```
Formatierte Ausgabe in Python               (C) 2019-2020 T.Birnthaler OSTC GmbH
============================

Doku --> http://docs.python.org/3/library/functions.html#format                format Function
         http://docs.python.org/3/library/stdtypes.html#old-string-formatting printf-style String Fo
rmatting
         http://docs.python.org/3/library/stdtypes.html#str.format             String Methods
         http://docs.python.org/3/library/string.html#formatspec               Format-specification M
ini-Language
         http://docs.python.org/3/library/string.html#formatstrings            Format String Syntax
         http://docs.python.org/3/library/string.html#string-formatting        Custom String Formatti
ng
         http://docs.python.org/3/library/string.html#string.Formatter         Formatter Class
         http://docs.python.org/3/reference/lexical_analysis.html#f-strings    Formatted String Liter
als

Es gibt in Python 3 Arten, formatierte Zeichenketten mit festen Textteilen und
variablen Datenteilen darin zu erzeugen. Alle 3 beruhen auf dem Prinzip, in
eine SCHABLONE/ein TEMPLATE (String mit PLATZHALTERN) Werte im passenden Format
einzubauen und den Ergebnisstring zurÃ¼ckzugeben.

  A) C-Stil        (VERALTET analog printf-Funktion mit %-Operator)
  B) Python-Stil 1 (MODERN mit String-Funktion STR.format(...))
  C) Python-Stil 2 (MODERN mit Format-String F"...")

Hier ein identisches Beispiel fÃ¼r alle 3 Stile: Folgende 3 Werte

  zahl = 5678      # Ganzzahl
  wert = 3.1415    # FlieÃM-^_komma
  text = "hallo"   # String

werden per Platzhalter mit passendem Formattyp (d=decimal, f=float, s=string)
in einen String eingefÃ¼gt:

  print( "Zahl %6d, Wert %6.2f, Text %-10s." % (zahl, wert, text))
  print( "Zahl {0:6d}, Wert {1:6.2f}, Text {2:10s}.".format(zahl, wert, text))
  print(F"Zahl {zahl:6d}, Wert {wert:6.2f}, Text {text:10s}.")

Es ergibt sich jeweils der gleiche Ergebnisstring:

  "Zahl   5678, Wert   3.14, Text hallo     ."
        ------       ------       ----------
          6d          6.2f           -10s

Die Platzhalter werden also per % oder {} dargestellt, der Typ-Buchstabe am
Ende eines Platzhalters legt den erwarteten Datentyp des einzufÃ¼llenden Wertes
fest (Python Ã¼berprÃ¼ft dies auch). Die Werte zu den Platzhaltern werden durch
ihre Reihenfolge, ihre Positionsnummer oder ihren Variablennamen festgelegt.

Folgende Formattyp-Buchstaben sind mÃ¶glich:

  +-----+-------------+-------------------------------+-----------------------+
  | Bst | Name        | Bedeutung                     | Beispiel              |
  +-----+-------------+-------------------------------+-----------------------+
  | s   | string      | Zeichenkette                  |                       |
  | c   | character   | Zeichen per Unicode           |                       |
  +-----+-------------+-------------------------------+-----------------------+
  | d   | decimal     | Ganzzahl im Dezimalformat     |                       |
  | i   | integer     | Ganzzahl im Dezimalformat     |                       |
  +-----+-------------+-------------------------------+-----------------------+
  | x X | hexadecimal | Ganzzahl im Hexadezimalformat |                       |
  | o   | octal       | Ganzzahl im Oktalformat       |                       |
  | b   | binary      | Ganzzahl im BinÃ¤rformat       |                       |
  +-----+-------------+-------------------------------+-----------------------+
  | f F | float       | FlieÃM-^_kommazahl            |                       |
  | e E | exponent    | FlieÃM-^_kommazahl            |                       |
  | g G | general     | FlieÃM-^_kommazahl            |                       |
  | n   | number      | FlieÃM-^_kommazahl            |                       |
  +-----+-------------+-------------------------------+-----------------------+

Vor dem Typbuchstaben kÃ¶nnen folgende Angaben stehen (Standard ist Leerzeichen
als FÃ¼llzeichen und rechtsbÃ¼ndige Ausrichtung; bei FlieÃM-^_kommazahlen wird auf
die angegebenen Nachkommastellen gerundet):

  * Ausrichtung (- oder < linksbÃ¼ndig, > rechtsbÃ¼ndig, ^ zentriert)
  * FÃ¼llzeichen (Leerzeichen, 0-Zeichen oder andere per =Z)
  * Minimale Breite (wird "gesprengt" falls Wert breiter ist)
  * Maximale Breite (nach Punkt, nur Strings)
  * Anzahl der Nachkommastellen (nach Punkt, nur FlieÃM-^_kommazahlen)

Im Python-Stil 1 steht im Platzhalter vor einen ":" eine Indexposition (mit 0
beginnend) der Werte im .format(...)-Aufruf (Reihenfolge der Nummern beleibig,
```

```
gleiche Nummer mehrfach erlaubt). Ohne Nummer werden die Werte von links nach
rechts in die jeweiligen Platzhalter eingefÃ¼llt (":" ist trotzdem notwendig!).

  "Zahl {0:6d}, Wert {1:6.2f}, Text {2:10s}.".format(zahl, wert, text)
  "Zahl {:6d}, Wert {:6.2f}, Text {:10s}.".format(zahl, wert, text)

Ergibt:

  "Zahl   5678, Wert   3.14, Text hallo     ."
  "Zahl   5678, Wert   3.14, Text hallo     ."
        ------       ------       ----------

Im einfachsten Fall genÃ¼gt ein "{}" als Platzhalter, diese werden von links
nach rechts mit den Werten in .format(...) in der notwendigen Breite gefÃ¼llt:

  "Zahl {}, Wert {}, Text {}.".format(zahl, wert, text)

Ergibt:

  "Zahl 5678, Wert 3.1415, Text hallo."
        ----       ------       -----

Im Python-Stil 2 steht im Platzhalter vor einem ":" ein Variablenname oder ein
Rechenausdruck, dessen Wert eingefÃ¼llt wird:

  F"Zahl {zahl:6d}, Wert {wert:6.2f}, Text {text:10s}.")
  F"Zahl {zahl*5+2:6d}, Wert {wert/5.0:6.2f}, Text {text+'X':10s}.")

Ergibt:

  "Zahl   5678, Wert   3.14, Text hallo     ."
  "Zahl  28392, Wert   0.63, Text halloX    ."
        ------       ------       ----------

Im Python Stil 2 lassen sich mit weiteren (eingeschachtelten) Klammern {...}
beliebige Teile der Formatangabe per Variable oder Rechenausdruck fÃ¼llen:

  breite = 6
  nkst = 2
  typ = "d"
  F"Zahl {zahl:{breite}{typ}}, Wert {wert:{breite}.{nkst}f}, Text {text:{breite}s}.")

Ergibt:

  "Zahl   5678, Wert   3.14, Text hallo ."
        ------       ------       ------
```