

# LU03b - PyTest



PyTest ist ein Test-Framework basierend auf Python. Es eignet sich vor allem, um die Schnittstellen einzelner Funktionen zu testen.

## Einstieg

Die Nutzung von PyTest in unserer IDE PyCharm ist relativ einfach.

### Paket "pytest" installieren

Zuerst musst du das Paket `pytest` im Virtual Environment deines Projekts installieren. Gehe in den Tab Terminal von PyCharm. Achte auf die Anzeige `(venv) ...` am Anfang des Prompts.

```
(venv) PS C:\BZZ\Python\MyProject>
```

Falls diese Anzeige fehlt, musst du die virtuelle Umgebung zuerst aktivieren. Gib dazu unter Windows den Befehl `.\venv\Scripts\activate.ps1` ein.

Nun kannst du `pytest` installieren.

```
(venv) PS C:\BZZ\Python\MyProject> pip3 install pytest
```

Ich empfehle dir, die Pakete immer im Virtual Environment zu installieren. Dadurch gibt es weniger Probleme mit der Kompatibilität verschiedener Projekte.

### Modul für Testfälle

Nachdem `pytest` installiert ist, erstellst du ein neues Modul (Python File) für deine Tests. Als Dateinamen ergänze ich den Namen des zu testenden Moduls (z.B. `main.py`) um `test_` (z.B. `test_main.py`).

### Unit Tests

Die einzelnen Unit Tests werden als Funktionen in Python programmiert. Zum Beispiel:

```
def test_normal():  
    result = factorial(7)  
    assert result == 5040
```

- Der Funktionsname für den Test muss mit `test_` beginnen. Andernfalls erkennt `pytest` die Funktion nicht.
- Wir rufen die zu testende Funktion (im Beispiel `factorial`) mit den Testdaten auf.
- Der Befehl `assert` vergleicht das tatsächliche Resultat mit dem erwarteten Resultat.

## Assert

Der Befehl `assert` ist eine spezielle Bedingung, die wir zum Testen und Debuggen von Code verwenden. Falls die Bedingung erfüllt ist, wird `true` zurück gegeben. Sonst wird eine `AssertionError`-Exception geworfen. Wir könnten das gleiche Resultat auch mit `if/else` erreichen:

<b>assert</b>	<b>if / else</b>
<code>assert result == 5040</code>	<pre>if result == 5040:     return true else:     raise AssertionError</pre>

## Beispiel

Anhand dieses einfachen Beispiels siehst du, wie eine Python-Funktion getestet werden kann.

### factorial.py

Dieses Modul enthält eine Funktion um die Fakultät einer Zahl zu berechnen.

```
def factorial(number):
    fact = 1
    for count in range(1, number + 1):
        fact = fact * count
    return fact
```

### test\_factorial.py

Dieses Modul enthält meine Unit Tests.

```
from factorial import factorial

def test_normal():
    result = factorial(7)
    assert result == 5040

def test_zero():
    result = factorial(0)
    assert result == 1
```

# Tutorials

- [Tutorialpoints](#)
- [Youtube](#)

---

## M450-LU03



Marcel Suter

From:

<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:

<https://wiki.bzz.ch/modul/m450/learningunits/lu03/pytest?rev=1711631267>

Last update: **2024/03/28 14:07**

