



Die Programmiersprache Python - eine kurze Einführung

Der Python-Prompt lautet `>>>` und bedeutet, dass Python bereit ist, eine Anweisung auszuführen oder eine Frage zu beantworten.

Python arbeitet interaktiv – wir können eine Anweisung schreiben und sofort die Auswertung erhalten. Python antwortet also mit der ausgewerteten Eingabe.

Grundrechenarten:

Anweisung	Ausgabe/Erklärung
<code>>>> 5 + 2</code>	7 (Addition)
<code>>>> 5 - 2</code>	3 (Subtraktion)
<code>>>> 5 * 2</code>	10 (Multiplikation)
<code>>>> 5 / 2</code>	2.5 (Division)
<code>>>> 5 // 2</code>	2 (Ganzzahldivision)
<code>>>> 6.0 // 2</code>	3.0 (Ergebnis ist float)
<code>>>> 5 ** 2</code>	25 (Potenzieren)

Arbeiten mit Strings (Zeichenketten)

Anweisung	Ausgabe/Erklärung
<code>>>> einstring = "Zeichenkette"</code>	immutable, d. h. einzelne Zeichen können nicht verändert werden
<code>>>> einstring[0]</code>	'Z' erstes Zeichen des Strings
<code>>>> einstring[0] = 'W'</code>	ERROR Fehlermeldung, da immutable
<code>>>> 3 * einstring</code>	'ZeichenketteZeichenketteZeichenkette'
<code>>>> einstring[-1]</code>	'e' erstes Zeichen von hinten
<code>>>> einstring[1:6]</code>	'eiche' 1. bis 6. Zeichen (jedoch ohne 6.)
<code>>>> einstring[:-1]</code>	'Zeichenkett' letzter Buchstabe wird gelöscht

Ausgabe (print-Befehl)

Anweisung	Ausgabe/Erklärung
<pre>>>> augen = 2 >>> nase = 1 >>> print("ich habe", augen, "Augen und", nase, "Nase!")</pre>	<p><i>Trennung durch „Komma“ (gemischte Datentypen, keine Typumwandlung)</i></p> <p><i>ich habe 2 Augen und 1 Nase!</i></p>
<pre>>>> print("ich habe " + str(augen) + " Augen und " + str(nase) + " Nase!")</pre>	<p><i>Trennung durch „Plus“ (Typumwandlung erforderlich)</i></p> <p><i>ich habe 2 Augen und 1 Nase!</i></p>



Quelltext

Kommentare beginnen mit #-Zeichen oder sind zwischen je 3 Anführungszeichen
Zusammengehörende Code-Blöcke werden eingerrückt In der Regel 4 Leerzeichen Beginn immer in gleichen Spalte keine geschweiften Klammern, Strichpunkte etc.
Am Ende einer Anweisung steht ein Doppelpunkt , danach Einrückung
Anweisung können durch Backslash (\) in der nächsten Zeile fortgeführt werden

Wiederholungen

Vorbemerkung:	
Zahlenbereiche	
<code>>>> range(8)</code>	die ersten 8 Zahlen ab 0 Ergebnis: 0,1,2,3,4,5,6,7
<code>>>> range(3,6)</code>	alle ganzen Zahlen von 3 bis 6 (ohne die 6) Ergebnis: 0, 1, 2, 3, 4, 5, 6, 7
Feste Anzahl an Wiederholungen (for-Schleife) (die Laufvariable i nimmt nacheinander jeden Wert an)	
<code>for i in range(0,3): print(i)</code>	Ausgabe: 0 1 2
<code>for i in ("Auto", 2, 3.14): print(i, type(i))</code>	Ausgabe: ('Auto', <type 'str'>) (2, <type 'int'>) (3.14, <type 'float'>)
Bedingte Wiederholungen (while-Schleife)	
<code>i = 0 while i < 3: print(i) i = i + 1</code>	Ausgabe: 0 1 2
Weitere Anweisungen in Wiederholungen	
<code>break</code>	verlässt diese Schleife sofort und macht nach der Schleife weiter
<code>continue</code>	beendet diesen Schleifendurchlauf und macht sofort mit dem nächsten Durchlauf weiter

Bedingungen

<code>if ... :</code> <code>elif ...:</code> <code>else:</code>	(einseitige Bedingung) (für Mehrfachauswahl) (Alternative)	Beispiel: <code>from random import choice optionen = ['Sonne', 'Wolken', 'Regen', 'Sturm'] wetter = choice(optionen) if wetter == 'Sonne': badengehen() elif wetter == 'Wolken': radeln() else: regenschirmEinpacken()</code>
-----------------------------------------------------------------------	------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



Funktionen

Methoden beginnen mit dem Schlüsselwort <code>def</code>	Beispiel: <pre>from math import sqrt def gibWurzel(zahl=0): „optionaler Kommentar, der angezeigt wird“ if zahl < 0: return False else: return sqrt(zahl)</pre>
Optional: Kommentar, der bei Aufruf angezeigt wird, folgt in „	
Rückgabewert folgt nach Schlüsselwort <code>return</code>	

Klassen

Klassen beginnen mit dem Schlüsselwort <code>class</code>	Beispiel: <pre>class Quader(): def __init__(self): self.laenge = 1 self.breite = 1 self.hoehe = 1 def volumen(self): vol = self.laenge * self.breite * self.hoehe return vol</pre>
Als Konstruktor dient die Methode <code>__init__(self)</code> <code>self</code> bezeichnet dabei den Selbstbezug, also die Klasse selbst.	
Attribute der Klasse werden mit Hilfe von <code>self</code> gesetzt.	
Weitere Methoden können mit Hilfe des Schlüsselwortes <code>def</code> hinzugefügt werden.	

Vererbung

Oberklassen werden bei der Klassendeklaration in die Klammer nach dem Klassennamen gesetzt.	Beispiel: (Klasse Quader erbt von Klasse box) <pre>class Quader(box): def __init__(self): box.__init__(self) self.laenge = 1 self.breite = 1 self.hoehe = 1</pre>
Mehrfachvererbung ist möglich, die einzelnen Oberklassen werden durch ein Komma voneinander getrennt.	
Im Konstruktor einer Klasse müssen auch die Konstruktoren der Oberklassen aufgerufen werden.	