LU05c - Eigene Exceptions auslösen

Wie erwähnt, sind es oft Benutzereingaben, die zu einem Fehlverhalten bei einer Software führen können - sofern man diese Eingaben eben nicht im Voraus prüft. Dazu folgendes Beispiel.

Beispiel: Personendaten erfassen

In eine Eingabefeld kann die Körpergrösse eines Menschen eingegeben werden.

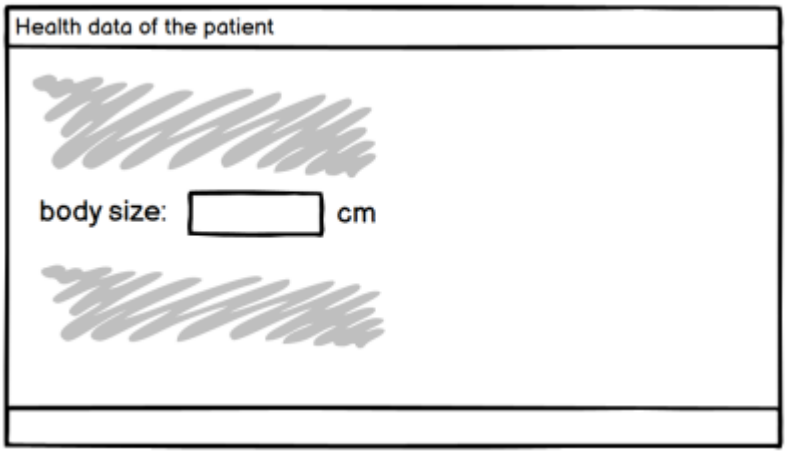

Abb: Eingabe mit begrenztem Wertebereich

Es ist klar ersichtlich, dass für die Eingabe der Körpergrösse negative Werte aber auch Werte über ca. 250cm keinen Sinn machen. Diese Situation kann übewacht und mit einem eigenen Fehlertyp (z.B. BodySizeError) signalisiert werden.

Warum wirft man eigene Exception?

Ein Fehler stellt immer eine Ausnahme im Ablauf des Programms dar. Natürlich könnte man z.B. den Wert -1 (bei Zahlen) oder ein Leerzeichen als Fehlercode zurückgeben. Aber was, wenn eben auch negative Werte zulässig sind?

Dieses Beispiel haben wir in der bankaccount bei der Klasse `BankAccount` gesehen. Dort darf der Saldo (`balance`) bis zu einem gewissen Wert negativ sein. Aber welcher negative Wert soll dann einen Fehler signalisieren? Es ist also um vieles besser, wenn im Fehlerfall ein entsprechendes Fehlerobjekt erzeugt und geworfen wird. Auf dieses Fehlerobjekt kann dann an anderer Stelle im Code mittels `try-except` reagiert werden.

Beispiel: Körpergrösse überwachen

```
class BodySizeError(Exception):
    """
    Dieser Fehlerfall wird ausgelöst, wenn die vom
    Benutzer eingegebene Körpergrösse nicht im Bereich
    20...250 cm liegt.
```

```
    """
    def __init__(self, size):
        super().__init__(f'Der Wert {size} liegt nicht im Bereich 20...250
cm')


class Person():
    def __init__(self, name):
        self._name = name
        self._body_size = 0

    @property
    def body_size(self):
        return self._body_size

    @body_size.setter
    def body_size(self, size):
        if 50 <= size <= 250:
            self._body_size = size
        else:
            raise BodySizeError(size)

if __name__ == '__main__':
    jonathan = Person('Jonathan')
    try:
        size = int(input('Körpergrösse: '))
        jonathan.body_size = size
    except BodySizeError as body_size_error:
        print(body_size_error)
    # OK hier gehts weiter
    print('...')
```

Kopieren Sie den Code in Ihre Entwicklungsumgebung und setzen Sie einige Breakpoints. Starten Sie den Debugger und probieren ein paar Eingabewerte aus. Schauen Sie, was der jeweilge Effekt ist.

Eigene Exception erstellen

Im Folgenden betrachten wir die Codesequenzen, die wichtig sind und bei denen neue Techniken angewendet werden.

Deklaration

Die Deklaration der eigenen Fehlerklasse als Ableitung der Klasse `Exception`.

```
class BodySizeError(Exception):
    # Vererbung der Eigenschaften der Klasse Exception an die Klasse
BodySizeError
```

Das Thema Vererbung werden Sie erst im Detail kennenlernen. Dennoch brauchen wir diese Technik hier, um eine eigene Exception-Klasse zu erstellen.

Konstruktor der Oberklasse

In der Regel verwendet man den Konstruktor der Oberklasse, um das Objekt vollständig zu initialisieren.

```
def __init__(self, size):
    super().__init__(f'Der Wert {size} liegt nicht im Bereich 20...250 cm')
    # Aufruf des super-Konstruktors mit den entsprechenden Parametern.
```

Anhand der Dokumentation wissen wir, dass die Klasse `Excpetion` eine Fehlermeldung ausgeben kann. Es ist also wichtig, an Hand einer Dokumentation die Funktionalität der Oberklasse in Erfahrung zu bringen.

Auslösen der Exception

```
if 50 <= size <= 250:
    self._body_size = size
else:
    raise BodySizeError(size)
    # hier wird das Fehlerobjekt erzeugt und "geworfen"
```

Mittels einer Bedingung wird die Gültigkeit eines Wertes überprüft. Im Fehlerfall wird durch `raise` die Exception zuerst erzeugt und dann geworfen. Im Code ist ersichtlich, dass hier nicht eine Objektreferenz sondern die Klasse (`BodySizeError`) angeschrieben wird. Das heisst, dass in diesem Moment der Programmausfürhung ein entsprechendes Objekt erzeugt wird. (vrgl dazu die Skizze unten)

Fehler fangen und verarbeiten

```
except BodySizeError as body_size_error:
    # das geworfene Objekt vom Typ BodySizeError benennen
    print(body_size_error)
```

Wir geben bei `except` an, welchen Fehlertyp wir erwarten und wie das Fehlerobjekt heissen soll. Anschliessend codieren wir das gewünschte Verhalten unseres Programms, wenn der Fehlertyp auftreten sollte.

Ablauf im Code

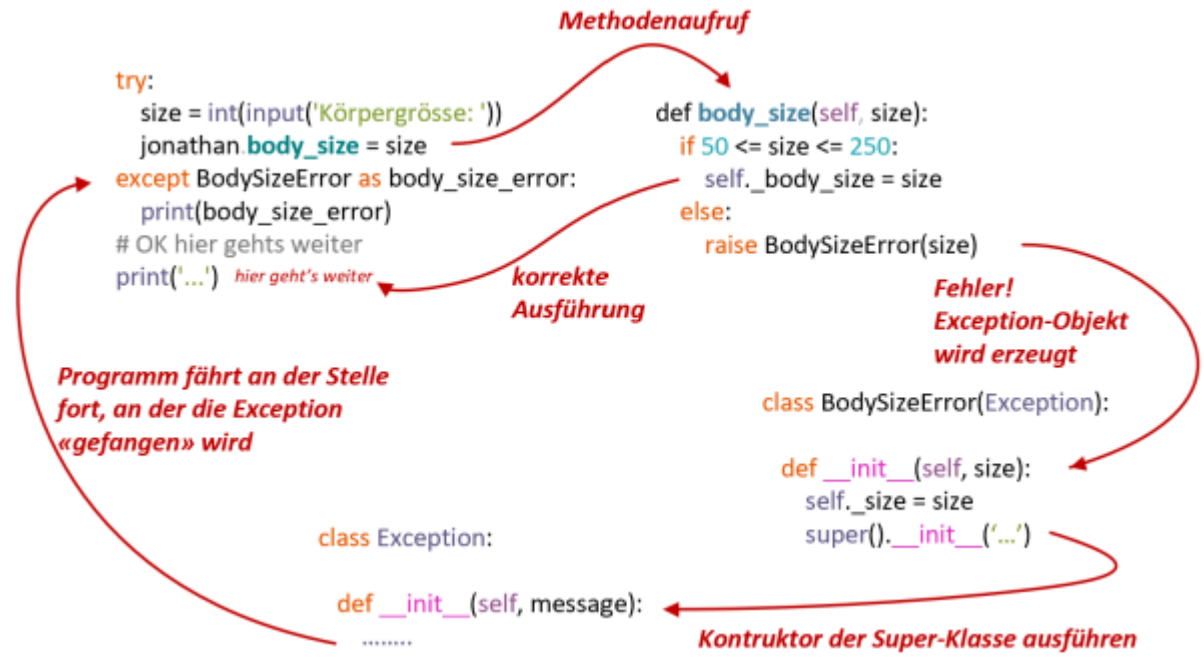

Abb: Ablauf bei einer Exception

M320-LU05