

Informatik I: Einführung in die Programmierung

2. Erste Schritte in Python

Albert-Ludwigs-Universität Freiburg



**UNI
FREIBURG**

Prof. Dr. Peter Thiemann

20. October 2021



Allgemeines

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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

Wir verwenden **Python** (genauer Python 3), eine

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



- Anfang der 90er Jahre als Skriptsprache für das verteilte Betriebssystem Amoeba entwickelt;
- gilt als einfach zu erlernen;
- wurde kontinuierlich von Guido van Rossum bei Google (seit 2013 Dropbox; seit 2019 i.R.) weiterentwickelt.
- bezieht sich auf die Komikertruppe *Monty Python*.



Guido van Rossum (Foto: Wikipedia)

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Hier eine Auswahl von Lehrbüchern zu Python3.

- Allen Downey, *Think Python: How to Think Like a Computer Scientist*, O'Reilly, 2nd edition, 2015
- als PDF herunterladbar oder als HTML lesbar (Green Tea Press):
<https://greenteapress.com/wp/think-python-2e/>
- als deutsche Version: Programmieren lernen mit Python, O'Reilly, 2013.
- Mark Lutz, *Learning Python*, O'Reilly, 2013 (deutsche Ausgabe ist veraltet!)
- Michael Weigend, *Python ge-packte Referenz*, mitp Verlag, 8. Auflage 2020 (als Nachschlagwerk, V3.8)
- Viele Videos und Online-Kurse
- Allgemeiner Hintergrund: Perdita Stevens, *How to Write Good Programs. A Guide for Students*, Cambridge University Press, 2020.

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Warum Python?

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



- 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, ...)

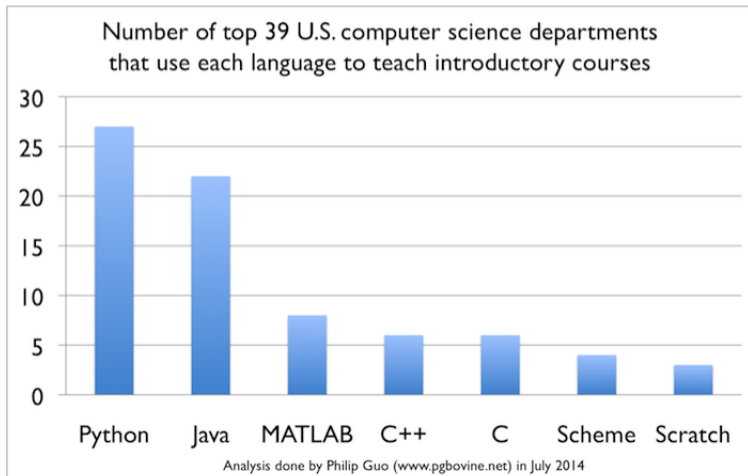
Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

Python ist #1



UNI
FREIBURG



Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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/>

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



- Python ist „langsamer“ als Java und C++
- Wieviel langsamer?
<https://benchmarkgame-team.pages.debian.net/benchmarkgame/index.html>
- Eignet sich nicht für das Schreiben von Gerätetreibern
- Eignet sich nicht direkt für die Programmierung von (kleinen) Mikrocontrollern (*bare metal programming*)
- “Python ... is not considered ideal for mobile app development and game development due to the consumption of more memory and its slow processing speed while compared to other programming languages.”
<https://squareboat.com/blog/advantages-and-disadvantages-of-python>



Unter <http://python.org/> befinden sich die aktuelle Dokumentation und Links zum Herunterladen (uns interessiert Python 3.X, $X \geq 9$) 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!**

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

Interpreter- versus Compiler-Sprachen



Allgemeines

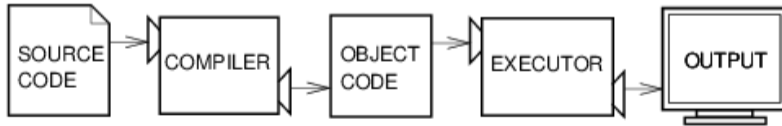
Warum
Python?

Python-
Interpreter

Shell

Rechnen

Interpreter- versus Compiler-Sprachen



Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Der Python-Interpreter kann auf folgende Arten gestartet werden:

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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.

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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**)



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.



Interaktives Nutzen der Shell

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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

```
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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"
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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'
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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 '
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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

Python-Interpreter

```
>>> print(7 * 6)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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

Python-Interpreter

```
>>> print(7 * 6)
42
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

Erste Schritte: „Drucken“



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

Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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.

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Ein weiteres Detail zu print:

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.

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

Exkurs: Hello-World-Programme



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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

Exkurs: Hello-World-Programme



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

Python

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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!");  
    }  
}
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Rechnen

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python kennt drei verschiedene Datentypen für Zahlen:

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

int



Schreibweise für Konstanten vom Typ `int`:

Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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.

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python benutzt für Arithmetik die folgenden Symbole:

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> 14 * 12 + 10
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> 14 * 12 + 10  
178  
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> 14 * 12 + 10
178
>>> 14 * (12 + 10)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> 14 * 12 + 10
178
>>> 14 * (12 + 10)
308
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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  
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



- Syntax von `float`-Konstanten: mit Dezimalpunkt und optionalem Exponent:
2.44, 1.0, 5., 1.5e+100 (bedeutet $1,5 \times 10^{100}$)
- Syntax von `complex`-Konstanten: 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: %

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> print(1.23 * 4.56)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> print(1.23 * 4.56)
5.6088
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> print(1.23 * 4.56)
5.6088
>>> print(17 / 2.0)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> print(1.23 * 4.56)
5.6088
>>> print(17 / 2.0)
8.5
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> print(1.23 * 4.56)
5.6088
>>> print(17 / 2.0)
8.5
>>> print(23.1 % 2.7)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> print(1.23 * 4.56)
5.6088
>>> print(17 / 2.0)
8.5
>>> print(23.1 % 2.7)
1.5
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

Wieviel ist $2 - 2.1$?



Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

Wieviel ist $2 - 2.1$?



Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

Wieviel ist $2 - 2.1$?



Python-Interpreter

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

- Die meisten Dezimalzahlen können **nicht** exakt als Gleitkommazahlen dargestellt werden (!)

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen

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.

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> print(2+3j + 4-1j)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> print(2+3j + 4-1j)
(6+2j)
>>>
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

```
>>> print(2+3j + 4-1j)
(6+2j)
>>> 1+2j * 100
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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)
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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).

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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

Python-Interpreter

```
>>> 1e-999
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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

Python-Interpreter

```
>>> 1e-999  
0.0
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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

Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



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

Python-Interpreter

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

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



- Ganze Zahlen können beliebig groß (und klein) werden.
- Gleitkommazahlen haben einen eingeschränkten Wertebereich (meist gemäß dem 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
```

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



- Ganze Zahlen können beliebig groß (und klein) werden.
- Gleitkommazahlen haben einen eingeschränkten Wertebereich (meist gemäß dem 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!

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen



- Python ist ein **objektorientierte, dynamisch getypte, interpretierte** und **interaktive höhere** Programmiersprache.
- Python ist sehr **populär** und wird in den USA als die **häufigste Anfängersprache** genannt. Betriebssystemen.
- Für manche Anwendungen ist Python zu **langsam** und **verbraucht zu viel Speicher**.
- Es gibt drei **numerische Typen** in Python: `int`, `float`, und `complex`.
- Es werden die üblichen **arithmetischen Operationen** unterstützt.

Allgemeines

Warum
Python?

Python-
Interpreter

Shell

Rechnen