Python Mengen (set und frozenset) (C) 2020 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 (None, bool, int, float, complex, str, tuple) bzw. eigentlich "HASHABLE" Objekte sein. D.h. der gleiche Key darf nur 1x vorkommen).

```
  s1 = set({})                                  # Leere Menge
  s2 = {"abc", True, 123, -3.14, (1,2,3)}
  s3 = {1, 2, 3}
```

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

Von Sets (mutable) + Frozensets (immutable) unterstÃ¼tzte Operationen:

len(S)		Anzahl Elemente
S2 = S.copy()		Shallow/Flace 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?
S2 = S.union(T)	S | T	Vereinigungsmenge
S2 = S.intersection(T)	S & T	Schnittmenge
S2 = S.difference(T)	S - T	Differenzmenge
S2 = S.symmetric_difference(T)	S ^ T	Symmetrische Differenzmenge

Nur von Sets (mutable) unterstÃ¼tzte Operationen:

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: Seit Python 3.7 entspricht die Elementreihenfolge im Set der EinfÃ¼gereihenfolge, vorher war die Reihenfolge der Elemente zufÃ¤llig und konnte sich jederzeit Ã¤ndern.