PENJELASAN DARI CODING TERSEBUT :

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

- Membuat kelas Node yang digunakan untuk menyimpan data pada masing-masing elemen (node) dari **double linked list**.
- self.data: menyimpan data aktual.
- self.prev: menunjuk ke node sebelumnya (kiri).
- self.next: menunjuk ke node selanjutnya (kanan).

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

- Membuat kelas DoubleLinkedList untuk mengelola linked list.
- self.head menyimpan referensi ke node pertama dari list.

```
def append(self, data):
    new_node = Node(data)
    if self.head is None:
        self.head = new_node
        return
    cur = self.head
    while cur.next:
        cur = cur.next
    cur.next = new_node
    new_node.prev = cur
```

- append digunakan untuk menambahkan node di akhir.
- Jika list kosong (head adalah None), new_node jadi head.
- Jika tidak, cari node terakhir (cur.next is None) lalu hubungkan node baru di akhir.

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

- Menampilkan seluruh data dalam list.
- Setiap data dipisahkan dengan " <-> " jika ada node setelahnya.

```
def delete_first(self):
    if self.head is None:
        print("List kosong")
        return
    print(f"Menghapus node awal: {self.head.data}")
    self.head = self.head.next
    if self.head:
        self.head.prev = None
```

- Menghapus node pertama.
- Jika list kosong, tampilkan pesan.
- Ubah head ke node kedua.
- Jika node kedua ada, hapus referensi ke node pertama (prev = None).

```
def delete_last(self):
    if self.head is None:
        print("List kosong")
        return
    cur = self.head
    if cur.next is None:
        print(f"Menghapus node terakhir: {cur.data}")
        self.head = None
        return
    while cur.next:
        cur = cur.next
    print(f"Menghapus node terakhir: {cur.data}")
    cur.prev.next = None
```

- Menghapus node terakhir.
- Jika list kosong, tampilkan pesan.
- Jika hanya ada satu node, kosongkan list (head = None).
- Jika lebih, cari node terakhir dan hilangkan koneksi dari node sebelumnya ke node terakhir.

```
def delete_by_value(self, value):
    if self.head is None:
        print("List kosong")
        return
    cur = self.head
    while cur:
        if cur.data == value:
            print(f"Menghapus node dengan nilai: {value}")
```

```
            if cur.prev:
                cur.prev.next = cur.next
            else:
                self.head = cur.next
            if cur.next:
                cur.next.prev = cur.prev
            return
        cur = cur.next
    print(f"Data {value} tidak ditemukan dalam list.")
```

- Menghapus node berdasarkan data yang dicari (value).
- Jika node ditemukan:
- Jika node punya prev, sambungkan prev.next ke cur.next.
- Jika node adalah head, ubah head ke node selanjutnya.
- Jika node punya next, sambungkan next.prev ke cur.prev.
- Jika tidak ditemukan, tampilkan pesan bahwa data tidak ada.

dll = DoubleLinkedList()

- Membuat objek dll dari kelas DoubleLinkedList.

dll.append(10)

dll.append(20)

dll.append(30)

dll.append(40)

- Menambahkan empat node ke list: 10, 20, 30, 40.

print("Isi awal:")

- Menampilkan teks bahwa isi list akan ditampilkan.

dll.display()

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

dll.delete_first()

- Menghapus node pertama (10). Head sekarang adalah 20.

dll.display()

- Menampilkan list: 20 <-> 30 <-> 40.

last()

- Menghapus node terakhir (40).

dll.display()

- Menampilkan list: 20 <-> 30.

dll.delete_by_value(20)

- Menghapus node dengan nilai 20.

dll.display()

- Menampilkan list: 30.

dll.delete_by_value(99)

- Mencoba menghapus node dengan nilai 99, tapi tidak ada. Muncul pesan: "Data 99 tidak ditemukan dalam list."