Zerlegung einer Punktwolke entlang einer Koordinatenachse.

```
def split_cloud_of_points(cloud, subset, axis):
    """Gegeben sei eine "Punktwolke" von m d-dimensionalen Punkten
    in Form einer d x m Matrix (numpy-array) cloud, und eine
    Teilmenge dieser "Punktwolke" in Form eines eindimensionalen
    numpy-arrays (dtype=boolean) subset, das die (Spalten-)Indizes der
    Punkte der Teilmenge subset codiert (im Sinne einer charakteristischen
    Funktion der Teilmenge): Die Funktion liefert eine
    Zerlegung der Teilmenge gemäß dem _Median_ der Projektion
    auf die Koordinaten-Achse axis.
    """

    # Projektion: Koordinaten-Achse axis; das bool-Array subset
    # "wählt daraus die Spalten aus, die zu subset gehören":
    coords = cloud[axis][subset]

    # coords kann im allgemeinen "Doubletten" (also gleiche Elemente)
    # enthalten: Die numpy-Funktion "unique" returns the sorted unique
    # elements of an array.
    # Hier wird also ("im Hintergrund") sortiert - O(n log(n)), und es
    # werden "Doubletten" entfernt:
    unique_sorted_coords = np.unique(coords)

    nof_unique_coords = len(unique_sorted_coords)
    if nof_unique_coords <= 1:
        # Hier gibt's nichts mehr zu "zerschneiden":
        return unique_sorted_coords[0], subset, np.zeros(
            cloud.shape[1],
            dtype=int
        )

    # Implicit else:

    # Bestimme den Median der Koordinaten (ohne Wiederholungen!):
    median = unique_sorted_coords[(nof_unique_coords+1)//2-1]

    # Auch das folgende wird mit numpy sehr einfach:
    # "Im Hintergrund" wird das ganze array durchlaufen - O(n) -;
    # wir erhalten damit die charakteristische Funktion aller
    # Spalten, deren axis-Koordinate <= median ist:
    boolean_array_indicating_left_block = (cloud[axis] <= median)

    # Erzeuge zwei neue Blöcke (als _Kopien_ der entsprechenden Zerlegung):
    # numpy logical_and ist eine _koordinatenweise_ UND-Verknüpfung; für
    # die charakteristischen Funktionen von Teilmengen entspricht diese
    # logische Operation der Durchschnittsbildung:
    left_or_lower_block = np.copy(
        np.logical_and(
            subset,
            boolean_array_indicating_left_block
        )
    )
    # numpy logical_not "flippt" jede bool-Eintragung: True->False, und
    # vice versa; für die charakteristischen Funktionen von Teilmengen
    # entspricht diese logische Operaton der Komplementbildung:
    right_or_upper_block = np.copy(
        np.logical_and(
            subset,
            np.logical_not(boolean_array_indicating_left_block)
        )
    )

    # Gib Median und Blöcke zurück:
    return median, left_or_lower_block, right_or_upper_block
```