





P y t h o n - S p i c k z e t t e l f ü r P H P - E n t w i c k l e r

	
Arrays	
array()	
<pre>\$aTest = array('a','b','c');</pre>	<pre>aTest = 'a','b','c' # Tupel aTest = ('a','b','c') # Tupel aTest = ('a',) # (!!!) Tupel aTest = ['a','b','c'] # Liste Liste: Dynamisches "Array", Tupel: Statisches "Array"</pre>
in_array()	
<pre>if in_array('a',array('a','b','c')) { { echo 'Ist drin'; }</pre>	<pre>if 'a' in ('a','b','c'): print 'Ist drin'</pre>
count()	
<pre>\$aTest = array('a','b','c'); echo count(\$aTest);</pre>	<pre>aTest = 'a','b','c' print len(aTest)</pre>
implode()	
<pre>\$aTest = array('a','b','c'); echo implode('-', \$aTest); // 'a-b-c'</pre>	<pre>aTest = 'a','b','c' print '-'.join(aTest)</pre>
explode()	
<pre>\$aTest = explode('-', 'a-b-c');</pre>	<pre>aTest = 'a-b-c'.split('-')</pre>
Assoziative Arrays	
<pre>\$aTest['bla'] = array('a','b','c');</pre>	<pre>aTest = {'bla': ('a','b','c')} oder: aTest = {} aTest['bla'] = 'a','b','c'</pre>
Zeichenketten	
Mehrzeilige Zeichenketten	
<pre>echo <<<EOT Zeile 1 Zeile 2 EOT;</pre>	<pre>print "" Zeile 1 Zeile 2 ""</pre>
substr()	
<pre>echo substr('abcdef',1); // 'bcdef' echo substr('abcdef',1,3); // 'bcd' echo substr('abcdef',0,4); // 'abcd' echo substr('abcdef',0,20); // 'abcdef'</pre>	<pre>print 'abcdef'[1:] # 'bcdef' print 'abcdef'[1:4] # 'bcd' (!!!) print 'abcdef'[:4] # 'abcd' print 'abcdef'[:20] # 'abcdef'</pre>
strlen()	
<pre>echo strlen('abcd');</pre>	<pre>print len('abcd');</pre>
strpos()	
<pre>echo strpos('abcde','b'); // '2' if (strpos('abcde','abc') === 0) { echo 'Beginnt mit Needle.'; }</pre>	<pre>print 'abcde'.index('b') # '2' if 'abcde'.startswith('abc'): print 'Beginnt mit Needle.'</pre>
str_replace()	
<pre>echo str_replace('a','b','abc');// 'bbc'</pre>	<pre>print 'abcd'.replace('a','b') # 'bbc'</pre>

	
Kontrollstrukturen, Operatoren etc.	
for-Schleife	
<pre>for(a=0;\$a<10;\$a++) { echo \$a; }</pre>	<pre>for a in range(10): print a Nutzung von range(): range(3) # 0,1,2 range(2,5) # 2,3,4 range(5,25,5) # 5,10,15,20</pre>
Operatoren etc.	
!	not
.=	+=
Einzeilige Kommentare	
//	#
Mehrzeilige Kommentare	
<pre>/* Blablaba */</pre>	<pre>''' Blablaba '''</pre>
Zeichenverkettung	
<pre>\$nTest = 1; echo 'Die Zahl ist '.\$nTest;</pre>	<pre>nTest = 1 print 'Die Zahl ist ' + str(nTest) besser (automatische Typkonvertierung): print 'Die Zahl ist ', nTest print 'Die Zahl ist %s' % nTest</pre>
Typkonvertierung	
<pre>intval(), floatval(), strval()</pre>	<pre>int(), float(), str()</pre>
Boolscher Typ	
<pre>True, true, TRUE</pre>	<pre>True</pre>
<pre>False, false, FALSE</pre>	<pre>False</pre>
Wahrheitswerte von Objekten werden mit bool(<i>Objekt</i>) verglichen	
Trinitäts-Operator	
<pre>echo ('a'=='a' ? 'Gleich' : 'Ungleich');</pre>	<pre>print 'a'=='a' and 'Gleich' or 'Ungleich'</pre>
Besonderheiten	
Rechenprobleme bis einschliesslich Version 2.4	
<pre>echo 7/4; // ergibt 1.75</pre>	<pre>print 7/4 # ergibt 1 (!!!) richtig: print 7.0/4.0 # ergibt 1.75 print float(7)/float(4) # ergibt 1.75 oder: from __future__ import division</pre>