

LU09.L02 - Inhalte in Methoden auslagern

1. Menüauswahl in Methode 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 Menue
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 unsere 3 Schritte durch:

1. Funktionsblock definieren

- **Name:** Die Methode druckt die Menüauswahl auf die Konsole aus, mögliche Namen wären z.B. `print_menuue`, `print_dialog`, etc.
- **Parameter:** Der Methode müssen keine Informationen übergeben werden. Die Parameterliste ist also Leer ()
- **Return:** Die Methode gibt keinen Wert zurück, ist also `None`

```
def print_menuue():
```

2. docstring erstellen

Ein kurzer beschreibender Satz.

```
def print_menuue():
    """
    Prints the menu to the user
    :return: None
    """
```

3. Logik Implementieren

`main.py`

```
def print_menuue():
```

```
"""
Prints the menu to the user
:return: None
"""

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!')

def talk_to_user():
    """
    Talks to the user to determine what he wants.

    :return: None
    """

    print('Lieber Benutzer, herzlich willkommen zu diesem Programm')

    # Show Menue
    print_menu()
    selection = input('>> ')

    while selection != 'x':
        if selection == 'f': # Fibonacci
            print('Which position of the Fibonacci-List do you want to know?')

            position = int(input(''))
            number1 = 0
            number2 = 1
            counter = 3
            while counter <= position:
                next_number = number1 + number2
                number1 = number2
                number2 = next_number
                counter += 1

            print(number2)
        elif selection == 'e': # Kleines Einmaleins
            print('Which number do you want to learn?')
            number = int(input(''))
            for counter in range(0, 11):
                print(number * counter)
        elif selection == 'g': # Odd or Even
            print('Enter a number to check for odd/even?')
            number = int(input('Give a number:\n'))
            if number % 2 == 0:
```

```

        print(f'Number {number} is even.')
    else:
        print(f'Number {number} is odd.')
    else: # Wrong input
        print('Befehl nicht verstanden, bitte erneut eingeben.')

    print_menue()
    selection = input('>> ')

if __name__ == '__main__':
    talk_to_user()

```

2. Unterprogramme auslagern

Um die Übersichtlichkeit des Codes zu erhöhen, lagern wir die Unterprogramme () in eigene Methoden aus.

1. Funktionsblock definieren

	Name	Parameter	Funktion
Fibonacci	fibonacci	()	def fibonacci():
Einmaleins	multiplication_table	()	def multiplication_table():
Gerade/Ungerade	odd_or_even	()	def odd_or_even():

2. docstring erstellen

	docstring
Fibonacci	Prints the value from the Fibonacci series at position x.
Einmaleins	Prints the multiplication table for the number x.
Gerade/Ungerade	Prints if the value is even or odd

3. Logik implementieren

main.py

```

def print_menue():
    """
    Prints the menu to the user
    :return: None
    """

    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!')
```



```
def fibonacci():
    """
    Prints the value from the Fibonacci series at position x.
    :return: None
    """
    print('Which position of the Fibonacci-List do you want to know?')
    position = int(input(''))
    number1 = 0
    number2 = 1
    counter = 3
    while counter <= position:
        next_number = number1 + number2
        number1 = number2
        number2 = next_number
        counter += 1

    print(number2)
```



```
def multiplication_table():
    """
    Prints the multiplication table for the number x.
    :return:
    """
    print('Which number do you want to learn?')
    number = int(input(''))
    for counter in range(0, 11):
        print(number * counter)
```



```
def even_or_odd():
    """
    Prints if the value is even or odd
    :return:
    """
    print('Enter a number to check for odd/even?')
    number = int(input('Give a number:\n'))
    if number % 2 == 0:
        print(f'Number {number} is even.')
    else:
        print(f'Number {number} is odd.')
```



```
def talk_to_user():
    """
```

```
Talks to the user to determine what he wants.

:return: None
"""

print('Lieber Benutzer, herzlich willkommen zu diesem Programm')

# Show Menue
print_menu()
selection = input('>> ')

while selection != 'x':
    if selection == 'f': # Fibonacci
        fibonacci()
    elif selection == 'e': # Kleines Einmaleins
        multiplication_table()
    elif selection == 'g': # Odd or Even
        even_or_odd()
    else: # Wrong input
        print('Befehl nicht verstanden, bitte erneut eingeben.')

    print_menu()
    selection = input('>> ')

if __name__ == '__main__':
    talk_to_user()
```



© Kevin Maurizi

From:

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

Permanent link:

<https://wiki.bzz.ch/de/modul/m319/learningunits/lu09/loesungen/funktionen>Last update: **2025/06/23 07:45**