21世纪软件工程专业规划教材

Python测试技术

周元哲　编著

清华大学出版社

21世纪软件工程专业规划教材

Python测试技术

清华大学出版社
北京

内 容 简 介

本书讲述了与 Python 语言有关的三大测试——单元测试、网络测试和移动测试，主要包括软件测试基础、自动测试技术、Python 与软件测试、Python 与 unittest 单元测试、Python 与 Selenium 网络测试、Python 与 DDT 数据驱动测试、Python 与 UIAutomator 测试、Python 与 Appium 移动测试等相关内容。附录介绍了前端测试、Jest 和 Monkey 等相关知识。

本书内容精练、由浅入深，注重知识的连续性和渐进性，适合作为高等院校相关专业教材或教学参考书，也可以供从事计算机应用开发的各类技术人员参考，还可作为全国计算机等级考试、软件技术资格与水平考试的培训资料。

图书在版编目(CIP)数据

Python 测试技术/周元哲编著. —北京：清华大学出版社，2019.12
21 世纪软件工程专业规划教材
ISBN 978-7-302-54195-0

Ⅰ. ①P…　Ⅱ. ①周…　Ⅲ. ①软件工具—程序设计—高等学校—教材　Ⅳ. ①TP311.561

中国版本图书馆 CIP 数据核字(2019)第 255961 号

责任编辑：张　玥
封面设计：何凤霞
责任校对：胡伟民
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-83470236

印 装 者：三河市少明印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：12.25　　**字　　数**：310 千字
版　　次：2019 年 12 月第 1 版　　**印　　次**：2019 年 12 月第 1 次印刷
印　　数：1～1500
定　　价：40.00 元

产品编号：085349-01

前言

PREFACE

Python是一种解释型、面向对象、动态数据类型的高级程序设计语言，外挂各种库，在大数据、数据分析、科学计算等方面功能卓越。本书讲述了与Python语言相关的测试技术，包括单元测试unittest、网络测试Selenium和移动测试Appium。具体章节包括软件测试基础、自动测试技术、Python与软件测试、Python与unittest单元测试、Python与Selenium网络测试、Python与DDT数据驱动测试、Python与UIAutomator测试、Python与Appium移动测试等相关内容。附录介绍了前端测试、Jest和Monkey等相关知识。

本书采用Python 3版本，所有程序都在PyCharm、Anaconda中进行调试和运行。

本书内容精练，文字简洁，结构合理，实训题目经典实用，综合性强，特别适合作为高等院校Python语言测试的教材或教学参考书，也可以供从事计算机应用开发的各类技术人员应用参考。

西安邮电大学的孟伟君、焦继业、邓万宇、孔韦韦、包志强等阅读部分手稿。西安睿博智能股份有限公司的周鑫、西安玺奥信息安全技术有限公司的谭小琴、清华大学出版社张玥编辑对本教材的写作大纲、写作风格等提出了很多宝贵的意见。软件工程专业18级学生卓越调试了部分代码。衷心感谢上述各位的支持和帮助。本书写作时参阅了大量中英文专著、教材、论文、报告及网上的资料，由于篇幅所限，未能一一列出，在此一并向相关作者表示敬意和衷心的感谢。

由于作者水平有限，时间紧迫，本书难免有疏漏之处，恳请广大读者批评指正。

编　者

2019年9月

图书资源支持

感谢您一直以来对清华版图书的支持和爱护。为了配合本书的使用，本书提供配套的资源，有需求的读者请扫描下方的“书圈”微信公众号二维码，在图书专区下载，也可以拨打电话或发送电子邮件咨询。

如果您在使用本书的过程中遇到了什么问题，或者有相关图书出版计划，也请您发邮件告诉我们，以便我们更好地为您服务。

资源下载、样书申请

书圈

我们的联系方式：

地　　址：北京市海淀区双清路学研大厦 A 座 701

邮　　编：100084

电　　话：010-83470236　010-83470237

资源下载：http://www.tup.com.cn

客服邮箱：tupjsj@vip.163.com

QQ：2301891038（请写明您的单位和姓名）

扫一扫，获取最新目录

课 程 直 播

用微信扫一扫右边的二维码，即可关注清华大学出版社公众号“书圈”。

目 录

CONTENTS

软件测试基础

本章介绍软件测试的基本知识、软件测试的发展历程、软件测试分类等。软件测试方法分为白盒测试和黑盒测试，讲解了白盒测试的逻辑覆盖法、路径分析法等方法和黑盒测试的等价类划分法、边界值分析法、决策表和因果图等方法。

1.1 软件测试概述

软件测试是控制软件质量的重要手段和关键活动，一般具有如下作用：

(1) 测试不仅仅是为了找出错误，而是通过分析错误产生的原因和错误的发生趋势，帮助项目管理者发现当前软件开发过程中的缺陷，以便及时改进。

(2) 帮助测试人员设计出有针对性的测试方法，改善测试的效率和有效性。

(3) 没有发现错误的测试也是有价值的，完整的测试是评定软件质量的一种方法。

软件测试的分类方法很多。按测试阶段和层次划分，分为单元测试、集成测试、确认测试、系统测试、验收测试；按所采取的技术划分，分为白盒测试、黑盒测试和灰盒测试；按所关注的质量属性和目的划分，分为功能测试、性能测试、压力测试、安全性测试、兼容性测试、可靠性测试、容错性测试、安装/卸载测试、恢复测试等。

1.2 软件测试历程

软件测试伴随着软件的产生而产生。早在 20 世纪 50 年代，英国著名的计算机科学家图灵就给出了软件测试的原始含义。他认为，测试是程序正确性证明的一种极端实验形式。在早期软件开发过程中，软件规模小，复杂程度低，软件开发过程相当混乱无序，软件测试的含义也比较窄，等同于“调试”，目的是纠正软件的故障。它常常由软件开发人员自己进行，主要是针对机器语言和汇编语言设计特定的测试用例，运行被测试程序，将所得结果与预期结果进行比较，从而判断程序的正确性。早期对测试的投入极少，测试介入也晚，常常是等到形成代码，产品已经基本完成时才进行测试。

直到 1957 年，软件测试首次作为发现软件缺陷的活动，与调试区分开来。1972 年，北卡罗来纳大学举行首届软件测试会议，John Good Enough 和 Susan Gerhart 在 IEEE 上发表《测试数据选择的原理》，确定软件测试是软件的一种研究方向。1975 年，John

Good Enough 首次提出了软件测试理论，从而把软件测试这一实践性很强的学科提高到了理论的高度。1979 年，Glenford Myers 在《软件测试艺术》一书中提出“测试是为发现错误而执行的一个程序或者系统的过程”。

20 世纪 80 年代早期，软件和 IT 行业进入了大发展时代，软件趋向大型化、高复杂度，软件的质量越来越重要。一些软件测试的基础理论和实用技术开始形成，软件开发的方式也逐渐由混乱无序过渡到结构化的开发过程，以结构化分析与设计、结构化评审、结构化程序设计以及结构化测试为特征，软件测试性质和内容也随之发生变化。测试不但是一个单纯的发现错误的过程，而且是具有软件质量评价的内容。软件工程的概念逐步形成，软件开发模型产生。1983 年，Bill Hetzel 在《软件测试完全指南》中指出，测试是以评价一个程序或者系统属性为目标的任何一种活动，是对软件质量的度量。IEEE 给软件测试的定义为“使用人工或自动手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果直接的差别”。这个定义明确地指出，软件测试的目的是为了检验软件系统是否满足需求。软件测试不再是一个一次性的，也不只是开发后期的活动，而是与整个开发流程融合成一体。

20 世纪 90 年代，随着面向对象分析和面向对象设计技术的逐渐成熟，面向对象软件测试技术逐渐受到人们重视。1994 年 9 月，Communication of ACM 出版了《面向对象的软件测试专集》，内容涉及类测试、集成测试和面向对象软件的可测试性等问题。1989 年，Fiedler 从面向对象的测试与传统测试的不同点出发，提出了面向对象单元测试的解决方案，开始从事面向对象软件测试的研究工作。

1996 年，测试成熟度模型 TMM(Testing Capability Maturity Model)等一系软件测试相关理论提出。到了 2002 年，Rick 和 Stefan 在《系统的软件测试》一书中对软件测试做了进一步描述：测试是为了度量和提高软件的质量，对软件进行工程设计、实施和维护的整个生命周期过程。近 20 年来，随着计算机和软件技术的飞速发展，软件测试技术的研究也取得了很大突破。许多测试模型(如 V 模型等)产生，单元测试、自动化测试等方面涌现出了大量的软件测试工具。在软件测试工具平台方面，产生了很多商业化的软件测试工具，如捕获/回放工具、Web 测试工具、性能测试工具、测试管理工具、代码测试工具等。一些开放源码社区中也出现了许多软件测试工具，它们得到了广泛应用，且相当成熟和完善。

1.3 软件测试分类

从不同的角度考虑软件测试，可以有不同的划分方法。图 1.1 展示了从不同角度进行软件测试的分类。

下面主要讲解白盒测试和黑盒测试。

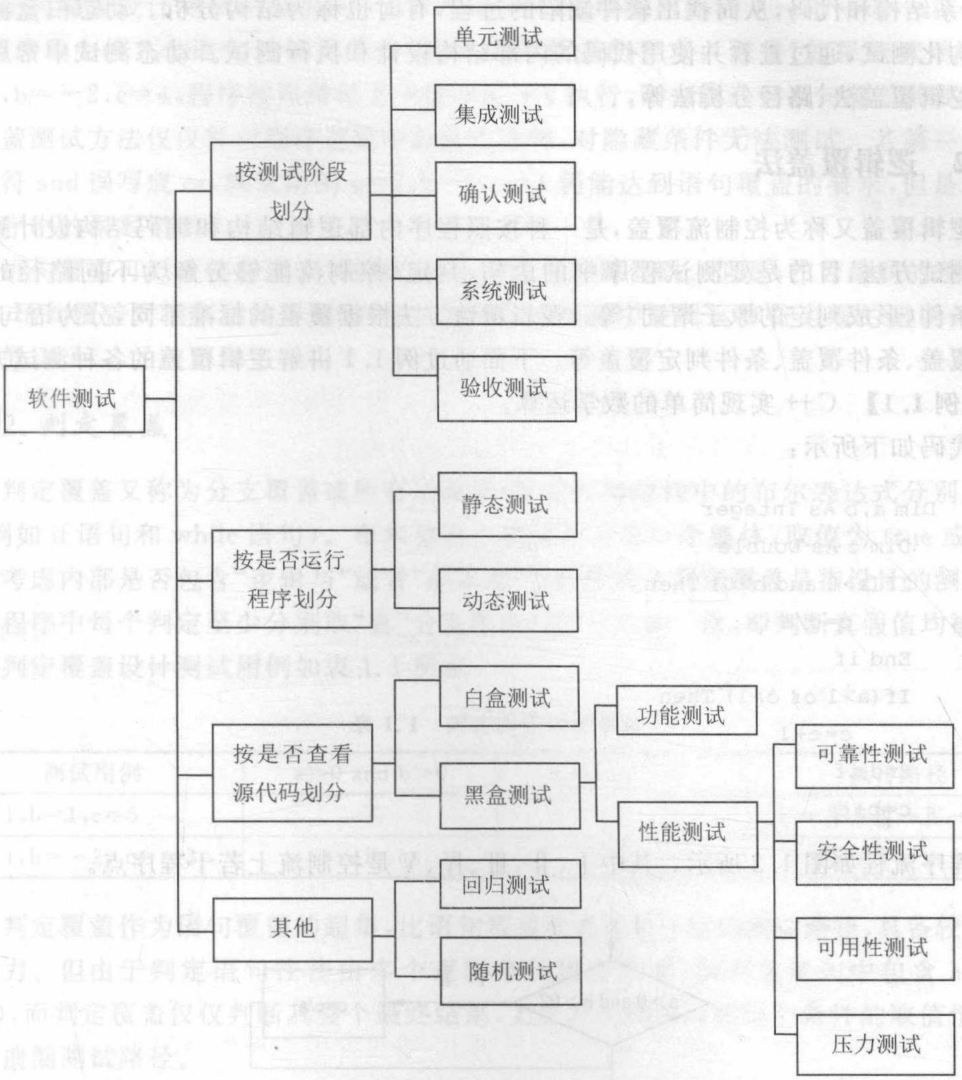

图 1.1 软件测试分类

1.4 白盒测试

1.4.1 概述

白盒测试是把测试对象看作一个打开的盒子，允许测试人员利用程序内部的逻辑结构及有关信息设计或选择测试用例，通过在不同点检查程序状态确定实际状态是否与预期的状态一致。白盒测试用于纠正软件系统在描述、表示和规格上的错误，是进一步测试的前提。

白盒测试测试软件产品的内部结构和处理过程，而不测试软件产品的功能，白盒测试分为静态测试和动态测试。静态白盒测试是在不执行的条件下有条理地仔细审查软件设

计、体系结构和代码，从而找出软件缺陷的过程，有时也称为结构分析。动态白盒测试也称结构化测试，通过查看并使用代码的内部结构设计和执行测试。动态测试中常用的方法有逻辑覆盖法、路径分析法等。

1.4.2 逻辑覆盖法

逻辑覆盖又称为控制流覆盖，是一种按照程序内部逻辑结构和编码结构设计测试用例的测试方法，目的是要测试程序中的语句、判定（控制流能够分解为不同路径的程序点）、条件（形成判定的原子谓词）等。逻辑覆盖方法根据覆盖的标准不同，分为语句覆盖、判定覆盖、条件覆盖、条件判定覆盖等。下面通过例 1.1 讲解逻辑覆盖的各种测试方法。

【例 1.1】 C++ 实现简单的数学运算。

代码如下所示：

```
1  Dim a,b As Integer
2    Dim c As Double
3    If(a>0 and b>0) Then
4        c=c/a
5    End if
6    If(a>1 or c>1) Then
7        c=c+1
8    End if
9    c=b+c
```

程序流程如图 1.2 所示。其中Ⅰ、Ⅱ、Ⅲ、Ⅳ、Ⅴ是控制流上若干程序点。

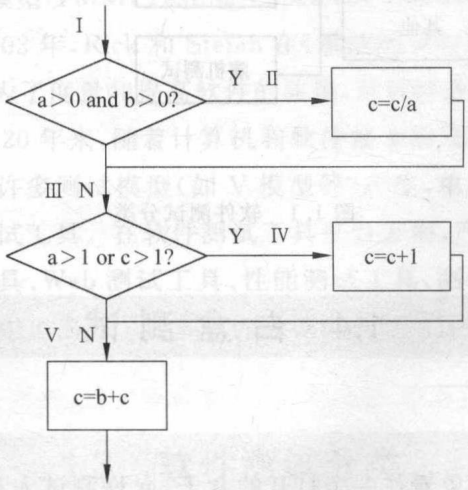

图 1.2 程序流程图

1. 语句覆盖

语句覆盖又称为线覆盖面或段覆盖面。其含义是指，选择足够数目测试数据，使被测程序中每条语句至少执行 1 次。

语句覆盖设计测试用例为 a=2,b=2,c=4,程序按照路径 Ⅰ→Ⅱ→Ⅲ→Ⅳ→Ⅴ 执行,程序段中的 5 个语句均执行 1 次,符合语句覆盖的要求。但是,如果测试用例选择 a=2,b=-2,c=4,程序按照路径 Ⅰ→Ⅲ→Ⅳ→Ⅴ 执行,则未能达到语句覆盖的要求。语句覆盖测试方法仅仅针对程序逻辑中的显式语句,对隐藏条件无法测试。若第一个逻辑运算符 and 误写成 or,测试用例 a=2,b=2,c=4 仍能达到语句覆盖的要求,但是无法发现程序中的误写错误。

语句覆盖可以直接应用于目标代码,不需要处理源代码。但是,作为最弱的逻辑覆盖方法,语句覆盖对控制结构不敏感,往往发现不了判断中逻辑运算符出现的错误,逻辑覆盖很低。

2. 判定覆盖

判定覆盖又称为分支覆盖或所有边覆盖,测试控制结构中的布尔表达式分别为真和假(例如 if 语句和 while 语句)。布尔型表达式被认为是一个整体,取值为 true 或 false,而不考虑内部是否包含“逻辑与”或者“逻辑或”等操作符。判定覆盖是指设计的测试用例使得程序中每个判定至少分别取“真”分支和取“假”分支各一次,即判断真假值均被满足。

判定覆盖设计测试用例如表 1.1 所示。

表 1.1 判定覆盖测试用例

测试用例	a>0 and b>0	a>1 or c>1	执行路径
a=1,b=1,c=5	T	T	Ⅰ→Ⅱ→Ⅲ→Ⅳ→Ⅴ
a=1,b=-2, c=-3	F	F	Ⅰ→Ⅲ→Ⅴ

判定覆盖作为语句覆盖的超集,比语句覆盖要多几乎一倍的测试路径,具备较强的测试能力。但由于判定语句往往由多个逻辑条件组合而成(如判定语句中包含 and、or、case),而判定覆盖仅仅判断其整个最终结果,无法发现判定内部每个条件的取值情况,必然会遗漏测试路径。

3. 条件覆盖

条件覆盖是指每个判断内部中的每个条件的所有可能取值至少满足 1 次。

针对 a>0 and b>0 判定条件表达式:a>0 取值为“真”,记为 T1;a>0 取值为“假”,记为 F1;b>0 取值为“真”,记为 T2;b>0 取值为“假”,记为 F2。针对条件表达式 a>1 or c>1,a>1 取值为“真”,记为 T3;a>1 取值为“假”,记为 F3;c>1 取值为“真”,记为 T4;c>1 取值为“假”,记为 F4。测试用例如表 1.2 所示。

表 1.2 条件覆盖测试用例

测试用例	覆盖条件	具体取值条件	执行路径
a=2,b=-1,c=-2	T1,F2,T3,F4	a>0,b<=0,a>1,c<=1	Ⅰ→Ⅲ→Ⅳ→Ⅴ
a=-1,b=2,c=3	F1,T2,F3,T4	a<=0,b>0,a<=1,c>1	Ⅰ→Ⅲ→Ⅳ→Ⅴ

条件覆盖比判定覆盖增加了对符合判定情况的测试,增加了测试路径。但条件覆盖

只能保证每个条件至少有一次为真，而不考虑所有的判定结果。表 1.2 中的测试用例 a=2,b=-1 和测试用例 a=-1,b=2 满足了条件覆盖的测试用例，保证了 a>0 and b>0两个条件各取 true 和 false 一次，但是其判定结果都是 false，并没有满足判定覆盖。故条件覆盖不一定包含判定覆盖。

4. 条件判定覆盖

由于判定条件不一定包含条件覆盖，条件覆盖也不一定包含判定覆盖，自然会有一种能同时满足两种覆盖标准的逻辑覆盖，这就是条件判定覆盖或者判定-条件覆盖(Condition/Decision Coverage，缩写 C/DC)。判定－条件覆盖的含义是通过设计足够的测试用例，使得所有条件的可能结果至少执行一次取值，同时，所有判断的可能结果至少执行一次。因此，判定-条件覆盖的测试用例满足如下条件：

(1) 所有条件的可能结果至少执行一次取值。

(2) 所有判断的可能结果至少执行一次。

条件判定覆盖设计测试用例如表 1.3 所示。

表 1.3 条件-判定覆盖测试用例

测试用例	覆盖条件	执行路径
a=2,b=1,c=5	T1,T2,T3,T4	Ⅰ→Ⅱ→Ⅲ→Ⅳ→Ⅴ
a=-1,b=-2,c=-3	F1,F2,F3,F4	Ⅰ→Ⅲ→Ⅴ

判定-条件覆盖同时满足判定、条件两种覆盖标准，是判定和条件覆盖设计方法的交集。

1.4.3 路径分析法

路径分析测试法是在程序控制流图的基础上，通过分析控制构造的环路复杂性导出独立路径集合，设计测试用例的方法。程序的所有路径作为一个集合，在这些路径集合中必然存在一个最短路径，这个最短的路径称为基路径或独立路径。

路径分析测试法的主要步骤如下所示。

步骤 1：绘制控制流图。

以详细设计或源代码作为基础，导出程序的控制流图。

步骤 2：计算圈复杂度 V(G)。

圈复杂度 V(G)是为程序逻辑复杂性提供定量的测度，该度量用于计算程序的基本独立路径数目，是所有语句至少执行一次的上界。

$$V(G)=\text{边数}-\text{节点数}+2$$

步骤 3：确定独立路径的集合。

独立路径是指至少引入程序的一个新处理语句集合或一个新条件的路径，即独立路径必须包含一条在定义之前不曾使用的边。

步骤 4：测试用例生成。