Source Code 1. Knoten für Doubly Linked Lists (doppelt verkettete Listen).

```
# Ein "Hilfs-Dictionary" für eine "Spiegelung"
# (Vertauschung von links und rechts), die sich hier
# als nützlich erweist:
mirror = {'left' : 'right', 'right' : 'left'}

class DLLNode:
    """A node to be used in a doubly linked list"""
    def __init__(self, data):
        # Store the data
        self.contents = data
        # Initially, the "pointers" to the left/right neighbours
        # are None
        self.neighbour = {'left' : None, 'right' : None}

    def __str__(self):
        """String representation of the node"""
        return f'{self.contents} {self.neighbour}'

    # "Hineinquetschen" eines neuen Knotens, links oder rechts
    # von self:
    def squeeze_in(self, what, where):
        """Insert a new node (containing "what") to side "where".
        """
        # Make the new node containing "what":
        new_node =  DLLNode(what)
        # Self's neighbour on side "where" becomes new node`s neighbour:
        new_node.neighbour[where] = self.neighbour[where]
        if self.neighbour[where]:
            self.neighbour[where].neighbour[mirror[where]] = new_node
        # Make this new node self's neighbour on side "where":
        self.neighbour[where] = new_node
        new_node.neighbour[mirror[where]] = self
        return new_node

    # Entfernen von self aus seiner "Nachbarschaft":
    def detach(self):
        """Remove self."""
        # Nachbarn von self:
        left, right = [self.neighbour[where] for where in ["left", "right"]]
        if left:
            left.neighbour["right"] = right
        if right:
            right.neighbour["left"] = left
        return self
```