LU03c - PyTest: Input und Output

PyTest kann Ausgaben des Programms und Eingaben des Benutzers im Terminal simulieren.

Ausgaben

Manche Funktionen erzeugen Ausgaben im Terminal. Besonders bei Fehlersituationen werden solche Ausgaben als Teil eines Logfiles genutzt, um Fehler zu protokollieren. Andere Funktionen schreiben Werte in eine Datei. Beide Arten von Ausgaben können wir im Rahmen von Unit Tests mit PyTest überprüfen.

Indem wir unserer Testfunktion `capsys` als Argument mitgeben, können wir auf den Output einer Funktion zugreifen:

```
def test_normal(capsys):
    factorial(7)
    output = capsys.readouterr().out
    assert output == '5040\n'
```

In diesem Beispiel holen wir mittels `capsys.readouterr().out` den Output der Funktion `factorial(x)`. Diesen Output können wir dann mittels `assert` mit dem erwarteten Output vergleichen.

Beispiel

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

factorial.py

Ich verwende wieder das Beispiel aus dem letzten Kapitel. Diesmal gibt die Funktion das Resultat im Terminal aus.

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

test_factorial.py

Dieses Modul enhält meine Unit Tests.

```
from factorial import factorial


def test_normal():
    factorial(7)
    output = capsys.readouterr().out
    assert output == '5040\n'

def test_zero():
    factorial(0)
    output = capsys.readouterr().out
    assert output == '1\n'
```

Eingaben

Zum Simulieren von Eingaben steht uns `monkeypatch` zur Verfügung. Neben diversen anderen Funktionen, kann `monkeypatch` die Eingaben des Benutzers simulieren.

```
def test_normal(capsys, monkeypatch):
    inputs = iter([7])
    monkeypatch.setattr('builtins.input', lambda _: next(inputs))
    factorial()
    captured = capsys.readouterr()
    assert captured.out == '5040\n'
```

In diesem Beispiel wird die Zahl 7 als Benutzereingabe an die getestete Funktion gesendet. Der Befehl `monkeypatch.setattr('builtins.input', lambda _: next(ITERATOR))` erwartet eine Aufzählung von Eingabewerten. Diese Aufzählung erzeugen wir mit dem Befehl `inputs = iter([7])`.

Beispiel

Nun passen wir das Beispiel um den Einsatz von Benutzereingaben an.

factorial.py

```
def factorial():
    number = int(input('Number >'))
    fact = 1
    for count in range(1, number + 1):
        fact = fact * count
    print fact
```

test_factorial.py

Dieses Modul enhält meine Unit Tests.

```
from factorial import factorial


def test_normal(capsys, monkeypatch):
    inputs = iter([7])
    monkeypatch.setattr('builtins.input', lambda _: next(inputs))
    factorial()
    captured = capsys.readouterr()
    assert captured.out == '5040\n'


def test_zero(capsys, monkeypatch):
    inputs = iter([0])
    monkeypatch.setattr('builtins.input', lambda _: next(inputs))
    factorial()
    captured = capsys.readouterr()
    assert captured.out == '1\n'
```

M450-LU03