Python Sequenzen (str, tuple, list) (C) 2019-2021 T.Birnthaler OSTC GmbH
==========

Doku --> http://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range

Sequenzen sind eine "geordnete" Folge von Elementen, jedem Element wird ein INDEX zugeordnet (Ganzzahl beginnend mit 0), Ã¼ber den es angesprochen werden Kann. Der Index lÃ¤uft AUFSTEIGEND von 0 bis Sequenz-LÃ¤nge-1 bzw. ABSTEIGEND von -1 bis -Sequenz-LÃ¤nge:

```
  s1 = "hallo"                      s2 = ""     # String + leerer String
  t1 = ( True, "hallo", 3.14 )      t2 = ()     # Tupel + leerer Tupel
  l1 = [ 123, 456, 789 ]            l2 = []     # Liste + leere Liste

   01234  # Positiver Index (0 .. LÃ¤nge-1)
  "hallo" # Sequenz (str)
      -1  # Negativer Index (-1 .. -LÃ¤nge) = AbkÃ¼rzung fÃ¼r len(s) - 1
     -2   #                                              len(s) - 2
    -3    #                                              len(s) - 3
   -4     #                                              len(s) - 4
  -5      #                                              len(s) - 5
```

Die Zugriffsgeschwindigkeit auf ein bestimmtes ELEMENT Ã¼ber seinen INDEX ist unabhÃ¤ngig von der Sequenz-LÃ¤nge immer gleich schnell (RANDOM ACCESS).

Die Zugriffsgeschwindigkeit auf eine bestimmtes ELEMENT Ã¼ber seinen WERT (z.B. welchen Index hat ein bestimmter Wert oder wie hÃ¤ufig kommt ein WERT in der Sequenz vor) ist abhÃ¤ngig von der Sequenz-LÃ¤nge und immer langsam (erfordert das Durchsuchen der gesamten Sequenz)

Es gelten folgende BeschrÃ¤nkungen:

Datentyp		Elemente	Elementanzahl
str	immutable	UTF-8 codiertes Zeichen	fest
tuple	immutable	Python-Objekt	fest
list	mutable	Python-Objekt	variable
bytes	immutable	Byte 00-FF	fest
bytearray	mutable	Byte 00-FF	variabel
array	mutable	Zahlen gleichen Typs	fest

Folgende Operationen werden von (fast) allen Sequenz-Typen (str, tuple, list, bytes, bytearray) unterstÃ¼tzt (sowohl mutable als auch immutable). S und T sind Sequenzen des gleichen Typs, I, J, K, N sind Ganzzahlen und E ist ein beliebiges Objekt, das die BeschrÃ¤nkungen der Sequenz S erfÃ¼llt.

ACHTUNG: Die Sequenz S bleibt unverÃ¤ndert!

Operation	Bedeutung
E in S	Element E in S vorhanden?
E not in S	Element E in S nicht vorhanden?
S + T	Verkettung von S und T
S * N N * S	Sequenz N-mal verketten
S[I]	Element mit Index I (Start bei 0)
S[I:J]	Slice von I bis J-1 (mit Schrittweite 1)
S[I:J:K]	Slice von I bis J-1 mit Schrittweite K
len(S)	LÃ¤nge von S (Anzahl Elemente)
min(S)	Kleinstes Element in S (gemÃ¤Ã-^_ Vergleich "<")
max(S)	GrÃ¶Ã-^_tes Element in S (gemÃ¤Ã-^_ Vergleich "<")
sum(S)	Summe der Elemente in S (mÃ¼ssen Zahlen sein)
all(S)	True falls alle Elemente True (oder S leer), sonst False
any(S)	False falls alle Elemente False (oder S leer), s. True
S.index(E[,I[,J]])	Index des 1. Elements gleich E in S (von Index I bis J-1; Standard: 0 bis len(S)-1)
S.count(E)	Anzahl Vorkommen von Element E in S
reversed(S)	Iterator mit Elementen in umgekehrter Reihenfolge
sorted(S,key,reverse)	Sortierte Liste der Elemente (gemÃ¤Ã-^_ Vergleich "<") (key=MAPFUNC, reverse=True --> absteigend)

HINWEIS: Indexwerte I und J sowie Schrittweite K dÃ¼rfen auch negativ sein (auch gemischt).

```
HINWEIS: Die Formulierung "gemÃ¤ÃM-^_ Vergleich <" bedingt, dass ALLE Elemente per
Operator "<" miteinander vergleichbar sind. D.h. entweder alle Elemente sind
Zahlen (bool, int, float, complex, Decimal, Fraction) oder alle Elemente haben
den gleichen Datentyp str, tupel, list, ...

HINWEIS: Der Indexzugriff wird Ã¼berwacht, d.h. bei Zugriff Ã¼ber den Rand hinaus
         erfolgt eine Exception "IndexError". Indexzugriffe und Slicing erzeugen
         eine Kopie der Sequenz und dÃ¼rfen aus einer beliebigen Kombination von
         positiven und negativen Indices bestehen. Der Startindex 0, der
         Endindex len(S)+1 und die Schrittweite 1 dÃ¼rfen weggelassen werden.

           S[:]            = 1:1-Kopie der Sequenz
           S[::]           = 1:1-Kopie der Sequenz
           S[::1]          = 1:1-Kopie der Sequenz
           S[0:]           = 1:1-Kopie der Sequenz
           S[:len(S)+1]    = 1:1-Kopie der Sequenz
           S[0:len(S)+1]   = 1:1-Kopie der Sequenz
           S[0:len(S)+1:1] = 1:1-Kopie der Sequenz
           S[::2]          = Jedes 2. Element ab 1. Element (Index 0,2,4,...)
           S[1::2]         = Jedes 2. Element ab 2. Element (Index 1,3,5,...)
           S[5:]           = Ohne erste 5 Elemente (Index 5..*)
           S[:5]           = Erste 5 Elemente (Index 0..4)
           S[3:5]          = Element 4 und 5 (Index 3+4)
           S[1:]           = Erstes Element weglassen (Index 0)
           S[:-1]          = Letztes Element weglassen (Index -1)
           S[1:-1]         = Erstes + letztes Element weglassen (Index 0,-1)
           S[::-1]         = Alle Elemente in umgekehrter Reihenfolge
           S[0:5:-1]       = Erste 5 Elemente in umgek. Reihenfolge (Index 0..4)
           S[:5:-1]        = Erste 5 Elemente in umgek. Reihenfolge (Index 0..4)
           S[1:-1:-1]      = Ohne Erstes + letztes Element in umgek. Reihenfolge
           S[-5:]          = Letzte 5 Elemente (Index -5..-1)
           S[-5::-1]       = Letzte 5 Elemente in umgek. Reihenfolge (Index -5..-1)


Immutable Sequenzen
-------------------
Folgende Operation wird von Immutable Sequenzen (str, tuple, bytes) zusÃ¤tzlich
zu obigen unterstÃ¼tzt (S ist eine immutable Sequenz):

ACHTUNG: Die Sequenz S bleibt unverÃ¤ndert!

+--------------------+-------------------------------------------------------+
| Operation          | Bedeutung                                             |
+--------------------+-------------------------------------------------------+
| hash(S)            | Hashwert erstellen (eindeutig)                        |
+--------------------+-------------------------------------------------------+


Mutable Sequenzen
-----------------
Folgende Operationen werden von mutable Sequenzen (list, bytearray) zusÃ¤tzlich
zu den obigen Sequenz-Operationen unterstÃ¼tzt (S ist eine mutable Sequenz, T
ist eine Sequenz oder ein Iterator des gleichen Typs, I, J, K, N sind
Ganzzahlen und E ist ein beliebiges Objekt, das die BeschrÃ¤nkungen der Sequenz
S erfÃ¼llt):

ACHTUNG: Sequenz S wird verÃ¤ndert (bleibt aber das gleiche Objekt)!

ACHTUNG: Operationen += *= sind auf String + Tupel anwendbar,
         erzeugen aber NEUEN String / Tupel!

+--------------------+-------------------------------------------------------+
| Operation          | Bedeutung                                             |
+--------------------+-------------------------------------------------------+
| S[I] = E           | Element an Position I durch E ersetzen                |
| S[I:J] = T         | Elemente von I..J-1 durch Inhalt von Iterable T ersetz.|
| del s[I:J]         | Elemente von I..J-1 aus Liste entfernen               |
| S[I:J:K] = T       | Elemente s[I:J:K] durch Inhalt von Iterable T ersetzen|
| del s[I:J:K]       | Elemente s[I:J:K] aus Liste entfernen                 |
+--------------------+-------------------------------------------------------+
| S.append(E)        | E am Sequenz-Ende anhÃ¤ngen:      S[len(S):len(S)] = [E]|
| S.extend(T)  s += T| Elemente von T anhÃ¤ngen:           S[len(S):len(S)] = T|
| S.insert(I, E)     | Element E an Position I einfÃ¼gen:          s[I:I] = [E]|
+--------------------+-------------------------------------------------------+
| E = S.pop([I])     | Element an Position I (STD: 0) entfernen und zurÃ¼ck   |
| S.remove(E)        | Erstes Element mit Wert E entfernen (Suche!)          |
+--------------------+-------------------------------------------------------+
| S *= N             | Sequenz N-mal vervielfachen + verketten               |
| S.reverse()        | Reihenfolge der Elemente umdrehen                     |
| S.clear()          | Sequenz leeren (alle Elemente entfernen)       del S[:]|
| S2 = S.copy()      | Shallow/Flache Kopie erzeugen:                     S[:]|
```

+--------------------+--+

HINWEIS: Indexwerte I und J sowie Schrittweite K dÃ¼rfen auch negativ sein
(auch gemischt).

list-Sequenz

Folgende Operation wird von Listen zusÃ¤tzlich zu den allgemeinen
Sequenz-Operationen und den Operationen fÃ¼r mutable Sequenzen unterstÃ¼tzt:

ACHTUNG: Die Liste L wird verÃ¤ndert (bleibt aber das gleiche Objekt)!

Operation	Bedeutung
L.sort(key,reverse)	Elemente aufsteigend sortieren (gemÃ¤ÃM-^_ Vergleich "<") (key=MAPFUNC, reverse=True --> absteigend)

HINWEIS: Die Formulierung "gemÃ¤ÃM-^_ Vergleich <" bedingt, dass ALLE Elemente per
Operator "<" miteinander vergleichbar sind. D.h. entweder alle Elemente sind
Zahlen (bool, int, float, complex, Decimal, Fraction) oder alle Elemente haben
den gleichen Datentyp str, tupel, list, ...