Informatik I: Einführung in die Programmierung

2. Erste Schritte in Python

Albert-Ludwigs-Universität Freiburg

Peter Thiemann

23. Oktober 2018

1 Allgemeines

Programmiersprachen

Ada, Basic, C, C++, C♯, Cobol, Curry, F♯, Fortran, Go, Gödel, HAL, Haskell, Java, Lisp, Lua, Mercury, Miranda, ML, OCaml, Pascal, Perl, Python, Prolog, Ruby, Scheme, Shakespeare, Smalltalk, Visual Basic, u.v.m.

Wir lernen hier Python (genauer Python 3), eine

- objektorientierte,
- dynamisch getypte,
- interpretierte und interaktive
- höhere Programmiersprache.

Die Programmiersprache Python …

- Anfang der 90er Jahre als Skriptsprache für das verteilte Betriebssystem Amoeba entwickelt;

Guido van Rossum (Foto: Wikipedia)

- gilt als einfach zu erlernen;
- wurde kontinuierlich von Guido van Rossum bei Google weiterentwickelt.
- bezieht sich auf die Komikertruppe *Monty Python*.

Literatur

Es gibt eine Menge von Lehrbüchern zu Python3. Wir werden im wesentlichen einsetzen

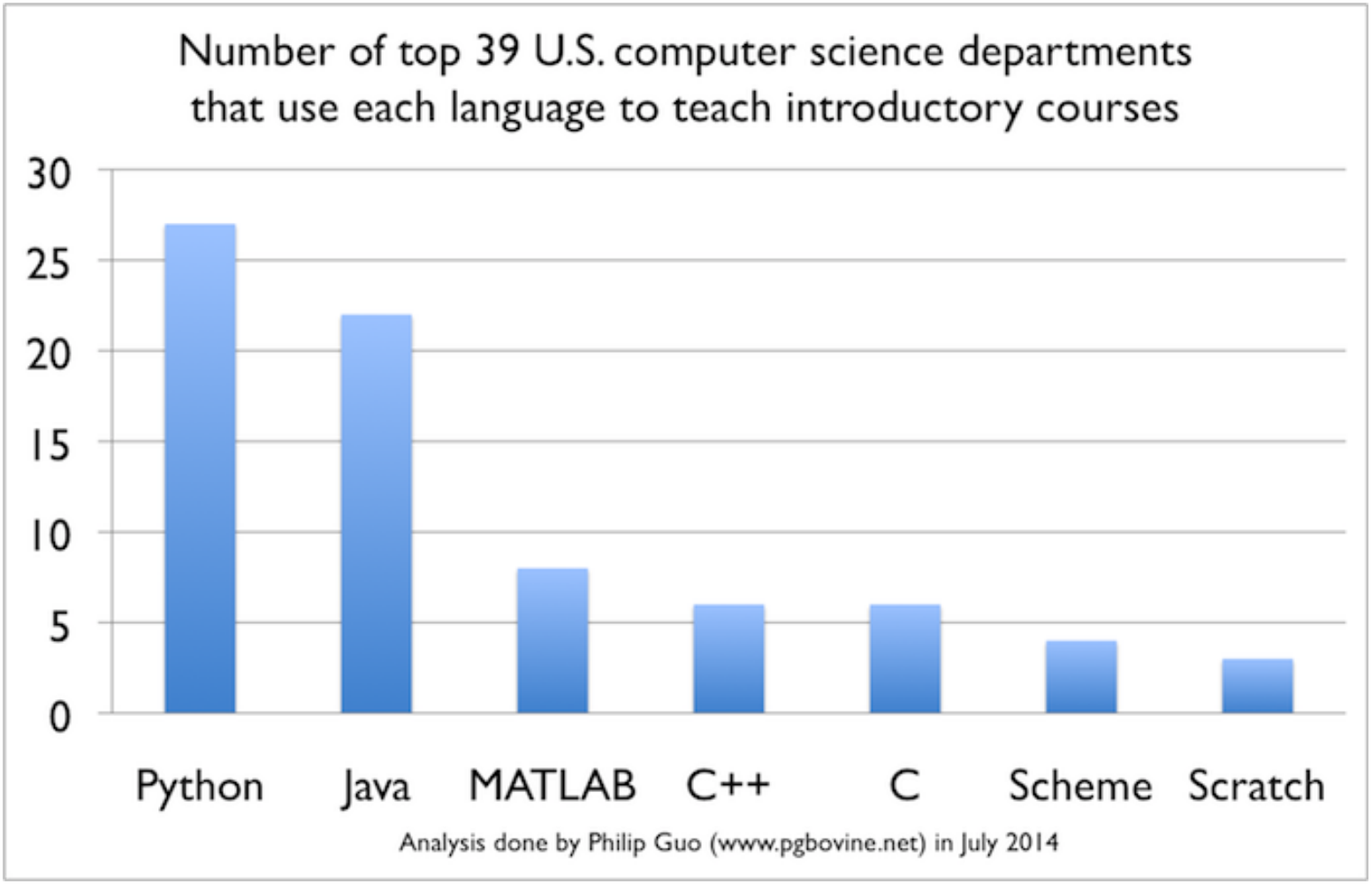
- Allen Downey, *Think Python: How to Think Like a Computer Scientist*, O'Reilly, 2013
- als PDF herunterladbar oder als HTML lesbar (Green Tea Press): `http://greenteapress.com/thinkpython/thinkpython.html`
- als deutsche Version: Programmieren lernen mit Python, O'Reilly, 2013.
- Marc Lutz, *Learning Python*, O'Reilly, 2013 (deutsche Ausgabe ist veraltet!)
- Marc Lutz, *Python kurz & gut*, O'Reilly, 2014 (als Nachschlagwerk)
- Viele Videos und Online-Kurse

2 Warum Python?

Warum Python benutzen?

- Softwarequalität
  - Lesbarkeit
  - Software-Reuse-Mechanismen (wie OOP)
- Programmierer-Produktivität
  - Python-Programme sind oft 50% kürzer als vergleichbare Java oder C++-Programme.
  - Kein Edit-Compile-Test-Zyklus, sondern direkte Tests
- Portabilität
- Support-Bibliotheken („Batterien sind enthalten“)
- Komponenten-Integrierbarkeit (Java, .Net, COM, Silverlight, SOAP, CORBA, . . .)

Einsteigersprachen in den USA



Python ist #1

http://www.youtube.com/watch?v=f3EbDbm8XqY&t=7m3s

Wer benutzt Python?

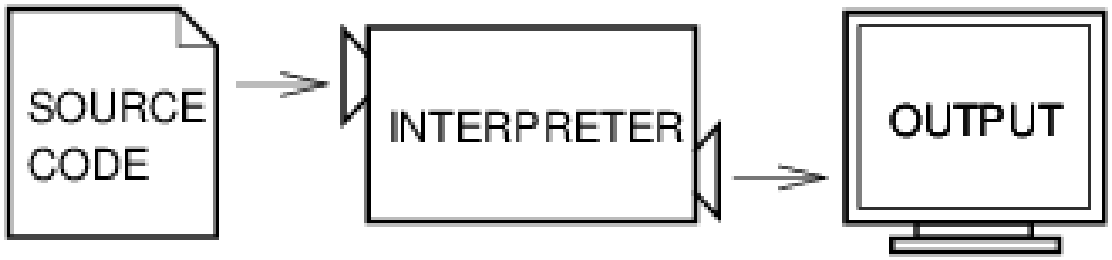
- Google: Web search, App engine, YouTube
- Dropbox
- CCP Games: EVE Online
- 2kgames: Civilization IV (SDK)
- Industrial Light & Magic: Workflow-Automatisierung
- ESRI: Für Nutzerprogrammierung des GIS
- Intel, Cisco, HP, Seagate: Hardwaretesting
- NASA, JPL, Alamos: Scientific Computing
- `...http://www.python.org/about/success/`

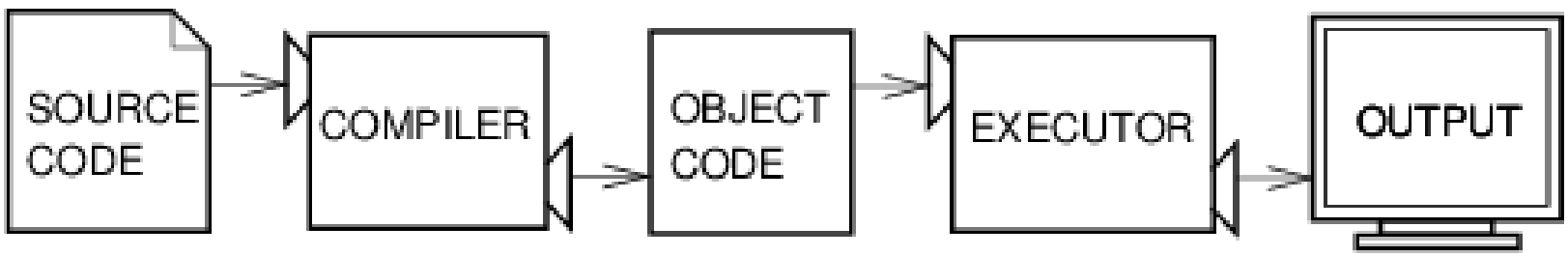
Was geht nicht?

- Python ist „langsamer“ als Java und C++
- Wieviel langsamer? http://benchmarksgame.alioth.debian.org/
- Eignet sich nicht für das Schreiben von Gerätetreibern
- Eignet sich nicht direkt für die Programmierung von (kleinen) Mikrocontrollern (*bare metal programming*)

3 Python-Interpreter

Interpreter- versus Compiler-Sprachen





Abbildungen aus Downey 2013

Woher nehmen?

Unter `http://python.org/` befinden sich die aktuelle Dokumentation und Links zum Herunterladen (uns interessiert Python 3.X, $X \geq 6$) für

- *Windows*,
- *MacOSX*,
- *Unixes* (Quellpakete),
- für aktuelle *Linux-Distributionen* gibt es Packages für die jeweilige Distribution, meistens bereits installiert!

Läuft u.a. auch auf dem Raspberry Pi!

Interaktiver und Skript-Modus

Der Python-Interpreter kann auf folgende Arten gestartet werden:

- im interaktiven Modus (ohne Angabe von Programm-Parametern)

→ Ausdrücke und Anweisungen können interaktiv eintippt werden, der Interpreter wertet diese aus und druckt ggf. das Ergebnis.

- im Skript-Modus (unter Angabe einer Skript-/Programm-Datei)

→ Ein Programm (auch Skript genannt) wird eingelesen und dann ausgeführt.

4 Interaktives Nutzen der Shell

Erste Schritte: Ausdrücke

Nach Starten des Interpreters erscheint das Prompt-Zeichen. Das Eintippen von Ausdrücken wertet diese aus und liefert ein Ergebnis.

Um dem Interpreter eine Ausgabe zu entlocken, gibt es zwei Methoden. Zum einen wertet der Interpreter jeden eingegebenen Ausdruck aus und gibt das Ergebnis aus:

Python-Interpreter

```
>>> 7 * 6
42
>>> "Hello world"
'Hello world'
>>> "spam " * 4
'spam spam spam spam '
```

Erste Schritte: „Drucken“

Zum anderen kann die `print`-Funktion den Wert eines Ausdrucks ausgeben:

Python-Interpreter

```
>>> print(7 * 6)
42
>>> print("Hello world")
Hello world
>>> print("spam " * 4)
spam spam spam spam
```

`print` ist der übliche Weg, Ausgaben zu erzeugen und funktioniert daher auch in „richtigen“ Programmen.

Exkurs: Hello-World-Programme

Hello-World-Programme dienen dazu, eine erste Idee vom Stil einer Programmiersprache zu bekommen.

Python

```
print("Hello World!")
```

Java

```
class HelloWorld {
  public static void main(String[] arg) {
    System.out.println("Hello World!");
  }
}
```

Brainfuck

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]
>++.>+.+++++++..+++.>++.<<+++++++++++++++.
>.+++.------.--------.>+.>.
```

Ausgaben des Interpreters

Es besteht ein kleiner aber feiner Unterschied zwischen „nackten“ Ausdrücken und Ergebnissen der `print`-Funktion:

Python-Interpreter

```
>>> print(7 * 6)
42
>>> print("Hello world")
Hello world
>>> print("oben\nunten")
oben
unten
>>> print(None)
None
```

Python-Interpreter

```
>>> 7 * 6
42
>>> "Hello world"
'Hello world'
>>> "oben\nunten"
'oben\nunten'

>>> None
>>>
```

Mehr dazu später . . .

Etwas mehr zu `print`

Wir werden die Möglichkeiten von `print` später noch ausführlicher behandeln. Ein Detail soll aber schon jetzt erwähnt werden:

Python-Interpreter

```
>>> print("2 + 2 =", 2 + 2, "(vier)")
2 + 2 = 4 (vier)
```

- `print` kann mehrere Ausdrücke durch Kommas getrennt verarbeiten.
- Die Ergebnisse werden in derselben Zeile durch Leerzeichen getrennt ausgegeben.

Die Hilfe-Funktion

Wenn Sie etwas zu einem Befehl oder einer Funktion in Python wissen möchten, dann nutzen Sie die help-Funktion:

Python-Interpreter

```
>>> help
Type help() for interactive help, or help(object) for
help about object.
>>> help(print)

Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', ...
```

5 Rechnen

Zahlen

Python kennt drei verschiedene Datentypen für Zahlen:

- `int` für ganze Zahlen;
- `float` für Gleitkommazahlen
  (eine verrückte Teilmenge der rationalen Zahlen);
- `complex` für komplexe Gleitkommazahlen.

int

Schreibweise für Konstanten vom Typ `int`:

Python-Interpreter

```
>>> 10
10
>>> -20
-20
```

Syntax

Die Schreibweise von Konstanten ist ein Aspekt der **Syntax** einer Programmiersprache. Sie beschreibt, welche Zeichen erlaubt sind, welche Worte vordefiniert sind und wie Sätze (Programme) in der Programmiersprache aussehen müssen.

Rechnen mit `int`

Python benutzt für Arithmetik die folgenden Symbole:

- Grundrechenarten: `+`, `-`, `*` `/`
- Ganzzahlige Division: `//`
- Modulo: `%`
- Potenz: `**`

Rechnen mit `int`: Beispiele

Python-Interpreter

```
>>> 14 * 12 + 10
178
>>> 14 * (12 + 10)
308
>>> 13 % 8
5
>>> 11 ** 11
285311670611
```

Integer-Division: Ganzzahlig oder nicht?

Der Divisionsoperator `/` liefert das Ergebnis als `float`.
Der Operator `//` rundet auf die nächste ganze Zahl ab.

Python-Interpreter

```
>>> 20 / 3
6.666666666666667
>>> -20 / 3
-6.666666666666667
>>> 20 // 3
6
>>> -20 // 3
-7
```

Gleitkommazahlen und komplexe Zahlen

- `float`-Konstanten schreiben sich Dezimalpunkt und optionalem Exponent:

  `2.44`, `1.0`, `5.`, `1.5e+100` (bedeutet $1{,}5 \times 10^{100}$)
- `complex`-Konstanten schreiben sich als Summe von (optionalem) Realteil und Imaginärteil mit imaginärer Einheit `j`:

  `4+2j`, `2.3+1j`, `2j`, `5.1+0j`

Die arithmetischen Operatoren für `float` und `complex` sind die gleichen wie für die ganzzahligen Typen:

- Grundrechenarten: `+`, `-`, `*`, `/`, `//`
- Potenz: `**`
- Rest bei Division für ganzzahliges Ergebnis: `%`

Rechnen mit `float`

Python-Interpreter

```
>>> print(1.23 * 4.56)
5.6088
>>> print(17 / 2.0)
8.5
>>> print(23.1 % 2.7)
1.5
>>> print(1.5 ** 100)
4.06561177535e+17
>>> print(10 ** 0.5)
3.16227766017
>>> print(4.23 ** 3.11)
88.6989630228
```

Wieviel ist 2 – 2.1?

Python-Interpreter

```
>>> 2 - 2.1
-0.10000000000000009
```

- Die meisten Dezimalzahlen können **nicht** exakt als Gleitkommazahlen dargestellt werden (!)
- Programmier-Neulinge finden Ausgaben wie die obige oft verwirrend — die Ursache liegt in der Natur der Gleitkommazahlen und ist unabhängig von der Programmiersprache.

Rechnen mit `complex`

Python-Interpreter

```
>>> print(2+3j + 4-1j)
(6+2j)
>>> 1+2j * 100
(1+200j) [Achtung, Punkt vor Strich!]
>>> (1+2j) * 100
(100+200j)
>>> print((-1+0j) ** 0.5)
(6.12303176911e-17+1j)
```

Automatische Typkonversionen

Haben die Operanden unterschiedliche Typen, wie in `100 * (1+2j)` oder `(-1) ** 0.5`, werden die Operanden vom "kleineren" Typ zum "größeren" hin konvertiert. Dabei werden die folgenden Bedingungen der Reihe nach geprüft, die erste zutreffende Regel gewinnt:

- Ist einer der Operanden ein `complex`, so wird der andere zu `complex` konvertiert (falls er das nicht schon ist).
- Ist einer der Operanden ein `float` (und keiner ein `complex`), so wird der andere zu `float` konvertiert (falls er das nicht schon ist).

Überläufe und Unterläufe

- Ganze Zahlen können beliebig groß (und klein) werden.
- Gleitkommazahlen haben einen eingeschränkten Wertebereich (meist IEEE 754 Standard, double precision).
- Durch Interpreter, aber nicht durch Python festgelegt.

Python-Interpreter

```
>>> 1e-999
0.0
>>> 1e+999
inf
>>> 1e+999 - 1e+999
nan
```

`inf` steht für *infinity* und `nan` für *not a number*. Mit beiden kann weiter gerechnet werden!

Zusammenfassung

- Python ist ein objektorientierte, dynamisch getypte, interpretierte und interaktive höhere Programmiersprache.
- Python wird immer populärer und wird in den USA als die häufgste Anfängersprache genannt.
- Python läuft auf praktisch allen Maschinen und Betriebssystemen.
- Es gibt drei numerische Typen in Python: `int`, `float`, und `complex`.
- Es werden die üblichen arithmetischen Operationen unterstützt.