Bagian 1: Kelas Node (untuk membuat simpul pada linked list)

```
# Node class untuk double linked-list
class Node:
```

- Kita membuat **kelas Node** untuk merepresentasikan **satu simpul** dalam *double linked list* (daftar yang bisa maju dan mundur).

```
  def __init__(self, data):
    self.data = data
    self.prev = None
    self.next = None
```

- Fungsi __init__ adalah **konstruktor**, otomatis dijalankan saat objek dibuat.
- self.data = data: Menyimpan nilai yang diberikan ke node.
- self.prev = None: Mengarah ke node sebelumnya (awalnya kosong).
- self.next = None: Mengarah ke node selanjutnya (awalnya kosong).

Bagian 2: Kelas DoubleLinkedList

```
# Double Linked List class
class DoubleLinkedList:
```

- Ini adalah kelas utama untuk membuat dan mengelola **daftar berantai ganda**.

```
  def __init__(self):
    self.head = None
```

- Konstruktor untuk memulai linked list dengan **kepala (head) kosong**.

Fungsi append (menambahkan node di akhir)

```
  def append(self, data):
    new_node = Node(data)
```

- Membuat simpul baru (new_node) dengan data yang diberikan.

```
    if self.head is None:
      self.head = new_node
      return
```

- Jika list masih kosong, langsung jadikan new_node sebagai kepala.

```
    cur = self.head
```

```
        while cur.next:
            cur = cur.next
```

- Kalau tidak kosong, kita **cari node terakhir** dengan while.

```
= new_node
        new_node.prev = cur
```

- Sambungkan new_node ke node terakhir, dan sebaliknya.

Fungsi display (menampilkan isi list)

```
    def display(self):
        cur = self.head
        while cur:
            print(cur.data, end=" <-> " if cur.next else "\n")
            cur = cur.next
```

- Mulai dari kepala (cur = self.head), selama masih ada node:
    - Tampilkan cur.data.
    - Jika ada node setelahnya, tampilkan <->, kalau tidak, ganti baris.
    - Pindah ke node selanjutnya (cur = cur.next).

Fungsi delete_first (hapus node pertama)

```
    def delete_first(self):
        if self.head is None:
            print("List kosong")
            return
```

- Jika list kosong, tampilkan pesan dan keluar.

```
        print(f"Menghapus node awal: {self.head.data}")
        self.head = self.head.next
        if self.head:
            self.head.prev = None
```

- Hapus node pertama dengan memindahkan head ke node berikutnya.
- Jika masih ada node, set prev dari kepala baru ke None.

Fungsi delete_last (hapus node terakhir)

```
def delete_last(self):
    if self.head is None:
        print("List kosong")
        return
```

- Cek apakah list kosong.

```
    cur = self.head
    if cur.next is None:
        print(f"Menghapus node terakhir: {cur.data}")
        self.head = None
        return
```

- Jika hanya ada satu node, hapus dengan mengatur head ke None.

```
    while cur.next:
        cur = cur.next
```

- Kalau lebih dari satu, cari node terakhir.

```
    print(f"Menghapus node terakhir: {cur.data}")
    cur.prev.next = None
```

- Putuskan sambungan node terakhir dari node sebelumnya.

Fungsi delete_by_value (hapus berdasarkan nilai)

```
def delete_by_value(self, value):
    if self.head is None:
        print("List kosong")
        return
```

- Jika list kosong, tampilkan pesan.

```
    cur = self.head
    while cur:
        if cur.data == value:
```

- Cari node yang punya nilai value.

```
            print(f"Menghapus node dengan nilai: {value}")
            if cur.prev:
                cur.prev.next = cur.next
            else:
```

```
            self.head = cur.next
```

- Kalau node ada di tengah atau akhir: hubungkan node sebelumnya dengan node setelahnya.
- Kalau node yang dihapus adalah kepala: pindahkan head.

```
        if cur.next:
            cur.next.prev = cur.prev
        return
```

- Atur sambungan balik dari node berikutnya.
- Hentikan pencarian setelah node dihapus.

```
    cur = cur.next
print(f"Data {value} tidak ditemukan dalam list.")
```

- Jika tidak ditemukan sampai akhir, tampilkan pesan.

Contoh Penggunaan:

```
dll = DoubleLinkedList()
dll.append(10)
dll.append(20)
dll.append(30)
dll.append(40)
```

- Membuat objek dll dari DoubleLinkedList.
- Menambahkan 4 angka ke list.

```
print("Isi awal:")
dll.display()
```

- Menampilkan isi list: 10 <-> 20 <-> 30 <-> 40

```
dll.delete_first()
dll.display()
```

- Hapus node pertama (10), tampilkan list.

```
dll.delete_last()
dll.display()
```

- Hapus node terakhir (40), tampilkan list.

```
dll.delete_by_value(20)
dll.display()
```

- Hapus node yang bernilai 20, tampilkan list.

dll.delete_by_value(99)

- Coba hapus data yang tidak ada (99), tampilkan pesan bahwa data tidak ditemukan