

## Tugas penghapusan double linked list

```
1 class Node:
2     def __init__(self, data):
3         self.data = data
4         self.prev = None
5         self.next = None
6
7 class DoublyLinkedList:
8     def __init__(self):
9         self.head = None
10
11 # Membuat objek double linked list
12 dll = DoublyLinkedList()
13
14 # Menambah node data 1, 2, 3, 4 ke list
15 for val in [1, 2, 3, 4]:
16     new_node = Node(val)
17     if dll.head is None:
18         dll.head = new_node
19     else:
20         # cari node terakhir
21         current = dll.head
22         while current.next is not None:
23             current = current.next
24         current.next = new_node
25         new_node.prev = current
26
27 # Mencetak isi list sebelum penghapusan
28 print("List sebelum penghapusan:")
29 current = dll.head
30 while current is not None:
31     print(current.data, end=" <-> ")
32     current = current.next
33 print("None")
34
35 # Menghapus node pertama
36 if dll.head is not None:
37     if dll.head.next is None:
38         dll.head = None
39     else:
40         dll.head = dll.head.next
41         dll.head.prev = None
42
43 # Mencetak isi list setelah menghapus node pertama
44 print("List setelah menghapus node pertama:")
45 current = dll.head
46 while current is not None:
47     print(current.data, end=" <-> ")
48     current = current.next
49 print("None")
50
```

```

51 # Menghapus node terakhir
52 if dll.head is not None:
53     if dll.head.next is None:
54         dll.head = None
55     else:
56         current = dll.head
57         while current.next is not None:
58             current = current.next
59         if current.prev:
60             current.prev.next = None
61
62 # Mencetak isi list setelah menghapus node terakhir
63 print("List setelah menghapus node terakhir:")
64 current = dll.head
65 while current is not None:
66     print(current.data, end=" <-> ")
67     current = current.next
68 print("None")
69
70 # Menghapus node berdasarkan nilai data = 2
71 val_to_delete = 2
72 current = dll.head
73 while current is not None:
74     if current.data == val_to_delete:
75         if current.prev is not None:
76             current.prev.next = current.next
77         else:
78             # node yang dihapus adalah head
79             dll.head = current.next
80         if current.next is not None:
81             current.next.prev = current.prev
82         break
83     current = current.next
84
85 # Mencetak isi list setelah menghapus node dengan nilai 2
86 print("List setelah menghapus node dengan nilai 2:")
87 current = dll.head
88 while current is not None:
89     print(current.data, end=" <-> ")
90     current = current.next
91 print("None")
92

```

## Outputnya

List sebelum penghapusan:

1 <-> 2 <-> 3 <-> 4 <-> None

List setelah menghapus node pertama:

2 <-> 3 <-> 4 <-> None

List setelah menghapus node terakhir:

2 <-> 3 <-> None

List setelah menghapus node dengan nilai 2:

3 <-> None

Penjelasannya

**Baris 1:** Mendefinisikan kelas Node sebagai struktur data dasar untuk elemen dalam double linked list.

**Baris 2:** Mendefinisikan konstruktor `__init__` untuk kelas Node.

**Baris 3:** Menyimpan nilai yang diterima dalam `self.data`.

**Baris 4:** Mengatur referensi `prev` dan `next` menjadi `None`.

**Baris 6:** Mendefinisikan kelas `DoublyLinkedList` sebagai wadah untuk menyimpan beberapa node.

**Baris 7:** Mendefinisikan konstruktor `__init__` untuk kelas tersebut.

**Baris 8:** Inisialisasi `self.head` menjadi `None`, yang berarti list masih kosong.

**Baris 10:** Membuat objek `dll` dari kelas `DoublyLinkedList`.

**Baris 12:** Memulai perulangan untuk menambahkan data 1, 2, 3, 4 ke dalam linked list.

**Baris 13:** Membuat objek `new_node` dari kelas Node dengan nilai `val`.

**Baris 14:** Mengecek apakah list masih kosong (`dll.head is None`).

**Baris 15:** Jika list kosong, `new_node` menjadi node pertama (`head`).

**Baris 16:** Jika list tidak kosong, node akan ditambahkan di akhir.

**Baris 17:** Inisialisasi `current` ke `dll.head` untuk mulai pencarian node terakhir.

**Baris 18:** Perulangan untuk mencari node terakhir (`current.next is not None`).

**Baris 19:** Berpindah ke node berikutnya.

**Baris 20:** Menambahkan node baru di akhir, dan menghubungkannya dua arah.

**Baris 23:** Menampilkan teks "List sebelum penghapusan".

**Baris 24:** Inisialisasi variabel `current` dengan `dll.head`.

**Baris 25:** Melakukan perulangan untuk mencetak isi list.

**Baris 26:** Mencetak nilai data node saat ini.

**Baris 27:** Berpindah ke node berikutnya.

**Baris 28:** Mencetak "None" sebagai penanda akhir list.

**Baris 31:** Mengecek apakah list tidak kosong sebelum menghapus node pertama.

**Baris 32:** Jika hanya ada satu node, langsung hapus.

**Baris 33:** Mengosongkan list jika hanya satu node.

**Baris 34:** Jika lebih dari satu node, lanjut ke penghapusan.

**Baris 35:** Geser `head` ke node kedua (`dll.head.next`).

**Baris 36:** Set `prev` dari node baru menjadi `None`.

**Baris 39:** Menampilkan teks "List setelah menghapus node pertama".

**Baris 40:** Inisialisasi ulang `current` ke `dll.head`.

**Baris 41–43:** Loop untuk mencetak isi list.

**Baris 44:** Mencetak "None" di akhir.

**Baris 46:** Mengecek apakah list tidak kosong sebelum menghapus node terakhir.

**Baris 47:** Jika hanya ada satu node, langsung kosongkan list.

**Baris 48:** Set `dll.head = None`.

**Baris 49:** Jika lebih dari satu node, cari node terakhir.

**Baris 50:** Inisialisasi `current` ke `dll.head`.

**Baris 51:** Cari node terakhir dengan perulangan.

**Baris 52:** Berpindah ke node berikutnya.

**Baris 53:** Cek apakah ada node sebelumnya.

**Baris 54:** Putuskan hubungan dengan node terakhir (`current.prev.next = None`).

**Baris 57:** Menampilkan teks "List setelah menghapus node terakhir".

**Baris 58–61:** Melakukan pencetakan isi list.

**Baris 62:** Cetak "None".

**Baris 64:** Menetapkan nilai yang ingin dihapus (`val_to_delete = 2`).

**Baris 65:** Inisialisasi `current` ke `dll.head`.

**Baris 66:** Mulai loop untuk mencari node dengan data 2.

**Baris 67:** Jika data cocok...

**Baris 68:** Jika node bukan head, sambungkan `prev.next` ke `next`.

**Baris 69:** Hapus hubungan dari node sebelumnya ke node saat ini.

**Baris 70:** Jika node adalah head, geser head ke node berikutnya.

**Baris 71:** Lanjutkan proses penghapusan.

**Baris 72:** Jika bukan node terakhir, sambungkan `next.prev` ke node sebelumnya.

**Baris 73:** Menyambung node selanjutnya ke node sebelumnya.

**Baris 74:** Keluar dari loop setelah penghapusan.

**Baris 75:** Jika data belum cocok, lanjut ke node berikutnya.

**Baris 77:** Menampilkan teks "List setelah menghapus node dengan nilai 2".

**Baris 78–81:** Melakukan pencetakan isi list.

**Baris 82:** Cetak "None".