Quicksort-Implementierung, rekursiv.

```
def quicksort(objekte):
    """Diese Funktion gibt eine sehr knappe Implementation des
    Quicksort-Algorithmus: Das Argument objekte sollte eine Liste
    (oder ein anderer Iterator) von Objekten sein, die paarweise
    vergleichbar sind (für die also die Operatoren '>' und '<='
    definiert sind), der Rückgabewert ist die sortierte Liste der
    Objekte."""
    # Illustration: Eine Liste oder ein Tupel ist
    # zwar an sich kein Wahrheitswert ("True" oder "False"),
    # aber: Eine Liste ist im Zusammenhang einer if-Bedingung
    # "falsy", wenn sie leer ist, sonst "truthy".
    if not objekte:
        # Für eine leere Liste ist natürlich nichts weiter zu tun
        return []
    # Illustration von _Unpacking_: Das erste Element von
    # objekte wird der variablen trenner zugewiesen, die restlichen
    # Elemente bilden die _neue_ Liste objekte:
    trenner, *objekte = objekte
    # Die restlichen Elemente werden in zwei Teile geteilt, je
    # nachdem ob sie kleinergleich oder größer
    kleinere = [o for o in objekte if o <= trenner]
    groessere = [o for o in objekte if o > trenner]
    # Rekursiver Aufruf der Funktion:
    return quicksort(kleinere) + [trenner] + quicksort(groessere)
```