|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Kurzübersicht über die Programmierung mit Python***  **Datentypen**   |  |  |  | | --- | --- | --- | | **Integer** | i=1 | Ganzzahl | | **Float** | f=0.1 | Gleitkommazahl | | **String** | s="Hallo" | Zeichenkette | | **Liste** | l=[1, 2, 3] | Veränderbare Liste mit Reihenfolge | | **Tupel** | t=(1, 2, 3) | Unveränderbare Liste mit Reihenfolge | | **Dictionary** | d={1: "eins", 2: "zwei", 3: "drei"} | Assoziatives Array |   **Umwandlung von Datentypen**  int(2.33)            liefert 2  float(3)              liefert 3.0  str(53)               liefert die Zahl 53 als Zeichenkette  **Ausdrücke und Operatoren, Bedeutung in Klammern**  **Arithmetik**  **+** (plus),  **-** (minus),  **\*** (mal),  **/**  (dividiert durch), **\*\*** (hoch), **%** (Modulo, Rest einer Division)  16/3 liefert 5 (Ganzzahldivision), 16.0/3 liefert  5.333333333333333 (Gleitkommazahldivision)  **Vergleiche**  **==**(gleich),  **!=** (ungleich), > (größer), < (kleiner), >= (größer oder gleich), <= (kleiner oder gleich)  **Logik**  **true** (wahr), **false** (falsch), **and** (und zugleich), **or** (oder), **not** (nicht)  **Element-Beziehung**  **in** (in der Menge), z.B.  3 **in** [1, 2, 3]  (true),  4 **not in** [1, 2, 3]  (true)  **Zeichenketten**  Zeichenketten können mit **+** verknüpft oder mit **\*** wiederholt werden, z.B.  “abc“**+**“def“  liefert  “abcdef“,   5**\***“A“  liefert   “AAAAA“  **Sequenzen**  Als Sequenzen werden Tupel (1, 2, 3) und Listen [1, 2, 3] bezeichnet. Im Gegensatz zu Dictionaries sind Sequenzen geordnet, d.h. jedes Element hat eine eindeutige Position, ab Position **0** (Null) für das erste Element der Liste,  **-1** kann für das letzte Element.gesetzt werden, z.B. (3, 4, 5)[0]  liefert  3,  (3, 4, 5)[-1]  liefert  5  **Listen-Generatoren**  **range( 10)** gleichwertig zu [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], **range(1,4)** entspricht [1,2,3]  **Funktions-Definition**  **def** quadrat(x):  **return** x \*\* 2  quadrat(3)         liefert  9  **Befehle bei Zahlen und Zeichenketten**  **round(a [, n])** - rundet eine Fließkommazahl. Optional kann die Anzahl der Nachkommastellen angegeben werden, z.B.  **round(5.59555, 2)**      liefert  5.6  **str()** - in String konvertieren,  **float()** - in Fließkommazahl konvertieren.  Nicht nur Zahlen, sondern generell alle Objekte kann man mittels **str()** in einen String konvertieren, z.B. **print** **(str**(167)[-1])        liefert  7  **len**() - Länge des Strings ermitteln, z.B.  test\_string=“Heute“  **len**(test\_string)              liefert  5  **Kontrollstrukturen**  **if-elif-else**        Beispiel:  x = 8  **if** x > 5:  **print** ("x ist größer als 5")  **elif** x < 5:  **print** ("x ist kleiner als 5")  **else**:  **print** ("x ist 5")                      Ausgabe:  x ist größer als 5    **while**               Beispiel:  i = 0  **while** i < 4:  **print** ("i ist", i)       i = i + 1  Ausgabe:  i ist 0  i ist 1  i ist 2  i ist 3    **for**                Beispiel:  **for** i **in** range(10,21):  **print** (i\*i,end=' ')  Ausgabe:                      100 121 144 169 196 225 256 289 324 361 400    **break** Beispiel:  **for** i **in** "Hallo Welt!":  **if** i == " ":  **break**  **print** (i,end='')  Ausgabe:  Hallo    **Module**  **from …  import …**  **z.B. from** math **import \***  Bedeutung: Vom Modul math wird alles (\*) importiert.  **Ein- und Ausgabe**  **a = input(**“Text“)             #Eingabe der Variable a  **print** “Text“, a                 #Ausgabe von Text und Wert der Variablen a  Beispiel:  i = 99  print ('%3d   %1.6f' % (i,1.0/i))  Ausgabe:  99   0.010101  **Schlüsselwörter**  **and,  as,  assert,  break,  class, continue,  def,  del,  elif ,  else,  except,  exec,  finally,  for,  from,  global,  if,  import,  in,  is,  lambda, not,  or,  pass,  print,  raise,  return,  try,  while,  with,  yield** |  |