

LU09.A02 - Inhalte in Funktionen auslagern



Lagern Sie Funktionalität in Funktionen aus um den Code besser lesbar zu machen.

Akzeptieren Sie das Github-Classroom-Assignment

Akzeptieren Sie das Github-Assignment und klonen Sie das Projekt in Ihre IDE.

Aufgaben

0. Programm ausführen

Führen Sie das Programm `main.py` aus und machen Sie sich mit den Funktionen vertraut.

Führen Sie auch die Testfälle `main_test.py` aus. Diese müssen alle auf PASSED stehen.

```
===== test session starts
=====
collecting ... collected 4 items

main_test.py::test_menu PASSED [
25%]
main_test.py::test_fibonacci PASSED [
50%]
main_test.py::test_einmaleins PASSED [
75%]
main_test.py::test_even_odd PASSED
[100%]

===== 4 passed in 0.01s
=====
```

1. Menüauswahl in Funktionen auslagern

Wir haben uns vorgenommen, dass wir Duplikate im Code vermeiden möchten. Im Programm `talk_to_user` listen wir dem User an zwei Stellen im Code seine Auswahlmöglichkeiten auf:

```
# Show Menu
print() # Newline
print('=====')
```

```
print('Was möchten Sie mit dieser Applikation machen?')
print('Wählen Sie \'f\' für Fibonacci-Reihe ausgeben')
print('Wählen Sie \'e\' für Kleines Einmaleins')
print('Wählen Sie \'g\' für Berechnung Gerade / Ungerade')
print('Wählen Sie \'x\' um das Programm zu beenden!')
```

Wir möchten diesen Programmteil nun in eine externe Methode packen. Wir arbeiten uns also durch die 3 Schritte durch:

1. Funktionsblock definieren
2. docstring erstellen
3. Logik implementieren

Ihre Änderung darf das Resultat der Tests nicht verändern, es müssen immer noch alle Tests auf PASSED stehen.

```
===== test session starts
=====
collecting... collected 4 items

main_test.py::test_menu PASSED [
25%]
main_test.py::test_fibonacci PASSED [
50%]
main_test.py::test_einmaleins PASSED [
75%]
main_test.py::test_even_odd PASSED [
100%]

===== 4 passed in 0.01s
=====
```



Setzen Sie dies im Code nun um. Erstellen Sie einen **Commit**, wenn Sie die Lösung implementiert haben.

Eine mögliche Lösung finden Sie hier: [Menüauswahl in Funktion auslagern](#)

2. Unterprogramme auslagern

Um die Übersichtlichkeit des Codes zu erhöhen, lagern wir die **Unterprogramme (Fibonacci, Einmaleins und Gerade/Ungerade) in eigene Funktionen** aus.

Um diese Unterprogramme nun in eine externe Methoden zu packen, arbeiten uns für jede Methode durch die 3 Schritte durch:

1. Funktionsblock definieren
2. docstring erstellen
3. Logik implementieren

Ihre Änderung darf das Resultat der Tests nicht verändern, es müssen immer noch alle Tests auf PASSED stehen.

```
===== test session starts
=====
collecting ... collected 4 items

main_test.py::test_menuue PASSED [
25%]
main_test.py::test_fibonacci PASSED [
50%]
main_test.py::test_einmaleins PASSED [
75%]
main_test.py::test_even_odd PASSED
[100%]

===== 4 passed in 0.01s
=====
```



Setzen Sie dies im Code nun um. Erstellen Sie einen **Commit**, wenn Sie die Lösung implementiert haben.

Eine mögliche Lösung finden Sie hier: [Unterprogramme auslagern](#)

⇒ *GitHub Repo für externe Besucher*

GitHub Repository <https://github.com/templates-python/m319-lu09-a02-define-functions>

Lernende am BZZ müssen den Link zum GitHub Classroom Assignment verwenden

[M319-LU09](#)



© Marcel Suter, Kevin Maurizi

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

Permanent link:
<https://wiki.bzz.ch/de/modul/m319/learningunits/lu09/aufgaben/funktionen>

Last update: **2025/06/23 07:45**



