Python BU

01_ Datentypen, Variablen, Zeichenketten

Numerische Datentypen und Literale

Python unterscheidet drei verschiedene grundlegende Arten von Zahlen:

1. Ganze Zahlen,
2. Gleitkommazahlen und
3. komplexe Zahl

Numerische Datentypen: Ganze Zahlen

Ganze Zahlen sind sozusagen das Brot- und Buttergeschäft der Computerei. In Python können Sie mit ihnen ziemlich intuitiv umgehen.

Ganz Konstanten im Programmtext (vulgo »Literale«) sind einfach Folgen von Ziffern, die dann als Dezimalzahlen interpretiert werden:

```
1
4711
3141926535
```

Wenn Sie maschinennahe Programmierung betreiben, ist es oft nützlich, Zahlen in einem anderen Zahlensystem als dem sonst üblichen Dezimalsystem anzugeben. Python unterstützt das Binär-, das Oktal- und das Hexadezimalsystem:

Numerische Datentypen: Gleitkommazahlen

Neben ganzen Zahlen unterstützt Python auch Gleitkommazahlen, salopp gesagt also Zahlen mit einem Nachkommaanteil.

Das Komma wird nach US-amerikanischer Manier aber als Punkt geschrieben: 1.5

Numerische Datentypen: Ganze Zahlen vs Gleitkommazahlen

Auch wenn ganzzahlige und Gleitkomma-Literale im Programmtext ziemlich ähnlich aussehen, stehen sie doch für sehr unterschiedliche Daten:

Normale Ganzzahlen werden im Computer genau dargestellt (mit den eingebauten Ganzzahlen des Prozessors, falls sie nicht zu groß sind, sonst mit Python-internen Datenstrukturen),

Gleitkommazahlen dagegen entsprechen den eingebauten Gleitkomma
zahlen »doppelter Genauigkeit«.

Auch bei Python gelten die üblichen Vorbehalte im Umgang mit computerisierten Gleitkommazahlen.
Zum Beispiel sollten Sie, wenn Sie zehnmal den Wert 0 .1 addieren, nicht damit rechnen, dass exakt 1. 0 herauskommt.

Überhaupt ist es in der Regel Keine Gute Idee, zwei Gleitkommazahlen auf Gleichheit zu testen. Dazu sagen wir später mehr.

Numerische Datentypen: Komplexe Zahlen

Elektrotechniker und andere Ingenieure werden mit Genugtuung feststellen, dass Python auch ohne Zusatzbibliotheken mit komplexen Zahlen umgehen kann. Imaginäre Literale bestehen einfach aus Gleitkomma- oder ganzzahligen Literalen mit einem angehängten »j« (oder »J«):

Numerische Ausdrücke

Python kommt mit den gängigen Grundrechenarten zu recht

Negative Zahlen sind auch kein Problem - setzen Sie einfach ein Minuszeichen direkt vor die Zahl.
In Python liefert eine Division immer ein Gleitkommaresultat, unabhängig vom Typ der Operanden.

Numerische Ausdrücke: Mathematische Funktionen

Python unterstützt auch einige nützliche mathematische Funktionen, die Sie auf Zahlen anwenden können:

```
>>> abs(-5)                                  Betrag
5
>>> int(3.14)                          Umwandeln in ganze Zahl
3
>>> int(-3.14)                 int rundet immer in Richtung Null
-3
>>> max(1,7,5)                               Maximum
7
>>> min(5,2,1)                               Minimum
1
>>> round(1.123)                             Runden
1.0
>>> round(1.234, 1)          Runden auf bestimmte Stellenzahl
1.2
```

Numerische Ausdrücke: Math Module

Für größere Geschütze der mathematischen Art müssen Sie auf das math-Modul zurückgreifen, ungefähr so

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sin(math.pi/2)
1.0
>>> math.log(10)                                    Basis e
2.302585092994046
```

Variable

Die allermeisten Programmiersprachen sind sich einig, dass es eine gute Idee ist, Objekten (wie Zahlenwerten) einen Namen geben zu können das macht Programme übersichtlicher, leichter lesbar und besser wartbar. Es ist allemal vernünftiger, etwas zu sagen wie

```
>>> pi = 3.141592654
>>> u = 2*pi*5                                  Umfang eines Kreises mit Radius 5
>>> a = pi*5*5                                  Fläche eines Kreises mit Radius 5
```

als etwas wie

```
>>> u = 2*3.141592654*5
>>> a = 3.141592654*5*5
```

Im ersten Beispiel haben Sie auch weniger Arbeit, wenn der Wert von n sich mal ändert.)

Benennungen im Python

Namen in Python dürfen
- Buchstaben,
- Ziffern
- und die Unterstreichung enthalten,

Das erste Zeichen eines Namens darf aber keine Ziffer sein.

Groß- und Kleinschreibung sind bedeutsam.

Die Länge eines Namens ist nicht beschränkt (außer durch Ihre persönliche Bequemlichkeit).

Sonderbedeutung

Einige Klassen von Namen haben eine besondere Bedeutung. Diese erkennt man an den Unterstreichungen am Anfang oder Ende:

_IRGENDWAS: Namen, die mit einer Unterstreichung anfangen, werden von einem Modul nicht automatisch exportiert.

Unterstreichungen

IRGENDWAS Namen mit je zwei Unterstreichungen am Anfang und am Ende haben eine »magische« Sonderbedeutung, die vom Python-Interpreter festgelegt wird. Sie können damit zum Beispiel Konstruktoren und Destruktoren für Objekte implementieren oder Operatoren überladen.

__IRGENDWAS Namen mit zwei Unterstreichung am Anfang (aber keinen zwei Unterstreichungen am Ende) gelten als »klassenprivat«.
Wenn Sie objektorientiert mit Klassen und Unterklassen arbeiten, können Sie so vermeiden, dass interne Attribute verschiedener Klassen miteinander kollidieren.

Zeichenketten- Literale

Zeichenketten-Literale werden entweder mit einfachen oder mit doppelten Anführungszeichen»eingerahmt«. Die Anführungszeichen sind selber nicht Teil des Literals.
Die betreffenden Anführungszeichen dürfen im Literal selbst nicht vorkomme

```
>>> "Er sagte "Huch" und fiel in Ohnmacht"
SyntaxError: invalid syntax
>>> 'Er sagte "Huch" und fiel in Ohnmacht'                    Besser.
'Er sagte "Huch" und fiel in Ohnmacht'
>>> "Er sagte 'Huch' und fiel in Ohnmacht"                    Auch gut.
"Er sagte 'Huch' und fiel in Ohnmacht"
```

```
>>> "Er sagte \"Huch\" und fiel in Ohnmacht"
'Er sagte "Huch" und fiel in Ohnmacht'
```

```
>>> """Er sagte "Huch"
... und fiel in Ohnmacht"""
'Er sagte "Huch"\nund fiel in Ohnmacht'
```

Alternativ können Sie »lange« Zeichenketten benutzen. Die beginnen mit drei Anführungszeichen (statt einem) und enden mit drei weiteren zu denen am Anfang passenden Anführungszeichen. In ihrem Innern dürfen nicht nur (einzeln oder paarweise) dieselben Anführungszeichen stehen, sondern auch Zeilen umbrüch

Rückstrich- Kombinationen in Zeichenketten-Literalen

Kombination	Bedeutung
`\\`	Rückstrich (\)
`\'`	Einfaches Anführungszeichen (')
`\"`	Doppeltes Anführungszeichen (")
`\a`	Signalton (ASCII BEL)
`\b`	Rückschritt (ASCII BS)
`\f`	Seitenvorschub (ASCII FF)
`\n`	Zeilenschritt (ASCII LF)
`\r`	Wagenrücklauf (ASCII CR)
`\t`	Horizontaler Tabulator (ASCII TAB)
`\v`	Vertikaler Tabulator (ASCII VT)
`\ooo`	Zeichen mit dem oktalen numerischen Code »ooo«
`\xhh`	Zeichen mit dem hexadezimalen numerischen Code »hh«
`\uxxxx`	Zeichen mit dem 16-Bit-Hexadezimalcode »xxxx«*
`\uxxxxxxxx`	Zeichen mit dem 32-Bit-Hexadezimalcode »xxxxxxxx«*
`\N{Name}`	Zeichen mit dem Namen *Name**

Zeichenketten

In Python können Sie mehrere Zeichenketten-Literale direkt nebeneinanderstellen. Diese gelten dann als ein großes Literal, das die Zeichenkette repräsentiert, die Sie erhalten, wenn Sie die Werte der einzelnen Literale lückenlos aneinanderhängen. Die einzelnen Literale dürfen zum Beispiel unterschiedliche Anführungszeichen-Konventionen verwenden:

```
>>> "Hallo" 'Welt'
'HalloWelt'
```

Auch eine Mischung aus Unicode- und einfachen Literalen ist erlaubt:

```
>>> u"Gr\u00fcezi" 'Welt'
u'Gr\xfceziWelt'
>>> print(u"Gr\u00fcezi" 'Welt')
GrüeziWelt
```

Zeichenketten-Ausdrücke

Auch Zeichenketten können Sie in Python gezielt manipulieren, etwa indem Sie sie aneinanderhängen:

```
>>> hw = "Hallo Welt"
>>> print(hw + 'enbummler')
Hallo Weltenbummler
```

Multiplikation« einer Zeichenkette mit einer Zahl ist auch erlaubt

```
>>> "Bla" * 3                                          So herum …
'BlaBlaBla'
>>> 3 * "Bla"                                          … oder so herum
'BlaBlaBla'
```

Die Länge einer Zeichenkette ermitteln Sie mit der Funktion len ():

```
>>> len("Hallo")
5
```

Zeichenketten sind in Python eine Form von »Folgen« (sequence types). Zu die- Folgen sen zählen auch Listen und Tupel, die wir später kennenlernen werden. Es gibt diverse nützliche Operationen, die Sie auf Listen und Tupel anwenden können und die auch mit Zeichenketten funktionieren - wir kommen darauf zurück.

Datentypen

Zusammenfassung

- Python unterstützt ganze Zahlen (»normale« und »lange«), Gleitkommazahlen und komplexe Zahlen.
- Python unterstützt die gängigen Grundrechenarten mit den üblichen Vorrangregeln (die Division ist ein bisschen trickreich).
- Diverse nützliche mathematische Funktionen sind entweder standardmäßig im Python-Interpreter realisiert oder über das Modul `math` zugänglich.
- Namen in Python dürfen Buchstaben, Ziffern und den Unterstrich enthalten (aber nicht mit einer Ziffer anfangen). Einige Schlüsselwörter sind tabu.
- Namen, die mit einem Unterstrich anfangen, haben eine Sonderbedeutung.
- Zeichenketten-Literale werden durch einfache oder doppelte Anführungszeichen begrenzt. »Lange« Literale dürfen Zeilenumbrüche enthalten.
- Zeichenketten unterstützen Aneinanderhängen, »Multiplikation« und Längenbestimmung.
- Zeichenketten in Python sind eine Art von Folgen