```
# 1. Klasse
class Node:
    nextid = 0
    def __init__(self, data):
        self.id = Node.nextid
        self.data = data
        self.next = None
        Node.nextid += 1

    def __str__(self):
        return "Node({}, {})".format(
            self.data, self.next.id if self.next else None)


# 2. Traversieren
def traverse(head):
    curr = head
    while curr:
        print(curr)
        curr = curr.next


# 3. Zum Schluss hinzufuegen
def add(head, data):
    if head:
        curr = head
        while curr.next:
            curr = curr.next
        curr.next = Node(data)
    else:
        # Liste leer, daher neuen head anlegen
        head = Node(data)
    return head


# 4. Suchen nach Daten
def search(head, data):
    curr = head
    while curr:
        if curr.data == data:
            break
        curr = curr.next
    return curr


# 4a. Liefert Index von gesuchtem Datum
def indexOf(head, data):
    curr = head
    idx = 0
    while curr:
        if curr.data == data:
            break
        idx += 1
        curr = curr.next
    if curr:
        # gefunden
        return idx
    else:
        # nicht gefunden
        return -1


# 5. LÃ¤nge bestimmen
def size(head):
    curr = head
    cnt = 0
    while curr:
        curr = curr.next
```



```
        cnt += 1
    return cnt


# 6. EinfÃ¼gen an beliebiger Stelle
def insert(head, data, idx):
    curr = head
    if curr:
        if idx == 0:
            # Am Anfang einfÃ¼gen
            head = Node(data)
            head.next = curr
        else:
            i = 1
            while curr.next and i < idx:
                curr = curr.next
                i += 1
            if i != idx:
                raise ValueError("wrong index")
            # curr zeigt auf Position davor!
            tmp = curr
            curr = Node(data)
            curr.next = tmp.next
            tmp.next = curr
    else:
        # Liste leer
        if idx == 0:
            # Index ok, daher als head einfÃ¼gen
            head = Node(data)
        else:
            raise ValueError("wrong index")

    return head



# ohne Fehlerueberpruefungen:
#   - idx != 0 und Liste leer => am Anfang einfuegen
#   - idx > Laenge der Liste => am Ende anhaengen
# Es muss gelten: idx >= 0
def insert2(head, data, idx):
    curr = head
    if curr and idx != 0:
        i = 1
        while curr.next and i < idx:
            curr = curr.next
            i += 1
        # curr zeigt auf Position davor!
        tmp = curr.next
        curr.next = Node(data)
        curr.next.next = tmp

        # Alternative ohne ".next.next"
        #tmp = curr
        #curr = Node(data)
        #curr.next = tmp.next
        #tmp.next = curr
    else:
        # Am Anfang einfÃ¼gen, da entweder Liste leer oder idx == 0
        head = Node(data)
        head.next = curr

    return head


# 7. LÃ¶schen eines Knotens
def remove(head, data):
    curr = head
    prev_curr = None
```

```
    while curr and curr.data != data:
        prev_curr = curr
        curr = curr.next
    if curr:
        # gefunden
        if prev_curr:
            prev_curr.next = curr.next
        else:
            head = curr.next
    return head


if __name__ == "__main__":
    head = None

    print("Suchen nach otto in leerer Liste:", search(head, "otto"))
    print()

    print("Index von otto in leerer Liste:", indexOf(head, "otto"))
    print()

    print("Liste aufbauen durch Anhaengen von 3 Knoten (aaa, bbb, ccc)!")
    head = add(head, "aaa")
    head = add(head, "bbb")
    head = add(head, "ccc")
    print()

    print("Index von aaa:", indexOf(head, "aaa"))
    print("Index von bbb:", indexOf(head, "bbb"))
    print("Index von ccc:", indexOf(head, "ccc"))
    print("Index von otto:", indexOf(head, "otto"))
    print()

    print("Traversieren!")
    traverse(head)
    print()

    print("LÃ¤nge bestimmen:", size(head))
    print()

    print("Suchen nach Knoten mit aaa:", search(head, "aaa"))
    print("Suchen nach Knoten mit bbb:", search(head, "bbb"))
    print("Suchen nach Knoten mit ccc:", search(head, "ccc"))
    print("Suchen nach nicht existenten Knoten:", search(head, "xxx"))
    print()

    print("Neue Liste anlegen!")
    head = None
    print()

    print("LÃ¤nge bestimmen:", size(head))
    print()

    print("Versuchen Knoten mit nicht existentem Index in "
          "leerer Liste anlegen:")
    try:
        head = insert(head, "zzz", 9)
    except ValueError:
        print("  ValueError beim Einfuegen mit Index 9 in leerer Liste")
    print()

    print("Einfuegen von 3 Knoten jeweils an Index 0 (bbb, ddd, fff)!")
    head = insert2(head, "bbb", 0)
    head = insert2(head, "ddd", 0)
    head = insert2(head, "fff", 0)
    print()

    print("Traversieren!")
    traverse(head)
```



```
    print()

    print("Einfuegen von einem Knoten an Index 1 (eee)!")
    head = insert2(head, "eee", 1)
    print()

    print("Traversieren!")
    traverse(head)
    print()

    print("Einfuegen von einem Knoten an vorletzter Position! (ccc)")
    head = insert2(head, "ccc", 3)
    print()

    print("Traversieren!")
    traverse(head)
    print()

    print("Einfuegen von einem Knoten an letzter Position! (aaa)")
    head = insert2(head, "aaa", 5)
    print()

    print("Traversieren!")
    traverse(head)
    print()

    print("Einfuegen von einem Knoten an ungÃ¼ltigem Index:")
    try:
        head = insert(head, "XXX", 7)
    except ValueError:
        print("  ValueError beim Einfuegen mit Index 7")

    print("LÃ¶schen des ersten Knotens! (fff)")
    head = remove(head, "fff")

    print("Traversieren!")
    traverse(head)
    print()

    print("LÃ¶schen eines inneren Knotens! (ccc)")
    head = remove(head, "ccc")

    print("Traversieren!")
    traverse(head)
    print()

    print("LÃ¶schen des letzten Knotens! (aaa)")
    head = remove(head, "aaa")

    print("Traversieren!")
    traverse(head)
    print()

    print("LÃ¶schen eines nicht existenten Knotens! (xxx)")
    head = remove(head, "xxx")

    print("Traversieren!")
    traverse(head)
    print()
```