

# Hilfsfunktionen zum Zahlenrechnen

Hier findest Du einige grundlegende Python-Funktionen, die beim Programmieren der Beispiele von 'Jede Menge Zahlen' nützlich sein können. Wie immer ist jede Verbesserung erlaubt und erwünscht!

Erst einmal die Wandlung zwischen einer Zahl (ihrem **Wert**) und ihrer **Darstellung** als String von Ziffern:

<pre>## Zahl (dezimal) in Ziffernstring wandeln  def numstr(n):     return str(n)  ## Ziffernstring (dezimal) in Zahl wandeln  def numval(n_str):     return long(n_str)</pre>	<p>Rückgabe long für alle Fälle, sonst reicht auch 'int(n_str)</p>
<pre>&gt;&gt;&gt; numstr(246) '246' &gt;&gt;&gt; numval("246") 246L</pre>	

Und nun das gleiche in einer beliebigen Zahlenbasis:

<pre>## Zahl (Basis) in Ziffernstring wandeln  DIGITS="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"  def numstrbase(num,base=10):     s = ""     while num&gt;0:         num,d = divmod(num,base)         s = DIGITS[d]+s     return s  ## Ziffernstring (Basis) in Zahl wandeln  def numvalbase(n_str,base=10):     return long(n_str,base)</pre>	<p>elegant: ist die Basis nicht angegeben, rechnen wir automatisch dezimal</p>
<pre>&gt;&gt;&gt; numstrbase(1234,16) '4D2' &gt;&gt;&gt; numvalbase("101100111000111100001111100000",2) 753124320L &gt;&gt;&gt; numvalbase("101100111000111100001111100000",4) 310840756082660352L</pre>	

Eine Zahl in die Liste ihrer Ziffern wandeln

<pre>def digitlist(n):     l = list(str(n))     return map(int,l)</pre>	n in Liste von Ziffern-Strings wandeln auf jedes Element 'int' anwenden
<pre>&gt;&gt;&gt; digitlist(2468753) [2, 4, 6, 8, 7, 5, 3]</pre>	

Der größte gemeinsame Teiler:

<pre># ggT  def gcd(a,b):     while b&gt;0: a,b = b,a%b     return a</pre>	
<pre>&gt;&gt;&gt; gcd(140,105) 35 &gt;&gt;&gt; gcd(141,105) 3 &gt;&gt;&gt; gcd(141,106) 1</pre>	teilerfremde Zahlen!

Gelegentlich brauchen wir die Liste der Primzahlen bis zu einer erwünschten Obergrenze. Hier eine sehr flotte, wenn auch ein wenig trickreiche Variante:

<pre>## sieve of Eratosthenes, returns list of primes 1&lt;primes&lt;=n  def eratosthenes(n):     sieve = [0,0]+[1]*n      # [0 0 1 1 1 ...]     prime = 2      while prime**2 &lt;= n:         for i in range(prime**2, n+1, prime):             sieve[i] = 0         prime = prime+1 + sieve[prime+1:].index(1)      return filter(lambda i, sieve=sieve: sieve[i],range(n+1))</pre>	[0,0,1,1,1,1,...]  Liste durchwandern Vielfache markieren nächste Primzahl  Filter ergibt range- Werte, falls Sieve- Wert=1
<pre>&gt;&gt;&gt; eratosthenes(30) [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]</pre>	

Funktionen, die man zum Umkehren von Strings und Zahlen brauchen kann (Palindromaufgaben!)

<pre>## String umkehren  def revstr(s):     l = list(s)     l.reverse()     return "".join(l)  ## Zahl reversieren  def revnum(n):     return long(revstr(str(n)))</pre>	
<pre>&gt;&gt;&gt; revstr("abcde") 'edcba' &gt;&gt;&gt; revstr("9901") '1099' &gt;&gt;&gt; revnum(9901) 1099L</pre>	

**Bildschirmausgabe:**

einmal simpel, dann mit den Formatierungsanweisungen aus C. Beachte: Die Formatierungen liefern einen String zurück, der sogar nach Belieben weiterverarbeitet werden kann!

<pre>print 'a', 'b' print 'a'+ 'b'  '%s---%s'%( 'a', 'b' ) '%3d'%1 '%-3d'%1  '%10.4f'%x</pre>	<pre>zeigt 'a b' zeigt 'ab'  ergibt 'a---b' ergibt ' 1'    auf 3 Plätze rechtsbündig formatiert ergibt '1  '   auf 3 Plätze linksbündig formatiert  ergibt Fließkommazahl x mit vier Nachkommastellen auf 10 Plätze rechtsbündig</pre>
---	--
