Sets in Python

In Python sind Sets eine Sammlung von eindeutigen Elementen. Sie bieten verschiedene Methoden, um mit diesen Sammlungen zu arbeiten. Hier sind einige der wichtigsten Set-Methoden:

1. **add(elem)**: Fügt ein Element zum Set hinzu. Wenn das Element bereits vorhanden ist, wird es nicht erneut hinzugefügt.

```
my_set = {1, 2, 3}
my_set.add(4)
# my_set ist jetzt {1, 2, 3, 4}
```

2. **remove(elem)**: Entfernt ein bestimmtes Element aus dem Set. Wenn das Element nicht vorhanden ist, wird ein `KeyError` ausgelöst.

```
my_set = {1, 2, 3}
my_set.remove(2)
# my_set ist jetzt {1, 3}
```

3. **discard(elem)**: Entfernt ein bestimmtes Element aus dem Set, falls es vorhanden ist. Wenn das Element nicht vorhanden ist, passiert nichts.

```
my_set = {1, 2, 3}
my_set.discard(2)
# my_set ist jetzt {1, 3}
```

4. **clear()**: Entfernt alle Elemente aus dem Set, sodass es leer wird.

```
my_set = {1, 2, 3}
my_set.clear()
# my_set ist jetzt {}
```

5. **pop()**: Entfernt und gibt ein zufälliges Element aus dem Set zurück. Wenn das Set leer ist, wird ein `KeyError` ausgelöst.

```
my_set = {1, 2, 3}
element = my_set.pop()
# element könnte 1, 2 oder 3 sein, my_set hat jetzt ein Element weniger
```

6. **union(other_set)**: Gibt die Vereinigung des Sets mit einem anderen Set zurück, d.h. alle Elemente, die in mindestens einem der Sets vorhanden sind.

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
union_set = set1.union(set2)
# union_set ist {1, 2, 3, 4, 5}
```

7. **intersection(other_set)**: Gibt die Schnittmenge des Sets mit einem anderen Set zurück, d.h. alle Elemente, die in beiden Sets vorhanden sind.

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
intersection_set = set1.intersection(set2)
# intersection_set ist {3}
```

8. **difference(other_set)**: Gibt die Differenz des Sets mit einem anderen Set zurück, d.h. alle Elemente, die im ersten Set, aber nicht im zweiten Set vorhanden sind.

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
difference_set = set1.difference(set2)
# difference_set ist {1, 2}
```

9. **symmetric_difference(other_set)**: Gibt die symmetrische Differenz des Sets mit einem anderen Set zurück, d.h. alle Elemente, die in genau einem der Sets vorhanden sind.

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
sym_diff_set = set1.symmetric_difference(set2)
# sym_diff_set ist {1, 2, 4, 5}
```

10. **issubset(other_set)**: Gibt `True` zurück, wenn das Set eine Teilmenge des anderen Sets ist, d.h. alle Elemente des Sets sind auch im anderen Set vorhanden.

```
set1 = {1, 2}
set2 = {1, 2, 3}
is_subset = set1.issubset(set2)
# is_subset ist True
```

11. **issuperset(other_set)**: Gibt `True` zurück, wenn das Set eine Obermenge des anderen Sets ist, d.h. alle Elemente des anderen Sets sind auch im Set vorhanden.

```
set1 = {1, 2, 3}
set2 = {1, 2}
is_superset = set1.issuperset(set2)
# is_superset ist True
```

12. **isdisjoint(other_set)**: Gibt `True` zurück, wenn das Set und das andere Set keine gemeinsamen Elemente haben.

```
set1 = {1, 2}
set2 = {3, 4}
is_disjoint = set1.isdisjoint(set2)
# is_disjoint ist TrueDiese Methoden ermöglichen es, effizient mit Sets zu arbeiten und
# verschiedene Operationen durchzuführen, die in der Mengenlehre üblich sind.
```