Finde Zusammenhangskomponenten in einem einfachen Graphen.

```python
def get_components(am):
    """Finde die Zusammenhangskomponenten des Graphen mit
    (quadratischer und symmetrischen!) Adjazenzmatrix am;
    retourniere diese in einer Liste."""
    # Anzahl der Knoten:
    n = am.shape[0]
    # Liste der bereits gefundenen Zusammenhangskomponenten
    components = []
    # Menge der (Indizes von) Knoten, die noch nicht auf
    # Zusammenhangskomponenten verteilt wurden:
    vertices = set(range(n))
    # Solange vertices nicht leer ist:
    while vertices:
        # Wir suchen eine neue Komponente:
        new_component = set()
        # Wir geben "schichtenweise Knoten" zur neuen Komponente,
        # die "innerste Schicht" ist einpunktig:
        current_layer = set([min(vertices)])
        # Solange current_layer nicht leer ist:
        while current_layer:
            # Die "neue Schicht" kommt zur neuen Komponente dazu ...
            new_component.update(current_layer)
            # ... und wird von der Knotenmenge weggenommen:
            vertices.difference_update(current_layer)
            # Jetzt suchen wir die "nächste Schicht":
            # print(components, new_component, vertices)
            new_layer = set()
            # Mit der itertools-Funktion "product" durchlaufen wir alle
            # möglichen Kanten und schauen, ob die im Graphen vorhanden
            # sind:
            for i,j in itertools.product(current_layer, vertices):
                if am[i][j]:
                    new_layer.add(j)
            current_layer = new_layer
        # Nach der inneren while-Schleife speichern wir new_component:
        components+= [new_component]
    # Nach der äußeren while-Schleife retournieren wir die Komponenten:
    return components
```

Für einen (einfachen) Graphen, der durch seine Adjazenzmatrix gegeben ist, finden wir durch "direktes Suchen" seine Komponenten und retournieren diese als Liste von (Knoten-)Mengen.