Funktionen

- Python Funktionen definieren

Einleitung

- Ganz allgemein gesehen stellt eine Funktion ein Strukturierungselement in einer Programmiersprache dar, um eine Menge von Anweisungen zu gruppieren, damit man sie mehrmals im Programm verwenden kann.

- Die Benutzung von Funktionen erhöht maßgeblich die Verständlichkeit und Qualität eines Programmes.

- Funktionen kennt man unter unterschiedlichen Bezeichnungen in verschiedenen Programmiersprachen. So kennt man sie auch als Subroutinen, Routinen, Prozeduren, Methoden und **Unterprogramme**.

Definition einer Funktion

Funktionen werden mit dem Schlüsselwort def eingeleitet

```
def funktionsname(Parameterliste):
    Anweisung(en)
```

Beispiel

```
def summe(a, b):
        c = a + b
```

Bestandteile einer Funktion

Eine Funktion besteht aus zwei Teilen:

a) Kopf der Funktion mit Parameterliste und Namen der Funktion

b) dem Funktionskörper mit Anweisungen und Rückgabewert

```
def funktions_name(Parameterliste):
    Anweisung(en)
    return wert
```

Der Funktionsname

Der Funktionsname sollte so gewählt werden, dass er a) nicht mit einem Variablennamen kollidiert und b) erkennen lässt, was die Aufgabe der Funktion ist.

Er sollte keine Leer- und Sonderzeichen, wenn möglich auch keine Umlaute enthalten und in Snake-Case notiert sein.

Python Programmierer sind angehalten, sich an die Konvention der Schreibweise zu halten, die in PEP-8 angegeben ist!

def GETALPHABET(): Falsch

def getAlphabet(): Falsch

def get_alphabet(): Richtig

Die Parameterliste

Die Parameterliste besteht aus einem oder mehreren Bezeichnern, die durch Kommata getrennt sind.

Mit Hilfe der Parameterliste können Werte in Form von Argumenten an die Funktion übergeben werden.

```
def berechne_rauminhalt(a, b, c):
        rauminhalt = a * b * c


berechne_rauminhalt(20, 10, 3)
```

Funktionskörper

Der Funktionskörper (function body), also die Anweisungen, die ausgeführt werden, wenn die Funktion aufgerufen wird, wird in Python durch eine homogene Einrückung von 4 Leerzeichen markiert.

Also ebenso wie alle anderen Blöcke in Python.

```
def berechne_rauminhalt(a, b, c):
    rauminhalt = a * b * c
    return rauminhalt
```

Rückgabewert einer Funktion

Eine Funktion hat einen Rückgabewert. Dieser wird explizit durch das return-Keyword definiert. Wird kein return angegeben, ist der Rückgabewert per default None.

Der Wert, den die Funktion mit return zurückgibt, kann einer Variablen zugewiesen werden (volume).

```
def berechne_rauminhalt(a, b, c):
        ''' Diese Funktion berechnet den Rauminhalt eines Körpers
        und gibt den Wert via return zurück
        '''
        return a * b * c

volume = berechne_rauminhalt(10, 10, 10)
```

Funktionsaufruf

Eine Funktion wird definiert und dann ein- oder mehrmals aufgerufen.

```
def summe(a, b):
        return a + b

summe1 = summe(34, 22)
summe2 = summe(1, 2)
```

Die Parameter sind optional. Aber: wenn eine Funktion mit Parametern definiert wurde, muss sie in der Regel auch mit diesen Argumenten aufgerufen werden.

Mehrere Rückgabewerte

Eine Funktion kann mehrere Rückgabewerte definieren.

```
def is_odd(value):
        if value % 2:
                return True
        return False

if is_odd(3):
    print('ja, die Zahl ist ungerade')
```

Datentypen von Rückgabewerten

In Python muss im Gegensatz zu vielen anderen Sprachen der Rückgabewert einer Funktion nicht definiert werden.

Jeder Datentyp kann zurückgegeben werden: String, int, float, dict, list, tupel, set, Boolean, None.

Kommentieren von Funktionen

Jede Funktion sollte einen Kommentar haben. Dieser Kommentar wird doc-String genannt und ist über das __doc__ - Attribut auslesbar.

```
def average(a, b, c):
    """Berechne den Durchschnitt von drei Zahlen und runde ihn auf zwei Stellen."""
    return round((a + b + c) / 3, 2)
```

```
print(average.__doc__)
```

```
Berechne den Durchschnitt von drei Zahlen und runde ihn auf zwei Stellen
```

Beim automatisiertem Erstellen von Dokumentationen oder im Hilfesystem auf der Shell kann dieser Docstring ausgelesen werden.

Mathematische Funktionen vs Python Funktionen

Mathematische Funktionen zeichnen sich dadurch aus, dass sie bei gleichem Input immer denselben Output liefern.

Sie sind von der Umgebung isoliert, verändern also die Umgebung nicht.

$$f(x)=m\cdot x+b$$

Funktionen in Python dienen in erster Linie nur als Struktur, um Codeabschnitte zu verwalten.

Sie können durchaus die Umgebung verändern oder von Veränderungen in der Umgebung beeinflusst werden (Seiteneffekte).

Beispiel

```
p = 3

def calculate_something(a):
        return a * p


caluclate_something(3)
```

Die globale Variable p wurde in den Kontext der Funktion übernommen. Dies sollte, wenn möglich, vermieden werden.

Ausnahmen sind zum Beispiel Konstanten, Konfigurationsparameter, oder Methoden in der objektorientierten Programmierung.

Seiteneffekte

Wenn eine Instanz eines veränderbaren Datentypen als Funktionsparameter übergeben wird, zum Beispiel eine Liste, sollte bedacht werden, dass hier sogenannte Seiteneffekte auftreten. Da nicht der Wert an sich, sondern nur eine Referenz übergeben wird, nutzt Python diese Referenz und nicht den eigentlichen Wert

```
def fn(liste):
    liste.extend([1, 2, 3])
```

```
liste = [2,2]
fn(liste)
print(liste)
```

```
[2, 2, 1, 2, 3]
```

Unix Philosophie

Eine Funktion soll eine und genau eine Aufgabe haben. Diese Aufgabe soll sie bestmöglich ausführen.
https://en.wikipedia.org/wiki/Unix_philosophy

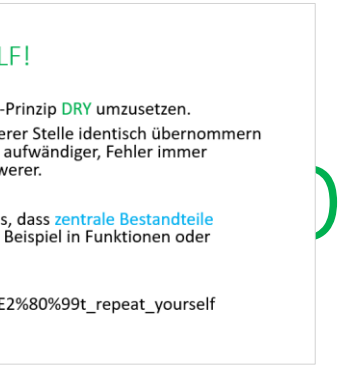
```
def summe(a, b):
        c = a + b
        mult = a * b
        log(f'Ergebnis von Summe: {c}')
        return c, mult
```

⇒ Schlecht (summe soll nur summieren und nicht auch noch multiplizieren und loggen)

```
def summe(a, b):
    return a + b

def multiply(a, b):
    return a * b
```

=> besser



ON‘T REPEAT YOURSELF!

Funktionen helfen, das Programmier-Prinzip DRY umzusetzen.

Code sollte nicht kopiert und an anderer Stelle identisch übernommern werden. Änderungen werden immer aufwändiger, Fehler immer wahrscheinlicher, Testing immer schwerer.

Guter Code zeichnet sich dadurch aus, dass zentrale Bestandteile eigenständig verwaltet werden. Zum Beispiel in Funktionen oder Klassen (Modulen).

https://de.wikipedia.org/wiki/Don%E2%80%99t_repeat_yourself

Tips für gutes Funktionsdesign

Nutze aussagekräftige, überschaubare Funktionsnamen in Snake-Case

Halte die Parameteranzahl gering

Vermeide Seiteneffekte

Schreibe die Funktionen robust und leicht lesbar

Schreibe für jede nicht selbsterklärende Funktion einen Doc-String

Eine Funktion soll nur eine Aufgabe haben (Single Responsibility Principle).

Falls eine Funktion mehrere Aufgaben erfüllt, splitte die Aufgaben in einzelne Funktionen

Halte die Anzahl der Zeilen in einer Funktion gering

Die Rückgabewerte einer Funktion sollten möglichst vom gleichen Typ sein

Bevorzuge Keyword-Argumente

Teste Deine Funktionen

Organisiere Funktionen in Modulen

Nutze Type-Hints

Mache Dir Gedanken um die Zeit- und Raum-Komplexität der Funktion