

Inhalt

1	Einleitung	1
1.1	Die Softwarekrise	3
1.2	Motivation für Softwaretests	5
1.2.1	Vor- und Nachteile	6
1.2.2	Varianten in der Namensgebung für Tests	7
1.2.3	Methodik: Wie kommen die Tests zustande?	9
1.2.4	Mathematik	10
1.3	Python	13
1.3.1	Schreibstil	14
1.3.2	Verzeichnisstruktur	16
1.3.3	Versionierung	17
1.3.4	Versionskontrollsysteme	18
1.3.5	Klinisch reine Umgebung	19
1.3.6	Dokumentation	20
1.3.7	Projektschnellstart	25
1.4	Interview: Dr. Mike Müller	26
2	Doctest	31
2.1	Definition: Docstring	31
2.2	Einfaches Beispiel	32
2.3	Der Interpreter	33
2.4	Eine Python-Datei	34
2.5	Dokumentierte Python-Datei	37
2.6	Tricks und Kniffe	42
2.6.1	Leerzeichen verbessern die Lesbarkeit	42
2.6.2	Variable Ergebnisse	43
2.6.3	Eine leere Zeile	44
2.6.4	Ausnahmebehandlung	46
2.6.5	Ausnahmebehandlung mit Details	48

2.6.6 Direkter Aufruf	49
2.6.7 Einen Test auslassen	51
2.7 Automatische Dokumentation	51
2.8 Interview: Dr. Stefan Schwarzer	52
3 Unittests machen Freude	57
3.1 Begriffe	58
3.1.1 Testfall	58
3.1.2 Testvorrichtung, test fixture	59
3.1.3 Testgruppe	60
3.1.4 Teststarter	60
3.1.5 Teststarter im Python-Modul	61
3.2 unittest Modul auf der Kommandozeile	62
3.2.1 Optionale Argumente	62
3.2.2 unittest in der Kommandozeile	63
3.2.3 Ablaufvereinfachung mit nosetests	63
3.2.4 Akzeptanz erwünscht	65
3.2.5 Fallunterscheidung	67
3.2.6 Ausnahmebehandlung	69
3.2.7 Vergleichsmöglichkeiten im Testfall	73
3.2.8 Assertions	73
3.2.9 Tests auslassen	75
3.3 Erweiterungen	77
3.3.1 Fixtures	77
3.3.2 Testabdeckung	78
3.3.3 Testabdeckung als HTML-Ausgabe	80
3.4 Vortäuschen falscher Tatsachen	82
3.4.1 Mock als Dekorator	82
3.4.2 Mock im Zusammenhang mit Kontextmanagern	82
3.4.3 Mock und die Nutzung im Testfall	84
3.4.4 Lern- und Spielwiese	85
3.5 Fingerübung I: Testgetriebene Entwicklung	86
3.5.1 Erster Testcode	87
3.5.2 Gültige Eingaben	89
3.5.3 Ungültige Eingaben	92
3.5.4 Tests erfolgreich?	95
3.5.5 Vollständige Testabdeckung?	97
3.6 Interview: Christian Theune	98

4	Nose	101
4.1	Hilfestellung	101
4.2	Konfiguration	102
4.3	Plugins	103
4.3.1	Plugin Beispiel: Test-Laufzeiten ermitteln	103
4.3.2	Plugin Integration in nosetests	112
4.3.3	Nur ein getestetes Plugin ist ein gutes Plugin	114
4.4	Interview: Stefan Hagen	115
5	pytest	117
5.1	Hilfestellung	117
5.2	Konfiguration	120
5.2.1	Markierungen	121
5.2.2	Testvorrichtungen	128
5.3	Testbeispiele	132
5.3.1	Aussagekräftige Fehlermeldungen	134
5.3.2	Ausnahmebehandlung	136
5.3.3	py.test mit unittests	140
5.4	Plugins	141
5.4.1	Plugin Beispiel: Bericht als csv-Datei erzeugen	142
5.4.2	Plugin Integration in py.test	146
5.4.3	Nur ein getestetes Plugin ist ein gutes Plugin	148
5.4.4	Die Benutzung des neuen Plugins	156
5.5	Fingerübung II: <code>sign(x)</code> , <code>csign(z)</code>	159
5.5.1	Teilung vor der Erweiterung	160
5.5.2	Signum für komplexe Zahlen	161
5.5.3	Der erste Testfall	163
5.5.4	Ungültige Eingabewerte	164
5.5.5	Doctests mit py.test	166
5.6	Interview: Holger Krekel	168
6	tox	171
6.1	Einstellungen	171
6.2	Ein Beispiel	172
6.3	Ein Testlauf	174
6.4	Interview: Bastian Ballmann	179
7	GUI Tests	183
7.1	PyQt4	183
7.1.1	Beispiel GUI	183

7.1.2	GUI Ansicht	186
7.1.3	GUI Test	186
7.1.4	Testabdeckung	187
7.2	Django: Testgetriebene Webentwicklung	188
7.2.1	Unittests und Funktionale Tests	189
7.2.2	Django Start	192
7.2.3	Django Entwicklungsserver	194
7.2.4	Eine Kurzgeschichte	196
7.2.5	Django Unittests	199
7.2.6	Unittest für eine View	203
7.2.7	View aus der Vorlage	206
7.2.8	Wo bleiben die Daten?	213
7.2.9	ORM und Persistenz	214
7.3	Interview: Guido Günther	221
8	Großes Python-Kino	223
8.1	SaltStack	223
8.1.1	Quellen und Unittests	224
8.1.2	Integrationstests	226
8.1.3	Dokumentation	228
8.2	OpenStack	228
8.2.1	Dokumentation, der Schlüssel zur Wolke	229
8.2.2	Keystone Tests	231
8.3	Interview: Julien Danjou	233
	Anhang	237
	A Abbildungsverzeichnis	237
	B Literaturhinweise	247
	C Stichwortverzeichnis	249



<http://www.springer.com/978-3-662-48602-3>

Softwaretests mit Python

Hubertz, J.

2016, IX, 254 S. 150 Abb., Hardcover

ISBN: 978-3-662-48602-3