览器一个 SessionID 信息，一般是一个长字符串，比如："sessionid=2sfs3221fe34324jk3243232k4j3"。当用户第二次访问网站的时候，浏览器会带着这个“SessionID”信息上送请求到网站的服务端，此时，服务端检测到上送的报文中有这个“SessionID”信息，就通过这个“SessionID”信息到自己内存中的 Session 存储的位置中查找是否存在同样信息，如果存在这样的“SessionID”信息，就会通过这个“SessionID”信息进而找到其他的会话信息比如：UserID=123456 等，这样就会判断出这个用户是一个已经登录过的用户，相当于知道了用户的当前状态。而 Session 在这里就完成了用来临时存储来访者信息的功能，而“SessionID”信息就相当于存储在服务端内存中的用户信息的一个索引号。如图 8.13 所示。

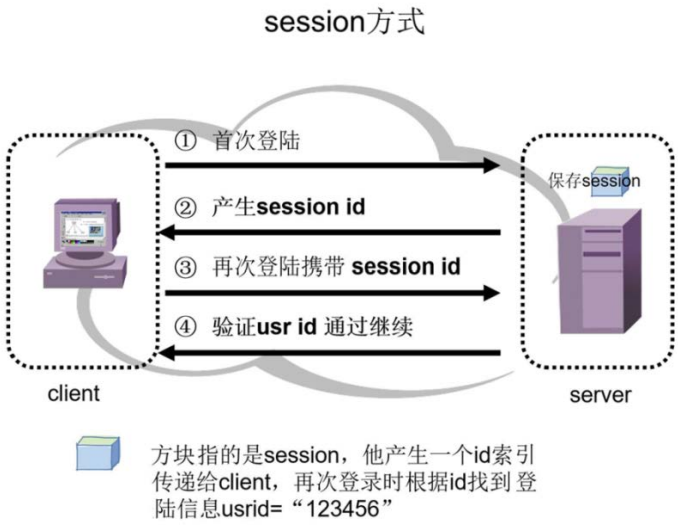


图 8.13 Session 方式

通过上面的关于 Cookie 和 Session 的描述，可以看出：

- Session 和 Cookie 都可以实现临时存储用户信息的功能，存储的目的也都是为了实现两个 HTTP 请求之间的数据共享。
- Cookie 将数据保存在用户端，通过第二次将保存的数据直接上送来实现两个 HTTP 请求之间数据的共享；Session 将数据保存在服务端，通过一个 Sessionid 作为索引，并将该索引传递给用户端，通过第二次上送这个索引来实现数据在两个 HTTP 请求之间的共享。
- Cookie 存储在客户端，数据的安全性不高。
- Cookie 存储在客户端，每次上送都要将所有需要在两次 HTTP 请求之间共享的数据全部上送，当交易量较大的时候，会占用网络带宽。
- 由于 Session 存储在服务端，并且一般在内存中，当用户在线数量较大的时候，会占用比较多的内存空间，对服务端的内存要求比较高。

在 Python 中使用 Session

在 Python 中使用 Session 非常简单，通过 requests.session()来获取一个 session 的句柄，然后，向使用 requests 一样使用新获取的 session 对象就可以了。

我们来通过实例代码体会一下 session 下的 get()方法和 requests 本身的 get()方面有什么不同。

```
>>> import requests
#调用 requests 的 get()方法，向/set 发送 GET 请求，设置 cookies，并输出返回结果
>>> result = requests.get("http://httpbin.org/cookies/set?name=akui")
>>> print(result.text)
{
  "cookies": {
    "name": "akui"
  }
}
#第二次调用 requests 的 get()方法，向/cookies 发送 GET 请求，获取当前的 cookies，并输出返回结果，发现刚才设置的 cookies 信息不见了。
>>> result = requests.get("http://httpbin.org/cookies")
>>> print(result.text)
{
  "cookies": {}
}
#======================我是分割线============================
#通过 requests.session()获取一个 session 对象
>>> session = requests.session()
#调用 session 对象的 get()方法，向/set 发送 GET 请求，设置 cookies，并输出返回结果
>>> result = session.get("http://httpbin.org/cookies/set?name=akui")
>>> print(result.text)
{
  "cookies": {
    "name": "akui"
  }
}
#第二次调用 session 对象的 get()方法，向/cookies 发送 GET 请求，获取当前的 cookies，并输出返回结果，发现刚才设置的 cookies 信息仍然是存在的。
>>> result = session.get("http://httpbin.org/cookies")
>>> print(result.text)
{
  "cookies": {
    "name": "akui"
  }
}
>>>
```

这就是神奇的 session 的作用，可以保持多个通过 session 对象发起的 HTTP 请求之间的状态连续，平滑实现跨 HTTP 请求的数据共享。

挑战问题

编写一个 Python 程序 playcookies.py，运行程序，打印出“请输入 cookies 指令：add key=value，用于增加 cookies del key，用于删除 cookies show，用于显示当前的 cookies quit，退出”。

当输入 add 指令，后面应该跟着 key-value 键值对，程序向 http://httpbin.org/cookies 网址的“/set?key=value”路径发送 get 请求，并以美化的 JSON 格式将返回报文内容输出。注意：key-value 要用从终端输入的实际内容替换。

当输入 del 指令，后面应该跟着 key，程序向 http://httpbin.org/cookies 网址的“/delete?key”路径发送 get 请求，并以美化的 JSON 格式将返回报文内容输出。注意：key 要用从终端输入的实际内容替换。

当输入 show 指令，程序向 http://httpbin.org/cookies 网址发送 get 请求，并以美化的 JSON 格式将返回报文内容输出。当输入 quit 指令的时候，退出程序。

如图 8.14 所示。

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /
请输入cookies指令:
add key=value , 用于增加cookies
del key       , 用于删除cookies
show          , 用于显示当前的cookies
quit          , 退出
add name=akui
{
    "cookies": {
        "name": "akui"
    }
}
add email=a_kui@163.com
{
    "cookies": {
        "email": "a_kui@163.com",
        "name": "akui"
    }
}
del name
{
    "cookies": {
        "email": "a_kui@163.com"
    }
}
show
{
    "cookies": {
        "email": "a_kui@163.com"
    }
}
quit

Process finished with exit code 0
```

图 8.14 Play Cookies

请读者通过对函数或者类的设计，让程序使用不超过 2 个 print()。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

（1） requests 库中的 session，会话对象让你能够跨请求保持某些参数。

（2） 细读并理解：

http://docs.python-requests.org/zh_CN/latest/user/advanced. html#advanced，的“会话对象”小节。

拓展

“在不考虑输入错误的情况下，实际代码行数不超过 30 行 60 列”，看看代码是不是满足这个要求？

不考虑上面的约束，看看满足如下条件应该如何修改代码：

（1） add 后面跟两个空格的时候会怎么样？如何处理？

（2） 加入如下特性：输入的指令不是 add、del、show、quit 的时候，不做任何处理重新打印开头的提示。

读 者 圈

第四部分 实践 HTTP 接口测试

第九章 先要测起来

9.1 认识自动化测试

学习目标

了解自动化测试的类型、意义和 xUnit 测试框架。

学习资源

https://martinfowler.com/bliki/TestPyramid.html

知识准备

自动化测试的意义

自动化测试，简单来说就是将软件测试工作由手工变成自动化进行，自动化测试和手工测试是两个平行的概念。从理论上讲，所有软件测试工作都应该进行自动化测试方面的改造。能够自动进行软件测试工作自然是再好不过的事情。但是，事实并非如此。纵观整个软件测试行业的现状，手工测试依然在实际的软件开发实施工作中占据了很大的比例。

自动化测试和手工测试的适应场景和特点。

（1） 自动化测试案例可以低成本、快速的反复运行。

对于那些需要反复进行手工验证的场景，通过进行自动化测试的改造，可以有效地节省成本，缩短验证的周期。在瀑布模型的软件开发生命周期中，功能测试阶段后期经常需要进行的**回归测试**，就是自动化测试的一个典型应用场景。

自动化测试案例的编写和维护成本较高。自动化测试案例的编写需要测试开发人员掌握一定的工具使用技能，最好再具备一定的编程技能，这就导致了自动化测试案例的编写和维护的成本较高。如果一个案例只需要运行很少的几次，自动化测试的改造便失去了价值，比如对于一个临时性功能（投产后，

短期启用后就会下线）的测试。

（2） 自动化测试可以做手工做不了、不好做的测试。

在实际开发中有很多场景手工测试并不擅长，这时也是自动化测试发挥威力的时候。典型的场景有：

- 针对接口级别的测试。
- 针对一些特殊的输入数据（比如不可见的字符，或者二进制数据）的测试。
- 测试的输入数据和需要检验的输出数据特别多的时候（比如针对报表的测试）。

（3） 自动化测试不能彻底的替代手工测试。

虽然自动化测试相对于手工测试存在着诸多的优点，但是，自动化测试并不能完全的替代手工测试，手工测试作为一种测试方法，并不会消失。

虽然，自动化测试并不能彻底的替代手工测试，但是，自动化测试作为一种高效、快速和对测试人员要求更高的软件测试方法，已经越来越多的被各个行业的软件开发组织所采纳。对于软件测试人员，特别是从事手工测试的软件测试人员来说，掌握自动化测试的相关技术已经成为一种基本的技能要求。如图 9.1 所示。

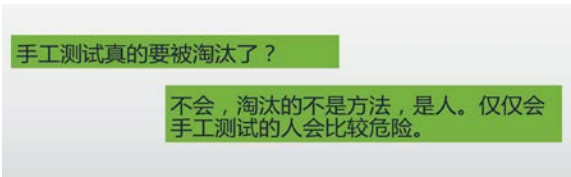


图 9.1 Test Automation

自动化测试的类型

说到自动化测试的类型，不得不提到 Martin Fowler 关于测试金字塔的论述。

“The test pyramid is a concept developed by Mike Cohn, described in his book Succeeding with Agile. Its essential point is that you should have many more low-level unit tests than high level end-to-end tests running through a GUI.”

“测试金字塔是由 Mike Cohn 提出的一个概念，在他的书《Succeeding with Agile》（中文版：《Scrum 敏捷软件开发》）中有详细的描述。测试金字塔的重点是相对于高层次的通过驱动界面执行的端到端的自动化测试，你应该更多的编写低层次的单元测试级别的自动化测试。”

——Martin Fowler，测试金字塔，https://martinfowler.com/bliki/TestPyramid.html

测试金字塔如图 9.2 所示。

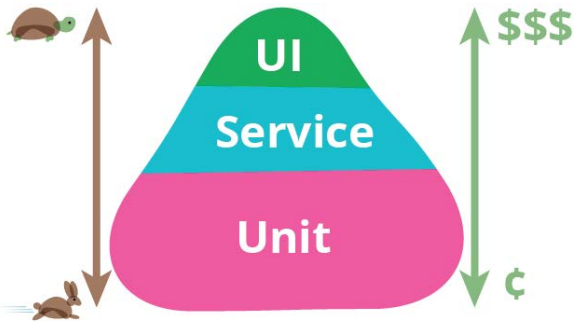


图 9.2 测试金字塔

注释：UI：界面测试，Service：服务测试，Unit：单元测试

测试金字塔模型主要聚焦于自动化测试的验证层次以及在进行自动化案例编写和资源投入的时候，应该如何将精力在各个验证层次之间分布。

一个更细化的测试金字塔的模型来自 Alister Scott，他在 2012 的一篇文章《Introducing the software testing ice-cream cone (anti-pattern)》中对自动化测试金字塔进行了更为细致的分层，如图 9.3 所示。

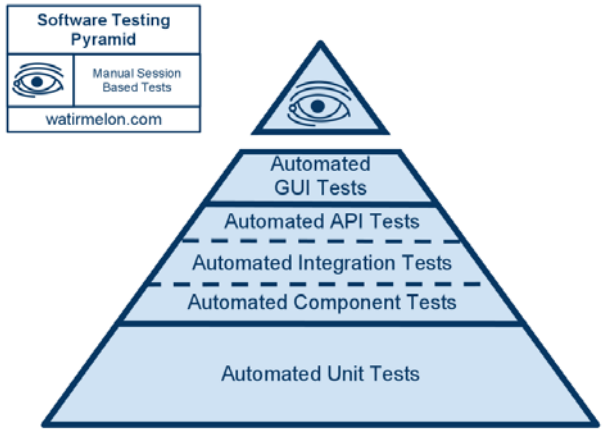


图 9.3 自动化测试金字塔

注释：Software Testing Pyramid：软件测试金字塔，Manual Session Based Tests：手工测试，Automated GUI Tests：自动化界面测试，Automated API Tests：自动化 API 测试，Automated Integration Tests：自动化集成测试，Automated Component Tests：自动化组件测试，Automated Unit Tests：自动化单元测试，

在 Alister Scott 的金字塔模型中，他用一个处于金字塔最高点部分的“上帝之眼”来代表基于人工会话的测试（Manual Session Based Test），下面的三层，基本上与 Mike Cohn 的测试金字塔相似。不同的是，原来的 Service 层被更详细的分成了：自动化 API 测试，自动化集成测试和自动化组件测试。

小练习：

请读者自行查找资料，进行学习，并回答如下问题：

1. 单元测试是什么？
2. 集成测试是什么？
3. GUI 是什么？
4. 集成测试和组件测试的区别是什么？
5. 在以下的自动化测试中，（ ）应该运行的最快。

A. 单元测试　　B. 组件测试　　C. 集成测试　　D. GUI 测试

6. 在以下的自动化测试中，（ ）的测试案例应该投入最大，案例写的最多？

A. 单元测试　　B. 组件测试　　C. 集成测试　　D. GUI 测试

xUnit 测试框架准备

测试框架

以下代码是一个加法函数，用于根据输入的两个参数 num1 和 num2，返回一个字符串等式。

```
def add(num1, num2):
    return "{0} + {1} = {2}".format(num1, num2, num1 + num2)
```

为了测试这个函数，我们编写一个程序，调用这个函数并验证其结果是否符合预期。

```
def add(num1, num2):
    return "{0} + {1} = {2}".format(num1, num2, num1 + num2)
if(__name__ == "__main__"):
    if("1 + 2 = 3" == add(1,2)):
        print("测试通过")
    else:
        print("测试不通过")
```

每次测试都要编写上面这么多的代码，会导致很多的代码重复，并且代码的可读性和可维护性也会堪忧。

为了解决这个问题，我们就需要一个单元测试框架。

单元测试框架的目的是为单元测试案例的编写提供必要的辅助和支持，使单元测试的代码具有更好的可读性和可维护性。

为了完成下面的挑战问题，读者需要学习了解如下内容：

继承与 self

继承的概念在前面的面向对象的小节已经学习过，但是并没有实战练习，这里读者要重新学习一下《简明 Python 教程》中“面向对象编程”小节的“继承”部分。其实，继承最主要的目的就是为了复用，通过继承的机制，子类可以继承和复用父类的成员变量和方法。

同样，对于 self 的概念，这里也不再多做介绍，读者可以参考《简明 Python 教程》中“面向对象编程”小节的“self”部分。

dir()函数

在 Python 3.6 中一共有 68 个内建函数(Built-in Functions)，dir()函数是其中一个很重要的内建函数。

以下内容引用自 Python 官方文档：https://docs.python.org/3.6/library/functions.html#dir

dir([object])

Without arguments, return the list of names in the current local scope. With an argument, attempt to return a list of valid attributes for that object.

没有参数的时候，dir()函数返回一个包含当前作用域下所有名字的列表；当有一个参数的时候，dir()函数会尝试返回参数对象的所有的有效属性组成的列表。

```
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__']
#定义一个 Test 类
>>> class Test:
```

```
...     def methodA(self):
...         pass
...
#声明一个 Test 类的 test 对象
>>> test = Test()
>>> dir(test)
#带有两个下划线开头的都是私有的预定义变量或者方法，注意找一下是否有我们上面代码中自定义的 methodA 方法。
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'methodA']
>>>
```

小练习：

请读者编写一个小程序，打印出列表数据类型的所有以 i 和 r 开头的方法。

getattr()函数

getattr()函数也是一个很重要的内建函数。

以下内容引用自 Python 官方文档：https://docs.python.org/3.6/library/functions.html#getattr

getattr(object, name[, default])

Return the value of the named attribute of object. name must be a string. If the string is the name of one of the object's attributes, the result is the value of that attribute. For example, getattr(x, 'foobar') is equivalent to x.foobar. If the named attribute does not exist, default is returned if provided, otherwise AttributeError is raised.

返回对象被指定名字的属性的值，name 参数必须是一个字符串，如果字符串是对象的一个属性的名字，则返回的结果是该属性的值。比如，getattr(x, 'foobar')等同于 x.foobar 的效果。如果被指定名字的属性不存在，有 default 参数的时候，则返回 default 参数指定的值；没有 default 参数的时候会报一个 AttributeError 的错误。

如果字符串对象是一个方法的名字，则会返回这个方法的指针。

```
>>> class Test:
...     def methodA(self):
```

```
...         print("这个方法是 methodA")
...         pass
...
>>> test = Test()
#获取 test 对象的 methodA()方法的方法指针
>>> method = getattr(test,"methodA")
>>> method()
这个方法是 methodA
>>> method = getattr(test,"methodB")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
#找不到 test 对象的 methodB()方法，没有提供缺省的返回值，所以报错了
AttributeError: 'Test' object has no attribute 'methodB'
>>> method = getattr(test,"methodB","methodA")
>>> method()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object is not callable
#缺省值是一个字符串，所以，当 methodB 方法或者属性都找不到的时候，就返回了一个字符串“methodA”
>>> method
'methodA'
#设置缺省值为 test 对象的 methodA()方法
>>> method = getattr(test,"methodB",getattr(test,"methodA"))
#此时，虽然找 methodB 仍然找不到，但是，不会报错，会返回缺省的 methodA()方法
>>> method()
这个方法是 methodA
>>>
```

挑战问题

实现一个简易的属于自己的自动化测试框架 TestFrame 类，文件名为 testframe.py。让其他开发者通过继承该自动化测试框架的父类 TestFrame，来创建一个自动化测试案例子类（如下所示的 MyTestCase 类）。通过在该子类中编写 test 开头的方法来进行具体的测试案例的编写。测试案例编写完成后，

通过实例化该子类为一个对象，并运行继承自父类(TestFrame)的 runTest()方法，就可以运行该子类中定义的所有以 test 开头的方法。

要求在自动化测试框架的父类 TestFrame 中实现一个 assertEqual (expected, tested)方法，该方法可以判断第二个被测试的参数(tested)是否与第一个参数(expected)相等：

- 如果相等就输出“【方法名字】测试通过!”;
- 如果不相等就输出“【方法名字】测试失败! 预期结果是：1 + 2 = 3，实际结果是：1 + 1 = 2”。

MyTestCase 类，文件名 MyTestCase.py 中的代码如下：

```
from testframe import *

def add(num1, num2):
    return "{0} + {1} = {2}".format(num1, num2, num1 + num2)

class MyTestCase(TestFrame):
    def testAdd(self):
        self.assertEqual("1 + 2 = 3",add(1,2))

    def testAdd2(self):
        self.assertEqual("1 + 2 = 3", add(1, 1))

if(__name__ == "__main__"):
    testcase = MyTestCase()
    testcase.runTest()
```

读者完成自动化测试框架 TestFrame 类(文件名为 testframe.py)的编写之后，将其放在与 MyTestCase.py 相同的目录下，然后运行 MyTestCase.py 会得到如图 9.4 所示的输出结果。

图 9.4 Test Frame

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

（1） 类的继承概念和实际使用。

（2） dir()和 getattr()。

（3） self 变量的含义。

自动化测试

（1） 自动化测试的意义。

（2） 自动化测试的分类。

9.2 unittest（一）

学习目标

学会用 unittest 测试 Python 类。

学习资源

https://docs.python.org/3/library/unittest.html
https://docs.python.org/3/library/math.html
https://pythonguidecn.readthedocs.io/zh/latest/writing/tests.html

知识准备

unittest 是 python 中最流行的单元测试框架之一，已经被集成到 python 标准库。下面对 unittest 进行入门级介绍。

unittest 的概念

以下内容翻译整理自 https://docs.python.org/3/library/unittest.html

The unittest unit testing framework was originally inspired by JUnit and has a

similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

unittest 单元测试框架的灵感来自 Java 语言的主要测试框架 Junit。unittest 支持测试自动化，在测试案例之间共享测试准备和测试清理代码，支持将测试案例分组聚合管理以及测试案例与报告框架独立。

To achieve this, unittest supports some important concepts in an object-oriented way:

为了实现以上目标，unittest 以面向对象的方式支持以下重要的概念

1. test fixture 测试夹具

A test fixture represents the preparation needed to perform one or more tests, and any associate cleanup actions. This may involve, for example, creating temporary or proxy databases, directories, or starting a server process.

测试夹具是执行测试方法前所需要做的准备工作，以及测试方法完成后所有清理活动的统称。包括，创建临时数据库、数据库代理、文件目录或启动服务进程。

实际上，就是通常所说的 setUp 和 tearDown，setUp 用来进行测试准备工作，tearDown 用来进行测试清理工作。先记住这两个词，在下面的代码里读者会有更直观的认识。

2. test case 测试案例

A test case is the individual unit of testing. It checks for a specific response to a particular set of inputs. unittest provides a base class, TestCase, which may be used to create new test cases.

一个测试案例就是一个独立的测试单元。它会检查被测试对象在一系列特定的输入后，是否给出特定的响应。unittest 提供了一个基类 TestCase，这个类可以用来创建新的测试案例。

也就是说，unittest 里所有的测试案例，即测试方法，都继承自 TestCase 类。

3. test suite 测试套件

A test suite is a collection of test cases, test suites, or both. It is used to aggregate tests that should be executed together.

测试套件是一些测试案例或者测试套件的集合，用来将一些需要一起执行的测试案例聚合在一起，以方便执行。

4. test runner 测试运行器

A test runner is a component which orchestrates the execution of tests and provides the outcome to the user. The runner may use a graphical interface, a textual interface, or return a special value to indicate the results of executing the tests.

测试运行器是一个组件，可以编排多个测试案例的执行并且给使用者提供执行后的输出。启动器可以是图形界面、字符界面或者通过返回一个特定的值来表示测试案例的执行结果。

第一个 unittest 类

首先看一个实际的测试代码。

```
import unittest
def addnum(num1,num2):
    return num1+num2
class TestComputer(unittest.TestCase):
    #应该支持负数
    def test_should_good_for_negative_number(self):
        self.assertEqual(0,addnum(-1,1))
    #应该支持大于 10000000000 的数
    def test_should_good_for_bignum_morethan_10000000000(self):
        self.assertEqual(10000000001,addnum(1,10000000000))
    #应该支持小于-10000000000 的数
    def test_should_good_for_bignegativenum_morethan10000000000(self):
        self.assertEqual(-10000000001,addnum(-1,-10000000000))

if __name__=='__main__':
    unittest.main()
```

上面的代码，首先定义了一个 addnum 的函数，然后定义了一个 TestComputer 的测试类，这个类里定义的三个方法就是三个针对 addnum 函数的测试案例，见代码中的注释部分。

最后一个条件判断语句需要特别说明一下。

这个条件判断语句的目的是为了从命令行运行测试案例提供支持。当用户从命令行运行 python 代码的时候，__name__变量会被赋值为'**main**'。此时，调用 unittest.main()方法，就会将上面 TestComputer 中定义的所有测试案例都执行一遍。

上文提到，测试案例会检查被测试对象在一系列特定的输入后，是否会给出特定的响应。要对响应或者输出进行检查，就需要用到 assert 方法，下表是一些常见的 assert 方法。

方　法	等价于	描　述
assertEqual(a, b)	a == b	是否相等
assertNotEqual(a, b)	a != b	是否不相等
assertTrue(x)	bool(x) is True	是否为真
assertFalse(x)	bool(x) is False	是否为假
assertIs(a, b)	a is b	是否相同
assertIsNot(a, b)	a is not b	是否不同
assertIsNone(x)	x is None	是否是 None
assertIsNotNone(x)	x is not None	是否不是 None
assertIn(a, b)	a in b	a 是否在 b 里
assertNotIn(a, b)	a not in b	a 是否不在 b 里
assertIsInstance(a, b)	isinstance(a, b)	a 是否是类型 b
assertNotIsInstance(a, b)	not isinstance(a, b)	a 是否不是类型 b

当 assert 方法为真，即结果为是的时候，测试案例会通过；反之，测试案例不通过。

运行 unittest

在 PyCharm 中测试类定义的行，如上面代码中“class TestComputer (unittest.TestCase):”所在的行，单击右键，选择绿色三角菜单，即“run unittest in xxxxxxxx”，即可以运行当前测试类中的所有测试案例。如图 9.5 所示。

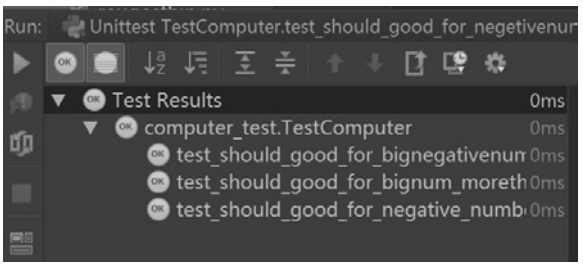


图 9.5 运行所有 unittest 案例

在某个测试案例方法上，右键选择“run unittest in xxxxxxx”，即可运行当前选定的测试案例。如图 9.6 所示。

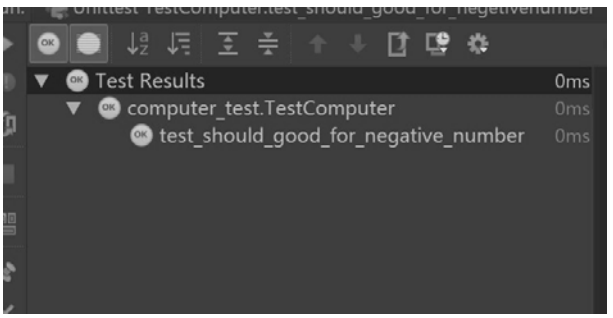


图 9.6 运行当前选定的 unittest 案例

unittest 有两种运行方法。一种是在 PyCharm 中运行，一种是通过命令行运行。通过命令行运行测试案例，我们下一个小节再做介绍。

math 模块

在“5.2 模块是枝干”小节，我们已经知道了模块的概念，并且自己编写了一个模块。下面向大家介绍一个很重要的模块，math 模块。

math 是一个 python 自带的模块，该模块主要提供针对 c 标准库中定义的数学功能的访问。

下面介绍两个常用 math 模块的功能，以下英文信息来自 Python3 官方网站。

math.fabs(x)

Return the absolute value of x.
返回 x 的绝对值。

```
>> import math
>>> math.fabs(-3)
3.0
```

math.trunc(x)

```
Return the Real value x truncated to an Integral (usually an integer).
返回实数 x 的整数部分。

>>> import math
>>> math.trunc(3.5)
3
```

isinstance()函数和 type()函数

下面介绍两个有用的内建函数：isinstance()和 type()。

以下英文信息来自 Python3 官方网站。

isinstance(object, classinfo)

Return true if the object argument is an instance of the classinfo argument, or of a (direct, indirect or virtual) subclass thereof. If object is not an object of the given type, the function always returns false. If classinfo is a tuple of type objects (or recursively, other such tuples), return true if object is an instance of any of the types. If classinfo is not a type or tuple of types and such tuples, a TypeError exception is raised.

当 object 参数是一个 classinfo 参数的实例（或者是其子类）的时候，返回 true；否则，返回，false；当 classinfo 是一个元组类型的时候，object 参数是元组中任何一个元素对应的类型的实例都会返回 true；如果 classinfo 不是一个类型或者类型组成的元组时，返回一个类型错误异常：TypeError。

```
>>> num1 = 4
>>> str1 = "hello"
>>> isinstance(num1,int)
True
>>> isinstance(str1,int)
False
>>> isinstance(str1,str)
```

```
True
>>> isinstance(str1,string)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'string' is not defined
>>> isinstance(str1,"string")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: isinstance() arg 2 must be a type or tuple of types
```

注意体会上面两个不同错误出现的原因。

class type(object)

With one argument, return the type of an object. The return value is a type object and generally the same object as returned by object.__class__.

当带一个参数的时候，返回 object 参数对应的类型。返回值是一个类型对象，与 object.__class__的返回值一致。

```
>>> num1 = 4
>>> str1 = "hello"
>>> type(num1)
<class 'int'>
>>> num1.__class__
<class 'int'>
>>> type(str1)
<class 'str'>
```

挑战问题

编写一个 Python 的 unittest 程序 TestMyCalculator.py，对“5.3 面向对象是另一种看待世界的视角”小节的运算器类 MyCalculator 的除法方法 divide()进行测试。

现在 MyCalculator 类的 divide()方法需满足以下要求：

- 除法运算得到的结果，要用去尾法取整。

- 当被零除的时候，返回字符串“除数不能为零”，不能直接抛ZeroDivisionError 异常到输出界面。
- 要求输入的两个参数均为整数型或者字符串类型。
- 当输入的参数为字符串类型时，如果该字符串能够正确的转换为整数型，则将其转换为整数型，并继续后续处理。
- 输入的参数既不是整数型，又不是能够转换为整数型的字符串时，返回字符串“请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串”。

如图 9.7 所示。

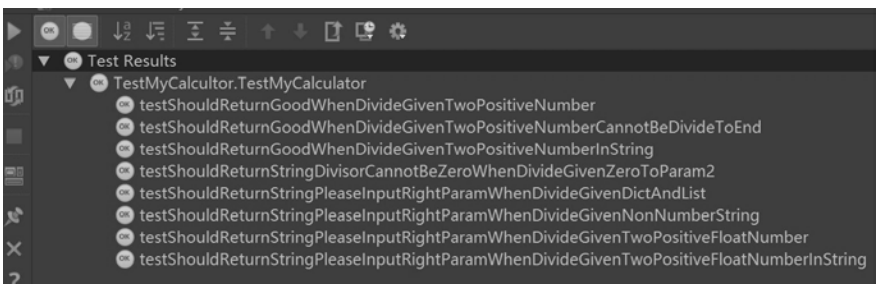


图 9.7 Test Mycalculator

请按照以下步骤解决挑战问题：

（1） 假设 MyCalculator 类的 divide()方法已经满足了要求。

（2） 以测试人员的视角考虑一下，为了测试这个 divide()方法至少需要多少个单元测试案例。

（3） 写好单元测试案例，暂时不要修改 MyCalculator 类的代码。

（4） 运行所有写出来的测试案例，保证其全部是 fail，没有 Error。

（5） 根据写好的测试案例的运行情况，修改 MyCalculator 类的 divide()方法，让没有通过的测试案例都一一通过。

遵守上面的步骤和要求，你会有不一样的收获。

如果你针对 divide()方法设计的测试案例少于 8 个，请好好看一下上面的要求，复习一下测试案例编写的方法。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，

重做一次。

难点提示

语言基础知识要点

（1） unittest 的中文资料可以参考简书中的文章，链接如下：

- http://www.jianshu.com/p/38948d0d73f5
- http://www.jianshu.com/p/0b04cb0450ee
- http://www.jianshu.com/p/8e22c1213260
- http://blog.csdn.net/xiaosongbk/article/details/52884837

（2） 每个测试方法均以 test 开头，否则不能被识别。

（3） Python 有很多其他的单元测试框架，建议读者聚焦其中一个，或仅学习本书推荐的 unittest 框架即可。

（3） try……except 是用来捕获代码中的异常。

（4） isinstance()用来判断一个变量的类型。

（5） math.trunc()用来对一个数值向靠近 0 的方向取整，即去尾法取整。了解一下 math 库的其他方法，对于数值计算类问题的解决会很有帮助。

9.3 unittest（二）

学习目标

学会用 unittest 测试 Python 类，学会通过命令行运行 unittest。

学习资源

https://docs.python.org/3/library/unittest.html
https://pythonguidecn.readthedocs.io/zh/latest/writing/tests.html

知识点

通过命令行运行 unittest

Python 的 unittest 模块可以用于从命令行运行测试模块、测试类甚至某一个测试方法，python 命令的-m 参数用于以脚本的方式运行一个库模块。下面

的命令就是以脚本的方式运行 unittest 库模块，这样 unittest 就会将后面的文件、类、模块或者方法作为单元测试来运行，并且给出输出结果：

```
python -m unittest test_module1 test_module2
python -m unittest test_module.TestClass
python -m unittest test_module.TestClass.test_method
python -m unittest tests/test_something.py
```

你可以通过传递一个 –v 的参数在运行测试案例的时候，显示更多的细节，比如：

```
python -m unittest -v test_module
```

在自己的源代码路径下，尝试运行一下已经编写的测试案例。

下面是 python 单元测试模块的命令行格式：

```
python -m unittest [-h] [-v] [-q] [--locals] [-f] [-c] [-b]
                   [tests [tests ...]]
```

tests 参数是多个测试模块、测试类或者测试方法的列表。

常用的两个可选参数：

```
-v, --verbose   显示详情
-f, --failfast  在第一个失败或者报错的案例处停止执行
```

unittest 模块支持测试案例的自动发现，即 discover 子命令。

```
python -m unittest discover [-h] [-v] [-q] [--locals] [-f] [-c]
                            [-b] [-s START] [-p PATTERN] [-t TOP]
```

注意：作为一种简写形式，python -m unittest 和 python -m unittest discover 是等同的，但是，当你希望传递参数给 discover 子命令的时候，命令行中的 discover 不能省略。可选参数:

```
-v, --verbose 显示详情
-s, --start-directory directory 从指定的开始目录启动发现(缺省不写为当前目录.)
-p, --pattern pattern 采用模式匹配测试文件名的方式发现(缺省不写是 test*.py)
-t, --top-level-directory directory 指定项目的最高层目录(缺省不写是开始目录)
```

-s –p 和 –t 选项是可以省略不写的，但是，要求后面指定的具体内容，必须按照上述顺序指定。

下面的两个命令行是等同的（注意：模式匹配要用引号引起来，第二个省略了-s 和-p）：

```
python -m unittest discover -s project_directory -p "*_test.py"
python -m unittest discover project_directory "*_test.py"
```

运行一下这个命令：

```
python -m unittest discover -v "test*.py"
```

看看报什么错误？为什么？在命令行中间只添加一个字符就可以了，是哪个字符？

多尝试，多探索，多练习！！！

测试案例编写补遗

单元测试案例类需要继承自 unittest.TestCase。

- setUp 用来进行测试准备工作。
- tearDown 用来进行测试清理工作。

想想以下几个方法的作用：

- setUpClass()。
- tearDownClass()。
- setUpModule()。
- tearDownModule()。

小练习：

通过查找资料，学习如何采用@skip 修饰符跳过一个测试案例或一个测试类的执行。找到一个自己写测试类，在其中一个测试方法定义的前一行增加@skip 修饰符，运行一下整个测试类，检查该测试方法是不是真的被跳过了？

挑战问题

在完成“7.2 状态码的五个分类”小节的时候，假设读者已经有一个 StatusCodeType 的类，并且该类有一个 getStatusType()方法，可以接收一个字

符串类型的状态码，并满足如下要求：

（1） 如果状态码在 100-599 范围内，返回字符串，说明状态码的归属状态分类，以及分类描述信息，格式如下：

“你输入的状态码 121 属于 1XX 类
分类描述：信息，服务器收到请求，需要请求者继续执行操作”

注意：字符串是两行，中间的 121 根据输入码的不同而不同。

（2） 如果状态码不在 100-599 范围内，返回字符串提示错误信息“输入错误，请输入 100-599 之间的数字”。

编写一个 Python 的 unittest 程序 TestStatusCodeType.py，对 StatusCodeType 类的 getStatusType 方法上面描述的两个功能进行测试。

如图 9.8 和图 9.9 所示。

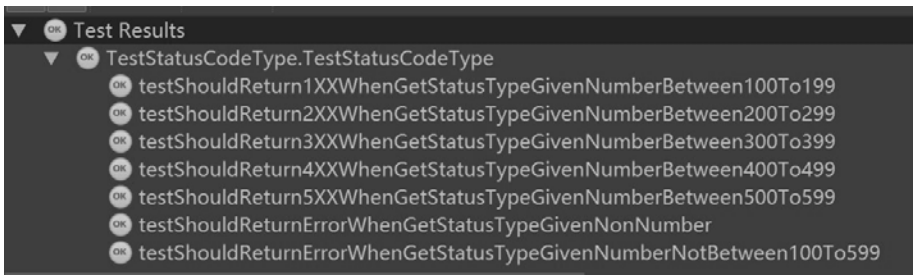


图 9.8 Test StatusCode Type

图 9.9 通过命令行运行的 Test StatusCode Type

考虑到这仅仅是练习，建议编写不少于 7 个测试案例。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

难点提示

如果“7.2 状态码的五个分类”小节的代码不是用类实现的，或者你写的类没有 getStatusType 方法，不要急于修改状态码分类的程序代码。先假设类和方法都有了。用测试人员视角考虑一下，针对一个假设已经存在的 StatusCodeType 类，其 getStatusType 方法需要多少测试案例，才能验证其功能的正确性。

请读者根据自己的理解，提前写好测试案例，然后通过创建和编写 StatusCodeType 类的代码，让测试案例一个一个的通过，然后来逐渐形成完整的 StatusCodeType 类。

严格遵守上面的过程，你会有不一样收获。

你应该有不少于 7 个测试案例，其中针对 5 个状态码分类每个都编写一个测试案例之外，还应该有 2 个异常场景的测试案例。如果考虑边界值，每类应该有 3 个测试案例，一共应该有不少于 17 个测试案例。

知识点

语言基础知识要点

（1） Python 命令行中 unittest 模块以及 discover 子命令的使用。

（2） 测试案例的命名也是一种学问，好的命名可以让测试案例一目了然，可以采用：test+ShouldReturn[预期结果]When[被调用的方法]Given[输入的参数]的格式命名测试案例。

（3） 一个测试案例最好只有一个测试场景，这样测试运行起来，结果会更加清晰，出错了也易于问题定位。

（4） 测试案例也要注意代码的简洁，避免重复。

拓展

在命令行中不带-v 参数运行 unittest 的时候，每一个“.”代表一个成功的案例，试试不成功的会显示出什么？

第十章 HTTP 接口测试（无状态）

10.1 接口约定

学习目标

学会用 unittest 测试 HTTP 接口。

学习资源

https://docs.python.org/3.6/library/time.html#module-time
http://www.runoob.com/python/python-date-time.html
https://docs.python.org/3.6/library/unittest.html#unittest.main

知识准备

time 模块

time 模块提供了和时间相关的各种功能。主要有以下函数（以下内容整理自 Python 官方网站）：

time.time()函数

Return the time in seconds since the epoch as a floating point number. The specific date of the epoch and the handling of leap seconds is platform dependent. On Windows and most Unix systems, the epoch is January 1, 1970, 00:00:00 (UTC)

以一个浮点数的形式，返回从新纪元开始到现在的秒数。新纪元的具体时间和闰秒的处理方式是平台相关的。在 Windows 和大多数的 Unix 系统上，新纪元的时间是 1970 年 1 月 1 日零点零分零秒。

```
>>> import time
>>> time.time()
```

```
1502506209.6767917
```

time.sleep(secs)

Suspend execution of the calling thread for the given number of seconds. The argument may be a floating point number to indicate a more precise sleep time.

让调用这个函数的线程暂停给定的秒数。secs 参数可以用浮点数来指定一个比秒更精确的暂停时间。

```
>>> import time
>>> time.sleep(5)
```

运行上面的代码看看暂停的效果。

time.asctime([t])

Convert a tuple or struct_time representing a time as returned by gmtime() or localtime() to a string of the following form: 'Sun Jun 20 23:21:05 1993'.

将一个给定的元组或者 struct_time 数据结构的数据变量转换成一个以下格式的字符串：'Sun Jun 20 23:21:05 1993'。

gmtime() or localtime()方法，会返回 struct_time 数据格式。

不提供参数的情况下，asctime()函数默认返回当前时间的字符串形式。

```
>>> import time
>>> time.asctime()
'Sat Aug 12 14:43:44 2017'
>>>
```

关于 gmtime() or localtime()方法的使用，读者可以自行查找相关资料进行学习。

除了 time 模块，calendar 模块也是一个常用的模块，主要用于日历计算方面，如年、月、日、星期的运算，读者可以根据自己的需要了解和学习。

unittest.main()方法

相信大家对以下两行代码并不陌生

```
if __name__=='__main__':
    unittest.main()
```

这两行代码的含义是，如果 python 脚本是被从命令行运行，此时__name__变量会被设置为‘**main**’，就会运行 unittest 中 main()方法，用来执行所有上面定义的测试案例。

unittest.main()方法有很多的参数可用，我们在后续的章节中会学到，这里主要来认识一下 verbosity 参数：

verbosity 参数：

You can run tests with more detailed information by passing in the verbosity argument.

你可以通过给 verbosity 参数赋值，在运行测试的时候显示更多更详细的信息。

verbosity 参数的默认值为 1，此时会用“.”表示案例通过；“F”表示案例失败。如果需要显示每一个案例方法的名和案例的运行情况，需要对其赋值为 2。

以上内容整理翻译自 Python 官方网站：https://docs.python.org/3.6/library/unittest.html#unittest.main

接口探索与约定

根据“8.1 我知道你是哪里人”小节中对查询接口（http://ip.taobao.com/service/getIpInfo.php）的描述：

（1） 接口的输入采用 GET 请求 url 参数的方式给出：/service/getIpInfo.php?ip=[ip 地址字串]。

（2） 响应信息为 JSON 格式，包括国家、省（自治区或直辖市）、市（县）、运营商信息。

（3） 返回数据格式示例如下：{"code":0,"data":{"ip":"210.75.225.254", "country":"e2d6fd","area":"34e317", "region":"317eace02","city":"317eace02", "county":"","isp":"535fe1", "country_id":"86","area_id":"100000","region_id":"110000","city_id":"110000", "county_id":"-1","isp_id":"100017"}}

（4） 返回格式中，code 值的含义为：0，成功；1，失败。

以上内容整理自“淘宝 IP 地址库”网站，链接为：http://ip.taobao.com/instructions.php

关于查询接口(http://ip.taobao.com/service/getIpInfo.php)的描述，实际上就是一种接口约定，接口约定中一般会包含以下内容。

（1） 请求的方式：getIpInfo 接口的请求方式是 HTTP 的 GET 请求。

（2） 数据的传入方式：getIpInfo 接口的数据传入方式是通过 url 参数传入。

（3） 响应信息格式：getIpInfo 接口的响应信息格式为 JSON 格式，并且给出了具体的 JSON 示例。

（4） 异常响应信息的形式：getIpInfo 接口返回的 JSON 格式中有一个叫做 code 的键值，用 0 和 1 分别代表成功和失败。

通过上面的描述我们可以看出，异常情况下，比如 IP 地址不合法，接口约定中并没有给出返回数据的格式示例。此时，我们可以使用 curl 工具进行一些简单的接口探索。

```
curl -v http://ip.taobao.com/service/getIpInfo.php?ip=badip
*   Trying 140.205.157.1...
- TCP_NODELAY set
- Connected to ip.taobao.com (140.205.157.1) port 80 (#0)
> GET /service/getIpInfo.php?ip=badip HTTP/1.1
> Host: ip.taobao.com
> User-Agent: curl/7.52.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Tengine
< Date: Sat, 12 Aug 2017 07:24:09 GMT
< Content-Type: text/html
< Transfer-Encoding: chunked
< Connection: keep-alive
< Vary: Accept-Encoding
< X-Powered-By: PHP/5.3.6
<
{"code":1,"data":"invaild ip."}* Curl_http_done: called premature == 0
- Connection #0 to host ip.taobao.com left intact
```

从中可以看到，在给这个查询接口传递一个不合法的 IP 地址“badip”时，接口响应的状态码部分仍然为“200 OK”，但是，返回的 JSON 数据格式发生了变化。

{"code":1,"data":"invaild ip."}

code 的值为 1，data 的值不再是一个 JSON 对象，而变成了一个字符串："invaild ip."

挑战问题

根据指示准备中的信息，编写一个 Python 的 unittest 程序 TestGetIPInfo.py，对“8.1 我知道你是哪里人”小节中查询 IP 地址所在地信息的接口进行测试。至少设计两个测试案例，并运行成功，示例如图 10.1 所示。

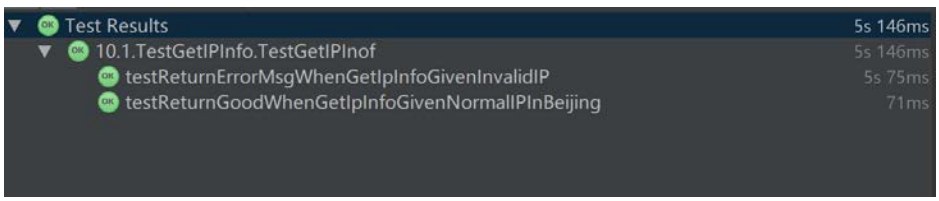


图 10.1 通过 Run Unittests 方式运行的 test getipinfo

图 10.1 是通过在 PyCharm 的测试类定义的代码行右键点击运行"Run Unittests in TestGetIPInfo"菜单项的结果。

图 10.2 是通过在 PyCharm 直接运行测试类的结果。

图 10.2 通过 Run TestGetIPInfo 方式运行的 test getipinfo

具体操作是：在"**main**"所在的行，点击鼠标右键，然后，点击运行"Run TestGetIPInfo"菜单项。

要求能够显示每一个案例方法的名称和案例的运行情况。

为了避免短时间内大量发送请求，请在第二个案例的开始位置等待 3 秒的时间。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

难点提示

（1） 从输入参数的角度进行等价类、边界值的案例设计。

（2） 考虑异常场景有哪些？

知识点

语言基础知识要点

为了防止 IP 地址被阻止，可以考虑在每一个案例运行前添加一个 sleep，暂停 3 秒钟。学习一下 time.sleep()，以及 time 里的其他方法对于时间操作类问题很有帮助。

10.2 案例编写

学习目标

学会用 unittest 测试 HTTP 接口，学习数据与代码分离。

学习资源

https://docs.python.org/3/library/unittest.html#distinguishing-test-iterations-using-subtests

https://docs.python.org/3/library/unittest.html#unittest.TestCase.subTest

知识准备

迭代运行同类测试案例

在测试过程中，经常会出现针对同一个测试接口或者功能，需要设计和运行多个测试案例的情况，而在这些测试案例之间，除了测试的输入数据和预期的结果数据有差异外，整体的测试运行步骤基本相同。

针对这种情况，我们会尝试将测试数据和测试的执行步骤进行分离，这就

是常说的“测试数据与代码分离”。

让我们以 9.2 Unittest（一）小节的测试案例为例。

以下的代码，是作者在 GitHub 上给出的 TestMyCalculator.py 的参考源代码。

```
import unittest
from MyCalcultor import MyCalculator

class TestMyCalculator(unittest.TestCase):
    def testShouldReturnGoodWhenDivideGivenTwoPositiveNumber(self):
        myCalculator = MyCalculator()
        self.assertEqual(4,myCalculator.divide(8,2))
    def
testShouldReturnGoodWhenDivideGivenTwoPositiveNumberCannotBeDivideToEnd(self):
        myCalculator = MyCalculator()
        self.assertEqual(4,myCalculator.divide(9,2))
    def testShouldReturnGoodWhenDivideGivenTwoPositiveNumberInString(self):
        myCalculator = MyCalculator()
        self.assertEqual(4,myCalculator.divide("9","2"))
    def  testShouldReturnStringPleaseInputRightParamWhenDivideGivenTwoPositive
FloatNumberInString(self):
        myCalculator = MyCalculator()
        self.assertEqual("请输入正确的参数，参数必须是整数型或者可以转换为整数
型的字符串",
                         myCalculator.divide("9.8","2.1"))
    def
testShouldReturnStringPleaseInputRightParamWhenDivideGivenTwoPositiveFloatNumbe
r(self):
        myCalculator = MyCalculator()
        self.assertEqual("请输入正确的参数，参数必须是整数型或者可以转换为整数
型的字符串",
                         myCalculator.divide(9.8,2.1))
    def  testShouldReturnStringDivisorCannotBeZeroWhenDivideGivenZeroToParam2
(self):
        myCalculator = MyCalculator()
        self.assertEqual("除数不能为零",myCalculator.divide(9,0))
```

```
    def testShouldReturnStringPleaseInputRightParamWhenDivideGivenNonNumberString(self):
        myCalculator = MyCalculator()
        self.assertEqual("请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串",
                         myCalculator.divide("ssd","dd"))
    def testShouldReturnStringPleaseInputRightParamWhenDivideGivenNonNumberString(self):
        myCalculator = MyCalculator()
        self.assertEqual("请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串",
                         myCalculator.divide("ssd","11"))
    def testShouldReturnStringPleaseInputRightParamWhenDivideGivenDictAndList(self):
        myCalculator = MyCalculator()
        self.assertEqual("请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串",
                         myCalculator.divide({"as":"value"},[11,2,2]))
```

这 9 个测试案例，每一个都是先创建一个 myCalculator 对象，然后使用 assertEqual()方法来验证 myCalculator.divide()方法在给定的输入下，是否能够得到预期的结果。

我们尝试改写一下这个测试类，首先将 9 个测试案例的输入数据和预期的输入结果进行整理，我们得到了以下的表格：

序　号	案例名称	输入参数 1	输入参数 2	预期结果
1	两个都是整数，且能够被整除的情况下结果正常	8	2	4
2	两个都是整数，且不能被整除的情况下结果为整数部分	9	2	4
3	两个输入参数为可以转换为整数的字符串，能够正常处理	"9"	"2"	4
4	两个输入参数为不能转换为整数的字符串，返回错误信息	"9.8"	"2.1"	"请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串"
5	两个输入参数为不是整数的数字型，返回错误信息	9.8	2.1	"请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串"

（续表）

序　号	案例名称	输入参数 1	输入参数 2	预期结果
6	除数为零的时候，返回错误信息	9	0	"除数不能为零"
7	两个输入参数为不是数字的字符串，返回错误信息	"ssd"	"dd"	"请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串"
8	一个输入参数为不是数字的字符串，返回错误信息	"ssd"	"11"	"请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串"
9	参数不是整数型也不是字符串，返回错误信息	{"as":"value"}	[11,2,2]	"请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串"

通过上面的表格我们看到，9 个测试案例的确只有在输入参数和预期结果上有差异。为了简化测试案例，我们将测试案例进行一下优化，编写一个 TestMyCalculatorByList.py。

首先定义一个列表数据结构，用于保存 9 个测试案例的测试数据：

```
testDataList = [
{
"name":"两个都是整数，且能够被整除的情况下结果正常",
"param1":8,
"param2":2,
"expect":4
},
{
"name":"两个都是整数，且不能被整除的情况下结果为整数部分",
"param1":9,
"param2":2,
"expect":4
},
{
"name":"两个输入参数为可以转换为整数的字符串，能够正常处理",
"param1":"9",
"param2":"2",
"expect":4
},
{
```

```
"name":"两个输入参数为不能转换为整数的字符串，返回错误信息",
"param1":"9.8",
"param2":"2.1",
"expect":"请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串"
},
{
"name":"两个输入参数为不是整数的数字型，返回错误信息",
"param1":9.8,
"param2":2.1,
"expect":"请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串"
},
{
"name":"除数为零的时候，返回错误信息",
"param1":9,
"param2":0,
"expect":"除数不能为零"
},
{
"name":"两个输入参数为不是数字的字符串，返回错误信息",
"param1":"ssd",
"param2":"dd",
"expect":"请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串"
},
{
"name":"输入参数为不是数字的字符串，返回错误信息",
"param1":"ssd",
"param2":"11",
"expect":"请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串"
},
{
"name":"参数不是整型也不是字符串，返回错误信息",
"param1": {"as":"value"} ,
"param2": [11,2,2] ,
"expect":"请输入正确的参数，参数必须是整数型或者可以转换为整数型的字符串"
}
]
```

我们可以看到 testDataList 列表中包含了 9 个元素，每个元素都是一个字典数据类型，包含：name，param1，param2 和 expect 四个键/值对。

下面我们通过编写一个测试方法 testRunTests()，在该测试方法中，通过一个 for 循环来迭代运行 testDataList 列表中的每一个测试案例。

代码如下：

```
from MyCalcultor import MyCalculator
import unittest

class TestMyCalculatorByList(unittest.TestCase):

    testDataList = [.....] #此处省略了上面的列表中的数据定义

    def testRunTests(self):
        calc = MyCalculator()
        for testdata in self.testDataList:
            self.assertEqual(testdata["expect"],calc.divide(testdata["param1"],testdata
["param2"]))
```

注意，记得将 9.2 unittest（一）小节中的 MyCalculator.py 源文件拷贝到当前目录下。

代码运行的结果如图 10.3 所示。

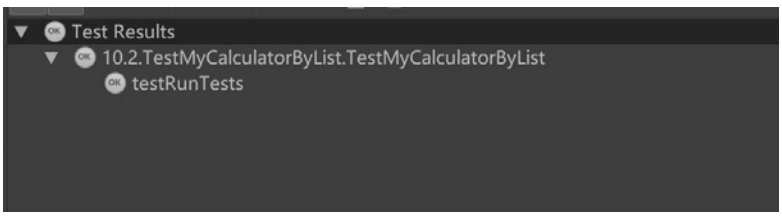


图 10.3 Test MyCalculator By List

至此，我们实现了测试数据和测试代码的分离。但是，上面的实现代码有一个问题，9 个测试案例只能体现为 1 个测试案例，这样如果有 1 个测试案例出现了失败，整个测试案例就失败了。测试执行被中断，后面的测试案例就不能被执行了，这种方法不利于针对导致测试案例执行失败问题的定位和修复。

为了解决这个问题，自 Python 的 3.4 版本开始，引入了 subTest()方法，

该方法用于在同一个测试方法中运行多个只有很小差异的子测试案例时，可以清楚地区分出各个子测试案例的运行情况。

subTest(msg=None, **params)

Return a context manager which executes the enclosed code block as a subtest. msg and params are optional, arbitrary values which are displayed whenever a subtest fails, allowing you to identify them clearly.

返回一个内容管理器，在内容管理器之下运行的测试代码会被作为一个子测试。msg 和 params 参数是可选的。子测试运行失败的时候，提示任何可以提供的信息，以便开发者定位问题。msg 可以理解为用于标识子测试的描述性信息。

下面我们用 subTest()方法，对上面通过 for 循环迭代运行的 9 个测试案例的代码进行一下优化，优化后的代码如下：

```
from MyCalcultor import MyCalculator
import unittest

class TestMyCalculatorByList(unittest.TestCase):

    testDataList = [......]

    def testRunTests(self):
        calc = MyCalculator()
        for testdata in self.testDataList:
            with self.subTest(msg=testdata["name"]):
                self.assertEqual(testdata["expect"],calc.divide(testdata["param1"],testdata["param2"]))
```

提醒：“9.2 unittest（一）”小节中的 MyCalculator.py 源文件需要拷贝到当前目录下。

运行 testRunTests()测试方法，会得到如图 10.4 所示的运行结果。

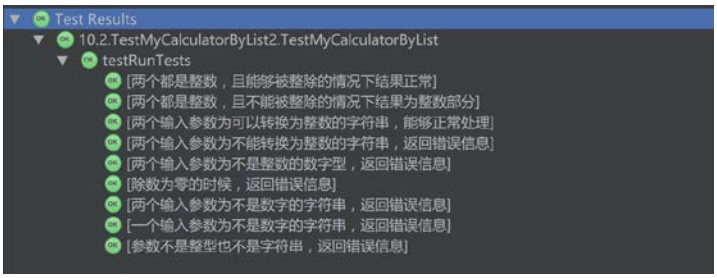


图 10.4 Test MyCalculator By List With SubTest

我们看到 9 个子测试案例也都有了自己的描述和运行状态。

这里介绍一下 with 关键字。该关键字是自 Python 2.5 引入的一种语法，更准确的说，是一种上下文的管理协议。自 python 2.6 开始，成为默认关键字。

如果读者对 with 关键字感兴趣，可以自行参考以下资料进行了解。

http://www.jb51.net/article/60666.htm

http://www.jb51.net/article/64023.htm

https://docs.python.org/3/reference/compound_stmts.html#with

https://docs.python.org/3/reference/datamodel.html#context-managers

挑战问题

编写一个 Python 的 unittest 程序 TestGetIPInfoWithList.py，对“8.1 我知道你是哪里人”小节中查询 IP 地址所在地信息的接口进行测试。设计两个测试案例，并运行成功，要求将包含如下两个 unittest 的测试案例：

（1） 给定一个正常的北京的 IP 地址，预期结果为：返回一个 JSON 数据结构，其 data 键元素对应的值元素是一个 JSON 对象，该 JSON 对象中有一个 city 键元素，且对应的值为“北京市”。

（2） 给定一个不完整的、异常的 IP 地址，预期结果为：返回一个 JSON 数据结构，其 data 键元素对应的值元素是一个字符串“invaild ip.”。

测试数据已经被抽离出来定义成了一个 list，内容如下：

```
testcaseList=
[{
"msg":"ReturnGoodWhenGetIpInfoGivenNormalIPInBeijing",
"ip":"124.126.228.193",
"expect":"北京市"
```

```
},
{
"msg":"ReturnErrorMsgWhenGetIpInfoGivenInvalidIP",
"ip": "124.126.228.",
"expect": "invaild ip."
}]
```

通过 TestGetIPInfoWithList 类的一个 test 方法 testRunTheCase 读取上面 testcaseList 列表中的两个案例的测试数据，来实现测试代码和测试数据的初步分离。

每个案例运行之前，暂停 3 秒钟。

运行的结果如图 10.5 所示。

图 10.5 Test MyCalculator By List

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

难点提示

getIpInfo 接口在两个测试案例中都会返回一个 JSON 格式的数据。但是，JSON 数据对象中的 data 键元素对应的值元素的数据类型，在两个案例中是不同的。

第一个案例中，data 键元素对应的值元素为 JSON 类型，其中的 city 值预期为“北京市”；第二个案例中，data 键元素对应的值元素为字符串类型，字符串的内容为："invaild ip."

当在同一个方法中要处理两个案例的数据，并且数据的来源不同时，可以考虑先判断 data 键元素对应的值元素的类型，再进行后续的不同处理。

知识点

语言基础知识要点

（1） 为了防止 IP 地址被阻止，可以考虑在每一个案例运行前添加一个 sleep，暂停 3 秒钟，学习一下 time.sleep()，以及 time 里的其他方法，对于时间操作类问题会有帮助。

（2） testRunTheCase 作为一个测试案例方法，在运行的时候能够显示同一个方法中两个案例各自的运行情况。subTest 方法用于在同一个测试案例方法中显示多个子案例的运行情况。

（3） subTest 一般与 with 一起使用，使用方法如下：with self.subTest (msg=“可以描述子案例的字符信息或者特征信息”): self.assertEqual (expectedResult, doSomething())。

10.3 数据外化到文件

学习目标

学会用 unittest 测试 HTTP 接口，学习将测试数据外化到文件中，学习读取文件中的内容，学习用正则表达式的方式对文件中的内容进行处理。

学习资源

http://www.runoob.com/python3/python3-reg-expressions.html
http://www.runoob.com/python3/python3-inputoutput.html
http://www.runoob.com/python3/python3-file-methods.html
http://www.runoob.com/python3/python3-file-readlines.html

知识准备

I-O、屏幕和文件

业界人士大都知道一个著名的关于程序的论断：“算法+数据结构=程序”。这个公式出自 Pascal 语言之父，图灵奖的获得者，瑞士的 Nicklaus Wirth。他有一本书籍就叫做《算法+数据结构=程序》（《Algorithms+Data

Structures=Programs》，Prentice-Hall，1976）。

我们看到，现在的很多程序除了算法和数据结构之外，I-O 已经成为其重要的组成部分。所谓 I-O，就是 INPUT 和 OUTPUT，即输入和输出。

简单的输入方面的例子，我们从第二个练习就已经开始在用了，即让程序具备获取键盘输入的信息的能力。

最简单的输出方面的例子，就是屏幕显示。

随着计算机的发展，输入和输出的含义已经非常广泛。输入就是从文件读取数据，而输出就是将加工处理后的数据保存到文件中。

首先，在本地的当前目录创建一个文件，名为：hello.txt。文件的内容如下：AAAAA BBBBB CCCCC。

我们现在将其读取出来，并打印到屏幕上，文件的代码如下：

```
helloFile = open("hello.txt","r",encoding="utf-8")
strLine = helloFile.readlines()
for line in  strLine:
   print(line)
helloFile.close()
```

是不是很简单。上面的第一行代码，用 open()方法打开一个文件，该方法会返回一个文件读写的句柄对象，然后，调用这个对象的相应的方法就可以读写文件。读写完毕之后，一定要记得调用 close()方法关闭文件。

关于 open()和 close()方法的详细用法，可以参见已给出的学习资料中的内容。只要记住，open()方法的第一个参数是文件的路径名，第二个参数是读写模式。读写模式一共有八种，分别由 8 个字符作为参数传递给 open()方法，这八个字符分别是：‘r’，‘w’，‘x’，‘a’，‘b’，‘t’，‘+’，‘U’。

小练习：

请通过查阅相关资料，分别找到以上八个字符的含义和混搭的用法。

encoding 是文件的编码格式，默认的编码格式是与平台相关的。

下面重点介绍一下 readlines()方法，方法的定义如下：

file.readlines(sizehint)

Read and return a list of lines from the stream. hint can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all

lines so far exceeds hint.

读取和返回从数据流（文件）中读取的列表类型数据，列表中保存的是每一行的内容。sizehint 可以用于指定一个数字来控制读取内容的行数，如果总行数超过 sizehint 的大小，只会读取 sizehint 指定的数值的行数。

Note that it's already possible to iterate on file objects using for line in file: ... without calling file.readlines().

注意，已经可以直接在文件对象之上使用迭代器，而不需要调用 file.readlines()方法。

也就是说，上面的代码可以改成如下：

```
helloFile = open("hello.txt","r",encoding="utf-8")
for line in  helloFile:
   print(line)
helloFile.close()
```

优化后的代码与之前的代码运行的效果是一样的。在对文本文件进行按行读取的时候，我们可以直接在 helloFile 文件句柄对象上运行 for 迭代器。

现在我们编写一个小程序，将“DDDDD”，附加到 hello.txt 文件的最后一行，也就是说，我们编写的小程序运行之后，hello.txt 文件的内容应该是：

```
AAAAA
BBBBB
CCCCC
DDDD
```

代码如下：

```
helloFile = open("hello.txt","w")
helloFile.write("DDDDD")
helloFile.close()
```

运行一下上面的代码，看看 hello.txt 文件是否和我们预想的一样？

显然是不一样的。

小练习：

修改上面小程序的代码中的一行，正确实现我们的需求，即“将’DDDDD’，附加到 hello.txt 文件的最后一行”。

初识正则表达式

正则表达式是一个特殊的字符串序列，被称之为“模式”，可用于检查一个给定的字符串是否符合某种模式。

比如：我们定义一个日期的模式是：由四位数字代表年，加一个“-”，两位数字代表月，加一个“-”，两位数字代表日，例如：2017-05-08。

那么我们就可以定义一个模式，用于匹配一个给定的字符串是不是符合我们要求的日期模式。

在正式表达式的模式定义语法中：\d 用于指定一个数字。

所以，我们可以将日期的正则表达式的模式定义为：“\d\d\d\d-\d\d-\d\d”

现在我们看一下运行的效果。

```
>>> import re
>>> print(re.match("\d\d\d\d-\d\d-\d\d","2017-05-08"))
<_sre.SRE_Match object; span=(0, 10), match='2017-05-08'>
>>>
```

在 Python 语言中，re 模块用于提供正则表达式的全部功能。

re 模块有两个重要的函数，re.match()和 re.search()。

这里我们使用 re.match()方法，这个方法会尝试采用参数中给定正则表达式模式，从第一个字符开始匹配字符串是否符合要求，如果不符合，就返回一个 None 对象；如果符合就会返回一个匹配对象。

我们发现，re.mach()方法返回了一个匹配对象，表明给定的字符串“2017-05-08”符合我们定义的日期模式。

re.match()方法的定义如下：

```
re.match(pattern，string，flag=0)
```

- 第一个参数：pattern，用于传入需要匹配的模式定义字符串。
- 第二个参数：string，用于传入被匹配和检查的字符串内容。
- 第三个参数：flag，用于传入一些标识，比如是否区分大小写等，后文会提到。

我们知道，re.match()方法会从第一个字符串匹配是否符合给定的模式，如果不符合，就会返回一个 None 对象；如果符合就会像上面的例子一样返回一个匹配对象。再看下面的例子：

```
>>> import re
>>> print(re.match("\d\d\d\d-\d\d-\d\d","2017-5-8"))
None
>>>
```

我们看到返回了 None 对象。因为模式中要求月和日都要有两个数字。那么，如何让模式既可以识别一位数字的月份，又可以识别两位数字的月份呢？

这里就要用到{}，{}在正则表达式中有特殊的含义，用于表示重复，最常用的方式是{m,n}的格式，第一个 m 是最少重复的次数，第二个 n 是最多重复的次数。比如，如果希望正则表达式模式可以同时匹配一位和两位数字的月份和日期，可以将模式修改成如下的样子：

```
\d\d\d\d-\d{1,2}-\d{1,2}
```

重新运行一下上面的例子：

```
>>> import re
>>> print(re.match("\d\d\d\d-\d{1,2}-\d{1,2}","2017-05-08"))
<_sre.SRE_Match object; span=(0, 10), match='2017-05-08'>
>>> print(re.match("\d\d\d\d-\d{1,2}-\d{1,2}","2017-5-8"))
<_sre.SRE_Match object; span=(0, 8), match='2017-5-8'>
>>>
```

我们看到，此时一位和两位的月份和日期都可以正确的匹配了，re.match()方法返回了匹配对象，而不是 None。其实，{}还可以只用一个数字作为参数，这个时候，表示{m}前面的字符必须重复 m 次。

```
>>> print(re.match("\d{4}-\d{1,2}-\d{1,2}","2017-5-8"))
<_sre.SRE_Match object; span=(0, 8), match='2017-5-8'>
>>>
```

也可以采用{m,}的格式作为重复次数的定义，表示至少重复 m 次。同时，为了简化一些重用的重复次数模式，正则表达式中还定义了重复次数的简化

字符：

- “*”字符代表{0,}，表示重复 0 次或者多次。
- “+”字符代表{1,}，表示重复 1 次或者多次。
- “?”字符代表{0,1}，表示重复 0 次或者 1 次。

比如，我们将被匹配的日期字符串改成” 2017-05-08”，在日期字符串的前面增加了多个空格。

```
>>> import re
>>> print(re.match("\d{4}-\d{1,2}-\d{1,2}","   2017-05-08"))
None
>>>
```

此时，由于 re.match()方法是从第一个字符开始匹配，但是给定的日期字符串前面加了多个空格，导致第一个字符匹配不成功，所以返回了 None。

为了兼容日期前面可能出现多个空格的情况，我们就需要用到“*”字符，将正则表达式的模式修改如下：

```
\s*\d{4}-\d{1,2}-\d{1,2}
```

在正则表达式的模式定义语法中：用于指定一个数字，在这里的，是用于代表一个空格。代表 0 个或者多个空格。

```
>>> import re
>>> print(re.match("\s*\d{4}-\d{1,2}-\d{1,2}","   2017-05-08"))
<_sre.SRE_Match object; span=(0, 13), match='   2017-05-08'>
>>>
```

我们看到，现在又匹配成功了。与此类似的在正则表达式中的特殊定义还有很多，具体可以参见学习资源中给出的链接进行学习。

当然，学了这么多我们一定要知道，其实最简单的模式就是字符串本身，千万不要忘了这一点。

```
>>> import re
>>> print(re.match("success","success in registation"))
<_sre.SRE_Match object; span=(0, 7), match='success'>
>>>
```

小练习：

请自行根据学习资料，掌握如下特殊定义在正则表达式中的含义。

\D：_____

\S：_____

\w：_____

\W：_____

例句“2017-05-08 是很好的一天”，我们希望能够将其中的日期提取并打印出来，应该怎么做？看看下边的例子：

```
>>> import re
>>> result = re.match("\s*(\d{4}-\d{1,2}-\d{1,2})","  2017-05-08 是很好的一天")
>>> print(result.group(1))
2017-05-08
```

这里，我们运用已学过的知识成功匹配了这个字符串，在此基础上，通过调用返回的匹配对象的 group()方法，来提取我们希望得到的日期字符串。

我们在“\d{4}-\d{1,2}-\d{1,2}”的两边增加了一对()，用这种方法定义了一个匹配子字符串。

前文我们提到了 re.match()方法匹配成功后，会返回一个匹配对象。该对象有几个方法用于对匹配的字符串内容进行查询，包括：group()，start()，end()，span()等。

这里我们重点介绍 group()方法：该方法用于返回匹配成功的子字符串的内容。

- 当参数为 0 的时候，返回整个字符串的内容。
- 当参数为 1 的时候，返回第 1 个子字符串的内容，以此类推。

子字符串的定义是通过给特定的模式字符串加上()确定的。下面的例子，来自 https://docs.python.org/3.6/library/re.html#match-objects：

```
>>> m = re.match(r"(\w+) (\w+)", "Isaac Newton, physicist")
>>> m.group(0)       # 返回这个匹配成功的字符串
'Isaac Newton'
>>> m.group(1)       # 第一个加上括弧的子字符串
'Isaac'
```

```
>>> m.group(2)      # 第二个加上括弧的子字符串
'Newton'
>>> m.group(1, 2)   # 多个参数会返回一个元组
('Isaac', 'Newton')
```

我们看到，上面模式字符串中有两个(+)，中间有空格，确定了两个子字符串。match.group()方法可以跟多个数字参数的时候，会返回一个元组。

以上仅仅是关于正则表达式的初步感性认识，建议读者认真学习给出的学习资源后，再进行挑战问题的解决。

小练习：

正则表达式有 14 个元字符，我们已经介绍了其中一部分，请回答下面元字符的含义：

^ _____

$ _____

* _____

+ _____

? _____

\ _____

| _____

大括号{ } _____

方括号[ ] _____

小括号( ) _____

上面例子中 re.match(r"(+) (+)", "Isaac Newton, physicist")语句中的第一个参数，模式字符串，前面的 r 的作用是什么？

_____

我们上面只介绍了 re.match()方法，请回答，re.match()方法和 re.search()方法的区别是什么？

_____

挑战问题

编写一个 Python 的 unittest 程序 TestGetIPInfoByFile.py，测试案例的要求与 10.2 小节相同。但是，要求数据从一个测试案例文本文件中读取，也就是

10.2 小节的 testcaseList 列表中的数据，将来自一个文件，文件的内容示例如下：

```
#案例目的：测试获取 IP 地址所在地理信息 API
"案例意图":"给定一个北京地区的 IP 地址，应该能够正确返回地理信息，并且 city 值为：北京市",
"msg":"ReturnGoodWhenGetIpInfoGivenNormalIPInBeijing",
"ip":"124.126.228.193",
"expect":"北京市"
---------------------------------
"案例意图":"给定一个非法的 IP 地址，应该返回错误信息：invaild ip.",
"msg":"ReturnErrorMsgWhenGetIpInfoGivenInvalidIP",
"ip": "124.126.228.",
"expect": "invaild ip."
---------------------------------------
```

测试案例文本文件的文件格式满足以下要求：

（1） 测试案例文本中#开头的行是注释，#前面可能有多个空格。

（2） 一行中如果包含三个以上连续的'-'，并且除此之外没有其他可见字符（可以有空格），表示一个案例的结束。

（3） 每个案例都采用“Key”:“Value”格式，Key 的内容和 Value 的内容都会用双引号引起来，也就是说双引号里面的内容才是案例的内容。

访问 github：https://github.com/akuing/python-and-http-interface-test，点击“Clone or download”按钮下载 ZIP 文件，下载后解压在 10.3 文件夹下有三个测试案例文件，要求将三个测试案例文件按顺序全部加载并运行。

运行结果如图 10.6 所示。

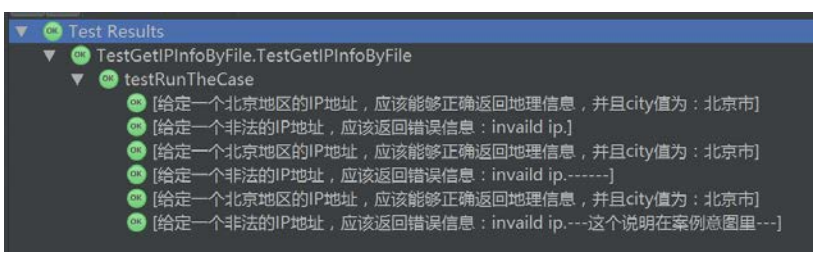


图 10.6 Test Getipinfo By File

本小节建议使用 re.match 或者 re.search，在文件之间增加 5 秒等待时间，以防止 IP 地址被封。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

难点提示

匹配每一行的数据时，考虑使用[^\"]的方式进行冒号两边的 Key-Value 的匹配。

知识点

语言基础知识要点

（1） 为什么示例中 pattern 字符串前面要加 r，加和不加的区别是什么？

（2） 正则表达式的 14 个字元，常用的表达式模式和特殊符号需要牢记。

（3） re.match 与 re.search 的区别。

（4） match.group()方法，第一个加上括弧的子字符串在 match.group(1)的位置，不是 0。

（5） 提示：文件 readLines()方法返回的是一个字符串列表，是 list 类型的。

10.4 数据外化到 Excel

学习目标

学会用 unittest 测试 HTTP 接口，学习将测试数据外化到 Excel 文件中并学习读取 Excel 文件中的内容。

学习资源

Pyexcel 官网首页，网址如下：

https://pyexcel.readthedocs.io/en/latest/

一名来自国外的数据科学记者使用 Python 操作 Excel 的文章，Karlijn

Willems，网址如下：

https://www.datacamp.com/community/tutorials/python-excel-tutorial

pyexcel 官方网站中的数据转换章节：

https://pyexcel.readthedocs.io/en/latest/tutorial_data_conversion.html

知识准备

Excel 操作

在仅仅考虑数据处理的情况下，所有让开发人员一个一个 cell 的操作 Excel 的方法完全不符合 Python 的设计哲学！

> Simple is better than complex.
> 简单胜过复杂。
> Complex is better than complicated.
> 复杂胜过繁复。
>
> ————Zen of Python

用 Python 操作 Excel，如不考虑格式化、字体、颜色和图表的情况，我只推荐 pyexcel 库。

首先看一下 pyexcel 的口号：

> pyexcel - Let you focus on data, instead of file formats.
> Pyexcel – 让你聚焦数据，而不是文件格式。

还是太抽象，来点感性认识。以下示例内容来自 pyexcel 官网首页：

Suppose you want to process the following excel data：假设你要处理如图 10.7 所示的 Excel 数据。

Name	Age
Adam	28
Beatrice	29
Ceri	30
Dean	26

图 10.7 姓名和年龄数据

注释：Name：姓名，Age：年龄

Here are the example usages：下面是使用示例：

```
import pyexcel as pe
records = pe.iget_records(file_name="your_file.xls")
for record in records:
    print("%s is aged at %d" % (record['Name'], record['Age']))
```

输出结果如下：

```
Adam is aged at 28
Beatrice is aged at 29
Ceri is aged at 30
Dean is aged at 26
```

直接从 Excel 文件读取成字典，这才是 Python 的处理哲学！

Pyexcel 的安装

Pyexcel 的简易安装通过 pip 进行，如图 10.8 所示。

```
C:\Users\akui>pip install pyexcel
Collecting pyexcel
  Using cached pyexcel-0.4.5.tar.gz
Collecting pyexcel-io>=0.3.0 (from pyexcel)
  Using cached pyexcel-io-0.3.2.tar.gz
Collecting texttable>=0.8.2 (from pyexcel)
  Downloading texttable-0.8.7.tar.gz
Installing collected packages: pyexcel-io, texttable, pyexcel
  Running setup.py install for pyexcel-io ... done
  Running setup.py install for texttable ... done
  Running setup.py install for pyexcel ... done
Successfully installed pyexcel-0.4.5 pyexcel-io-0.3.2 texttable-0.8.7
```

图 10.8　pip install pyexcel

安装完毕后，还需要记住一点：Pyexcel 是通过插件的方式支持多种 Excel 的文件格式的，要支持不同的 Excel 文件格式，如 csv, xls, xlsx, ods，只需要安装对应的插件就可以了。插件的列表如下。

Package name	Supported file formats
pyexcel-io	csv, csvz [1], tsv, tsvz [2]
pyexcel-xls	xls, xlsx(read only), xlsm(read only)
pyexcel-xlsx	xlsx
pyexcel-xlsxw	xlsx(write only)
pyexcel-ods3	ods
pyexcel-ods	ods
pyexcel-odsr	ods(read only)
pyexcel-text	(write only)json, rst, mediawiki, html, latex, grid, pipe, orgtbl, plain simple
pyexcel-handsontable	handsontable in html
pyexcel-chart	svg chart

图 10.9 Package in excel-prepare-plugin

注释：Package name：包名，Supported file formats：支持的文件格式

由于后面的示例中要处理 xlsx 的 Excel 格式文件，所以我们需要安装 pyexcel-xlsx 插件。如图 10.10 所示。

```
C:\Users\akui>pip install pyexcel-xlsx
Collecting pyexcel-xlsx
  Downloading pyexcel-xlsx-0.3.0.tar.gz
Collecting openpyxl>=2.2.2 (from pyexcel-xlsx)
  Downloading openpyxl-2.4.5.tar.gz (180kB)
    100% |████████████████████████████████| 184kB 21kB/s
Requirement already satisfied: pyexcel-io>=0.3.0 in c:\python\python36\lib\site-packages (from pyexcel-xlsx)
Collecting jdcal (from openpyxl>=2.2.2->pyexcel-xlsx)
  Downloading jdcal-1.3.tar.gz
Collecting et_xmlfile (from openpyxl>=2.2.2->pyexcel-xlsx)
  Downloading et_xmlfile-1.0.1.tar.gz
Installing collected packages: jdcal, et-xmlfile, openpyxl, pyexcel-xlsx
  Running setup.py install for jdcal ... done
  Running setup.py install for et-xmlfile ... done
  Running setup.py install for openpyxl ... done
  Running setup.py install for pyexcel-xlsx ... done
Successfully installed et-xmlfile-1.0.1 jdcal-1.3 openpyxl-2.4.5 pyexcel-xlsx-0.3.0
```

图 10.10 pip install pyexcel-xlsx

使用 pyexcel 读取.xls 或者.xlsx 文件

To get your data in an array, you can use the get_array() function that is contained within the pyexcel package:

要将数据加载为数组 Array，你可以直接使用 pyexcel 包里的 get_array()方法。

```
# Import `pyexcel`
import pyexcel
# Get an array from the data
my_array = pyexcel.get_array(file_name="test.xls")
```

You can also get your data in an ordered dictionary of lists. You can use the get_dict() function:

你也可以将数据加载为一个有序的字典列表，可以使用 get_dict()方法。

```
# Get your data in an ordered dictionary of lists 将数据加载为一个有序的字典列表
my_dict = pyexcel.get_dict(file_name="test.xls", name_columns_by_row=0)
备注：name_columns_by_row  用于指定列的标题所在的行，默认为 0。

# Get your data in a dictionary of 2D arrays 将数据加载为一个二维数组的字典
book_dict = pyexcel.get_book_dict(file_name="test.xls")
```

备注：get_book_dict()和 get_dict() 的区别是前者默认加载整个 Excel 文件的所有 Sheet，后者默认只加载第一个 Sheet，两个都可以通过 sheet_name =”Sheet 的名字”，来指定加载的 Sheet 页。

Lastly, you can also just retrieve the records with pyexcel thanks to the get_records() function. Just pass the argument file_name to the function and you should be getting back a list of dictionaries:

最后，你也可以通过 pyexcel 的 get_records()方法加载 Excel 中的记录。仅仅通过给这个方法传递一个文件名作为参数，你就会得到一个字典的列表，也就是说，得到的返回值是一个列表，列表中的每条记录都是一个字典。

```
# Retrieve the records of the file 得到文件中的记录
records = pyexcel.get_records(file_name="test.xls")
```

备注：get_records 和上面的 iget_records 没什么区别，后者使用的内存更少，如果是特别大的 Excel 文件，建议采用后一个方法。

小练习：

读者自己建立一个 Excel，将上面提到的方法一个一个的试验一下，将返回的结果 print()出来。

方法的列表如下：

- pyexcel.get_records()
- pyexcel.iget_records()
- pyexcel.get_dict()
- pyexcel.get_book_dict()
- pyexcel.get_array()

以上方法，都可以跟以下两个常用参数：

file_name=和 sheet_name=，在不指定 sheet_name 的情况下，除了 get_book_dict()方法默认会加载所有的 Sheet 页中的数据，其他四个方法默认只会加载第一个 Sheet 页的数据。

用 pyexcel 写文件

Just like it's easy to load your data into arrays with this package, you can also easily export your arrays back to a spreadsheet. Use the save_as() function and pass the array and the name of the destination file to the dest_file_name argument:

就像你可以轻松的用这个包将 Excel 数据加载为数据阵列（如 dict、list 等）一样，你也可以轻松的将数据阵列导出到 Excel 数据页。使用 save_as()方法，将数组和目标文件名传递给 dest_file_name 参数。

```
# Get the data 声明一个数据阵列，如数组列表 data
data = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

# Save the array to a file 保存数组到文件
pyexcel.save_as(array=data, dest_file_name="array_data.xls")
```

Note that if you want to specify a delimiter, you can add the dest_delimiter argument and pass the symbol that you want to use as a delimiter in between "".

注意：如果你想指定分隔符，可以增加一个 dest_delimiter 参数，将希望使用的分隔符放在引号("")之间，传递给该参数。

If, however, you have a dictionary, you'll need to use the save_book_as() function. Pass the twodimensional dictionary to bookdict and specify the file name and you're good:

如果你的数据是字典，需要通过 save_book_as()方法，将二维字典变量传递给 bookdict 参数，并指定好文件名。

```
# The data
2d_array_dictionary = {'Sheet 1': [
                            ['ID', 'AGE', 'SCORE']
                            [1, 22, 5],
                            [2, 15, 6],
                            [3, 28, 9]
                           ],
                    'Sheet 2': [
                             ['X', 'Y', 'Z'],
                             [1, 2, 3],
                             [4, 5, 6]
                             [7, 8, 9]
                            ],
                    'Sheet 3': [
                             ['M', 'N', 'O', 'P'],
                             [10, 11, 12, 13],
                             [14, 15, 16, 17]
                             [18, 19, 20, 21]
                            ]}

# Save the data to a file
pyexcel.save_book_as(bookdict=2d_array_dictionary, dest_file_name="2d_array_data.xls")
```

Something that you should keep in mind when you use the code that is printed in the code chunk above is that the order of your data in the dictionary will not be kept. If you don't want this, you will need to make a small detour. You can read all about it here.

应该记住的是，当你采用上面的大块代码进行数据保存时，字典中的数据顺序是不保证能够按照原来的顺序保存的。如果希望保存字典中的数据顺序，则需要采取一些小的绕行方法。就是采用 OrderedDict。

下面的英文请读者自己读一下，希望读者能够理解我保留所有翻译内容原文的用心！

If you want to preserve the order of sheets in your dictionary, you have to pass on an ordered dictionary to the function itself. For example:

```
data = OrderedDict()
data.update({"Sheet 2": a_dictionary_of_two_dimensional_arrays['Sheet 2']})
data.update({"Sheet 1": a_dictionary_of_two_dimensional_arrays['Sheet 1']})
data.update({"Sheet 3": a_dictionary_of_two_dimensional_arrays['Sheet 3']})
pyexcel.save_book_as(bookdict=data, dest_file_name="book.xls")
```

挑战问题

编写一个 Python 的 unittest 程序 TestGetIPInfoByExcel.py，测试案例的要求与“10.2 案例编写”相同。但是，要求数据从一个包含测试案例 Sheet 页的 Excel 文件中读取。文件的内容示例如图 10.11 所示。

案例编号	案例意图	输出案例描述	ip	expect
1	给定一个正确的IP地址，应该能够正确返回地理信息	给定一个北京地区的IP地址，应该能够正确返回地理信息，并且city值为：北京市	124.126.228.193	北京市
2	给定一个非法的IP地址，应该返回错误信息：invaild ip.	给定一个非法的IP地址，应该返回错误信息：invaild ip.	124.126.228.	invaild ip.

图 10.11 Test Getipinfo By Excel

注意：不要短时间内大量发送这个请求，短时间发送大量请求会导致自己的 IP 被网站认为是恶意的，网站对该 IP 请求的响应时间会变长。

两个 unittest 的测试案例同上一小节。

将测试数据抽离出来写成一个文本文件，通过一个 test 方法 testRunTheCase 运行文本文件中的测试案例。

参见 10.3 小节从 github 下载的练习资料，在下载的资料 10.4 文件夹中，有一个 Excel 文件作为测试案例文件。

建议文件之间增加 5 秒等待时间。运行的结果如图 10.12 所示。

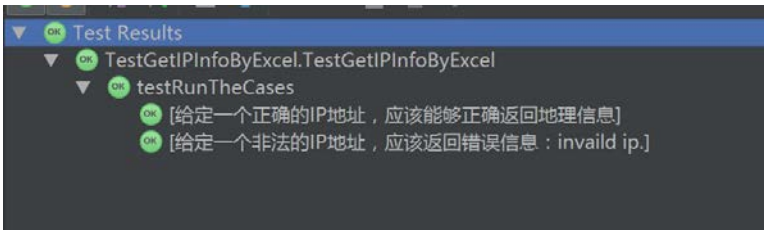


图 10.12 Test Getipinfo By Excel Output

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

语言基础知识要点

Excel 文件的读写有很多第三方库，包括 openpyxl、xlrd 和 pyexcel 等，这时只推荐 pyexcel。

拓展

到这里，读者可能已经会做 HTTP 接口测试了，在自己的日常工作中找几个 HTTP 的接口试一下吧！

读 者 圈

第十一章 普通 Web 接口测试（有状态）

11.1 接口探索

学习目标

下载并安装 custom-blog 网站，本地启动网站，并进行接口探索。

知识准备

安装与功能探索

访问 github：https://github.com/akuing/custom-blog 页面如图 11.1 所示。

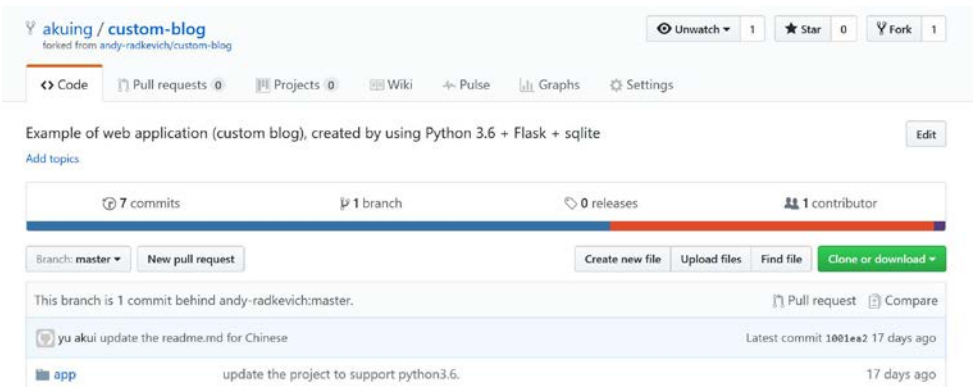


图 11.1 github 上的定制博客

点击“Clone or download”按钮，下载 ZIP 文件，解压后，按照 README 中的步骤安装和运行 custom-blog 应用。Custom-blog 是一个定制博客，用于 Web 应用示例，基于 Python3.6 + Flask 微框架 + sqlite 创建。

运行后的终端界面如图 11.2 所示。

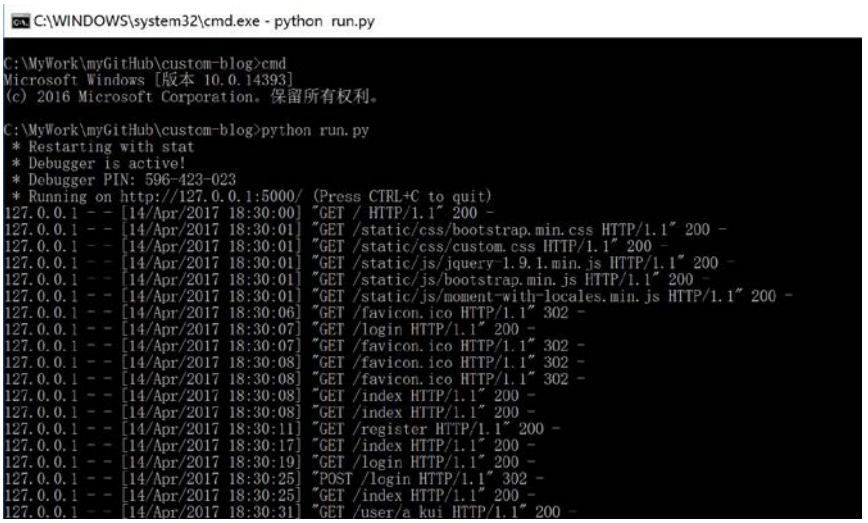


图 11.2 运行定制博客网站

运行后，按照要求打开浏览器，输入网址：http://127.0.0.1:5000 出现如图 11.3 所示的界面。

图 11.3 浏览定制博客网站

小练习：

请读者浏览 custom-blog 网站，列举一下网站的基本功能项。

注释：Log in：登录，Log out：退出，Register：注册，Edit Profile：编辑用户资料，Write New Post：发新贴子，Update：更新，Delete：删除，The title has a forbidden symbols：标题中包含被禁止的标识符，Field must be between 10 and 140 characters long：字段的长度必须在 10 到 140 之间。

接口探索

接口探索可以采用浏览器进行，Chrome 和 Firefox 均可。下面我用 Firefox 演示一个接口探索的例子：

用 Firefox 浏览器登录后，点击上面自己名字的超级链接，进入用户的 Profile 页，如图 11.4 所示。

图 11.4 个人配置信息

点击 Firefox 浏览器右侧的菜单按钮，会出现功能列表，选择点击“开发者”按钮或直接使用快捷键“ctrl+shift+I”。如图 11.5 所示。

图 11.5 Firefox 中的开发者模式-打开

出现“开发者”模式界面，点击开发者出现了一个菜单，选择菜单中的“网络”子菜单项或者使用快捷键 ctrl+shift+Q，如图 11.6 所示。

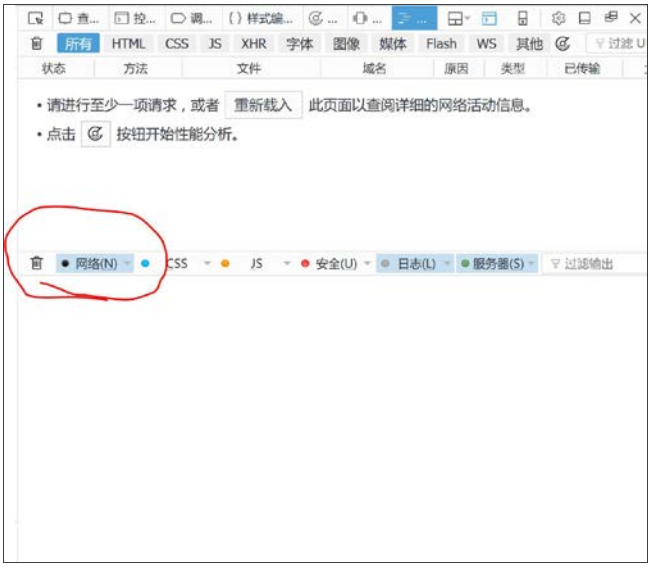


图 11.6 Firefox 中的开发者模式-网络

其他的对于接口探测工作暂时不必太过关注。点击左侧 custom-blog 网站页面中的“Edit Profile”超级链接，用户界面如图 11.7 所示。

图 11.7 更新个人配置信息

而“开发者”部分的界面会变成如图 11.8 所示。

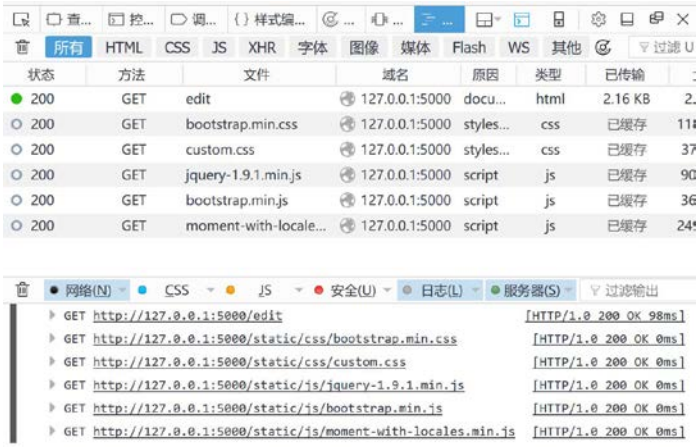


图 11.8 Firefox 中的开发者模式-请求

我们看到，虽然用户只是点击了左侧页面中的“Edit Profile”超级链接，浏览器中却发生了六个 Get 请求，这六个请求的后五个都是对 css 和 js 文件的请求，只有第一个是对 http://127.0.0.1:5000/edit 路径的 Get 请求。

鼠标点击 GET 前面的三角折叠，将请求的信息打开，出现如图 11.9 所示的界面。

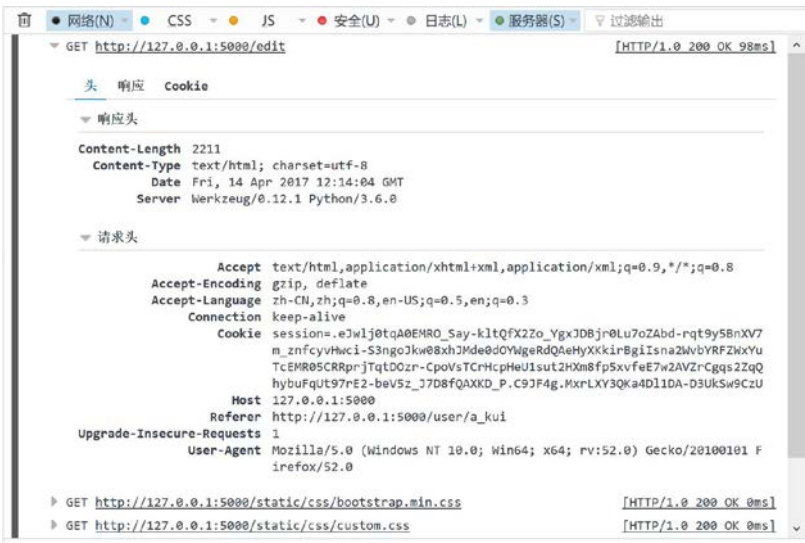


图 11.9 Firefox 中的开发者模式-请求数据

我们看到，这个 GET 请求的功能很明显就是查询用户的个人信息用于后续的编辑，请求的地址为指定的地址：http://127.0.0.1:5000/edit，而请求头中的 Referer 字段为：http://127.0.0.1:5000/user/a_kui，表明这个 GET 请求来自的页面链接的 URL。

响应报文头中 Content-Length 2211 是响应报文的长度，点击“响应”Tab 页，会看到返回报文的具体内容。报文中的 Content-Type text/html;charset=utf-8，表明这个响应报文的内容是 html 格式的采用 utf-8 编码。我们还看到 Cookie 中有 session，可以推测该接口需要登录后才能正常使用，因为，请求报文中没有指明是哪个用户，但是返回的页面却可以正确的返回用户 Profile 信息，说明服务器端是通过 session 中的信息得到当前用户的用户名或者用户 ID 的。

下面，我们探测了一个 GET 接口：http://127.0.0.1:5000/edit ，以此来回答一下，我称之为“接口八问”的八个问题。

该接口的请求地址是什么？

回答：**http://127.0.0.1:5000/edit 。**

该接口的功能描述是什么？

回答：**查询用户的个人信息。**

该请求接口是 GET 还是 POST？

回答：**GET 接口。**

该接口需要在登录情况下才有用吗？

回答：**是的。**

该接口有上送数据吗？上送的数据是什么？

回答：**没有上送数据。**

该接口返回的状态码是多少？

回答：**200**

该接口返回报文体的格式和编码是什么？

回答：**text/html;charset=utf-8，HTML 格式，编码为 UTF8。**

该接口返回的内容是什么？

回答：**用户的个人信息，具体可以通过响应报文中的报文体来查看。**

此时读者可以在简单地填写用户的 Profile 资料后，点击“Save”按钮，重点看一下通过网络监控到的请求和响应情况，你会看到一个 POST 接口。

尝试回答一下接口描述的“八个问题”，空出来的部分请读者根据自己的探测回答：

该接口的请求地址是什么？
回答：http://127.0.0.1:5000/edit。
该接口的功能描述是什么？
回答：修改用户的个人信息。
该请求接口是 GET 还是 POST？
回答：POST 接口。
该接口需要在登录情况下才有用吗？
回答：是的。
该接口有上送数据吗？上送的数据是什么？
回答：_____
该接口返回的状态码是多少？
回答：_____
该接口返回报文体的格式和编码是什么？
回答：_____
该接口返回的内容是什么？
回答：_____

挑战问题

（1） 本地安装 custom-blog 网站，通过终端启动 custom-blog 网站。

（2） 通过 http://127.0.0.1:5000 路径登录 custom-blog 网站，点击右上的 Log in，开始对网站的功能进行探测，并列举出至少六个 custom-blog 网站的功能接口。

（3） 修改自己的信息，写一个帖子，并发布。

这时的网站页面应该变成类似图 11.10 所示的样子。

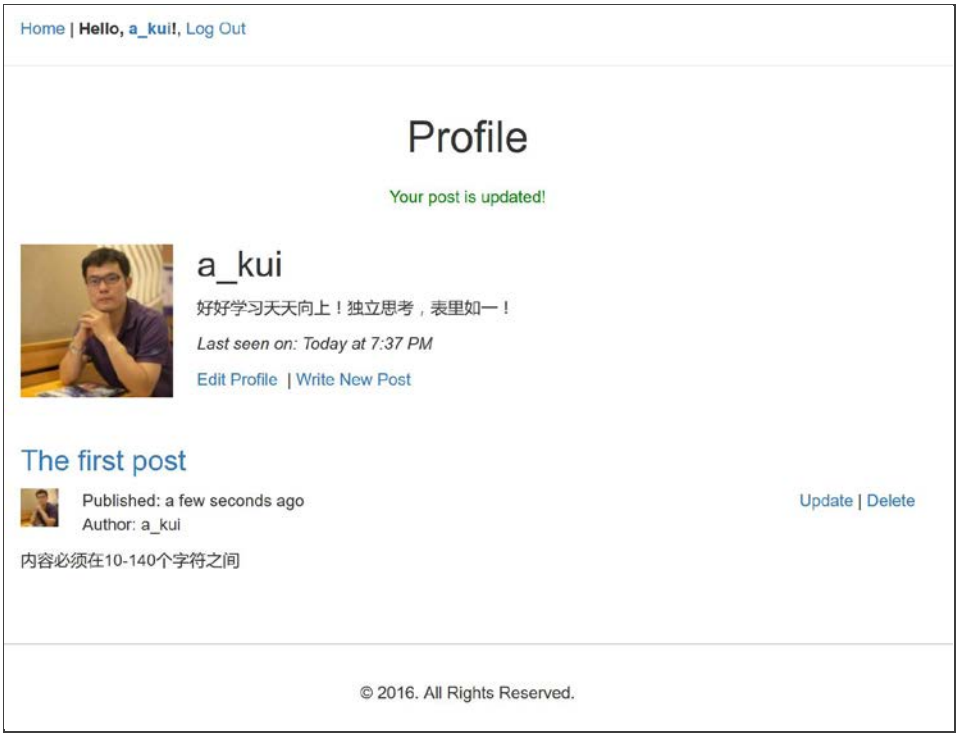


图 11.10 定制博客个人信息

尝试对网站中六个以上的 HTTP 接口进行定义，对每个发现的 HTTP 接口完成“八个问题”的回答。

知识点

接口探测要点

（1） 对网站的功能进行探测，并进行功能的列举，是测试人员的必备技能，只有了解了功能才能进行功能的测试。

（2） 通过 Firefox 进行 HTTP 接口探测是进行接口测试的必备技能，需要多看，多练习。

（3） 对 HTTP 协议不够了解的，在进行接口探测的时候会遇到一些困难，建议回到前面的**第三部分 初识 HTTP** 相应的章节进行复习。

拓展

是不是对于上面的头像怎么显示出来的很感兴趣？

添加头像的做法是：在 http://cn.gravatar.com/网站，用你在定制“博客网站”中注册的同一个邮箱注册一个用户，然后，上传你的头像图片，就可以啦。

只要你启动本地的定制博客，保持电脑处于联网状态，头像就会出现。有兴趣的读者可以阅读一下定制博客网站的代码，秘密就在 models.py 文件。

尝试退出登录后，直接在浏览器的地址栏输入 http://127.0.0.1:5000/edit 链接，观察网站的响应，并通过开发者模式中的网络监控查看详细的请求和响应信息，确认一下是否一定需要在登录的情况下，才能够正常使用这个接口，以及在出错的情况下网络监控到的信息是什么。

11.2 在返回页面中定位检查点

学习目标

学习使用 Python 的 BeautifulSoup 库解释 HTML 页面，定位取得其中的 csrf_token 元素的值。

学习资源

HTML 资源 http://www.runoob.com/html/html-tutorial.html
http://www.w3cschool.cn/html/
BeautifulSoup 资源 https://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/
http://beautifulsoup.readthedocs.io/zh_CN/latest/

知识准备

HTML 语法测试

本节内容假设为你对 HTML 已有初步的了解，如果你不知道 HTML 为何物，看不懂下面的代码，回答不了下面这段代码的三个问题，请先用一到两天的时间学习一下 HTML。学习资料如下：

http://www.runoob.com/html/html-tutorial.html
http://www.w3cschool.cn/html/

学习后，请参与如下测试：

http://www.w3cschool.cn/html/html-html_quiz.html

如果你的得分不及格（即 20 道题目中没有正确回答 12 道题以上），并且还是看不懂下面的内容，建议你继续学习 HTML，直到完全看懂下面的内容为止。

```
<form action="" method="post" name="login">
  <input id="csrf_token" name="csrf_token" type="hidden" value="1493480356##f5590d8719f6b312b28b76f8da59a8f8f2aff65e">
  <p>
    Email:<br>
    <input id="email" name="email" size="40" type="text" value=""><br>
    <br>
    Password:<br>
    <input id="password" name="password" size="40" type="password" value=""><br>
    <br>
    Password Confirmation:<br>
    <input id="cpassword" name="cpassword" size="40" type="password" value=""><br>
    <br>
  </p>
  <p><input class="btn btn-primary" type="submit" value="Register"></p>
</form>
```

为了证明你可以看懂上面的 HTML 代码，请回答以下三个问题：

（1） 上面密码域的最大输入长度为多少？

回答：_____

（2） 选择题，让上面的密码域可以对输入的字符进行“*”隐藏的关键字语句是哪句（ ）？

A. name="password"　B. id="password"　C. type="password"

（3） 判断题，上面的代码实现的 HTML 页面上能够看到"1493480356##f5590d8719f6b312b28b76f8da59a8f8f2aff65e"字符串。

上面的陈述是否正确，为什么？

回答：_____

请确定自己已经正确回答了上面的三个问题，否则，请继续学习 HTML 教材。

初识 Beautiful Soup

“Beautiful Soup 是一个可以从 HTML 或 XML 文件中提取数据的 Python 库。它能够通过你喜欢的转换器实现惯用的文档导航、查找、修改文档的方式。”

——Beautiful Soup 4.4.0 文档，http://beautifulsoup.readthedocs.io/zh_CN/latest/

简单来说，Beautiful Soup 就是一个可以解释 HTML 或者 XML 的 Python 库，本文主要讨论 HTML 的解释。所谓解释，就是读得懂文档的结构，并可以根据需要从文档中提取特定的内容。

为了演示 BeautifulSoup 的强大用途，我们先定义一段 HTML 代码。

```
<html><head>标题</head><body><b>正文</b><a href="http://example.com/link1" id="link1">链接一</a>
<a href="http://example.com/link2" id="link2"> 链 接 二 </a><a href="http://example.com/link3" id="link3">链接三</a>
</body></html>
```

首先，我们对这段看似很乱的 HTML 代码进行美化。

首先进行 BeautifulSoup 库的安装，BeautifulSoup 库并不是 Python 的标准库，需要安装后才能使用，安装命令如下：

```
$ pip install beautifulsoup4
```

安装完成后，在交互界面中尝试以下代码：

```
>>> from bs4 import BeautifulSoup
>>> htmltxt="'<html><head> 标 题 </head><body><b> 正 文 </b><a href="http://example.com/link1" id="link1" name="link1" >链接一</a>
... <a href="http://example.com/link2" id="link2" name="link2"> 链 接 二 </a><a href="http://example.com/link3" id="link3" name="link3">链接三</a>
... </body></html>'"
>>> soup=BeautifulSoup(htmltxt,"html.parser")
>>> print(soup.prettify())
<html>
```

```
 <head>
  标题
 </head>
 <body>
  <b>
   正文
  </b>
  <a href="http://example.com/link1" id="link1" name="link1">
   链接一
  </a>
  <a href="http://example.com/link2" id="link2" name="link2">
   链接二
  </a>
  <a href="http://example.com/link3" id="link3" name="link3">
   链接三
  </a>
 </body>
</html>
>>>
```

下面对示例中的代码进行解释：

```
from bs4 import BeautifulSoup
```

本行代码是引入 BS4（即 BeautifulSoup 4）库中的 BeautifulSoup 类，现在 BeautifulSoup 已经更新到 4.4 版本，建议不要再使用 BeautifulSoup2 或者 BeautifulSoup3 版本的库。

```
soup=BeautifulSoup(htmltxt,"html.parser")
```

本行代码将一个包含 HTML 文本的变量传递给 BeautifulSoup 类，用于构造一个 BeautifulSoup 对象，在解释 HTML 文本的时候，采用 html.parser 解释器。这个解释器是 Python 中标准的 HTML 解释器，还有一些其他的解释器，你可以通过阅读学习资料链接中的内容学习，一般情况下，这一个 HTML 解释器就足够使用了。

```
print(soup.prettify())
```

本行代码的作用就是打印经过美化的 HTML 源代码，而 soup.prettify()方法的作用就是将 Beautiful Soup 的文档树格式化后，以 Unicode 编码作为字符串返回，每个 XML/HTML 标签都独占一行。

注意，BeautifulSoup 并不是 Python 的标准库，需要安装后才能使用，安装命令如下：$ pip install beautifulsoup4。

在 HTML 中定位元素

BeautifulSoup 有很强大的解释、导航、搜索功能，本书仅仅介绍其中最常用的搜索定位功能。在 BeautifulSoup 中提供搜索功能的方法是 find_all()。

该方法的定义如下：

```
def find_all(name=None, attrs={}, recursive=True, text=None,
        limit=None, **kwargs)
```

该方法有 6 个参数，我们重点介绍其中的 name、**kwargs 和 attrs：

name 参数：

name 参数用于告诉 BeautifulSoup 只考虑特定名字的 Tag。这里引入了一个 Tag 的概念，这是 BeautifulSoup 中的一个重要概念。在 BeautifulSoup 中，所有对象可以归纳为 4 种，即 Tag，NavigableString，BeautifulSoup 和 Comment。Tag 对应的是 XML 或者 HTML 中的标签。

我们都知道，HTML 是一种标记语言，而在标记语言中，核心的概念就是标记的标签，在 HTML 中标签是由尖括号包围的关键词，如< html >，而标签一般是成对出现的，比如 < b > 和 < /b > ，我们一般称前一个为开始标签，后一个为结束标签。

回到 name 参数，这里需要传递给 name 的就是标签的名字。

还是使用本小节上文中的那个杂乱的 HTML。

```
>>> from bs4 import BeautifulSoup
>>> htmltxt="'<html><head> 标 题 </head><body><b> 正 文 </b><a href=
"http://example.com/link1" id="link1" name="link1">链接一</a>
... <a href="http://example.com/link2" id="link2" name="link2"> 链 接 二 </a><a
href="http://example.com/link3" id="link3" name="link3">链接三</a>
... </body></html>"'
```

```
>>> soup=BeautifulSoup(htmltxt,"html.parser")
>>> print(soup.find_all("a"))
[<a href="http://example.com/link1" id="link1" name="link1"> 链 接 一 </a>, <a href="http://example.com/link2" id="link2" name="link2"> 链 接 二 </a>, <a href="http://example.com/link3" id="link3" name="link3">链接三</a>]
```

通过这段代码我们看到，soup.find_all(“a”)返回了一个 list 类型的数据列表，里面是 HTML 源代码中所有标签标记的内容。

```
>>> for element in soup.find_all("a"):print(element)
...
<a href="http://example.com/link1" id="link1" name="link1">链接一</a>
<a href="http://example.com/link2" id="link2" name="link2">链接二</a>
<a href="http://example.com/link3" id="link3" name="link3">链接三</a>
>>>
```

如果用 for 循环一一打印出来会使结果更加清晰。

**kwargs 参数：

该参数要求指定一个键/值对，find_all()方法在搜索的时候，会根据键元素来搜索指定的 Tag 是否有匹配的属性，匹配到与键元素相同的属性后，还要求属性的值与参数中的值元素也相同，才算匹配成功。

继续上面的代码：

```
>>> print(soup.find_all("a",id="link2"))
[<a href="http://example.com/link2" id="link2" name="link2">链接二</a>]
>>>
```

此时，仅仅会显示 id="link2"的标签标记的内容。

我们看到，上面的超级链接的标记里也定义了 name 属性，但是，运行如下代码：

```
>>> print(soup.find_all("a",name="link2"))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: find_all() got multiple values for argument 'name'
>>>
```

我们却得到了 TypeError 的报错信息。原因是 find_all()方法的第一个参数名为 name，该参数传入的应该是标签的名字，而这里又传入了一个 name="link2"，这给 find_all()方法带来了困扰，不知道两个 name 参数应该使用哪一个，故而报错。

那么，如何才能通过 name 属性来定位标签要素呢？我们来看 attrs 参数。

attrs 参数：

attrs 参数是一个字典类型的参数，用来指定特定标签属性。find_all()方法在搜索的时候，就会将这个参数作为指定的 Tag 属性来进行匹配搜索。

继续上面的代码，如果我们要使用 name 属性进行标签要素的定位，代码如下：

```
>>> print(soup.find_all("a",attrs={"name":"link3"}))
[<a href="http://example.com/link3" id="link3" name="link3">链接三</a>]
>>>
```

这样就可以通过 name 属性定位元素了。一般来讲，掌握了以上 3 个参数的使用，读者就可以通过 BeautifulSoup 库来提取 HTML 报文中的特定标签内容了。

输出标签中的特定属性

在 BeautifulSoup 的 Tag 对象中有很多方法和属性，其中 name 和 attributes 两个属性最为重要。

每个 Tag 对象都有自己的名字，通过.name 来获取。

一个 Tag 可能有很多个属性。比如，下面的代码中，就是一个 Tag 对象。

```
<a href="http://example.com/link3" id="link3" name="link3">链接三</a>
```

这个 Tag 对象有一个 href 属性，一个 id 属性和一个 name 属性。Tag 对象的属性的操作方法与字典相同。

现在继续上面的例子。这次我们准备输出 name 为 link3 的超级链接标签中的超链接地址。

```
>>> print(soup.find_all("a",attrs={"name":"link3"})[0].attrs["href"])
http://example.com/link3
```

这段代码比较复杂，有偷懒之嫌，如果能够读懂这段代码的意思，说明读者已经较好地掌握了提取和输出 Tag 对象中的特定属性的要义。

下面是一个更加优雅的写法：

```
>>> taglist = soup.find_all("a",attrs={"name":"link3"})
>>> print(taglist[0].attrs["href"])
http://example.com/link3
```

尝试回答如下问题：

为什么上面的代码中要有[0]?

回答：_____

下面这行代码输出的结果是什么?

print(soup.find_all("a",attrs={"name":"link3"})[0].name)

回答：_____

挑战问题

编写一个 Python 程序 ShowCsrfToken.py，用 HTTP 的 GET 命令，获取 custom-blog 网站的登录页面内容(IP 地址为：http://127.0.0.1:5000/login)，并打印出登录页面中 name 为 csrf_token 的元素的 Value 值。结果如图 11.11 所示。

图 11.11 显示定制博客网站的 Csrf Token

注意运行 ShowCsrfToken.py 程序的时候，要保证 custom-blog 网站在本机处于启动状态。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要

闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

知识点

HTML 语法

- HTML 语言的定义和标签的概念。
- 对 HTML 源代码的认识。
- 知道以下标签的含义，并且了解这些标签的常用属性。
- <html> <head> <body> <a> <b> <p> <input>
 <form>

BeautifulSoup 库的使用

- 库的安装和引入。
- find_all()方法的使用。
- Tag 对象的 name 和 attrs 属性。

11.3 第一个测试案例

学习目标

编写一个有状态的测试案例，尝试使用 Python 接口方式登录到 custom-blog 网站。

知识准备

探测登录接口

通过 11.2 小节的接口学习，相信读者已经发现了一个登录接口。下面就一起进行登录接口的接口探测，尝试回答一下关于登录接口的“八个问题”。

登录 custom-blog 首页，点击最上方 Home 旁边的“Log in”链接，出现登录界面。打开“开发者”模式，选择“网路”，出现如图 11.12 所示的界面。

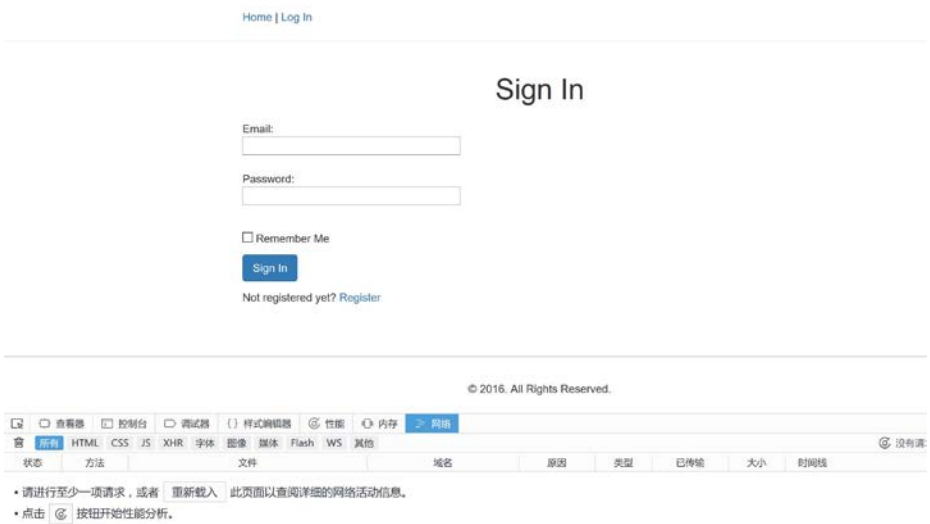


图 11.12 定制博客网站登录

下半部分是“开发者”模式界面，用于显示网络活动的信息。输入用户名和密码，点击“Sign In”按钮，看到如图 11.13 所示的界面。

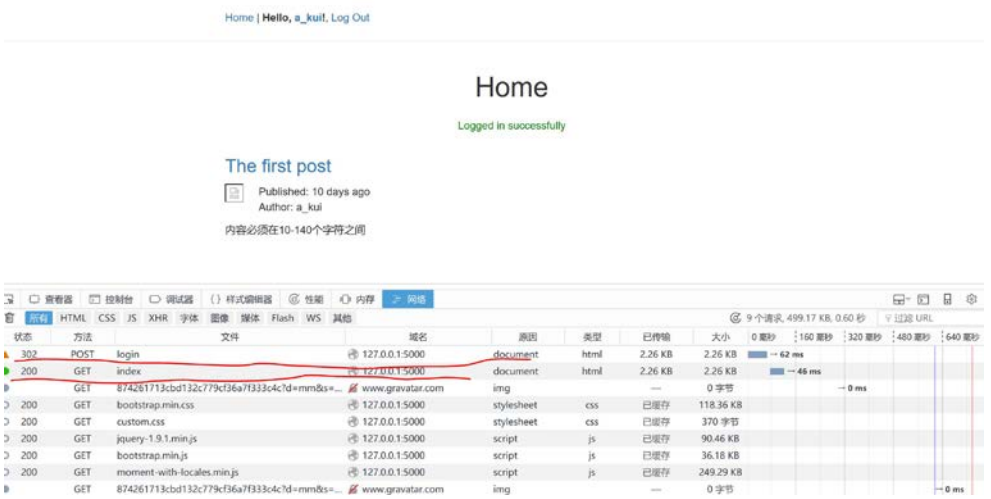


图 11.13 登录成功

绿色的“Logged in successfully”表示，你已经登录成功。注意下面“开发者”模式界面中的两个请求。第一个是向/login 的 POST 请求，返回码为 302；第二个是向/index 的 GET 请求，返回码是 200。

鼠标点击第一行的请求，会出现请求的详细信息，如图 11.14 所示。

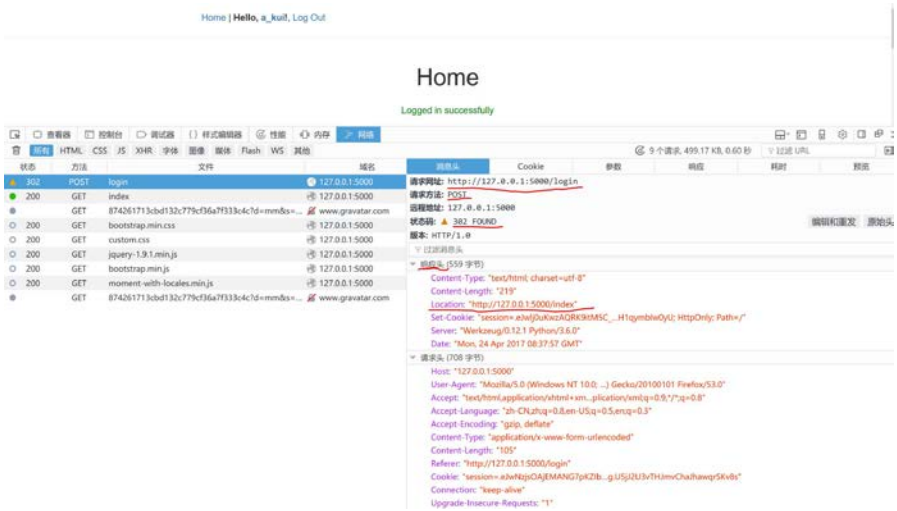


图 11.14 登录 POST 请求信息

请注意其中的划线部分，我们看到消息头中明确的表明，请求的网址为：http://127.0.0.1:5000/login，采用的请求方式是 POST，请求的返回状态码为 302。我们知道 302 是一个跳转状态码，浏览器收到这个状态码后就会向响应报文中的 Location 域指定的地址发起 GET 请求，在这里响应头中 Location 域的值为 http://127.0.0.1:5000/index。再看一下参数，“参数”TAB 页，如图 11.15 所示。

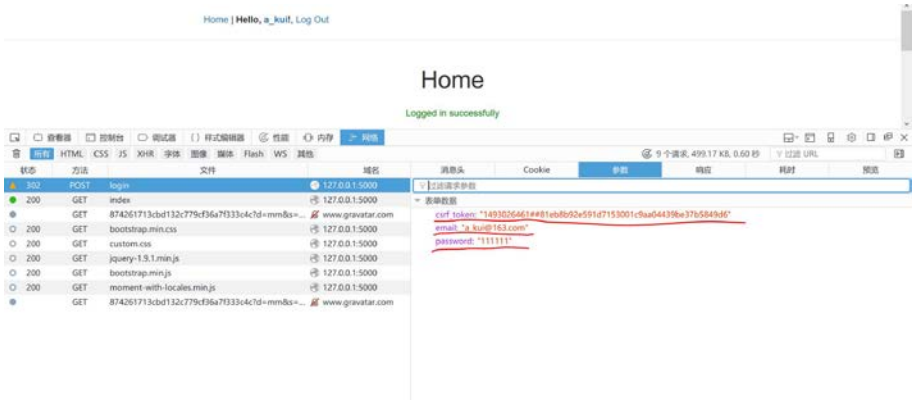


图 11.15 登录请求参数

我们看到上送的参数有三项内容：csrf_token，email，password。这就是我们探测到的上送数据。

至此，我们就可以回答出登录接口的“八个问题”了。

该接口的请求地址是什么？

回答：http://127.0.0.1:5000/login。

该接口的功能描述是什么？

回答：通过用户名密码登录到 custom-blog 网站。

该请求接口是 GET 还是 POST？

回答：POST

该接口需要在登录情况下才有用吗？

回答：不需要登录。

该接口有上送数据吗？上送的数据是什么？

回答：有，包括三个数据 csrf_token，email，password。

该接口返回的状态码是多少？

回答：302，Location 为 http://127.0.0.1:5000/index。

该接口返回报文体的格式和编码是什么？

回答：HTML 格式，编码为 UTF-8。

该接口返回的内容是什么？

回答：302，跳转到 index 页，界面会显示“Logged in successfully”。

挑战问题

编写一个 Python 的 unittest 程序 TestLogin.py，测试 custom-blog 网站在用户名、密码正确的时候，允许登录。如图 11.16 所示。

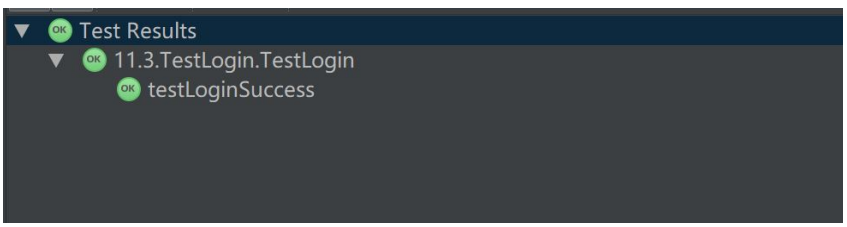


图 11.16 Test Login Success

验证测试案例通过需要达到以下两个条件：

（1） POST 请求返回的 HTTP 状态码为 200。

（2） 返回的报文内容中包含“Logged in successfully”。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

难点提示

登录 POST 请求的 csrf_token 要从登录界面的 HTML 中获取，登录界面的 HTML 内容通过向 http://127.0.0.1:5000/login 发送 GET 请求获取。

拓展

在验证返回的报文内容中包含“Logged in successfully”的时候，可以有多种验证的方法。

- self.assertTrue(XXXXXXXXXX)
- self.assertIn(xxxxx,xxxxxxx)

尝试分别用 assertTrue 和 assertIn 两个函数达到在验证返回的报文内容中包含“Logged in successfully”的目的。再看看自己是否能够找到更多的方法。

11.4 更多测试案例

学习目标

尝试使用 Python 接口方式在 custom-blog 网站成功注册一个新用户。

学习资源

百度百科 CSRF：
http://baike.baidu.com/item/CSRF
跨域 post 及使用 token 防止 csrf 攻击：
http://blog.csdn.net/u013934914/article/details/50428869

知识准备

探测注册接口

相信读者通过之前的接口探索，已经发现了一个注册接口，注册接口与上一小节的登录接口有很多相似的地方。

在 custom-blog 网站注册，页面如图 11.17 所示。

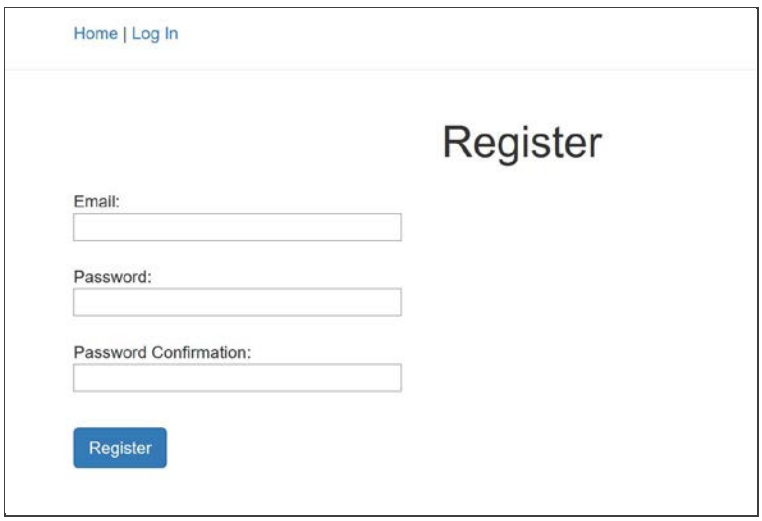


图 11.17 定制博客网站注册

作为一名测试人员，针对这样一个有三个输入框，一个按钮的注册页面，你会设计多少个测试案例场景？如果只让你测试三个案例，你会选择测试哪三个案例？

1、_____

2、_____

3、_____

下面我们尝试对注册接口进行探测，并给出接口的定义，回答注册接口的“八个问题”。

在注册页面打开“开发者”模式界面，显示网络活动的信息。输入用于注册的邮箱和两遍密码，点击“Register”按钮。

你会的看到如图 11.18 所示的界面。

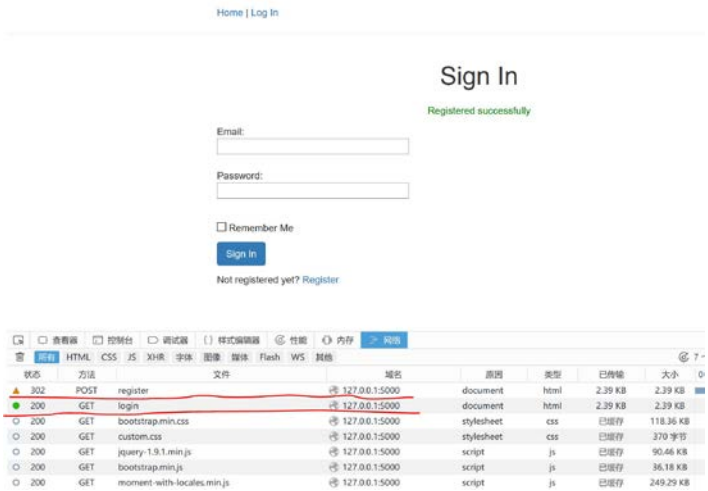


图 11.18 注册成功

“Sign In”下面出现绿色的“Registered successfully”表示，你已经注册成功。

注意在“开发者”模式界面中的两个请求：

- 第一个是向/register 的 POST 请求，返回码为 302。
- 第二个是向/login 的 GET 请求，返回码是 200。

点击第一行的请求，会出现请求的详细信息，如图 11.19 所示。

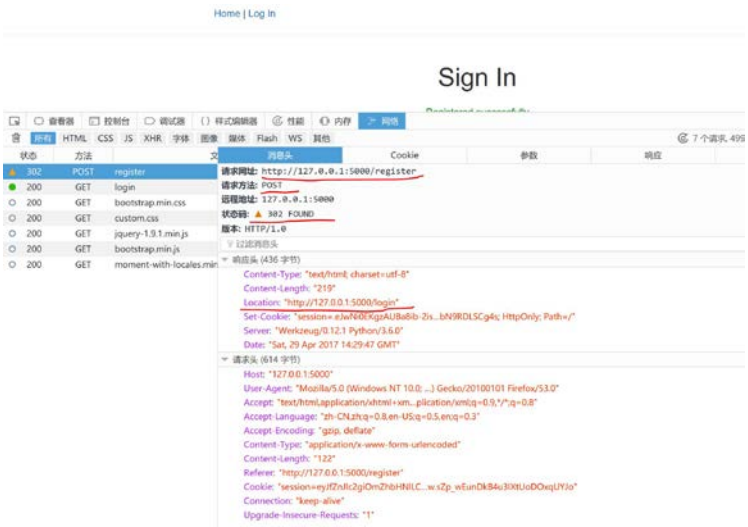


图 11.19 注册请求数据

请注意其中划线的部分，其明确的表明：

- 请求的网址为：http://127.0.0.1:5000/register。
- 采用的请求方式是 POST。
- 请求的返回状态码为 302。我们知道 302 是一个跳转状态码，浏览器收到这个状态码后就会向响应报文中的 Location 域指定的地址发起 GET 请求。
- 我们看到响应头中 Location 域的值为 http://127.0.0.1:5000/login。

再看一下请求上送的参数，点击“参数”页，如图 11.20 所示。

图 11.20 注册请求参数

我们看到上送的参数有四项内容：csrf_token，email，password，cpassword。

这就是我们探测到的上送数据。至此，我们就已经完成了接口的探索，相信读者也已经可以自信地回答描述接口定义的“八个问题”了。

小练习：

请读者根据以上探测的结果，回答针对“用户注册”接口的“八个问题”。

csrf_token 的由来

CSRF（Cross-site request forgery）跨站请求伪造，也被称为“One Click Attack”或者 Session Riding，通常缩写为 CSRF 或者 XSRF，是一种对网站的恶意利用。

——百度百科

CSRF 攻击可以简单地理解为，攻击者利用服务网站设计上的漏洞，伪造用户的请求，冒充用户在网站上发起操作，所以，这种攻击方式才被命名为“跨站请求伪造”。

也就是说：，攻击者盗用了合法用户的身份，以合法用户的名义在存在漏洞的服务网站上发起看似正常的操作。比如，攻击者可以用合法用户的身份，发送邮件，在论坛发帖，发起评论，修改密码进而盗取用户的账号，甚至购买商品，发起转账等等。

csrf_token，又被称做 One-Time Tokens，就是为避免这种攻击进行的一种设计。具体来说，就是服务网站在客户端获取请求表单的填写页面的时候，为每一个不同的表单分配一个伪随机值，不同的表单包含一个不同的伪随机值，只有伪随机值合法的表单才能被服务网站正确的处理。

我们通过上面的注册接口的探索可以看到，要通过发送到 http://127.0.0.1:5000/register 的 POST 完成注册，必须先得到上送参数 csrf_token 的值，而 csrf_token 的值包含在 POST 请求之前，对 http://127.0.0.1:5000/register 发送的 GET 请求得到的 HTML 中，具体可以参考上一小节的练习。

关于 CSRF 的更详细的内容可以参考如下文章：

http://www.cnblogs.com/hyddd/archive/2009/04/09/1432744.html

https://blog.tonyseek.com/post/introduce-to-xss-and-csrf/

BeautifulSoup 输出文本

接口测试返回 HTML 页面的时候，需要对页面中的内容进行检查，以便确定是否得到了预期的返回页面。最常见的就是检查 HTML 页面中是否包含了特定的文本。

BeautifulSoup 定位 Tag 后，有两种方式得到 Tag 的文本：string 和 get_text()。

string 方式

如果定位到的 Tag 对象只有一个节点，或者只有一个子节点，可以通过.string 的方式输出文本的内容。

我们还是用上一个小节中的 HTML 作为例子，读者可以到 github 地址：https://github.com/akuing/python-and-http-interface-test 下载全部代码，找到“11.4”文件夹中的 testfile.html。

在本地编写如下代码，命名为 test.py，与 testfile.html 放在同一个目录下：

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(open("testfile.html","r",encoding="utf-8"),"html.parser")
print(soup.find("a",id="link1").string)
```

运行程序，得到如下输出：

C:36.exe C:/MyWork/myBook/用 Python 做 HTTP 接口测试/source/3.8/test.py

```
链接一
```

通过 open()方法打开 testfile.html 文件，并且将文件句柄对象作为参数传递给 BeautifulSoup 方法，采用 html.parser 作为 HTML 的解释器。通过定位的 Tag 对象的.string 属性，获得对象的文本内容。

get_text()方式

如果 Tag 对象有多个包含文本内容的子对象，.string 属性会返回 None。

比如，我们将上文 test.py 源文件的最后一行改成：

```
print(soup.find("body").string)
```

再运行 test.py，就会得到“None”。因为 Body 标签下有多个链接子节点存在，每个子节点都有文本内容。此时，如果希望获取所有的子节点的文本内容，应该使用 get_text()方法。

将 test.py 源文件的最后一行修改为：

```
print(soup.find("body").get_text())
```

运行代码后，会得到如下输出：

```
正文
链接一
链接二
链接三
```

当然，即使上面 Tag 对象在只有一个节点的情况下，也可以使用 get_text()方法。

assertRegex()方法

assertRegex()方法是 TestCase 类提供的用于检查案例是否通过的方法，其定义如下：

```
assertRegex(text, regex, msg=None)
```

- text 参数：被检查的字符串，一般是从测试结果中提取出来的字符串。
- regex 参数：用于检查字符串是否符合预期的匹配模式，可以是普通字符串，也可以是正则表达式。
- msg 参数：可选参数，同于提醒这个检查的含义的备注信息。

需要提醒的是，如果 regex 参数中是普通的字符串，在 text 中的任何位置包含 regex 定义的字符串，该 assert 都会通过，这表明 assertRegex 不要求必须从 text 的最开始位置进行匹配。

如下面的案例是通过的：

```
import unittest
class TestRegex(unittest.TestCase):
    def testShouldPassGivenMatchInMiddle(self):
        self.assertRegex("haohaoxuexitiantianxiangshang","xuexi","应该通过")
```

regex 参数可以包含任何正则表达式内容，关于正则表达式的学习，这里就不做多余的描述了。

挑战问题

编写一个 Python 的 unittest 程序 TestCustomBlogRegister.py，测试如下场景。场景一：邮箱合法，密码两次一样，预期结果为注册成功。如图 11.21 所示。

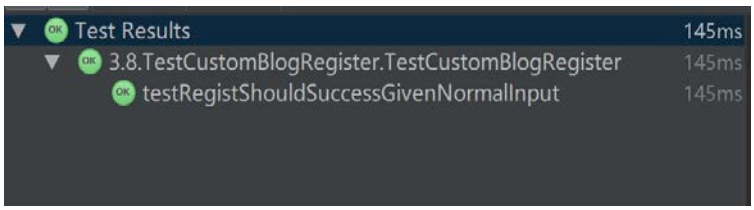


图 11.21 注册成功测试案例

注意运行 TestCustomBlogRegister.py 程序的时候，要保证 custom-blog 网站在本机是处于启动状态的。

读者可以通过接口探测，具体看一下注册成功的情况下，返回的 HTML 报文中是否包含“Registered successfully”字符串，并以此作为判断测试案例通过的验证条件之一。

建议使用 BeautifulSoup 进行字符串的定位，使用 assertRegex()方法作为判断条件。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

难点提示

或许，你编写的案例只有第一次是运行成功的，第二次运行的时候就会报错，修改一下注册邮箱又会成功一次。不过没有关系，这个问题，我们会在下一个小节进行解决。

知识点

BeautifulSoup 和 unittest

- BeautifulSoup 定位和输出文本的两种方式，以及两种方式之间的区别。
- assertRegex()方法的学习。

拓展

读者在进行注册成功的判断时，需要在返回的 HTML 中进行定位，并提取文本字符串，然后，判断文本字符串中是否包含“Registered successfully”字样。在上面的练习中已经推荐读者采用 assertRegex()方法进行判断。请读者思考一下，通过 assertEqual()方法和 assertTrue()方法是否也可以用于检查以上的字符串包含的逻辑判断？如何做到？

执行注册的测试案例的时候，是不是出现过如下的错误信息？想想如何解决？

```
Failure
Traceback (most recent call last):
  File "C:\MyWork\myBook\source\3.8\TestCustomBlogRegister.py", line 11, in
```

```
testRegistShouldSuccessGivenNormalInput
        self.assertRegex(returnStr,"Registered successfully")
    AssertionError: Regex didn't match: 'Registered successfully' not found in 'Such user already is avalable '
```

11.5 重复执行注册失败了

学习目标

尝试使用 Python 接口方式在 custom-blog 网站成功注册一个新用户。学习测试案例的场景准备和完成后的清理概念，初步了解 sqllite3。

学习资源

文件操作：
https://docs.python.org/3.6/library/os.html
https://docs.python.org/3.6/library/shutil.html#module-shutil
unittest：
https://docs.python.org/3.6/library/unittest.html#organizing-tests
SQLITE 数据库：
http://www.w3cschool.cn/sqlite/
http://www.runoob.com/sqlite/sqlite-tutorial.html
https://pysqlite.readthedocs.io/en/latest/sqlite3.html
SQL 教程：
http://www.w3cschool.cn/sql/

知识准备

关于 setUp()和 tearDown()

读者在编写上一小节案例的过程中已经发现，编写的案例只有第一次是运行成功的，第二次再运行时就会报错，修改一下注册邮箱又会成功一次。这主要是由于用户注册后，当再一次注册时，系统会报告该用户已经注册，不允许进行重复注册，所以，案例就失败了。

我们在上文介绍过，setUp()和 tearDown()两个方法是分别用来在测试案例执行之前进行测试准备工作和在测试案例执行之后清理测试现场的。以下代码来自：https://docs.python.org/3.6/library/unittest.html#organizing-tests。

```
import unittest
class WidgetTestCase(unittest.TestCase):
    def setUp(self):
        self.widget = Widget('The widget')

    def tearDown(self):
        self.widget.dispose()

    def test_default_widget_size(self):
        self.assertEqual(self.widget.size(), (50,50),
                         'incorrect default size')

    def test_widget_resize(self):
        self.widget.resize(100,150)
        self.assertEqual(self.widget.size(), (100,150),
                         'wrong size after resize')
```

在 test_default_widget_size()和 test_widget_resize()两个测试案例执行之前，unittest 测试框架都会先调用 setUp()方法，用于构建 widget 对象；同样，在测试案例执行之后，会调用 tearDown()方法，对构建的 widget 对象进行销毁。

与 setUp()和 tearDown()对应的，在类的层级和模块的层级也有相应的测试准备和清理测试现场的方法，运行的机制也是类似的。

- 类的层级上是：setUpClass()和 tearDownClass()。
- 模块的层级上是：setUpModule()和 tearDownModule()。

初识 SQLite 数据库

上文练习中用到的“定制博客 custom-blog”的例子，是我在 github 上 fork 自一位叫 Andy Radkevich 的程序编写者。原来的代码是基于 Python2.7 的，我将其迁移到了 Python3.6。

定制博客的网站结构非常简洁，基于 Flask 构建，采用了 SQLite3 数据库。

SQLite 数据库是一个在全球范围内广泛采用的、开源的轻量级数据库，

其官方网站为：http://www.sqlite.org。与其他数据库如 MySQL、Oracle 或者 SQLServer 不同，SQLite 属轻量级，不需要一个持续运行的守护进程，而是在需要的时候直接访问数据库存储文件。

在 custom-blog 工程文件夹中有一个 app.db 的文件，这个文件是“定制博客网站”的 SQLite 数据库存储文件。

SQLite 数据库的安装非常简单，可以直接通过网站下载，网站的地址为：http://www.sqlite.org/download.html。

- Windows 用户下载：sqlite-tools-win32-x86-xxxxxxxxx.zip
- MacOS 用户下载：sqlite-tools-osx-x86-xxxxxxxxx.zip
- Linux 用户下载：sqlite-tools-linux-x86-xxxxxxxxx.zip

下载的压缩包中包含三个可执行的文件：sqldiff、sqlite3 和 sqlite3_analyzer。

双击 sqlite3 可执行文件，终端输出如下：

```
SQLite version 3.18.0 2017-03-28 18:48:43
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

这表示我们已经成功的运行起来 sqlite3 的操作界面了。

```
sqlite> .open C:\MyWork\custom-blog\app.db
Error: unable to open database "C:MyWorkcustom-blogpp.db": unable to open database file
sqlite> .open C:\\MyWork \\custom-blog\\app.db
sqlite>
```

使用.open 命令打开“custom-blog 网站”的 sqlite 数据库存储文件(app.db)。

注意，在 Windows 的命令行界面下，需要用两个“”代表一个""，学过正则表达式的同学一定知道这是怎么回事。

使用.help 命令可以得到所有的命令提示，所有的 SQL 语句也都可以在这里运行。

下面我们简单地了解一下 custom-blog 网站数据库的结构。

```
sqlite> .tables
migrate_version  post          user
```

通过.tables 命令我们可以看到数据库中有三张表，分别是 migrate_version，post，user。

```
sqlite> .schema user
CREATE TABLE user (
        id INTEGER NOT NULL,
        email VARCHAR(100),
        password VARCHAR(100),
        nickname VARCHAR(100),
        role SMALLINT,
        about_me VARCHAR(140),
        last_seen DATETIME,
        PRIMARY KEY (id)
);
CREATE UNIQUE INDEX ix_user_email ON user (email);
CREATE UNIQUE INDEX ix_user_nickname ON user (nickname);
```

通过.schema [表名]命令，可以查看指定表的 SQL 定义。我们看到 user 表的主键是 ID，有 email，password 等字段，并且建有两个唯一索引 ix_user_email 和 ix_user_nickname，这意味着数据库要求 user 表中的 email 和 nickname 必须是唯一的。

```
sqlite> select email,password from user;
a_kui@163.com|pbkdf2:sha256:50000$dbW8j7hy$f78f7750a07edde0b99506b031463e6444a304ab60ad5e91dc99b58b4decd0d8
test@163.com|pbkdf2:sha256:50000$LuA8t5fJ$ddd5d7573bc0fa4b5b6f53a7450298be71246aa866d93f88fa9d72d8d6dbc0f2
sqlite>
```

select、insert、update、delete 等数据操作语句都是可以使用的，如果你对数据库操作语句不太熟悉，没有关系，这并不是本书的重点，有兴趣可以参见学习资源中给出的有关数据库部分的学习链接。

特别提醒，如果从命令行直接运行 sqlite3，可以将数据库存储文件的路径名作为命令行参数，比如下面的语句。

```
sqlite3 app.db
```

这样就可以直接打开数据库文件进行相关的后续操作了。

文件操作

说到文件操作不能不提的是 os 库。

我非常喜欢来自：https://docs.python.org/3.6/library/os.html 的关于 os 的这段介绍。

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

这个模块提供了一种可移植的方法，来使用依赖于操作系统的相关功能。如果你想读或者写一个文件，请看 open()方法；如果你想操作路径（paths），请看 os.path 模块；如果你想在命令行下读取所有文件的所有行的内容，请看 fileinput 模块；要创建临时文件或者目录，请看 tempfile 模块；如果要进行高层次的文件和目录操作，请看 shutil 模块。

这是多好的指路明灯！读者完全可以根据自己的需要查看对应模块的详细文档内容。

文件的创建、打开、读写我们前面的小节都已经有提到，但是，如何删除一个文件呢？这里就要用到 os 模块的 remove()方法，其定义如下：

```
os.remove(path, *, dir_fd=None)
```

该方法会删除 path 路径名指定的文件。如果 path 指向了一个目录，会报一个 OSError 的异常。删除目录应该使用 rmdir()方法。特别要提到的是，remove()方法的 path 参数是支持通过相对路径的方式指定文件名的。

那么，如果我们要拷贝和归档一个文件或者目录应该怎么做呢？

拷贝一个文件，一般的方法是：打开一个文件，将其写到另外一个新建的文件就完成了拷贝，如果是目录树就需要遍历。

```
Simple is better than complex.
简单胜于复杂。
```

这是 Python 之禅（The Zen of Python, by Tim Peters）的第三句。在我们使用 Python 的时候，如果感觉你的想法和设计过于复杂的时候，就需要自己反思一下，或者查一下资料，看一下是不是有更加简洁的做法。

而针对文件和目录树的拷贝、删除、归档的简洁的做法，由 shutil 库提供，我们重点看一下文件的拷贝。

```
shutil.copy(src, dst, *, follow_symlinks=True)
```

shutil.copy()方法，用于将 src 参数指定的文件拷贝到 dst 参数指定文件路径或者目录中。src 和 dst 都是字符串类型的路径文件名，如果 dst 指向的是一个目录，文件会保持名字不变，src 文件会被拷贝到 dst 目录中。这个方法的返回值是新创建的文件的路径名。

至于 shutil 的其他功能，感兴趣的读者可以参考学习资源中的文件操作部分的链接。

挑战问题

编写一个 Python 的 unittest 程序 TestCustomBlogRegister.py，测试如下场景。

- 场景一：邮箱合法，密码两次一样，预期结果为注册成功。
- 场景二：邮箱合法，密码两次不同，预期结果为注册失败。
- 场景三：邮箱合法，密码两次一样，但是，该邮箱已经注册过，预期结果为注册失败。

如图 11.22 所示。

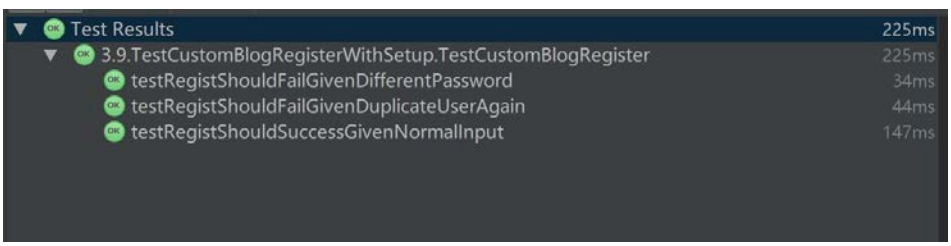


图 11.22 3 个注册测试案例

需要编写 setUp 和 tearDown，否则案例无法做到可重复执行。考虑直接对 sqlite3 数据库文件进行备份和回复的方式，简单直接。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超，请将编写的程序删除后，重做一次。

难点提示

Note：The order in which the various tests will be run is determined by sorting the test method names with respect to the built-in ordering for strings.

注意：不同测试案例的运行顺序是以测试方法名的顺序排序的，排序的方法采用的是字符串内置顺序。

——引用自 Python3 官方文档：https://docs.python.org/3/library/unittest.html

知识点

unittest

setUp()和 tearDown()方法的定义和作用

SQLite 数据库

建议通过.help 命令多了解一些 sqlite3 工具的命令。

SQL 语句中的增、删、改、查四个命令，建议读者熟练掌握。这四个命令对于在测试过程中查看数据的变化，验证测试效果很有帮助。

拓展

读者可以尝试一下是否可以针对登录也写出三个案例来。

11.6 命令行集成与 HTML 报告

学习目标

学会从命令行运行测试案例集，并生成 HTML 报告。

学习资源

unittest TestSuite：http://www.php.cn/python-tutorials-358252.html

http://www.cnblogs.com/yufeihlf/p/5707929.html
https://docs.python.org/3.6/library/unittest.html#organizing-tests

知识准备

关于测试案例集

到现在为止，关于 Python 的 unittest 我们学习了测试案例 unittest.TestCase 的编写，setUp()和 tearDown()方法的定义以及使用 unittest.main()启动测试案例。

本小节我们将学习测试案例的组织。所谓测试案例的组织，最主要的就是对测试案例进行分组和分类，让测试案例按照要求分组、分类执行，而不用每一次全部执行所有的测试案例。为了对测试案例进行组织，我们需要用到 TestSuite 和 TestLoader 两个类。

TestSuite 类

以下内容引用自 Python 官方文档：https://docs.python.org/3.6/library/unittest.html#unittest.TestSuite

This class represents an aggregation of individual test cases and test suites. The class presents the interface needed by the test runner to allow it to be run as any other test case. Running a TestSuite instance is the same as iterating over the suite, running each test individually.

TestSuite 类，即测试套件类，代表一组由多个独立的测试案例或者测试套件的集合。测试套件类提供了测试运行器（test runner）运行测试案例所需要的接口，这样 TestSuite 类就可以和其他测试案例一样被测试运行器加载运行了。运行一个测试套件的实例，和通过迭代器一个一个的运行测试套件里面的每一个单独的测试案例的效果是一样的。

If tests is given, it must be an iterable of individual test cases or other test suites that will be used to build the suite initially. Additional methods are provided to add test cases and suites to the collection later on.

如果要使用一些测试案例来初始化一个测试的套件对象，就必须给定一个包含多个单独的测试案例的迭代器或者另一个测试套件。测试套件类提供了向其集合中增加测试案例或者其他测试套件的方法。

TestSuite objects behave much like TestCase objects, except they do not

actually implement a test. Instead, they are used to aggregate tests into groups of tests that should be run together. Some additional methods are available to add tests to TestSuite instances:

除了不会真正的实现一个测试案例方法之外，测试套件对象的行为很大程度上和测试案例对象是一样。测试套件是用来对测试案例进行分组执行的，有一些附加的方法可以用于向测试套件实例增加测试案例。

以上内容，翻译自：Python 文档 unittest 小节。

总的来说，TestSuite 类是一个用于组织多个测试案例或者测试套件（TestSuite）的类，该类的大部分方法和 TestCase 一样。特别要提到的是，该类有几个专门用于向 TestSuite 对象自身添加测试案例的方法。

addTest(test)

功能：增加测试案例或者测试案例集到当前的测试套件（TestSuite）对象中。

addTests(tests)

功能：将一个包含了多个测试案例的迭代器或者测试套件实例中的案例，增加到当前的测试套件（TestSuite）对象中。

除了上面两个增加测试案例的方法，TestSuite 类和 TestCase 类一样，也有一个 run()方法，不同的是测试套件类的 run()方法需要将测试结果对象（TestResult）作为参数传入，而 TestCase.run()方法会返回一个测试结果对象。

让我们看一个来自 Python Doc 的例子：

```
def suite():
    suite = unittest.TestSuite()
    suite.addTest(WidgetTestCase('test_default_size'))
    suite.addTest(WidgetTestCase('test_resize'))
    return suite
```

上面的例子中，通过 TestSuite 类的 addTest 方法，分别将两个测试案例增加到了测试套件对象中。

TestLoader 类

测试案例集的组织还有一个很有用的类叫做 TestLoader，读者可以自行参考学习资源中的资料。阅读之后，完成下面的小练习。

小练习：

通过 TestLoader 类获取上面小节 TestCustomBlogRegister 测试案例类中所有测试方法的名字，并打印出来。

关于 HTML 测试报告

我们在运行测试案例时看到最多的是如图 11.23 这样的输出结果。

```
C:\MyWork\myBook\用Python做HTTP接口测试\source>python -m unittest TestComputer.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK

C:\MyWork\myBook\用Python做HTTP接口测试\source>python -m unittest -v TestComputer.py
test_should_good_for_bignegativenum_morethan10000000000 (TestComputer.TestComputer) ... ok
test_should_good_for_bignum_morethan_10000000000 (TestComputer.TestComputer) ... ok
test_should_good_for_negative_number (TestComputer.TestComputer) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.002s

OK
```

图 11.23 命令行运行测试案例

上面运行了两次 TestComputer.py 文件中的测试案例：

- 第一次是在命令行下，用默认参数运行得到输出情况，其中的三个“.”代表三个执行通过的测试案例。
- 第二次是在命令行下，用增加"-v"参数运行得到的输出更为详细，会打印输出每一个测试案例的名字和通过情况。

以上两种输出都是纯文本的，可视化效果并不好。那么，如何将测试的结果以更加可视化的形式输出呢？

要回答这个问题，我们需要先了解一下 unittest 运行测试案例和输出案例运行结果的机制。相信大家对以下两行代码并不陌生：

```
if __name__=='__main__':
    unittest.main()
```

unittest.main()方法可以有很多的参数。

```
unittest.main(module='__main__', defaultTest=None, argv=None, testRunner=None, testLoader=unittest.defaultTestLoader, exit=True, verbosity=1, failfast=None, catchbreak=None, buffer=None, warnings=None)
```

我们来认识其中常用的几个，内容来自：https://docs.python.org/3.6/library/unittest.html#unittest.main。

- testRunner 参数：

The testRunner argument can either be a test runner class or an already created instance of it. By default main calls sys.exit() with an exit code indicating success or failure of the tests run.

testRunner 参数可以是一个测试运行器类或者一个已经创建的迭代运行器的实例。默认情况下，main()方法会调用 sys.exit()，并通过退出码（exit code）来表示测试运行成功还是失败。如果不给 testRunner 参数赋值，默认使用 TextTestRunner 作为运行器，执行测试案例。

- testLoader 参数：

The testLoader argument has to be a TestLoader instance, and defaults to defaultTestLoader.

testLoader 参数必须是一个测试加载器类的实例，也就是说必须是一个对象，不能是类，缺省赋值为 defaultTestLoader。前面要求大家自行学习 TestLoader 类，不知道大家学习的怎么样了，是否学会了以一个列表返回一个测试案例类的所有测试案例方法的名字，并打印出来。

- verbosity 参数：

You can run tests with more detailed information by passing in the verbosity argument.

可以通过给 verbosity 参数赋值，在运行测试的时候显示更多、更详细的信息。默认值为 1，一般情况下如果需要显示案例方法名和案例的运行情况，可以对其赋值为 2。

注意：testRunner 参数即可以接受一个测试运行器类也可以接受一个对象作为值传入，如果向 main()方法传递了测试运行器的对象，而不是类，那么此时，在 main()方法中设置 verbosity 参数是不起作用的，需要在实例化的运行器对象上设置该参数。

其实，我们平时运行测试案例的时候，使用最多的是 TextTestRunner 类，该类是 unittest.main()方法的默认测试案例运行器，上面的纯文本的测试信息输出，就是来自于 TextTestRunner 类。

今天，我们要介绍一个 HTMLTestRunner 类。

这个类的作用是用于运行指定的测试案例，并产生可视化的 HTML 测试报告。

该类由 Wai Yip Tung 最初创建，GitHub 地址为：

```
https://github.com/tungwaiyip/HTMLTestRunner。
```

但是，这个库的代码已经长时间没有更新，并且是基于 Python2.7 的。后来，由一名 GitHub 贡献者 Rahul Yadav 对其进行了针对 Python3.0 的迁移，GitHub 地址为：

```
https://github.com/zac11/HTMLRunner。
```

但这个版本目前仍然有一些瑕疵，使用并不方便，所以我就就从 Rahul Yadav 的地址 Fork 了一个版本，并进行了修改，地址为：

```
https://github.com/akuing/HTMLRunner。
```

读者可以从我的 GitHub 地址，下载压缩包并解压缩，按照 Readme 文件中的提示，先运行一下，有一个感性认识。

对于 Windows 用户，首先，确保 Python3.6+已经安装，双击"runcmd.bat"，会出现命令行窗口。

在命令行窗口输入：

```
python TestComputer.py
```

你会看到 HTML 结果被打印到了终端上。

使用重定向将打印到终端上的 HTML 内容转存到 result.html 文件中。

```
python TestComputer.py > result.html
```

运行结束后，你会看到生成了一个 result.html 文件。双击 html 文件，就可以通过浏览器打开该文件。

直接运行 TestComputerSuite.py：

```
python TestComputerSuite.py
```

读者会看到生成来了一个 HTML 报告 computerTest.html。

请读者自行阅读 TestComputerSuite.py 的代码，体会测试案例套件的使用和测试运行器类的使用，以及测试案例代码与测试套件代码分离的代码组织方式。

挑战问题

编写一个 Python 的 unittest 程序，其中 TestCustomBlog.py 用于存放测试案例类，类名为 TestCustomBlog，里面包含六个以上的测试案例，并且保证测试案例可以多次运行；TestCustomBlogSuite.py 用于存放测试套件类。

这些案例中至少有三个测试注册，三个测试登录。

场景如下：

- 注册场景一：邮箱合法，密码两次一样，预期结果为注册成功。
- 注册场景二：邮箱合法，密码两次不同，预期结果为注册失败。
- 注册场景三：邮箱合法，密码两次一样，但是，该邮箱已经注册过，预期结果为注册失败。
- 登录场景一：邮箱合法，密码正确，预期结果为登录成功。
- 登录场景二：邮箱合法，密码不正确，预期结果为登录失败。
- 登录场景三：邮箱不合法，密码不正确，预期结果为登录失败。

将以上测试分成注册系列和登录系列两组测试案例，从命令行分别运行两个系列的测试案例，也可以合并运行，运行结束会生成一个 Html 的测试报告，如图 11.24 所示。

测试

Start Time: 2017-05-15 19:10:59
Duration: 0:00:00.601855
Status: Pass 6

用例执行情况

Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
TestCustomBlog	6	6	0	0	Detail
testLoginShouldFailedGivenBadPassword			pass		
testLoginShouldFailedGivenBadUserEmail			pass		
testLoginShouldSuccessGivenNormalUserAndPassword			pass		
testRegistShouldFailGivenDifferentPassword			pass		
testRegistShouldFailGivenDuplicateUserAgain			pass		
testRegistShouldSuccessGivenNormalInput			pass		
Total	**6**	**6**	**0**	**0**	

图 11.24 通过 HTMLTestRunner 运行测试案例

要求 HTML 测试报告的 Title 为“测试”，descriptions 为“用例执行情况”，要能够正常显示这两个参数的中文内容。

考虑使用 TestLoader 类，虽然准备中没有对其进行详细的介绍，但是使用该类来根据不同的名称加载不同的测试案例很便利。

注意：以上问题，请在十分钟内解决。切记，解决“挑战问题”过程中要闭卷完成。如果第一次不能闭卷完成或者完成超时，请将编写的程序删除后，重做一次。

难点提示

如果发现 HTMLTestRunner 模块或者类找不到，需要拷贝 HTMLTestRunner.py 到自己的测试案例源文件相同的目录。

知识点

语言基础知识要点

unittest 测试库

- unittest.main()方法的使用
- unittest.TestSuite 类的使用
- unittest.TestLoader 类的使用
- unittest.TextTestRunner 类和 unittest.HTMLTestRunner 类的使用

拓展

对测试案例的运行结果进行可视化或者结构化的输出非常重要，相关的资料很多，此处推荐几个项目供大家参考：

```
https://github.com/oldani/HtmlTestRunner
```

相比 HTMLTestRunner 而言，这是一个专注于将测试结果保存为 HTML 文件格式的项目，推荐读者采用，读者可以将本小节的挑战问题用 HtmlTestRunner 改写一下，作为拓展练习。

```
https://github.com/xmlrunner/unittest-xml-reporting
```

unittest-xml-reporting 是一个 unittest 测试运行器，可以产生 XML 的测试结果文件，这样测试结果以结构化数据的方式保存，进而可以被很多其他系统或者工具使用，比如构建系统 IDE 或者持续集成服务器等。

写在后面的话

这仅仅是一个开始

到此，关于 Python 编程基础和 HTTP 接口测试的学习就告一段落了。相信通过这一段时间的闯关式学习，你已经初步掌握了 Python 编程的基础技能，对 HTTP 和 JSON 有了较深入的理解，并且已经掌握了用 Python 进行 HTTP 接口测试的方法，可以用 Python 在自己的项目里进行一些 HTTP 接口的测试工作了。

但“这仅仅是一个开始”，后面接口测试学习的路还很长。正如开始的时候本书中所说的，**学习和掌握用 Python 进行 HTTP 接口自动化测试**是从手工测试工程师到自动化测试工程师转型的切入点，也仅仅是切入点。在实际工作中进行接口自动化测试，应该站在更高的视角，从团队和企业的角度考虑问题，多考虑使用和利用已有的工具和框架来辅助进行接口自动化测试工作，而不是执着于已经掌握的内容，一定要一切都用 Python 自行去实现。

正如在“3.3 摆脱“点点点”从哪里开始？”小节列表中所列举的内容所示，自动化测试领域有很多的内容需要学习，下面图表列举了当下比较流行的专用测试工具。读者可以根据自己的工作场景和需要选择适用的工具进行学习。

领　域	主　题	备　注
专用测试的工具	Selenium、WebDriver	WebDriver 主要用于驱动浏览器界面进行基于 UI 的 Web 测试
	Firefox、Chrome	开发者模式下，可以查看 Web 网络报文，进行 HTTP 接口探测
	Robotframework	一款 python 编写的功能自动化测试框架，采用关键字驱动的方式，使用自然语言来编写测试案例，支持多种类型的客户端和接口的测试
	Cucumber	一款用 Ruby 编写的功能自动化测试框架，支持行为驱动开发（BDD），也是一个能够使用自然语言来编写测试案例的自动化测试工具，支持 Java 和.Net 等多种开发语言

（续表）

领　域	主　题	备　注
	Appium	一个开源、跨平台的自动化测试工具，用于测试原生和轻量移动应用，支持 iOS, Android 和 FirefoxOS 平台
	QTP	Quick Test Professional 的简称，是一种商用的自动测试工具。支持录制和回放的方式编写测试案例，但是，录制和回放的方式编写测试案例或许会导致案例代码的可维护性差，主要是用于回归测试和测试同一软件的新版本
	Postman	一个非常有力的 Http Client 工具，可以用来测试 Web 接口，也可以独立安装也可以以 Chrome 插件的形式使用
	Watri	Web Application Testing In Ruby 的缩写，是一个面向功能的自动化测试框架，其原理与 Selenium+WebDriver 的方式类似，该框架用 Ruby 编写

附录：学习心得

——高园园

机缘巧合之下，参加了阿奎老师的课程《用 Python 做 HTTP 接口测试》，目前为止的状态：坚定不移地跟下去，自学+课程模式每天一小时！

1. 学习初衷

打算学习 Python,这个想法开始于 2017 年，和在网络上遇到的绝大多数测试人员不同，我的工作单位是一个大型国企，虽然从事软件测试工作已经有四年，可事实上，无论是测试理论还是实际项目经验，都处于一种停滞不前的状态，作为一个有理想的测试者，这样的工作状态让人心酸，因此，2017 年我的目标不再是仅仅完成单位的测试项目，而是还要提升自我！

所以说，人一旦有了梦想，挡都挡不住！

第一步，我关注了大量的软件测试类微信公众号，每天接收至少十条以上软件测试类短文的推送，内容涵盖了 APP 测试、UI 测试、测试基础理论等等各种类型，这种碎片化阅读持续了大概有一个月的时间，我发现自己彻底沦陷了，由于接收的信息太过琐碎，而自己目前并不具备整合、内聚的能力，导致的结果就是感觉每个人说得都对，每篇推文都好有道理，然而，对我个人而言，并没有什么……用！

第二步，在被碎片化知识淹没的情况下，我及时改变了战略，必须走向系统化的学习，因此，我断断续续买了十几本专业书，涵盖了测试理论、测试职业发展、测试策略、程序设计等等方面，在不断阅读、实验的过程中，我越来越意识到个人能力的薄弱，越来越渴望掌握更多的知识。

第三步，初识 Python，我已经不记得是在哪里第一次看到 Python，作为一个计算机专业毕业的测试人员，在这四年的测试生涯中，除了 C 语言，我已经彻底忘记了其他语言的语法之类，而转型自动化是我在这四年的手动测试过程中心心念念的方向，因此，学习一门新的语言势在必行，而选择 Python，就是个偶然吧，因为看到顾翔老师说他买了《跟老齐学 Python》，所以我也

购买了这一本，正式开启了我的 Python 之旅。

第四步，走近阿奎老师，在开始学习 Python 之后不久，我发现自己无法将 Python 语言的学习与软件测试连接起来（因为我做的是嵌入式软件的测试），此时，我有些迷茫，完全不知道后续该怎么走！正在这时，我看到了阿奎老师发布的课程《用 Python 做 HTTP 接口测试》，正如前面所说，我加入了课程的学习！

2. 学习目标

由于我目前的工作中并不涉及 HTTP 接口的测试，所以，我在这个课程中的学习目标有以下点：

（1） 通过“关卡”实战演习，加强 Python 的学习效果。在阿奎老师这本书的开头部分，有这样一段话：“本书采用“闯关式学习方法”编纂，阅读的过程中，需要读者边读边练习，每个里程碑包含多个小节，每个小节都是一个小的关卡，一般里程碑的最后一个小节是大关卡，有的大关卡是可选的，有的是必须通过的。”编程语言的学习很容易觉得枯燥，没有方向性，在我看来，“关卡”，其实就是阿奎老师的学习思路，这样的形式，给了初学者一条相对贯通的路去执行！

（2） 在练习的过程中复习 HTTP 协议、TCP/IP 协议。目前软件测试行业以互联网软件测试为主，而我工作中需要测试的软件与网络彻底隔离，长此以往，必然导致与行业脱节，所以我必须学习互联网相关软件的测试技能，阿奎老师的课程大纲正是以 HTTP 的自动化测试为最终目标，一步步前进，姑且认为是个人能力的版本迭代吧！

（3） 自动化测试。自动化测试的学习是我 2017 年的总目标，我希望能够在 HTTP 自动化测试的学习基础上，开发适用于公司嵌入式软件产品的自动化测试流程和框架，当然，这需要日复一日不断地修炼！

3. 学习内容及效果

目前，已经完成了 Python 语言基础和一部分 HTTP 相关内容的学习，涉及 Python 语言中基本对象类型、语句、函数、类库、方法等方面的内容。每天平均用时 1 小时左右（包括资料查找、看书的时间），实际上，阿奎老师会在每一小节写一句：以上问题，十分钟内就能解决！多么痛的领悟，当然，我是学生嘛，我习惯于先看问题，然后带着问题去思考，去查资料，自己解决之

后再对比老师写的难点、知识点，看自己是否有抓到老师说的重点，基本上都能八九不离十，只有一节完全脱离了老师的初衷，当然，这也让我再次深刻地理解“对于软件测试人员来说，正确理解需求有多么重要！”

就目前已经完成的课程来看，我个人认为，这种“闯关式”的学习方法非常有效，只要有一个有经验的老师带着你，按照既定的关卡走下去，可以少走很多弯路，毕竟，学习这件事情，真的是只有“熟能生巧”！希望未来我可以在十分钟内解决！

4. 感想

在不学习、不听、不看的情况下，人很容易固步自封，正如那句鸡汤文所说：“最可怕的是比你优秀的人还比你努力”，实际上更可怕的是：甩你几十条街的人比你还努力几条街！软件测试是一个需要不断学习、不断提升个人能力的行业，自动化测试的学习也好、测试理论的学习也好，哪怕是碎片化的阅读也好，只要还在进步，只要还能看到自己和他人的差距，我就不会慌张，剩下的就是练习、坚持、坚持练习！软件测试人员个人能力的迭代和软件的迭代一样，都是一个不断追求卓越的过程！

附录：参考资料

思特沃克公司. 技术雷达《Technology Rada Vol.16》. 2017 年 3 月

赖勇浩. 《Python 之禅》的翻译和解释. http://blog.csdn.net/gzlaiyonghao/article/details/2151918

阮一峰. 理解 RESTful 架构. http://www.ruanyifeng.com/blog/2011/09/restful.html

Swaroop C H.《简明 Python 教程》（《A Byte of Python》）. 最新中文版由漠伦翻译和维护. https://bop.molun.net. 采用 **知识共享 署名-相同方式共享 国际 4.0 协议（CC BY-SA Intl. 4.0）

菜鸟教程.Python3 教程.http://www.runoob.com/python3/python3-tutorial.html

w3cschool.cn.Python3 教程 .http://www.w3cschool.cn/python3/python3-tutorial.html

python.org.Python3.6 官方网站.https://docs.python.org/3.6/

Python 中文学习大本营.Python 入门指南.http://www.pythondoc.com/pythontutorial3/index.html

Python 中文开发者社区.Python3 中文手册.http://docs.pythontab.com/python/python3.4/

Python-Guide-CN.Python 最佳实践指南！.https://pythonguidecn. readthedocs.io/zh/latest/index.html

Kenneth Reitz.Requests: 让 HTTP 服务人类.http://cn.python-requests.org/zh_CN/latest/

C.W.pyexcel - Let you focus on data, instead of file formats.https://pyexcel.readthedocs.io/en/latest/

Karlijn Willems.Python Excel Tutorial: The Definitive Guide.https://www.datacamp.com/community/tutorials/python-excel-tutorial

delong.Beautiful Soup 4.4.0 文档.http://beautifulsoup.readthedocs.io/zh_CN/latest/