LU09.L03 - Erweiterte Aufgaben

Mehrere Return-Werte

main.py

```
def div(dividend, divisor):
    """
    Performs integer division.
    :param dividend: int
    :param divisor: int
    :return: Tuple (result of division, remainder) or None if divisor
is zero
    """
    if divisor == 0:
        print("Division durch Null ist nicht erlaubt")
        return None
    result = dividend // divisor
    rest = dividend % divisor
    return result, rest

def main():
    result, rest = div(34, 6)
    print(...)


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

2. Längenumrechner

1.Code anpassen

Anpassen der Defaultwerte:

length_calculator.py

```
def convert(length, from_unit=1, to_unit=2):
```

2. Erweiterung

Code Erklärung

In dem Codeabschnitt:

```
units = {"Meter": 1, "Meilen": 2, "Seemeilen": 3, "Yard": 4, "Inches": 5}

from_unit = units.get(from_unit, from_unit)
to_unit = units.get(to_unit, to_unit)
```

wird die `get`-Methode des Python-Wörterbuchs `units` verwendet. Diese Methode wird hier auf eine spezielle Weise eingesetzt, um sowohl mit numerischen als auch mit Texteingaben für die Einheiten umzugehen.

Das Wörterbuch `units`:

```
  Dieses Wörterbuch bildet die Namen der Einheiten (als Text) auf
entsprechende numerische Codes ab. Zum Beispiel: '''"Meter": 1, "Meilen": 2,
...''.
```

Die `get`-Methode:

```
  Die Methode ''get'' wird auf ein Wörterbuch angewendet und hat zwei
Parameter: den Schlüssel, dessen Wert abgerufen werden soll, und einen
Standardwert, der zurückgegeben wird, falls der Schlüssel im Wörterbuch
nicht existiert.
```

Anwendung in `from_unit = units.get(from_unit, from_unit)`:

- `from_unit` vor dem Gleichheitszeichen ist die Variable, die den aktualisierten Wert nach der Ausführung der Zeile speichert.
- `from_unit` im `get`-Aufruf (der erste Parameter) ist der Schlüssel, den wir im Wörterbuch `units` suchen.
- Der zweite `from_unit` im `get`-Aufruf ist der Standardwert, der zurückgegeben wird, falls der Schlüssel nicht im Wörterbuch gefunden wird.
- Falls `from_unit` ursprünglich eine Zahl ist (z.B. 1 für Meter), wird dieser Wert nicht im Wörterbuch gefunden. Daher wird der Standardwert zurückgegeben, der in diesem Fall derselbe Wert ist (1).
- Falls `from_unit` ein Text ist (z.B. „`Meter`“), wird der entsprechende numerische Code aus dem Wörterbuch geholt (in diesem Beispiel 1).

length_calculator.py

```
def convert(length, from_unit=1, to_unit=2):
    """
    Converts lengths. The following lengths can be converted: meters,
miles, nautical miles, yards, inches
    :length: length to convert
    :from_unit: 1=Meter, 2=Meilen, 3=Seemeilen, 4=Yard, 5=Inches or
```

```
equivalent strings
    :to_unit: 1=Meter, 2=Meilen, 3=Seemeilen, 4=Yard, 5=Inches or
equivalent strings
    :return: converted length
    """
    units = {"Meter": 1, "Meilen": 2, "Seemeilen": 3, "Yard": 4,
"Inches": 5}

    from_unit = units.get(from_unit, from_unit)
    to_unit = units.get(to_unit, to_unit)

    if (from_unit == 1):
        result = length
    elif (from_unit == 2):
        result = length * 1609.34
    elif (from_unit == 3):
        result = length * 1852.0
    elif (from_unit == 4):
        result = length * 0.9144
    elif (from_unit == 5):
        result = length * 0.0254

    if (to_unit == 1):
        return result
    if (to_unit == 2):
        result = result / 1609.34
        return result
    if (to_unit == 3):
        result = result / 1852.0
        return result
    if (to_unit == 4):
        result = result / 0.9144
        return result
    if (to_unit == 5):
        result = result / 0.0254
        return result

def main():
    print("1=Meter, 2=Meilen, 3=Seemeilen, 4=Yard, 5=Inches")
    print(convert(1, 1, 4))


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