```
class Graph:
    def __init__(self):
        self.graph = {}

    def add_edge(self, u, v):
        self.graph.setdefault(u, []).append(v)
        self.graph.setdefault(v, []).append(u)

    def dfs(self, node, visited, component):
        visited[node] = True
        component.append(node)

        for neighbor in self.graph.get(node, []):
            if not visited[neighbor]:
                self.dfs(neighbor, visited, component)

    def connected_components(self):
        visited = {node: False for node in self.graph}
        components = []

        for node in self.graph:
            if not visited[node]:
                component = []
                self.dfs(node, visited, component)
                components.append(component)

        return components

# Example Usage:
if __name__ == "__main__":
    # Create a sample graph
    g = Graph()
```

```
g.add_edge(0, 1)
g.add_edge(0, 2)
g.add_edge(1, 2)
g.add_edge(3, 4)

# Find connected components
components = g.connected_components()

print("Connected Components:")
for i, component in enumerate(components, 1):
    print(f"Component {i}: {component}")
```