Stack-basierte High-Level Programmiersprachen

Ulrich Hoffmann

Überblick

- Was ist Stack-basiert?
- Was ist High-Level?
- einige Beispiele in ausgewählten Sprachen

Was ist Stack-basiert?

- Expliziter Stack in der Sprache verfügbar
- Kommunikation zwischen Worten/Unterprogrammen/Funktionen erfolgt über den Stack nicht durch Parameter oder Register

Was ist high-level?

- Die von der Programmiersprache behandelten Datenstrukturen sind nicht-trivial
- Mehr als Zahlen, Adressen, Zeichen
- Sondern auch Strings, Hash-Maps, Files, große Zahlen, ...

Beispiele?

- Forth
  (ungetypt: Worte kennen die Typen)
  + D+ UM*
- JOY, factor,
  - cat, enchilada, ...
- Postscript, BibTeX, LIFO, ...

Forth: forth.py

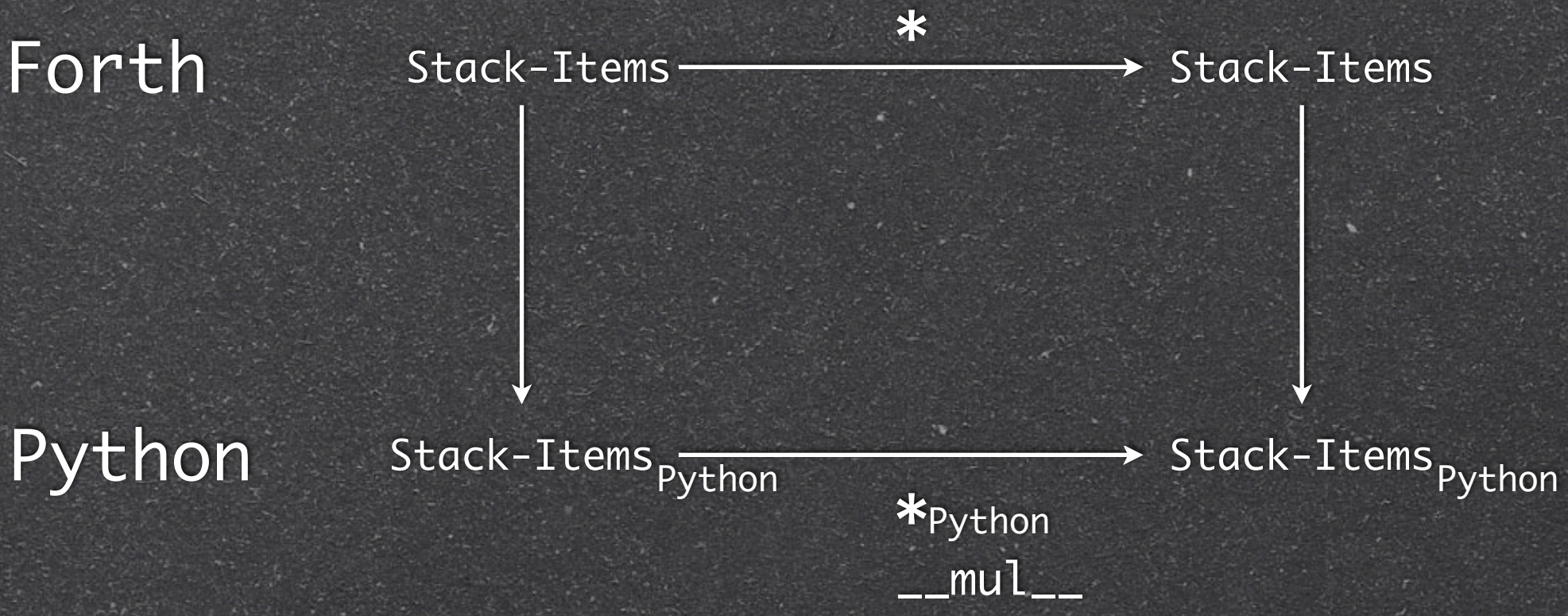
- ANS-Forth in Python implementiert
- portabel
- für Implementierungs-Experimente
- Auf was kann man verzichten?
- case-insensitiv, case-preserving
- forward

Forth:forth.py

Demonstration

Forth: forth.py

- Imlementierung der Zahlen-Arithmetik



Fakultätsfunktion

```
: fac ( n1 -- n2 )
   dup 0= IF drop 1
          ELSE dup 1- recurse * THEN ;

: fac ( n -- n )
   1 swap
   BEGIN ?dup
   WHILE swap over *  swap 1- REPEAT ;
```

Forth:forth.py

- Demonstration
- 1000 fac .

JOY

- Entwickelt von Manfred von Thun/Australien
- Rein funktionale Sprache
- Vorstellung: Worte sind Funktionen, die einen Stack als Parameter nehmen und einen neuen Stack liefern.
- Quotations: anonyme inline Worte
- Funktionen höherer Ordnung

JOY

```
fac == [ dup 0= ]
       [ pop 1 ]
       [ dup 1 - fac * ] ifte ;

fac ==  [ 1 ]  [ * ] primrec ;
```

Demonstration

factor

- entwickelt von Slava Pestov / Kannada
- www.factorcode.org
- inspiriert von JOY
- Aber Variablen und veränderbare Datenstrukturen (mehr wie Lisp)
- Portable Entwicklungsumgebung

factor

Demonstration

Fragen und Diskussion