Funktionale Programmierung

In Python

Definition

In der imperativen Programmierung besteht ein Programm aus einer Folge von Anweisungen, die in der Regel Variablen verändern – das ist die bekannte Methode.

Bei der funktionalen Programmierung bestehen Programme ausschließlich aus Funktionen. Funktionale Sprachen sind häufig deklarativ aufgebaut.

Deklarativ vs imperativ

Deklarative Sprachen: In deklarativen Sprachen beschreiben Sie, WAS getan werden soll, ohne explizit anzugeben, WIE es getan werden soll (SQL, Prolog, Haskell).

In imperativen Sprachen beschreiben Sie, WIE eine Aufgabe Schritt für Schritt ausgeführt werden soll. Sie geben genaue Anweisungen an den Computer, wie er die Aufgabe ausführen soll (C, C++, Java, Python..).

Rein funktionale Sprachen

Lambda-Kalkül (1930)

Lisp Familie(1958)

Erlang (1987)

Haskell (1990)

Sprachen, die funktionale Elemente implementieren sind zum Beispiel:

Python (1991)

JavaScript/ECMA-Script (1995)

Scala (2001)

Julia (2012)

Swift (2014)

Python setzt das Konzept der funktionalen Programmierung u.a. mit Hilfe von Lambda-Funktionen und Funktionen höherer Ordnung um.

Lambda-Funktionen (Anonyme Funktionen)

Mit Hilfe des lambda-Operators können anonyme Funktionen, d.h. Funktionen ohne Namen erzeugt werden.

Sie haben eine beliebe Anzahl von Parametern, führen einen Ausdruck aus und liefern den Wert dieses Ausdrucks als Rückgabewert.

Allgemein

lambda[<arg>...] :<ausdruck>

```
f = lambda x, y : x + y
f(1, 1)
```

Gebundene Variablen (bound variables)

Gebundene Variablen sind Variablen, die der lambda-Funktion als Argumente übergeben werden.

Im Gegensatz zu freien Variablen, die dem Funktionskörper von außerhalb übergeben werden können

lambda x: x + y

Nach dem Lambda-Kalkül und dem Paradigma der funktionalen Programmierung, sind freie Variablen in Funktionen verboten oder zumindest verpönt.

Anonyme Funktionen

Für Lambda-Funktionen gibt es diverse Begriffe aus anderen Programmiersprachen. Vom Konzept sind sie alle ähnlich:

Anonyme Funktionen

Lambda Funktionen

Lambda Ausdruck, Funktionsausdruck

Lambda Abstraktionen

Lambda Form

Funktionsliteral

Lambda Funktionen sofort aufrufen

Lambda Funktionen lassen sich an Ort und Stelle aufrufen, und zwar mit der Immediately Invoked Function Expression:

```
(lambda x, y: x + y)(2, 3)
```

Wofür werden Lambda Funktionen genutzt?

Sie werden häufig für Sortier-Aufgaben eingesetzt:

```
sorted(liste, key=lambda x: x%2==0)
```

Keine Seiteneffenkte

Funktionen sollten nach dem Paradigma der funktionalen Programmierung keine Seiteneffekte beinhalten. Sie sind somit wesentlich besser testbar. Ein Beispiel für das Updaten einer Liste ohne Seiteneffekt.

```
# gut, da expliziter Return der neuen, veränderten Liste.
def fn(seq: list[int], val: int) -> list[int]:
    return seq + val

seq = [2, 4]
seq = fn(seq, 5)

# schlechter, da Seiteneffekt:
def fn(val: int) -> None:
     seq.append(val)
seq = [2, 4]
fn(5)
```