(C) 2025 OSTC GmbH (http://www.ostc.de)


```
Python Mengen (set und frozenset)           (C) 2020-2021 T.Birnthaler OSTC GmbH
=================================

Doku --> http://docs.python.org/3/library/stdtypes.html#set

Mengen (set und frozenset) speichern nur eindeutige Keys (UNIQUE), diese mÃ¼ssen
IMMUTABLE Objekte (NoneType, bool, int, float, complex, str, tuple, bytes) bzw.
eigentlich HASHABLE Objekte sein. D.h. der gleiche KEY darf nur 1x vorkommen.

  s1 = set({})                              # Leere Menge, NICHT {} = leeres dict
  s2 = {"abc", True, 123, -3.14, (1,2,3)}  # 5 Elemente
  s3 = {1, 2, 3}                            # 3 Elemente

Sie lassen sich als "degeneriertes" Dictionary betrachten, bei denen zum KEY
der VALUE fehlt, d.h. die Zugriffsgeschwindigkeit auf einen bestimmten Key ist
unabhÃ¤ngig von der Set-GrÃ¶ÃM-^_e immer gleich schnell (RANDOM ACCESS).

Von Sets (mutable) + Frozensets (immutable) unterstÃ¼tzte Operationen (S, S2, T
sind Sets, K ist ein Key):

ACHTUNG: Nur lesende Operationen, das Set S bleibt unverÃ¤ndert!

+-----------------------------------+------------------------------+
| len(S)                            | Anzahl Elemente              |
| S2 = S.copy()                     | Shallow/Flache Kopie         |
+-----------------------------------+------------------------------+
| K in S                            | Ist Element enthalten?       |
| K not in S                        | Ist Element nicht enthalten? |
+---------------------------+--------+------------------------------+
| S.isdisjoint(T)           |        | Schnittmenge leer?           |
| S.issubset(T)             | S <= T | Ist Teilmenge (unecht)?      |
|                           | S < T  | Ist echte Teilmenge?         |
| S.issuperset(T)           | S >= T | Ist Obermenge (unecht)?      |
|                           | S > T  | Ist echte Obermenge?         |
+---------------------------+--------+------------------------------+
| S.union(T)                | S | T  | Vereinigungsmenge            |
| S.intersection(T)         | S & T  | Schnittmenge                 |
| S.difference(T)           | S - T  | Differenzmenge               |
| S.symmetric_difference(T) | S ^ T  | Symmetrische Differenzmenge  |
+---------------------------+--------+------------------------------+

Nur von Sets (mutable) unterstÃ¼tzte Operationen (S, T sind Sets, K ist ein Key):

ACHTUNG: Schreibende Operationen,
         das Set S wird verÃ¤ndert (bleibt aber das gleiche Objekt)!

ACHTUNG: Operationen |= &= -= ^= sind auf Frozenset anwendbar,
         erzeugen aber NEUEN Frozenset!

+-----------------------------------+--------------------------------------------+
| S.add(K)                          | Element hinzufÃ¼gen                         |
| S.remove(K)                       | Element entfernen (oder KeyError)          |
| S.discard(K)                      | Element entfernen (kein KeyError)          |
| S.pop()                           | Bel. Element entfernen + zurÃ¼ck (KeyError)|
+-----------------------------------+--------+-----------------------------------+
| S.update(T)                       | S |= T | Vereinigungsmenge zuweisen        |
| S.intersection_update(T)          | S &= T | Schnittmenge zuweisen             |
| S.difference_update(T)            | S -= T | Differenzmenge zuweisen           |
| S.symmetric_difference_update(T)  | S ^= T | Symmetrische Differenzmenge zuw.  |
+-----------------------------------+--------+-----------------------------------+
| S.clear()                         | Set leeren (alle Elemente entfernen)       |
+-----------------------------------+--------------------------------------------+

HINWEIS: Bei Zugriff auf eine Menge mit einem nicht vorhanden Key wird
         die Exception "KeyError" ausgelÃ¶st.

HINWEIS: Seit Python 3.7 entspricht die Element-Reihenfolge im Set der
EinfÃ¼ge-Reihenfolge und bleibt bei Operationen auf dem Set erhalten. Vorher war
die Reihenfolge der Elemente ZUFÃM-^DLLIG und konnte sich jederzeit Ã¤ndern.
```