

Das ist Python

Ein Tutorial

Mike Müller
mmueller@python-academy.de
Python Academy, Leipzig

Workshop

„Python im deutschsprachigen Raum“
8. September 2006
Leipzig

Überblick

- Entwicklungsgeschichte, Eigenschaften, Philosophie und Anwendungsbereiche
- interaktiver Prompt, Programme
- Namen, Datentypen, höhere Datentypen
- Syntax und Kontrollfluss
- prozedurale, objektorientierte und funktionale Programmierung
- Gelegenheitsprogrammierer und Vollzeit-Softwareentwickler
- Pythonimplementierungen und -nutzung
- Zusammenfassung

Entwicklungsgeschichte

- Geburtsjahr: 1989/1990 (veröffentlicht 1991)
- Schöpfer: Guido van Rossum
- Vorfahren: ABC, C/C++, Smalltalk. Modula, Eiffel, Bash, (später Java)
- Namen: Von Monty Python (Flying Circus)
- Aktuelle Version: 2.4 (2.5)

Eigenschaften

- vielseitige Programmiersprache
- einfach erlernbar
- Open Source (Python-Lizenz)
- interaktiv
- prozedurale, objektorientierte und funktionale Programmierung
- portabel
- sehr hohes Abstraktionsniveau (VHLL)
- „Batterien enthalten“

Philosophie

- Beschränkung des Sprachumfangs
- ein bevorzugter Weg, um ein Problem zu lösen
- konsistentes Verhalten
- „schmutzige“ Tricks sind verpönt
- Produktivität des Programmierers steht im Vordergrund (nicht die des Computers)

Anwendungsbereiche

- für fast alle Bereiche geeignet
- Web-Programmierung (CGI, Zope, Django, TurboGears, ...)
- GUI-Programmierung (Tkinter, wxPython, GTK, QT, MFC, ...)
- Rapid Prototyping
- Textverarbeitung
- Lehre

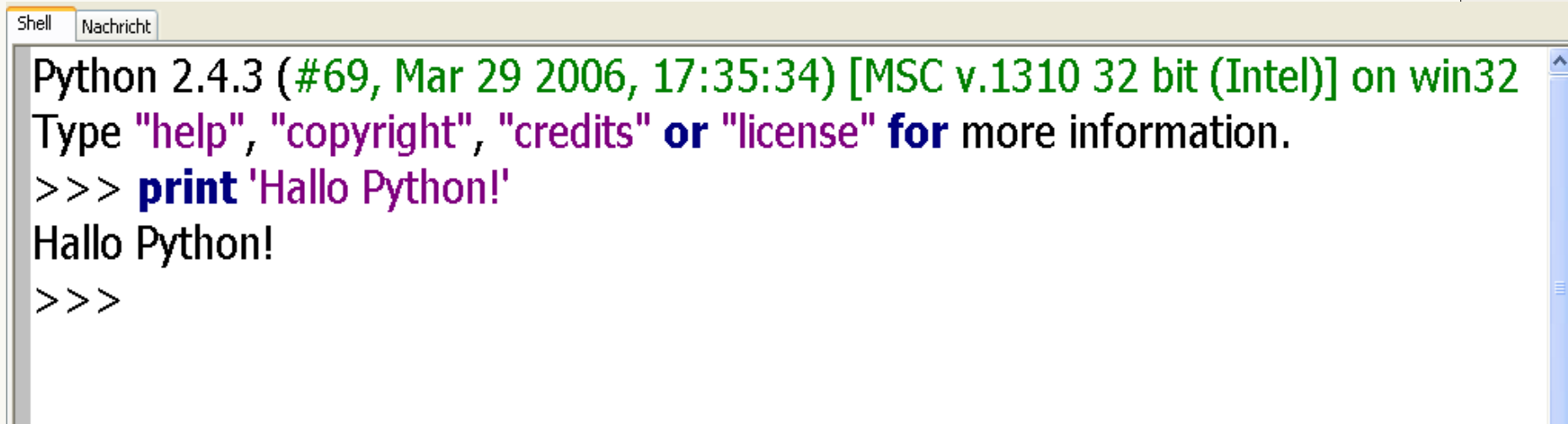
Mehr Anwendungsbereiche

- Systemadministration
- Automatisierung (Officeprogramme, COM, Gnome)
- Wissenschaftlicher Bereich
- Datenbankprogrammierung
- Einbetten in andere Programme
- plattformübergreifende Programme
- weniger geeignet für hardwarenahe Programmierung (Betriebssysteme, Treiber)

Python interaktiv

- interaktiver Modus zum Probieren und Veranschaulichen
- sofortige Rückmeldung an Programmierer ohne Kompilierung
- nach Test am Prompt in Datei *.py übernehmen
- ```
>>> print 'Hallo Python!'
Hallo Python!
```
- ```
hallo.py
print 'Hallo Python!'
```

Python interaktiv -Beispiel

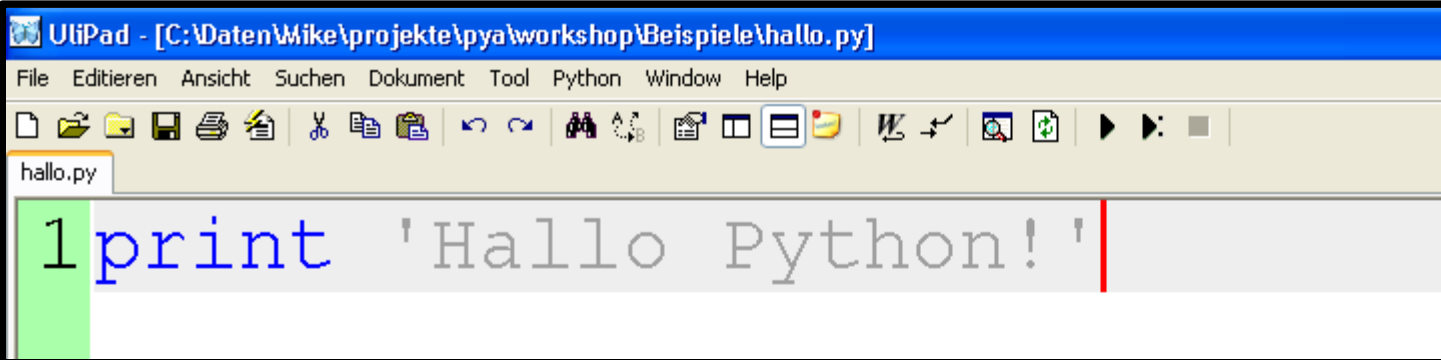
A screenshot of a Windows shell window titled "Shell" and "Nachricht". The window displays the Python 2.4.3 startup screen, including the version, date, time, and architecture. It shows the interactive prompt with the command `print 'Hallo Python!'` and its output.

```
Python 2.4.3 (#69, Mar 29 2006, 17:35:34) [MSC v.1310 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print 'Hallo Python!'
Hallo Python!
>>>
```

- interaktiver Prompt auf der Kommandozeile
- oder in IDE mit Syntaxhervorhebung

Python-Programme

Editor



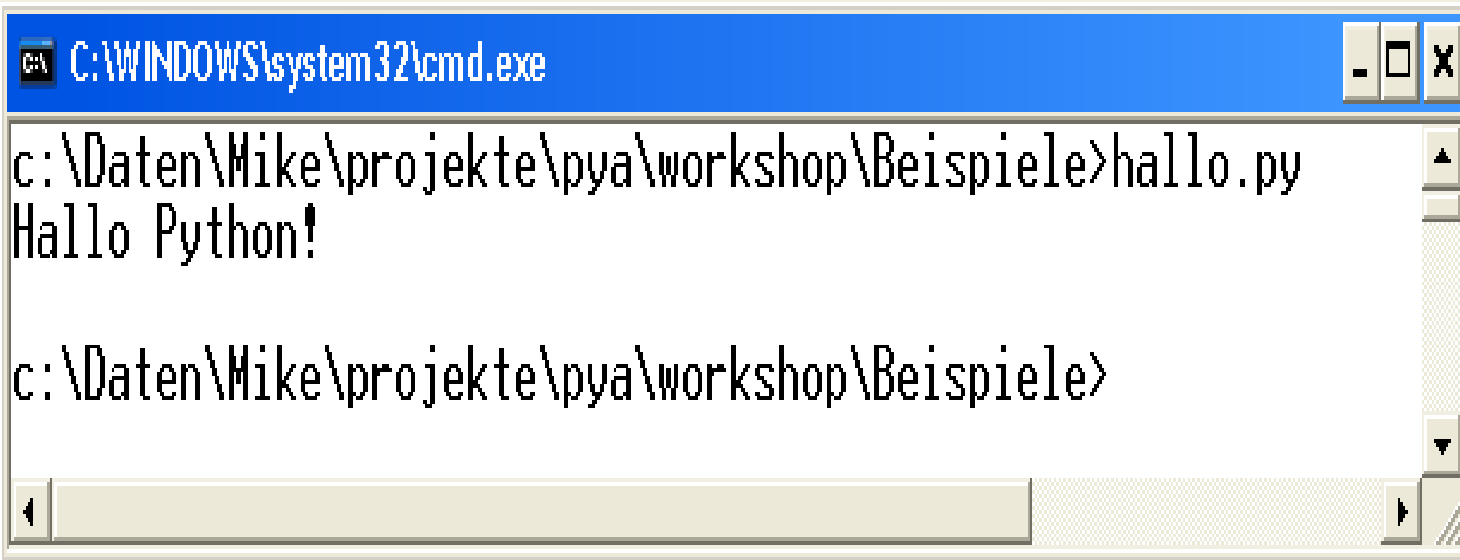
UtiPad - [C:\Daten\Mike\projekte\pya\workshop\Beispiele\hallo.py]

File Editieren Ansicht Suchen Dokument Tool Python Window Help

hallo.py

```
1 print 'Hallo Python!'
```

Kommandozeile



C:\WINDOWS\system32\cmd.exe

```
c:\Daten\Mike\projekte\pya\workshop\Beispiele>hallo.py
Hallo Python!

c:\Daten\Mike\projekte\pya\workshop\Beispiele>
```

Namen und Datentypen

- dynamische Typisierung, d.h. Typ wird zur Laufzeit bestimmt
- Namen können für Objekte anderen Typs wiederverwendet werden
- Ganzzahlen (`int`), Gleitkommazahlen (`float`), Zeichenketten (`string`)

```
zahl1 = 1 # Ganzzahl
```

```
zahl2 = 1.0 # Gleitkommazahl
```

```
text = 'Hallo Python!' # Text
```

Höhere Datentypen

- neben `int`, `float`, `string` gibt es z.B. Listen und Dictionaries
- sehr vielseitige Datentypen
- ermöglichen große Funktionalität
- konsistente Schnittstelle
- einfach
- werden in Python intern verwendet → Zugang zu Interpreterinformationen erleichtert

Höhere Datentypen - Beispiele

```
liste1 = [1, 2, 3]
```

```
dict1 = {'Peter': '0123 98765',  
        'Heike': '0456 12345'}
```

Alles ist ein Objekt

- Namen zeigen auf Objekte
- Auch Zahlen sind Objekte
- Eigenschaften und Methoden
- `id()`

Syntax

- Blöcke müssen eingerückt werden
- wenige Klammern
- keine Semikolons am Zeilenende
- gleiche Sicht auf Quelltext durch Programmierer und Interpreter
- „Executable Pseudo Code“

Kontrollfluss

- **Schleifen:** `for`, `while`
- `if`, `elif`, `else`
- `try`, `except`, `finally`
- **Vergleiche**

Prozedurale Programmierung

- Objektorientierte Programmierung ist nicht vorgeschrieben
- einfache Scripte durchaus ohne Strukturierung möglich
- Funktionen zur Strukturierung nutzen
- Beispiele

Objektorientierte Programmierung

- Klassen
- Methoden
- Vererbung
- Polymorphismus

Funktionale Programmierung

- map
- filter
- List comprehensions
- itertools

„Python mit Batterien“ (und Ladegerät)

- umfangreiche Standardbibliothek
- os, sys, shutil, ...
- Python Cookbook
- Cheese Shop

Python für Gelegenheitsprogrammierer

- leicht zu lernen
- auch nach längerer Nicht-Nutzung schnell wieder zu reaktivieren
- gut lesbar
- Sprachumfang überschaubar
- Automatisierung von Anwendungen
- Wissenschaftler/Ingenieure, Datenbank-/System-Administratoren und CMS-Nutzer
- Schnittstellen (COM, Erweiterungen)

Python für Gelegenheits- programmierer — Konzepte

- keine Überraschungen → **ein** bevorzugter Weg
- kein Zwang zur Objektorientierung
- Testen am interaktiven Prompt
- Bibliotheken für 99% der Aufgaben
- Philosophie: erst Bibliothek suchen, dann selbst programmieren
- höhere Datentypen mit großer Funktionalität

Python für Vollzeit-Softwareentwickler

- Integration in bestehende Systeme
- Zugang zu Interpreterinformationen
- skaliert für große Systeme
- agile Methoden gut anwendbar
- schnelle Entwicklung
- stabile Anwendungen

Python in bekannten Firmen

- Google
- NASA
- Red Hat
- Microsoft
- AstraZeneca
- Thawte Consulting (VeriSign)
- Industrial Light & Magic
- New York Stock Exchange

Andere Implementierungen IronPython

- CPython – C, Jython – Java, PyPy – Python
- IronPython - .NET
- Hauptentwickler Jim Hugunin (Jython)
- Version 1.0 RC2 (31. August)
- Zugang zu .NET-Plattform
- Potential: viele neue Entwickler
- können dann alle auch CPython
- Marketing-Schub zu erwarten

Nachteile von Python

- Keine
- Na gut, ein paar unwesentliche ...

Nachteile von Python

- teilweise unbekannt
- kein explizites Marketing wie z.B. Java
- Ausführungsgeschwindigkeit (JIT)
- Verteilung (py2exe)
- Bytecode leichter dekompilierbar als Maschinencode (kommerzielle Software)
- Mehr?

Zusammenfassung

- Python hat viele Vorteile
- wenige Nachteile
- gut zum Erlernen von Konzepten → gute erste Programmiersprache
- Auch gut für professionelle Softwareentwicklung-Entwicklung geeignet
- Nutzt Du Python oder kompilierst Du noch?