Source Code 1. Das Voronoi–Diagramm für eine Punktwolke liefert auch den „Rand“ der konvexen Hülle: Dazu sammeln wir einfach die Kanten des Voronoi–Diagramms, die *Strahlen* (also *nicht Strecken*!) entsprechen und ordnen Punkte der Wolke, die diesen Kanten entsprechen, in einem Kreis (im graphentheoretischen Sinn) an. Die hier gezeigte Implementierung dieser Idee passt zu der Konstruktion des Voronoi–Diagramms als `dict` in `make_voronoi_diagram`.

```
def convex_hull_voronoi(voronoi):
    # Die keys (Paare von Indices von Punkten) im so berechneten
    # Voronoi-Diagramm, die _nur einen_ zugeordneten Endpunkt haben,
    # bestimmen die Punkte, die den Rand der konvexen Hülle der
    # Punktwolke bilden:
    convex_hull_edges = [
        set(key) for key, value in voronoi.items() if len(value) < 2
    ]
    # Diese Liste von zweielementigen Teilmengen von Indices können
    # wir als Kanten in einem Graphen interpretieren, dessen Knoten
    # die _Indices_ der Punkte am Rand der Punktwolke sind, für die
    # das Voronoi-Diagramm berechnet wurde.
    # Diese Knoten wollen wir aber noch in die richtige Reihenfolge
    # bringen, sodass die so geordneten Knoten einem Kreis im Graphen
    # entsprechen):
    current_edge = convex_hull_edges[0]
    # Ok: Die Knoten dieser Kante liegen auf dem gesuchten Kreis:
    convex_hull_points = list(current_edge)
    # Diese Kante entfernen wir aus der Liste:
    convex_hull_edges.remove(current_edge)
    # Jetzt vervollständigen wir den Kreis:
    v0,v1 = current_edge
    while convex_hull_edges:
        for edge in convex_hull_edges:
            if v1 in edge:
                # "edge" enthält Knoten v1 ...
                convex_hull_edges.remove(edge)
                # ... und mit dem _anderen_ Knoten setzen wir fort:
                v1 = list(edge.difference(set([v1])))[0]
                convex_hull_points.append(v1)
    return convex_hull_points
```