TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Arbeitsblatt 2

Rekursion (endlos)

Rekursive Funktionen (d.h. Funktionen, die sich selbst aufrufen) benötigen eine Abbruchbedingung, um sich nicht in einer endlosen Aufrufkette zu verlieren. Bedingte Anweisungen sind das Mittel der Wahl, die Abbruchbedingung zu formulieren. Machen Sie es zunächst einmal bewusst falsch und programmieren sie eine Funktion endlos, die sich ohne weitere Abbruchbedingung selbst aufruft. Was passiert, wenn diese Funktion aufgerufen wird?

Rekursion (abgebrochen)

Um entscheiden zu können, ob die Rekursion abgebrochen werden muss oder nicht (d.h. für die Formulierung der Abbruchbedingung), müssen sich die Aufrufe in irgendeiner Form unterscheiden. Der beste Weg dazu ist die Verwendung von Parametern, mit deren Hilfe entschieden werden kann, ob abgebrochen werden soll.

Programmieren Sie eine Funktion `summe`, die einen ganzzahligen Parameter `x` erwartet. Das Ergebnis soll die Summe aller ganzen Zahlen von 1 bis `x` sein:

```
summe(x) := 0                    falls x == 0
summe(x) := summe(x-1)+ x        sonst
```

Wie kann mit Python eine rekursive Implementierung realisiert werden?

Was passiert, wenn die Funktion `summe` mit einem unzulässigen Parameter aufgerufen wird (z.B. mit einer Gleitkommazahl)? Wie sollte die Funktion reagieren?

Exceptions

Eine gute Reaktionsmöglichkeit auf fehlerhafte Parameter ist das Generieren eines Laufzeitfehlers, man spricht auch vom "Werfen einer Exception". Der Python-Befehl dazu heißt `raise`, z.B.:

```
raise TypeError("'int' expected")
```

Verbessern Sie Ihre Implementierung von `summe`, indem Sie zunächst prüfen, ob die Klasse des Parameters `int` ist und ggf. obige Ausnahme werfen.

Schleifen

Implementieren Sie die Funktion `summe` neu mit Hilfe

- einer `while`-Schleife
- einer `for`-Schleife