LU16.L01 – Behave Grundlagen

Musterlösung zu LU16.A01 – Behave Grundlagen.

Ergebnis

```
$ behave
2 features passed, 0 failed, 0 skipped
9 scenarios passed, 0 failed, 0 skipped
27 steps passed, 0 failed, 0 skipped
```

Hinweise

- **Platzhalter im Schritt:** Das Gerüst verwendet `{expected:float}`. Diesen Typ kennt behave nicht und meldet einen Fehler. Für Fliesskommazahlen schreibt man `{expected:g}` (allgemeine Zahl) oder `{expected:f}` (Zahl mit Dezimalpunkt). `{c:d}` steht für eine ganze Zahl, auch negativ.
- **Toleranz:** `37 °C` ergibt rechnerisch `98.60000000000001 °F`. Deshalb vergleichen die Then-Schritte mit einer Toleranz statt mit ==.
- **Exceptions:** Der When-Schritt fängt die Exception ab und speichert sie im `context`. Erst der Then-Schritt prüft sie. So bleibt die Trennung Given – When – Then erhalten.
- **bool ist auch int:** In Python ist `True` ein `int`. Die Prüfung schliesst `bool` deshalb ausdrücklich aus.
- **Zusatz:** Das Feature enthält eine `Scenario Outline` mit weiteren Werten (100 °C, −40 °C, 37 °C).

Projektstruktur

```
behave_project/
 ├── app/
 │    ├── __init__.py
 │    ├── converter.py
 │    └── auth.py
 └── features/
      ├── temperature.feature
      ├── login.feature
      └── steps/
           ├── temperature_steps.py
           └── login_steps.py
```

Auftrag 1 – Temperatur-Umrechnung

features/temperature.feature

```
Feature: Temperaturumrechnung
  Damit ich Temperaturwerte standardisiert anzeigen kann
  möchte ich zwischen Celsius und Fahrenheit korrekt umrechnen.

  Scenario: Celsius zu Fahrenheit
    Given eine Celsius-Temperatur von 0
    When ich nach Fahrenheit umrechne
    Then erhalte ich 32.0 Fahrenheit

  Scenario: Fahrenheit zu Celsius
    Given eine Fahrenheit-Temperatur von 212
    When ich nach Celsius umrechne
    Then erhalte ich 100.0 Celsius

  Scenario: Ungültige Eingabe (String)
    Given eine ungültige Eingabe "heiss"
    When ich eine Umrechnung ausführe
    Then wird ein ValueError ausgelöst

  Scenario Outline: Weitere Werte von Celsius zu Fahrenheit
    Given eine Celsius-Temperatur von <celsius>
    When ich nach Fahrenheit umrechne
    Then erhalte ich <fahrenheit> Fahrenheit

    Examples:
      | celsius | fahrenheit |
      | 100     | 212.0      |
      | -40     | -40.0      |
      | 37      | 98.6       |
```

features/steps/temperature_steps.py

```python
"""Schrittdefinitionen für temperature.feature"""
from behave import given, when, then

from app.converter import c_to_f, f_to_c

TOLERANCE = 1e-6


@given('eine Celsius-Temperatur von {c:d}')
def step_given_celsius(context, c):
    context.input_c = c
```

```
@given('eine Fahrenheit-Temperatur von {f:d}')
def step_given_fahrenheit(context, f):
    context.input_f = f


@given('eine ungültige Eingabe "{text}"')
def step_given_invalid(context, text):
    context.input_invalid = text


@when('ich nach Fahrenheit umrechne')
def step_when_to_f(context):
    context.result = c_to_f(context.input_c)


@when('ich nach Celsius umrechne')
def step_when_to_c(context):
    context.result = f_to_c(context.input_f)


@when('ich eine Umrechnung ausführe')
def step_when_convert_any(context):
    context.exc = None
    try:
        c_to_f(context.input_invalid)
    except ValueError as error:
        context.exc = error


@then('erhalte ich {expected:g} Fahrenheit')
def step_then_f(context, expected):
    assert abs(context.result - expected) < TOLERANCE, f'{context.result}
statt {expected}'


@then('erhalte ich {expected:g} Celsius')
def step_then_c(context, expected):
    assert abs(context.result - expected) < TOLERANCE, f'{context.result}
statt {expected}'


@then('wird ein ValueError ausgelöst')
def step_then_error(context):
    assert isinstance(context.exc, ValueError), 'Es wurde kein ValueError
ausgelöst'
```

app/converter.py

```
"""Umrechnung zwischen Celsius und Fahrenheit"""


def _check_number(value):
    """Akzeptiert nur int und float (bool ist ausgeschlossen)"""
    if isinstance(value, bool) or not isinstance(value, (int, float)):
        raise ValueError(f'Ungültige Temperatur: {value!r}')


def c_to_f(celsius):
    """Rechnet Celsius in Fahrenheit um: F = C * 9/5 + 32"""
    _check_number(celsius)
    return celsius * 9 / 5 + 32


def f_to_c(fahrenheit):
    """Rechnet Fahrenheit in Celsius um: C = (F - 32) * 5/9"""
    _check_number(fahrenheit)
    return (fahrenheit - 32) * 5 / 9
```

Auftrag 2 – Login

Die Feature-Datei `features/login.feature` bleibt wie in der Vorlage.

features/steps/login_steps.py

```
"""Schrittdefinitionen für login.feature"""
from behave import given, when, then

from app.auth import AuthService


@given('es existiert ein Benutzer "{username}" mit Passwort "{password}"')
def step_given_user_exists(context, username, password):
    context.auth = AuthService()
    context.auth.register(username, password)


@when('ich mich mit Benutzer "{username}" und Passwort "{password}"
anmelde')
def step_when_login(context, username, password):
    context.logged_in = context.auth.login(username, password)
```

```
@then('bin ich eingeloggt')
def step_then_logged_in(context):
    assert context.logged_in is True


@then('ist der Login abgelehnt')
def step_then_denied(context):
    assert context.logged_in is False
```

app/auth.py

```
"""Minimaler Login-Service ohne externe Abhängigkeiten"""


class AuthService:
    """Verwaltet Benutzer in einem Dictionary und prüft Logins"""

    def __init__(self):
        self._users = {}

    def register(self, username, password):
        """Registriert einen Benutzer. Leere Werte lösen einen ValueError
aus."""
        if not username or not password:
            raise ValueError('Benutzername und Passwort dürfen nicht leer
sein')
        self._users[username] = password

    def login(self, username, password):
        """Liefert True, wenn Benutzer und Passwort zusammenpassen, sonst
False"""
        return self._users.get(username) == password
```

`login` liefert für einen unbekannten Benutzer `False`, weil `dict.get` dann `None` zurückgibt und `None` nie einem Passwort entspricht.

M450-LU16, M450-B1A, M450-C2I

Kevin Maurizi

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

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu16/loesungen/behave

Last update: **2026/11/17 17:45**