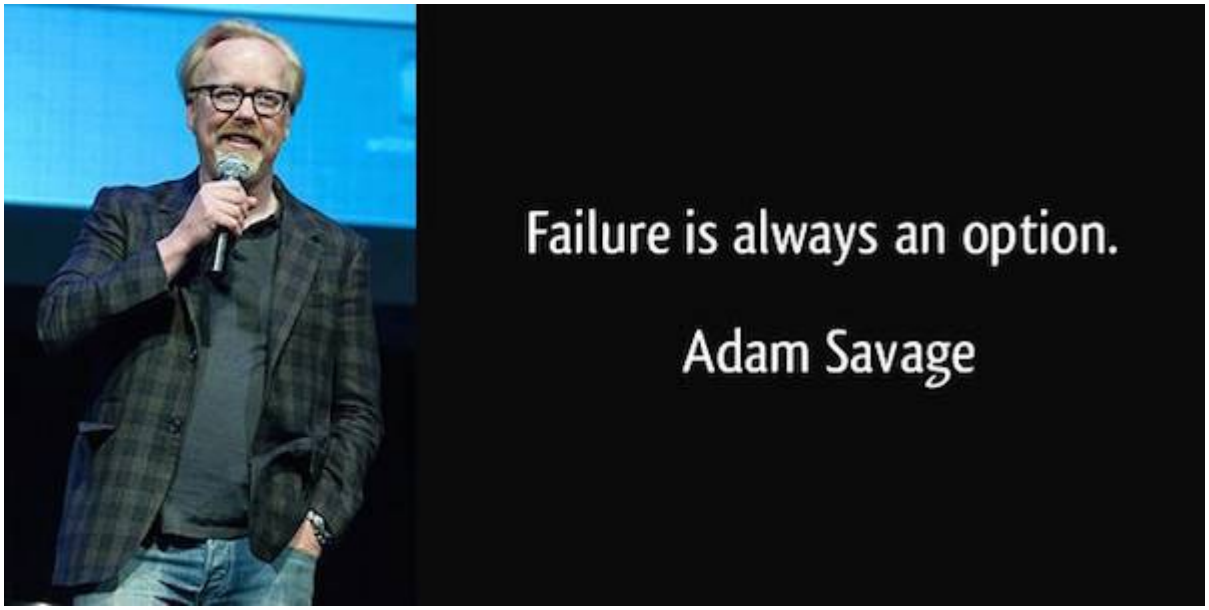
LU04d - Fehler erwartet

Es gibt Situationen, in den wir erwarten, dass ein Test scheitert. Anstatt den Test zu überspringen, markieren wir ihn mit `@pytest.mark.xfail`.



Es mag zunächst wenig intuitiv sein, einen Test auszuführen, der erwartungsgemäss scheitern soll. Wenn der Test sowieso fehlschlägt, kann ich den Test ja einfach vorerst überspringen.

Betrachten wir einmal folgende Situation in einem Projekt:

- Tag 1:
  - Andrea hat den Auftrag die Klasse `Customer` zu programmieren.
  - Raphael schreibt die Tests für die Applikation welche diese Klasse enthält. Da die Klasse `Customer` noch nicht lauffähig ist, markiert Bert die Tests mit `@pytest.mark.skip`.
- Tag 2:
  - Andrea hat die Klasse `Customer` teilweise fertig gestellt.
  - Raphael führt die Tests durch. Aufgrund der Markierung werden die Tests für `Customer` nicht ausgeführt und PyTest meldet keine Fehler.
- Tag 3:
  - Andrea ist fertig mit dem Programmieren der Klasse `Customer`.
  - Bei den Tests von Raphael tritt noch immer kein Problem auf, da die Tests ja übersprungen werden.

Im dümmsten Fall glauben Andrea und Raphael, dass die Applikation korrekt läuft, da ja keine Fehler bei den Tests erkannt wurden.

Nehmen wir nun an, Raphael hätte die Tests mit `@pytest.mark.xfail` markiert:

- Schon am Tag 2 würden einige Tests das Resultat `unexpectedly passing (XPASS)` melden.
- Am Tag 3 würden (fast) alle Tests XPASS melden.

Dadurch merkt Raphael automatisch, dass die Klasse `Customer` ganz oder teilweise umgesetzt ist.

pytest.xfail

Dekorator

```
@pytest.mark.xfail
def test_one():
   ...

@pytest.mark.xfail(reason='this is expected to fail')
def test_two():
    ...

@pytest.mark.xfail(condition=sys.version_info < (3, 7))
def test_three():
    ...
```

Innerhalb der Funktion

Wir können `pytest.xfail` auch innerhalb einer Testfunktion aufrufen. Im Gegensatz zum Dekorator wird dabei jedoch die Ausführung der Funktion abgebrochen und keine weiteren Befehle ausgeführt. Dadurch ist diese Variante weniger geeignet.

```
def test_read_entry(monkeypatch, capsys, attributes):
    if attributes not in ['BASIS', 'ERWEITERT']:
        pytest.xfail('Test kann erst durchgeführt werden, wenn die Attribute
der Klasse "Entry" definiert sind')
```

Exception erwartet

Beim Testen müssen wir immer auch die korrekte Verarbeitung von Ausnahmefällen prüfen. Daher schreiben wir Unittests, die bewusst eine Exception im Programm auslösen sollen.

```
def test_illegal_value():
    with pytest.raises(ValueError):
        account.balance = '3ex'
```

In diesem Beispiel geben wir einen formal falschen Wert für den Kontostand an. Sofern die Methode einen `ValueError` auslöst, gilt der Test als erfüllt. Ausführlicher wird `pytest.raises` in LU03e - Klassen und Fehlerfälle testen erklärt.

M450-LU04, M450-C1A

Marcel Suter

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu04/xfail?rev=1794898891

Last update: **2026/11/17 08:01**