Unser Pitch zur funktionalen Programmierung

mittels F#

Christian Parpart & Kei Thoma

Humboldt Universität zu Berlin

27. Juni 2019

$$i = i + 1$$

Wir werden nicht:

1. eine neue Programmiersprache lernen
2. funktionale Programmieren lernen

Unser Pitch zur funktionalen Programmierung

mittels F#

Christian Parpart & Kei Thoma

Humboldt Universität zu Berlin

27. Juni 2019

Inhalt

Immutability

Python

```
a = 0

i = 1
i = i + 1
```

F#

```
let a = 0

let mutable i = 1
i <- i + 1
```

Notation

Mathe

$$a = 1$$
$$f : x \mapsto x + a$$

$$g(f(x))$$
$$(g \circ f)(x)$$

F#

```
let a      = 1
let f (x) = x + a

g(f(x))

(g << f)(x)
```

Seiteneffektfreiheit

Code

```
let f(x) =
    let a = 2
    let b = 3
    a * b + x
```

Eine Funktion, die nur von ihren Argumenten abhängt, ist seiteneffektfrei!

Wir gewinnen die folgenden Garantien:

1. Vorherbestimmtheit (Determinismus)
2. Vereinfachte Code Verständlichkeit
3. Vereinfachte Refaktorisierung

Wiederverwendbarkeit

- folgt aus der Seiteneffektfreiheit
- nicht aller Code ist 25 Zeilen lang
  - rechts im Bild: 1023 Zeilen
  - links im Bild: 15894 Zeilen: Seite zu klein ;-(

Verifizierbarkeit

- Theorem Proving und Mutable Variablen (Z3, CVC4)
- FP Programme sind leichter zu verifizieren

Fazit

- Probleme von Funktionaler Programmierung
  - Schwierig sich rein zu denken
  - Langsamer als C/C++
    (aber immer noch schneller als Python)
  - wenn es einmal läuft, wie ein Stein
- Fun Fact: F# steht für ___

Fazit

- Probleme von Funktionaler Programmierung
  - Schwierig sich rein zu denken
  - Langsamer als C/C++
    (aber immer noch schneller als Python)
  - wenn es einmal läuft, wie ein Stein
- Fun Fact: F# steht für FUN

Bonusmaterial