LU04.L03 - Erwartete Fehler

Lösung zu LU04.A03 - Erwartete Fehler. Die vollständige Lösung liegt im Branch `solution` des Repositorys.

Teil 1 und 2: Tests mit xfail-Markierungen

So sieht `test_cart.py` am Ende von Teil 2 aus, solange `remove_item` und `apply_discount` noch Stubs sind:

```python
"""Unit Tests für den Warenkorb."""
import pytest

from cart import ShoppingCart


@pytest.fixture(name='cart')
def fixture_cart():
    """Ein leerer Warenkorb."""
    return ShoppingCart()


# Teil 1: Die Methoden add_item und total sind fertig programmiert.

def test_add_item(cart):
    """Ein Artikel wird mit Preis und Menge abgelegt."""
    cart.add_item('Apfel', 0.80, 3)
    assert cart.items['Apfel'] == {'price': 0.80, 'quantity': 3}


def test_add_item_twice(cart):
    """Derselbe Artikel erhöht nur die Menge."""
    cart.add_item('Apfel', 0.80)
    cart.add_item('Apfel', 0.80, 2)
    assert len(cart.items) == 1
    assert cart.items['Apfel']['quantity'] == 3


def test_total(cart):
    """Das Total summiert Preis mal Menge."""
    cart.add_item('Apfel', 0.80, 3)
    cart.add_item('Brot', 4.35)
    assert cart.total() == pytest.approx(6.75)


def test_add_item_negative_price(cart):
    """Ein negativer Preis wird abgelehnt."""
```

```
    with pytest.raises(ValueError):
        cart.add_item('Apfel', -0.80)


# Teil 2: remove_item und apply_discount sind erst Stubs.

@pytest.mark.xfail(raises=NotImplementedError, strict=True,
reason='remove_item noch nicht fertig')
def test_remove_item(cart):
    """Ein Artikel wird vollständig entfernt."""
    cart.add_item('Apfel', 0.80, 3)
    cart.add_item('Brot', 4.35)
    cart.remove_item('Apfel')
    assert list(cart.items) == ['Brot']


@pytest.mark.xfail(raises=NotImplementedError, strict=True,
reason='remove_item noch nicht fertig')
def test_remove_unknown_item(cart):
    """Ein unbekannter Artikel löst einen KeyError aus."""
    with pytest.raises(KeyError):
        cart.remove_item('Banane')


@pytest.mark.xfail(raises=NotImplementedError, strict=True,
reason='apply_discount noch nicht fertig')
def test_apply_discount(cart):
    """Der Rabatt wird vom Total abgezogen."""
    cart.add_item('Buch', 80.00)
    cart.apply_discount(25)
    assert cart.total() == pytest.approx(60.00)


@pytest.mark.xfail(raises=NotImplementedError, strict=True,
reason='apply_discount noch nicht fertig')
@pytest.mark.parametrize('percent', [-1, 51])
def test_apply_discount_invalid(cart, percent):
    """Rabatte ausserhalb von 0 bis 50 werden abgelehnt."""
    with pytest.raises(ValueError):
        cart.apply_discount(percent)
```

```
$ pytest -rxX
test_cart.py ....xxxxx
XFAIL test_cart.py::test_remove_item - remove_item noch nicht fertig
XFAIL test_cart.py::test_remove_unknown_item - remove_item noch nicht fertig
XFAIL test_cart.py::test_apply_discount - apply_discount noch nicht fertig
XFAIL test_cart.py::test_apply_discount_invalid[-1] - apply_discount noch
nicht fertig
XFAIL test_cart.py::test_apply_discount_invalid[51] - apply_discount noch
```

```
nicht fertig
4 passed, 5 xfailed
```

Teil 3: Stubs durch Code ersetzen

```
    def remove_item(self, name):
        """
        Entfernt einen Artikel vollständig aus dem Warenkorb.
        :param name: Bezeichnung des Artikels
        :raises KeyError: wenn der Artikel nicht im Warenkorb ist
        """
        del self.items[name]

    def apply_discount(self, percent):
        """
        Gewährt einen Rabatt in Prozent auf das Total.
        :param percent: Rabatt in Prozent, erlaubt sind 0 bis 50
        :raises ValueError: wenn der Rabatt kleiner als 0 oder grösser als
50 ist
        """
        if percent < 0 or percent > 50:
            raise ValueError('Der Rabatt muss zwischen 0 und 50 Prozent
liegen')
        self.discount = percent
```

Sobald die Methoden funktionieren, bestehen die markierten Tests. Wegen `strict=True` wertet PyTest das als Fehler:

```
$ pytest
FAILED test_cart.py::test_remove_item - [XPASS(strict)] remove_item noch
nicht fertig
...
5 failed, 4 passed
```

Das ist gewollt: PyTest meldet, dass der Code jetzt fertig ist und die Markierung entfernt werden muss. Ohne `strict=True` würden die Tests nur als XPASS erscheinen, und man könnte die Markierung leicht vergessen.

Nach dem Entfernen aller `@pytest.mark.xfail`-Zeilen:

```
$ pytest
test_cart.py .........
9 passed
```

Erläuterungen

- **raises=NotImplementedError:** Der Test gilt nur als erwarteter Fehlschlag, wenn genau der

Stub die Ursache ist. Ein Tippfehler im Test, der einen `NameError` auslöst, macht den Test sofort rot und fällt auf.

- **test_remove_unknown_item:** `del` auf einen fehlenden Schlüssel löst bereits einen `KeyError` aus. Es braucht dafür keinen zusätzlichen Code.
- **Fixture mit name:** `@pytest.fixture(name='cart')` verhindert die pylint-Warnung `redefined-outer-name`, die entsteht, wenn Fixture und Parameter gleich heissen.

M450-LU04

Marcel Suter

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

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu04/loesungen/xfail?rev=1794900261

Last update: **2026/11/17 08:24**