TDD WITH PYTHON

Test-driven Development with Python

Christian Janeczek 5AHITT

Inhaltsverzeichnis

Aufgabenstellung

Schreiben Sie die Klasse Bruch in einem Modul bruch
Nutzen Sie die Testklassen in PyCharm.

Ziel: Coverage > 95% und mindestens 50 von 63 Testfällen

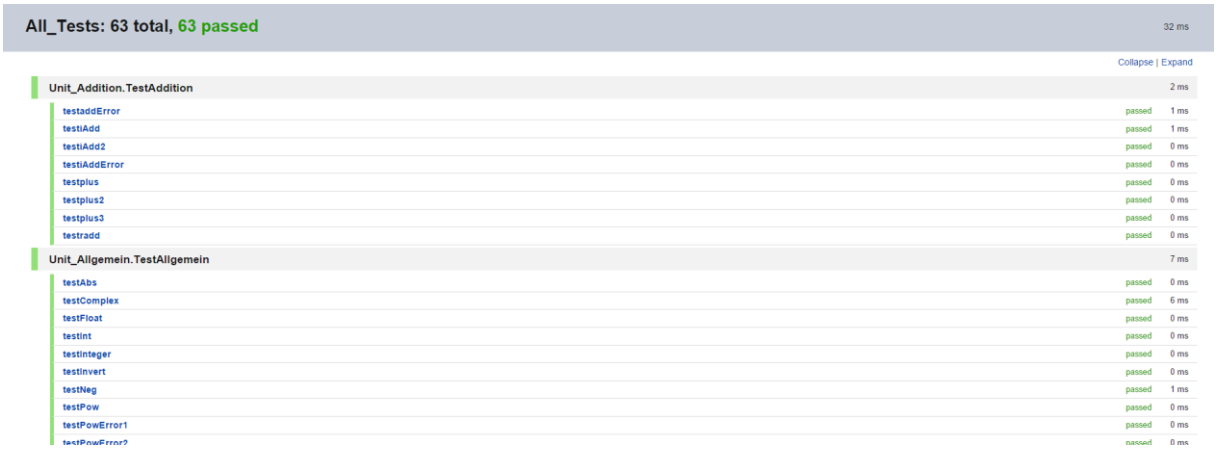
Empfohlene Vorgehensweise:

1. Projekt in PyCharm erstellen
2. Modul bruch erstellen
3. Klasse Bruch erstellen
4. Test-Ordner erstellen
5. Unit-Tests entpacken und lauffähig machen

Abgabe:

Protokoll mit Testreports; Bruch-Klasse; generierte Code-Dokumentation

Testreport



Coverage Report

Coverage report: 97%

Module	*statements*	*missing*	*excluded*	*coverage*
C:\Users\Chris\PycharmProjects\tdd-python\bruch	121	3	0	98%
C:\Users\Chris\PycharmProjects\tdd-python\test\Unit_Addition	35	1	0	97%
C:\Users\Chris\PycharmProjects\tdd-python\test\Unit_Allgemein	59	1	0	98%
C:\Users\Chris\PycharmProjects\tdd-python\test\Unit_Division	44	1	0	98%
C:\Users\Chris\PycharmProjects\tdd-python\test\Unit_Multiplikation	35	1	0	97%
C:\Users\Chris\PycharmProjects\tdd-python\test\Unit_String	20	1	0	95%
C:\Users\Chris\PycharmProjects\tdd-python\test\Unit_Subtraktion	35	1	0	97%
C:\Users\Chris\PycharmProjects\tdd-python\test\Unit_Vergleich	25	1	0	96%
C:\Users\Chris\PycharmProjects\tdd-python\test\Unit_Zusatz	20	1	0	95%
Total	**394**	**11**	**0**	**97%**

coverage.py v3.7.1

Generierte Dokumentation mit Sphinx

```
html:
	$(SPHINXBUILD) -b html $(ALLSPHINXOPTS) $(BUILDDIR)/html
	@echo
	@echo "Build finished. The HTML pages are in $(BUILDDIR)/html."
```

```
make html
```

bruch module

class bruch.**Bruch**(*arg1=None*, *arg2=None*)

Bases: builtins.object

Created on 15.01.2015 @author: Chris

_Bruch__makeBruch(*arg1*, *arg2=None*)

Parameters:
- **self** – it's the convention
- **arg1** – value of the zaehler
- **arg2** – value of the nenner

Returns: returns the created bruch of the 2 parameters zaehler and nenner

__abs__()

Parameters: **self** – it's the convention

Returns: returns the abstract value of the bruch object

__add__(*b1*, *b2*)

Parameters:
- **self** – it's the convention
- **b1** – first addend
- **b2** – second addend

Returns: returns the sum of the addends as a bruch object

__dict__ = *mappingproxy({'_Bruch__makeBruch': <function Bruch._Bruch__makeBruch at 0x043B3030>, '__module__': 'bruch', '__float__': <function Bruch.__float__ at 0x04383C00>, '__add__': <function Bruch._ 0x04383D20>, '__truediv__': <function Bruch.__truediv__ at 0x043B3078>, '__rsub__': <function Bruch.__rsub__ at 0x04383E40>, '__imul__': <function Bruch.__imul__ at 0x04383FA8>, '__rmul__': <function Bruc at 0x04383F60>, '__sub__': <function Bruch.__sub__ at 0x04383DF8>, '__itruediv__': <function Bruch.__itruediv__ at 0x043B30C0>, 'zaehler': 0, '__ge__': <function Bruch.__ge__ at 0x043839C0>, '__int__' Bruch.__int__ at 0x04383C90>, '__iadd__': <function Bruch.__iadd__ at 0x04383D68>, '__str__': <function Bruch.__str__ at 0x04383B70>, '__eq__': <function Bruch.__eq__ at 0x04383978>, '__init__' Bruch.__init__ at 0x04383A98>, '__rtruediv__': <function Bruch.__rtruediv__ at 0x043B3108>, '__doc__': '\n Created on 15.01.2015\n @author: Chris\n ', '__neg__': <function Bruch.__neg__ at 0x04383ED0>, <function Bruch.__isub__ at 0x04383E88>, '__weakref__': <attribute '__weakref__' of 'Bruch' objects>, '__abs__': <function Bruch.__abs__ at 0x04383C48>, '__dict__': <attribute '__dict__' of 'Bruch' objects> <function Bruch.__ne__ at 0x04383A08>, '__iter__': <function Bruch.__iter__ at 0x043B3198>, '__lt__': <function Bruch.__lt__ at 0x04383B28>, '__hash__': None, '__le__': <function Bruch.__le__ at 0x04383A50 <function Bruch.__gt__ at 0x04383AE0>, 'nenner': 1, '__radd__': <function Bruch.__radd__ at 0x04383DB0>, '__pow__': <function Bruch.__pow__ at 0x043B3150>, '__mul__': <function Bruch.__mul__ at 0x0 '__invert__': <function Bruch.__invert__ at 0x04383CD8>})*

__eq__(*other*)

Parameters:
- **self** – it's the convention
- **other** – the object which equality should be checked

Returns: True: if both objects are equal, else return False

__float__()

Parameters: **self** – it's the convention

Returns: returns the converted value of self as a float type

Arbeitszeitaufzeichnung

DATE	PHASE	TASK	ESTIMATION	ACTUAL	COMMENT
01.15.2014	Implementierung	Bruch-Klasse für die von Herr Professor	2:30:00	2:00:00	Testing is sexy
01.22.2014	Implementierung	Fertigstellung der Test-Cases + Code Dokumentation Generierung mit Sphinx	2:00:00	3:00:00	Sphinx on Windows is a b**ch
		SUM	4:30:00	5:00:00	

Conclusio

- Test-Driven Development in Verbindung mit Continous Integration ist anziehender als die magnetische Flussdichte eines Magnetars.

- Q: “Knock, knock.”
- A: “Who’s there?”
- Very long pause….
- Q: “Java.”

- Q: how many programmers does it take to change a light bulb?
- A: none, that's a hardware problem

- Programming is like sex:
- One mistake and you have to support it for the rest of your life.

- When your hammer is C++, everything begins to look like a thumb.

Quellenangabe

[1] **Beginning Test-Driven Development in Python,** David Sale**,**
http://code.tutsplus.com/tutorials/beginning-test-driven-development-in-python--net-30137

[2] **Test-Driven Development in Python,** Jason Diamond**,**
http://www.openp2p.com/pub/a/python/2004/12/02/tdd_pyunit.html

[3] **More Test-Driven Development in Python,** Jason Diamond**,**
http://www.onlamp.com/pub/a/python/2005/02/03/tdd_pyunit2.html

[4] **Magic Methods in Python,** Rafe Kettler**,** http://www.rafekettler.com/magicmethods.html