LU11.L01 - TDD-Kata: Passwortprüfer

Musterlösung zu LU11.A01 - TDD-Kata: Passwortprüfer.

Verlauf der Runden

Runde	Red	Green	Refactor
1	`test_valid_password`	`return []` (Fake it)	–
2	`test_too_short`, `test_length_limit`	`if len(password) < 8: return ['mindestens 8 Zeichen']`	Konstante `MIN_LENGTH`
3	`test_without_digit`	Liste `errors` anlegen, zweites `if` mit `any(char.isdigit() …)`	–
4	`test_without_uppercase`	drittes `if`	drei fast gleiche `if`: Regeln in eine Liste aus (Bedingung, Meldung) auslagern
5	`test_without_special_character`	eine Zeile in der Regelliste	Konstante `SPECIAL_CHARACTERS`
6	`test_several_errors`, `test_empty_password`	bereits grün, weil alle Regeln geprüft werden	–
7	`test_not_a_string`	`isinstance`-Prüfung am Anfang	Docstrings, pylint

Runde 6 ist sofort grün. Das ist erlaubt: Der Test dokumentiert eine Anforderung und schützt sie bei späteren Änderungen. Wer bei Runde 3 bereits `return` nach der ersten verletzten Regel programmiert hat, sieht hier einen roten Test.

Code

password.py

```
""" Prüft, ob ein Passwort die Regeln erfüllt """
MIN_LENGTH = 8
SPECIAL_CHARACTERS = '!?@#$%&*+-_.'


def check_password(password):
    """
    Prüft ein Passwort und liefert alle verletzten Regeln
    :param password: das Passwort
    :return: Liste mit Fehlermeldungen, leer wenn das Passwort gültig
ist
```

```
    :raises TypeError: wenn password kein String ist
    """
    if not isinstance(password, str):
        raise TypeError('Das Passwort muss ein String sein')
    rules = [
        (len(password) >= MIN_LENGTH, f'mindestens {MIN_LENGTH}
Zeichen'),
        (any(char.isdigit() for char in password), 'mindestens eine
Ziffer'),
        (any(char.isupper() for char in password), 'mindestens ein
Grossbuchstabe'),
        (any(char in SPECIAL_CHARACTERS for char in password),
'mindestens ein Sonderzeichen'),
    ]
    return [message for fulfilled, message in rules if not fulfilled]
```

test_password.py

```
""" Unit Tests für check_password, in der Reihenfolge, in der sie
entstanden sind """
import pytest

from password import check_password


def test_valid_password():
    """ Runde 1: ein gültiges Passwort liefert keine Fehler """
    assert check_password('Sommer24!') == []


def test_too_short():
    """ Runde 2: zu kurz """
    assert check_password('So24!') == ['mindestens 8 Zeichen']


def test_length_limit():
    """ Runde 2: Grenzwert, genau 8 Zeichen sind gültig """
    assert check_password('Sommer4!') == []


def test_without_digit():
    """ Runde 3: ohne Ziffer """
    assert check_password('Sommer!!') == ['mindestens eine Ziffer']


def test_without_uppercase():
    """ Runde 4: ohne Grossbuchstaben """
    assert check_password('sommer24!') == ['mindestens ein
Grossbuchstabe']
```

```
def test_without_special_character():
    """ Runde 5: ohne Sonderzeichen """
    assert check_password('Sommer245') == ['mindestens ein
Sonderzeichen']


def test_several_errors():
    """ Runde 6: alle verletzten Regeln werden gemeldet """
    assert check_password('abc') == [
        'mindestens 8 Zeichen',
        'mindestens eine Ziffer',
        'mindestens ein Grossbuchstabe',
        'mindestens ein Sonderzeichen',
    ]


def test_empty_password():
    """ Spezialfall: leeres Passwort """
    assert len(check_password('')) == 4


@pytest.mark.parametrize('password', [None, 12345678])
def test_not_a_string(password):
    """ Runde 7: kein String """
    with pytest.raises(TypeError):
        check_password(password)
```

Ergebnis

```
$ pytest -q
..........
10 passed

$ pylint --max-line-length=120 password.py test_password.py
Your code has been rated at 10.00/10
```

Zusatz: Regel 6

```
def test_with_space():
    """ Regel 6: keine Leerzeichen """
    assert check_password('Sommer 24!') == ['keine Leerzeichen']
```

In `password.py` genügt eine zusätzliche Zeile in der Regelliste:

```
        (' ' not in password, 'keine Leerzeichen'),
```

M450-LU11, M450-C2I, M450-C1I

Marcel Suter

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

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu11/loesungen/kata

Last update: **2026/11/23 09:25**