LU03.L03 - Rotation-Chiffre

Lösung zu LU03.A03 - Rotation-Chiffre. Der Code stammt aus dem Branch `solution` des Repositorys m450-lu03-a03-rotation.

main_test.py

main_test.py

```
""" Unit Tests für die Rotation-Chiffre """
import pytest

from main import prepare_text, split_blocks, rotate_right, encrypt,
decrypt


# ------------------------------------------------------------------------
------
# prepare_text
# ------------------------------------------------------------------------
------

def test_prepare_text_valid():
    """ Leerzeichen werden entfernt, Kleinbuchstaben werden gross """
    assert prepare_text('kurzer Text') == 'KURZERTEXT'


def test_prepare_text_no_string():
    """ Formal falsch: eine Zahl statt eines Texts """
    with pytest.raises(TypeError):
        prepare_text(42)


def test_prepare_text_empty():
    """ Logisch ungültig: nur Leerzeichen """
    with pytest.raises(ValueError):
        prepare_text('   ')


def test_prepare_text_invalid_chars():
    """ Logisch ungültig: Ziffern und Sonderzeichen """
    with pytest.raises(ValueError):
        prepare_text('Text 42!')


# ------------------------------------------------------------------------
```

```
------
# split_blocks
# ----------------------------------------------------------------------
------

def test_split_blocks_valid():
    """ Der Text geht genau in zwei Blöcke auf """
    assert split_blocks('KURZERTEXT', 5) == ['KURZE', 'RTEXT']


def test_split_blocks_padding():
    """ Der letzte Block wird mit X aufgefüllt """
    assert split_blocks('KURZERTEXT', 4) == ['KURZ', 'ERTE', 'XTXX']


def test_split_blocks_length_no_int():
    """ Formal falsch: die Blocklänge ist ein Text """
    with pytest.raises(TypeError):
        split_blocks('KURZERTEXT', '5')


def test_split_blocks_length_too_small():
    """ Logisch ungültig: Blocklänge 1 """
    with pytest.raises(ValueError):
        split_blocks('KURZERTEXT', 1)


def test_split_blocks_length_too_big():
    """ Logisch ungültig: Blocklänge grösser als der Text """
    with pytest.raises(ValueError):
        split_blocks('KURZERTEXT', 11)


# ----------------------------------------------------------------------
------
# rotate_right
# ----------------------------------------------------------------------
------

def test_rotate_right_valid():
    """ Jede Spalte wird von unten nach oben zu einer Zeile """
    assert rotate_right(['KURZE', 'RTEXT']) == ['RK', 'TU', 'ER', 'XZ',
'TE']


def test_rotate_right_different_lengths():
    """ Logisch ungültig: Blöcke mit unterschiedlicher Länge """
    with pytest.raises(ValueError):
        rotate_right(['KURZE', 'RTE'])
```

```
# ------------------------------------------------------------------------
------
# encrypt und decrypt
# ------------------------------------------------------------------------
------

def test_encrypt_valid():
    """ Beispiel von CrypTool """
    assert encrypt('kurzer Text', 5) == 'RKTUERXZTE'


def test_decrypt_valid():
    """ Umkehrung des Beispiels """
    assert decrypt('RKTUERXZTE', 5) == 'KURZERTEXT'


def test_decrypt_wrong_length():
    """ Logisch ungültig: 10 Zeichen passen nicht in Blöcke der Länge 3
"""
    with pytest.raises(ValueError):
        decrypt('RKTUERXZTE', 3)


def test_encrypt_decrypt():
    """ Verschlüsseln und Entschlüsseln ergibt den bereinigten Text,
aufgefüllt mit X """
    ciphertext = encrypt('Applikationen testen', 4)
    assert decrypt(ciphertext, 4) == 'APPLIKATIONENTESTENX'
```

Erläuterungen

- **Formal falsch** sind Werte mit dem falschen Datentyp, zum Beispiel eine Zahl statt eines Texts oder `'5'` statt 5. Das Programm löst dann einen `TypeError` aus.
- **Logisch ungültig** sind Werte mit dem richtigen Datentyp, die trotzdem nicht erlaubt sind: ein leerer Text, Ziffern im Text, eine Blocklänge von 1. Das Programm löst dann einen `ValueError` aus.
- **test_split_blocks_padding** prüft den Spezialfall, dass der Text nicht genau in die Blöcke passt. Gerade solche Fälle werden beim Testen oft vergessen.
- **test_encrypt_decrypt** prüft, dass Verschlüsseln und Entschlüsseln zusammenpassen. Weil der letzte Block aufgefüllt wird, steht am Ende ein zusätzliches X.
- **Tipp:** Die Tests für die Fehlersituationen lassen sich mit `@pytest.mark.parametrize` zusammenfassen (siehe LU03f).

M450-LU03

Marcel Suter

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

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu03/loesungen/rotation?rev=1794903340

Last update: **2026/11/17 09:15**