

SOURCE CODE 1. Inkrementeller Algorithmus zur Bestimmung der konvexen Hülle.

```
def convex_hull(lop):  
    """Bestimme die Eckpunkte des Polygons, das die konvexe Hülle der  
    Punktliste lop darstellt (geordnet im mathematisch positiven  
    Umlaufsinn): Berechne getrennt den oberen Teil und den unteren Teil  
    der konvexen Hülle."""  
    # Wandle die Liste in eine_lexikographisch geordnete_  
    # (len(lop) x 2) - Matrix um, als erstes einfach: Mache eine  
    # (len(lop) x 2) - Matrix aus der Liste:  
    dummy = np.array(lop)  
    # Als zweites: Transponiere diese Matrix  
    dummy_T = dummy.T  
    # Drittens: Bestimme jene _Permutation der Indices_, die die  
    # Zeilen der transponierten Matrix in lexikographische Ordnung  
    # bringt:  
    indices = np.lexsort((dummy_T[1], dummy_T[0]))  
    # Damit erhalten wir die Punkte (Zeilen der Matrix) in  
    # lexikographischer Ordnung:  
    lop_array = dummy[indices]  
  
    # Bestimme obere/untere Hälfte der konvexen Hülle (also des  
    # Polygons), von links nach rechts:  
    upper_points = [lop_array[0], lop_array[1]]  
    lower_points = [lop_array[0], lop_array[1]]  
    # Längen dieser "Hälften"  
    upl = 2  
    lpl = 2  
    # (D.h.: Die Vereinigungsmenge von upper_points und lower_points  
    # entspricht der Menge der Eckpunkte des Polygons; der linkeste  
    # und der rechteste Punkt in upper_points und lower_points stimmen  
    # überein, also upper_points[0] == lower_points[0] und  
    # upper_points[-1] == lower_points[-1].)  
  
    # For-Schleife startet beim dritten Punkt in lop_array (Index 2) ...  
    for p in lop_array[2:]:  
        # Innerhalb der upper_points: Suche den _am weitesten links  
        # liegenden_ Punkt q, sodass alle Punkte x _rechts_ von q  
        # _unterhalb_ der Verbindungsgeraden p,q liegen. Dass dazu  
        # die folgende einfache for-Schleife genügt, muß man sich  
        # kurz geometrisch überlegen!  
        for k,(q,x) in enumerate(  
            zip(  
                upper_points[-2::-1],  
                upper_points[-1::-1]  
            )  
        ):  
            if side_of_line(q, p, x) > 0:  
                k = k-1  
                break
```

```

upper_points = upper_points[:upl-k-1]
upper_points += [p]
upl = len(upper_points)

# Innerhalb der lower points: Suche den _am weitesten links
# liegenden_ Punkt q, sodass alle Punkte x _rechts_ von q
# _oberhalb_ der Verbindungsgeraden p,q liegen. Dass dazu die
# folgende einfache for-Schleife genügt, muß man sich (wie
# zuvor) kurz geometrisch überlegen!
for k,(q,x) in enumerate(
    zip(
        lower_points[-2::-1],
        lower_points[-1::-1]
    )
):
    if side_of_line(q, p, x) < 0:
        k = k-1
        break

lower_points = lower_points[:lpl-k-1]
lower_points += [p]
lpl = len(lower_points)

return upper_points[:-1]+[ppp for ppp in lower_points[1:][::-1]]

```