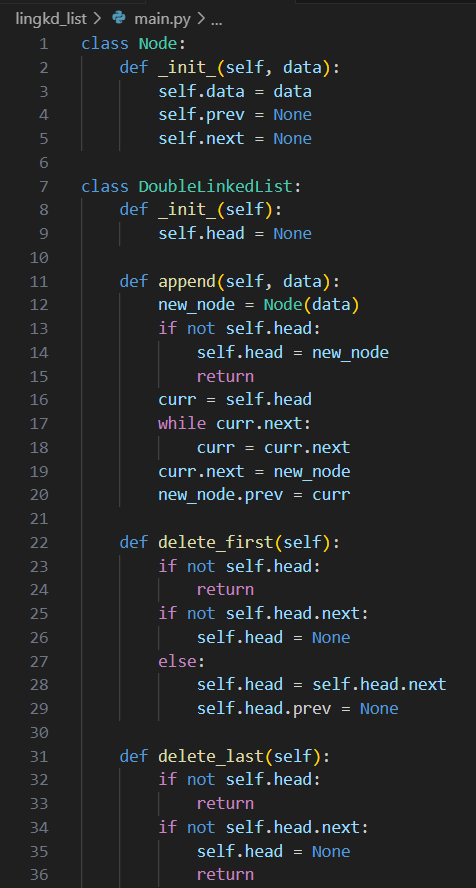
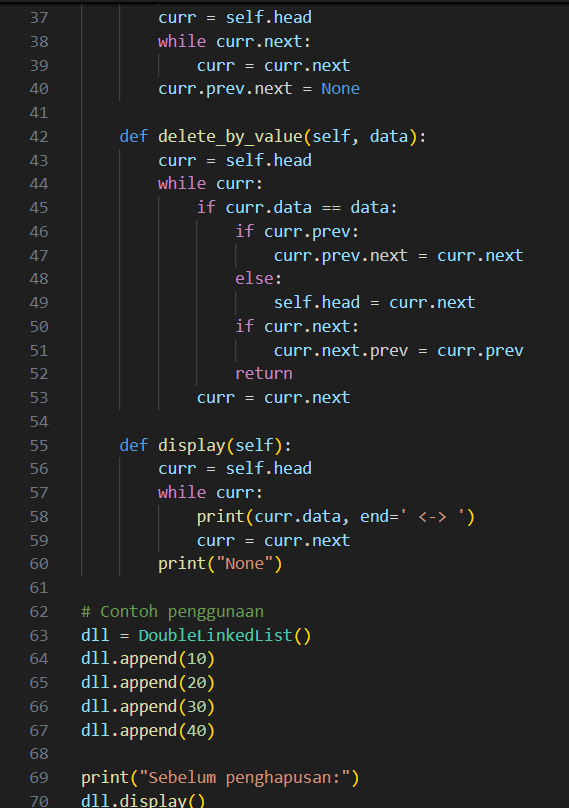
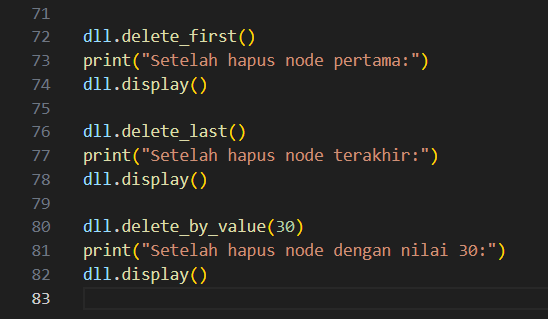
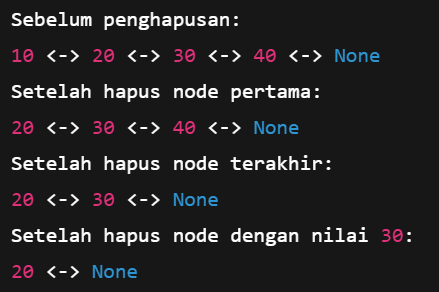
LINGKED LIST







HASIL



PENJELASAN

class Node:: Mendefinisikan kelas Node untuk menyimpan elemen dalam linked list.

def \_init\_(self, data):: Konstruktor yang dipanggil saat objek Node dibuat.

self.data = data: Menyimpan data dalam node.

self.prev = None: Penunjuk ke node sebelumnya (untuk double linked list).

self.next = None: Penunjuk ke node selanjutnya.

class DoubleLinkedList:: Kelas untuk linked list ganda (doubly linked list).

self.head = None: Awalnya list kosong, jadi kepala (head) di-set ke None.

def append(self, data):

new\_node = Node(data)

Membuat node baru dengan data yang diberikan.

if not self.head:

self.head = new\_node

return

Jika linked list kosong, node baru jadi head, dan keluar dari metode.

curr = self.head

while curr.next:

curr = curr.next

Jika list tidak kosong, iterasi ke node terakhir.

curr.next = new\_node

new\_node.prev = curr

Hubungkan node terakhir ke node baru, dan node baru menunjuk kembali ke node sebelumnya.

def delete\_first(self):

if not self.head:

return

Jika list kosong, keluar dari metode.

if not self.head.next:

self.head = None

Jika hanya satu node, kosongkan list.

else:

self.head = self.head.next

self.head.prev = None

Jika ada lebih dari satu node, pindahkan head ke node berikutnya dan putuskan hubungan ke node sebelumnya

def delete\_last(self):

if not self.head:

return

Jika list kosong, keluar.

if not self.head.next:

self.head = None

return

Jika hanya satu node, kosongkan list.

curr = self.head

while curr.next:

curr = curr.next

curr.prev.next = None

Iterasi ke node terakhir, lalu putuskan hubungannya dari node sebelumnya.

def delete\_by\_value(self, data):

curr = self.head

while curr:

Mulai dari head dan iterasi semua node.

if curr.data == data:

Jika ditemukan node dengan data yang dicari:

if curr.prev:

curr.prev.next = curr.next

Jika node bukan head, sambungkan node sebelumnya ke node sesudahnya.

else:

self.head = curr.next

Jika node adalah head, pindahkan head ke node berikutnya.

if curr.next:

curr.next.prev = curr.prev

return

curr = curr.next

Jika node setelahnya ada, hubungkan balik ke node sebelumnya. Keluar dari metode setelah penghapusan

def display(self):

curr = self.head

while curr:

print(curr.data, end=' <-> ')

curr = curr.next

print("None")

Mulai dari head, cetak semua data node dengan format data <->, sampai None.

8. Contoh Penggunaan

dll = DoubleLinkedList()

dll.append(10)

dll.append(20)

dll.append(30)

dll.append(40)

Membuat linked list dan menambahkan 4 node: 10 <-> 20 <-> 30 <-> 40

print("Sebelum penghapusan:")

dll.display()

Menampilkan isi list sebelum penghapusan.

dll.delete\_first()

print("Setelah hapus node pertama:")

dll.display()

Menghapus node pertama (10), hasil: 20 <-> 30 <-> 40

dll.delete\_last()

print("Setelah hapus node terakhir:")

dll.display()

Menghapus node terakhir (40), hasil: 20 <-> 30

dll.delete\_by\_value(30)

print("Setelah hapus node dengan nilai 30:")

dll.display()

Menghapus node dengan nilai 30, hasil: 20