

Einführung in Python

Seminar „Linux für Umsteiger“
Sommersemester 2008

Benjamin B. Rommel - 17.06.2008

Einführung in Python – Gliederung

I **Einleitung**

II **Features**

III **Programmiergrundlagen**

IV **Bekannte Python-Produkte**

I

Einleitung

- Geschichtlicher Abriss
- Ziele von Python

II

Features

III

Programmiergrundlagen

IV

Bekannte Python-Produkte

I Einleitung

II Features

- Einfach, portabel
- Erweiterbar, robust
- Speichermanagement
- Kompilierung

III Programmiergrundlagen

IV Bekannte Python-Produkte

I **Einleitung**

II **Features**

III **Programmiergrundlagen**

- Starten von Python
- Konsolenausgabe
- Variablen
- Listen
- Schleifen, bedingte Anweisungen
- Funktionen

IV **Bekannte Python-Produkte**

I **Einleitung**

II **Features**

III **Programmiergrundlagen**

IV **Bekannte Python-Produkte**

- Vollständig in Python geschrieben
- Teilweise in Python geschrieben

I

Einleitung

- Geschichtlicher Abriss
- Ziele von Python

II

Features

III

Programmiergrundlagen

IV

Bekannte Python-Produkte

Geschichtlicher Abriss:

- Entwickelt zum Anfang der 1990er
- Ursprünglich für Amoeba konzipiert
- Basierend auf ABC

Ziele von Python:

- Einfache, intuitive Syntax
- Große Mächtigkeit
- Freie Verfügbarkeit

I Einleitung

II Features

- Einfach, portabel
- Erweiterbar, robust
- Speichermanagement
- Kompilierung

III Programmiergrundlagen

IV Bekannte Python-Produkte

Features:

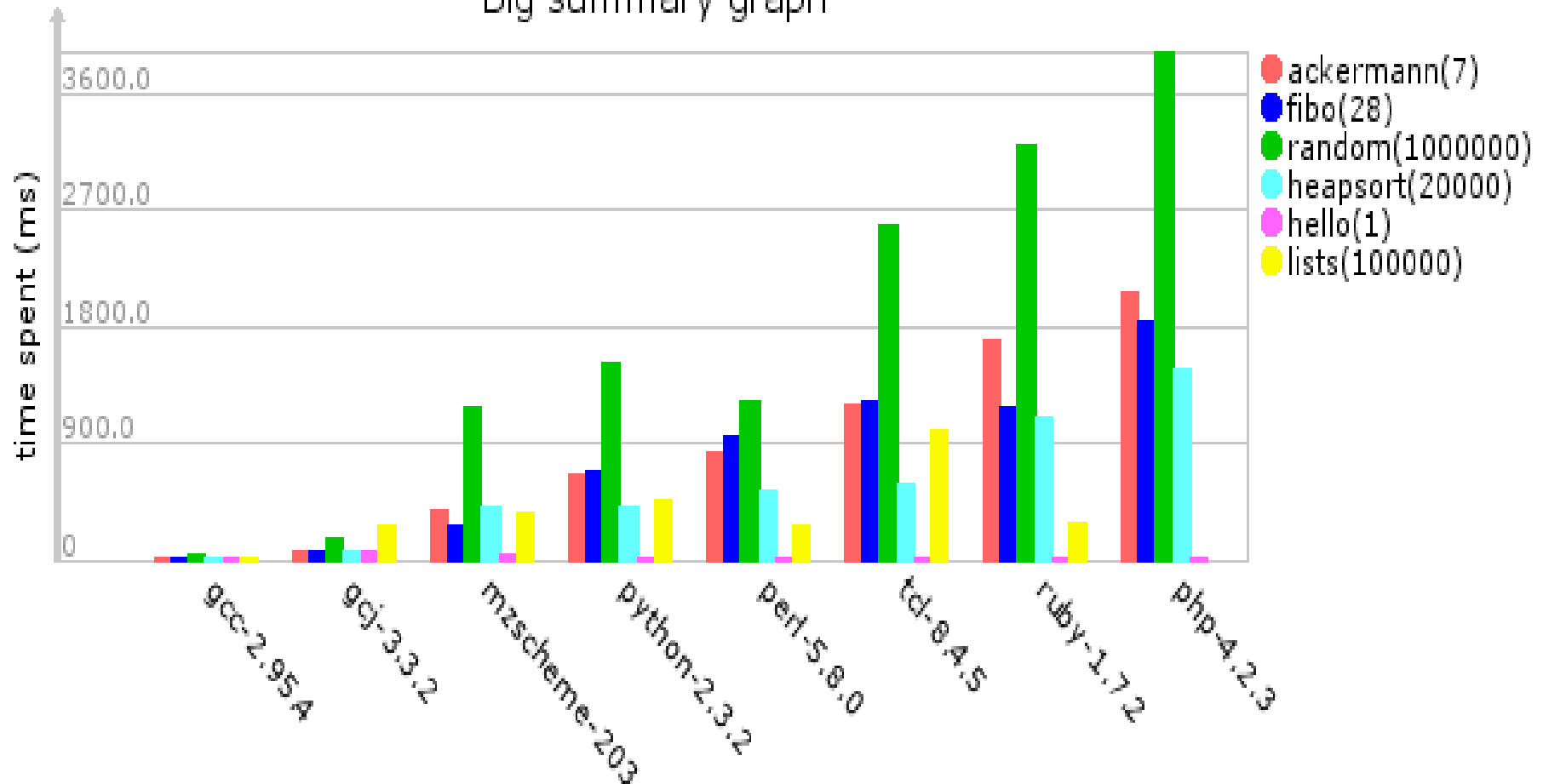
- Einfachheit
 - Reduzierte Syntax, daher leicht lesbar
 - Erzwungene Übersichtlichkeit
- Erweiterbarkeit
 - Quellcode kann auf Module verteilt werden
 - Module anderer Sprachen können geladen werden
- Portabilität
 - Auf allen Systemen mit ANSI C Compiler verfügbar
 - Mögliche Probleme bei neueren Python-Versionen

Features:

- Speichermanagement
 - Dynamische Datentypen
 - Garbage Collection
- Robustheit
 - Detaillierte Fehlerdiagnose
 - Ausgereifter Exception Handler
- Kompilierung
 - Zählt zu den interpretierten Sprachen
 - Wird byte-kompiliert
 - Langsamer als Programme in Maschinensprache

Übersicht:

Big summary graph



I **Einleitung**

II **Features**

III **Programmiergrundlagen**

- Starten von Python
- Konsolenausgabe
- Variablen
- Listen
- Schleifen, bedingte Anweisungen
- Funktionen

IV **Bekannte Python-Produkte**

Starten von Python:

- Starten der Python-Shell: `python`
- Starten eines Shell-Skriptes: `python skript.py`
- Verwendung einer IDE (z.B. IDLE)

Textausgabe:

- Ausgabe eines Wertes
`>>> print 1000`
- Ausgabe eines Strings
`>>> print "Hallo zusammen"`
`>>> print 'Hallo zusammen'`
- Kombinierte Ausgabe
`>>> print 2, '+', 2, '=', 4`

Variablen:

- Speicher
 - Dynamische Variablen, d.h. Typenfrei
 - Automatische Garbage Collection
- Variablennamen
 - Keine reservierten Namen, z.B. `def`, `in`, `for`, `if`
 - Variablen dürfen nicht mit Zahl beginnen
 - Namen sollten sinnvoll gewählt sein

Variablen:

- Deklaration, Wertzuweisung
 - Wertzuweisung geschieht durch Gleichheitszeichen
`>>> zahl = 500`
- Rechenoperationen
 - Es werden die Standard-Rechenregeln befolgt
`>>> zahl = (10 + 20 / 5) % 2 + 1`
- Ausgabe
 - Geschieht stets durch den `print`-Befehl
`>>> print 'zahl =', zahl`

Variablen:

- Strings
 - Wertzuweisung wie bei Zahlenvariablen
 - Interpretiert werden die Inhalte zwischen begrenzenden Zeichen
 - Beispiel: Es soll der Text **Hallo "Jack"** ausgegeben werden
 - Falsch: `>>> print "Hallo "Jack"`
 - Richtig: `>>> print 'Hallo "Jack"'`
 - Addition von Strings ebenfalls möglich:
`>>> print "Hallo " + "Jack"`

Listen:

- Variablen mit mehreren Elementen, vgl. Arrays in C
- Gleiche Regeln für Namensgebung wie normale Variablen
- Elemente könnten unterschiedlichen Typ haben

- Wertzuweisung der Werte in eckigen Klammern:

```
>>> liste = [1, 2, 'Hallo', 4]
```

- Ausgabe

- Die gesamte Liste:

```
>>> print liste
```

- Ein bestimmtes Element:

```
>>> print liste[0]
```

- Einen Bereich der Liste:

```
>>> print liste[1:3]
```

Bedingte Anweisungen:

- Realisiert als if-else-Bedingungen
- Variable wird mit Wert verglichen
 - Es stehen alle Standard-Vergleichsoperatoren zur Verfügung
- Ist die Bedingung erfüllt, wird der eingerückte Block ausgeführt
- Wurde die Bedingung nicht erfüllt, wird der unter else eingerückte Block ausgeführt oder alternativ elif-Abfragen durchgeführt

- Beispiel:

```
i = 10
if i > 20:
    print 'i > 20'
elif i > 10:
    print '10 < i <= 20'
else:
    print 'i <= 10'
```

while-Schleife:

- Solange eine Bedingung erfüllt ist, wird der eingerückte Block ausgeführt
- Verwendet werden die Standard-Vergleichsoperatoren

- Beispiel:

```
i = 15
while i > 10:
    print i
    i = i - 1
```

for-Schleife:

- In Python werden for-Schleifen durch Listen iteriert
- Alternativ kann eine Anzahl der Operationen angegeben werden

- Beispiel 1:

```
liste = [1, 2, 3, 4, 5]
for l in liste:
    print l
```

- Beispiel 2:

```
for l in range(5):
    print l+1
```

Funktionen:

- Dienen zur Codestrukturierung und Zusammenfassen von mehrfach verwendetem Code
- Funktionen werden nach den gleichen Regeln wie Variablen benannt
- Es können endlich viele Parameter übergeben werden
- Der Funktionsinhalt muss eingerückt sein

- Beispiel:

```
def function(n):  
    """Dies ist eine Funktion."""  
    n = n * 2000  
    print n  
  
function(5)
```

Funktionen - Rückgabewert:

- Funktionen können auch Werte zurückgeben, anstatt sie auszugeben
- Der Rückgabewert steht nach dem return-Befehl

- Beispiel:

```
def function(n):  
    """Dies ist eine Funktion."""  
    n = n * 2000  
    return n
```

```
print function(5)
```

Funktionen - Docstrings:

- Docstrings beschreiben den Sinn und Zweck einer Funktion
- Sie werden zu Beginn jeder Funktion geschrieben
- Docstrings zählen zum guten Programmierstil und sollten konsequent verwendet werden
- Beispiel anhand der letzten Funktion:
`print function.func_def` würde die folgende Ausgabe erzeugen:
Dies ist eine Funktion.

I Einleitung

II Features

III Programmiergrundlagen

IV **Bekannte Python-Produkte**

- Vollständig in Python geschrieben
- Teilweise in Python geschrieben

Bekannte Python-Produkte:

- Teilweise in Python geschrieben:
 - Battlefield 2: Konfigurationsskripte in Python
 - Blender: Python-Skript-Sprache
 - Morpheus: Client- / Server-Skripte
- Vollständig in Python geschrieben:
 - BitTorrent-Client
 - Django
 - EVE Online
 - Mojo Nation
 - Zope

Quelle: http://en.wikipedia.org/wiki/List_of_applications_written_in_Python

Zusammenfassung:

- Python ist ideal, um schnelle Ergebnisse zu erzielen
- Übersichtlichkeit des Codes, Erweiterbarkeit und Portabilität machen Python für vieles einsetzbar
- In der Leistung ist Python jedoch Maschinensprachen unterlegen
- Quellcode muss ggf. nach Versionsupdate aktualisiert werden

**Vielen dank für
eure Aufmerksamkeit!**

Quellen:

- <http://pytut.infogami.com/node1.html>
- http://de.wikipedia.org/wiki/Python_Programmiersprache
- http://de.wikipedia.org/wiki/Guido_van_Rossum
- <http://scutigena.sourceforge.net:80/>
- core Python programming, Wesley J. Chun