

Tugas Modul 2:

```
1 class Node:
2     def __init__(self, data):
3         self.data = data
4         self.prev = None
5         self.next = None
6
7
8 # Membuat linked list kosong
9 head = None
10
11 # Fungsi append di-inline tanpa def
12 data_to_add = [1, 2, 3, 4]
13
14 for data in data_to_add:
15     new_node = Node(data)
16     if head is None:
17         head = new_node
18     else:
19         current = head
20         while current.next:
21             current = current.next
22         current.next = new_node
23         new_node.prev = current
24
25 def print_list(head):
26     current = head
27     while current:
28         print(current.data, end=" <-> ")
29         current = current.next
30     print("None")
31
```

```
32 print("List sebelum penghapusan:")
33 print_list(head)
34
35 # Menghapus node awal (head)
36 if head is not None:
37     if head.next is None:
38         head = None
39     else:
40         head = head.next
41         head.prev = None
42
43 print("List setelah menghapus node pertama:")
44 print_list(head)
45
46 # Menghapus node akhir
47 if head is not None:
48     if head.next is None:
49         head = None
50     else:
51         current = head
52         while current.next:
53             current = current.next
54         # current adalah node terakhir
55         if current.prev:
56             current.prev.next = None
57
58 print("List setelah menghapus node terakhir:")
59 print_list(head)
60
61 # Menghapus node berdasarkan nilai tertentu (misal nilai 2)
62 value = 2
63 current = head
```

```
64 while current:
65     if current.data == value:
66         # Jika node yang dihapus adalah head
67         if current == head:
68             head = current.next
69             if head:
70                 head.prev = None
71         else:
72             if current.prev:
73                 current.prev.next = current.next
74             if current.next:
75                 current.next.prev = current.prev
76             break
77         current = current.next
78
79 print("List setelah menghapus node dengan nilai 2:")
80 print_list(head)
81
```

a. Penjelasa setiap baris:

1. Class node: sebuah class Node yang akan digunakan untuk membuat node dalam linked list.
def __init__(self, data): : Mendefinisikan method constructor untuk class Node, yang akan dipanggil ketika sebuah node dibuat.
self.data = data : Mengatur atribut data dari node dengan nilai yang diberikan. Atribut data ini digunakan untuk menyimpan nilai yang akan disimpan dalam node.
self.prev = None : Mengatur atribut prev dari node dengan nilai None. Atribut prev ini digunakan untuk menunjuk ke node sebelumnya dalam linked list.
self.next = None : Mengatur atribut next dari node dengan nilai None. Atribut next ini digunakan untuk menunjuk ke node berikutnya dalam linked list.
2. Membuat Linked List Kosong: - head = None : Membuat variabel head yang akan digunakan untuk menunjuk ke node pertama dalam linked list, dan mengaturnya dengan nilai None.
3. Fungsi Append di inline: bekerja dengan cara yang sama seperti fungsi append biasa. membuat node baru dan menambahkannya ke akhir linked list.
 - data_to_add = [1, 2, 3, 4] : Membuat list data yang akan ditambahkan ke dalam linked list.
 - for data in data_to_add: : Melakukan loop untuk setiap data dalam list data_to_add.
 - new_node = Node(data) : Membuat node baru dengan data yang diberikan.
 - if head is None: : Mengecek apakah linked list masih kosong (head adalah None).
 - head = new_node : Jika linked list kosong, maka node baru menjadi head.
 - else: : Jika linked list tidak kosong, maka node baru akan ditambahkan ke akhir linked list.
 - current = head : Membuat variabel current yang akan digunakan untuk menunjuk ke node saat ini dalam linked list.
 - while current.next: : Melakukan loop sampai node terakhir dalam linked list.
 - current = current.next : Mengatur current ke node berikutnya.
 - current.next = new_node : Mengatur next dari node terakhir ke node baru.
 - new_node.prev = current : Mengatur prev dari node baru ke node terakhir.
4. Menghapus node awal (head): proses menghapus node pertama dalam linked list. Ketika node awal dihapus, maka node berikutnya akan menjadi node awal yang baru.
 - if head is not None: : Mengecek apakah linked list tidak kosong.
 - if head.next is None: : Mengecek apakah linked list hanya memiliki satu node.
 - head = None : Jika linked list hanya memiliki satu node, maka head diatur ke None.
 - else: : Jika linked list memiliki lebih dari satu node.
 - head = head.next : Mengatur head ke node berikutnya.
 - head.prev = None : Mengatur prev dari node baru menjadi None.

5. Fungsi print list: sebuah fungsi yang digunakan untuk mencetak isi dari linked list. Fungsi ini memungkinkan kita untuk melihat data yang ada dalam linked list.

- def print_list(head): : Mendefinisikan fungsi print_list yang akan digunakan untuk mencetak linked list.
- current = head : Membuat variabel current yang akan digunakan untuk menunjuk ke node saat ini dalam linked list.
- while current: : Melakukan loop sampai node terakhir dalam linked list.
- print(current.data, end=" <-> ") : Mencetak data dari node saat ini.
- current = current.next : Mengatur current ke node berikutnya.
- print("None") : Mencetak None untuk menandakan akhir linked list.

6. Menghapus node akhir: proses menghapus node terakhir dalam linked list. Ketika node akhir dihapus, maka node sebelumnya menjadi node terakhir yang baru.

- if head is not None: : Mengecek apakah linked list tidak kosong.
- if head.next is None: : Mengecek apakah linked list hanya memiliki satu node.
- head = None : Jika linked list hanya memiliki satu node, maka head diatur ke None.
- else: : Jika linked list memiliki lebih dari satu node.
- current = head : Membuat variabel current yang akan digunakan untuk menunjuk ke node saat ini dalam linked list.
- while current.next: : Melakukan loop sampai node terakhir dalam linked list.
- current = current.next : Mengatur current ke node berikutnya.
- if current.prev: : Mengecek apakah node terakhir memiliki node sebelumnya.
- current.prev.next = None : Mengatur next dari node sebelumnya menjadi None.

7. Menghapus node berdasarkan nilai: proses menghapus node yang memiliki nilai tertentu dalam linked list. Ketika node dengan nilai tertentu dihapus, maka node sebelumnya dan node berikutnya dihubungkan kembali.

- value = 2 : Membuat variabel value yang akan digunakan untuk mencari node dengan nilai tertentu.
- current = head : Membuat variabel current yang akan digunakan untuk menunjuk ke node saat ini dalam linked list.
- while current: : Melakukan loop sampai node terakhir dalam linked list.
- if current.data == value: : Mengecek apakah data dari node saat ini sama dengan nilai yang dicari.
- if current == head: : Mengecek apakah node yang dihapus adalah head.
- head = current.next : Mengatur head ke node berikutnya.
- if head: : Mengecek apakah head tidak None.
- head.prev = None : Mengatur prev dari node baru menjadi None.
- else: : Jika node yang dihapus bukan head.
- if current.prev: : Mengecek apakah node yang dihapus memiliki node sebelumnya.

- `current.prev.next = current.next` : Mengatur next dari node sebelumnya ke node berikutnya.

b. output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python +v [icon] ... ^ X

List setelah menghapus node dengan nilai 2:
3 <-> None
PS D:\Pre pratikum\Modul 2> & C:/Users/ASUS/AppData/Local/Programs/Python/Python313/python.exe "d:/Pre pratikum/Modul 2/tugas.py"
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
PS D:\Pre pratikum\Modul 2>

```

Penjelasa setiap baris:

1. Class node: sebuah class Node yang akan digunakan untuk membuat node dalam linked list.
`def __init__(self, data):` : Mendefinisikan method constructor untuk class Node, yang akan dipanggil ketika sebuah node dibuat.
`self.data = data` : Mengatur atribut data dari node dengan nilai yang diberikan. Atribut data ini digunakan untuk menyimpan nilai yang akan disimpan dalam node.
`self.prev = None` : Mengatur atribut prev dari node dengan nilai None. Atribut prev ini digunakan untuk menunjuk ke node sebelumnya dalam linked list.
`self.next = None` : Mengatur atribut next dari node dengan nilai None. Atribut next ini digunakan untuk menunjuk ke node berikutnya dalam linked list.
2. Membuat Linked List Kosong: - `head = None` : Membuat variabel head yang akan digunakan untuk menunjuk ke node pertama dalam linked list, dan mengaturnya dengan nilai None.
3. Fungsi Append di inline: bekerja dengan cara yang sama seperti fungsi append biasa. membuat node baru dan menambahkannya ke akhir linked list.
 - `data_to_add = [1, 2, 3, 4]` : Membuat list data yang akan ditambahkan ke dalam linked list.
 - `for data in data_to_add:` : Melakukan loop untuk setiap data dalam list data_to_add.
 - `new_node = Node(data)` : Membuat node baru dengan data yang diberikan.
 - `if head is None:` : Mengecek apakah linked list masih kosong (head adalah None).
 - `head = new_node` : Jika linked list kosong, maka node baru menjadi head.

- else: : Jika linked list tidak kosong, maka node baru akan ditambahkan ke akhir linked list.
- current = head : Membuat variabel current yang akan digunakan untuk menunjuk ke node saat ini dalam linked list.
- while current.next: : Melakukan loop sampai node terakhir dalam linked list.
- current = current.next : Mengatur current ke node berikutnya.
- current.next = new_node : Mengatur next dari node terakhir ke node baru.
- new_node.prev = current : Mengatur prev dari node baru ke node terakhir.

4. Menghapus node awal (head): proses menghapus node pertama dalam linked list.
Ketika node awal dihapus, maka node berikutnya akan menjadi node awal yang baru.

- if head is not None: : Mengecek apakah linked list tidak kosong.
- if head.next is None: : Mengecek apakah linked list hanya memiliki satu node.
- head = None : Jika linked list hanya memiliki satu node, maka head diatur ke None.
- else: : Jika linked list memiliki lebih dari satu node.
- head = head.next : Mengatur head ke node berikutnya.
- head.prev = None : Mengatur prev dari node baru menjadi None.

5. Fungsi print list: sebuah fungsi yang digunakan untuk mencetak isi dari linked list.
Fungsi ini memungkinkan kita untuk melihat data yang ada dalam linked list.

- def print_list(head): : Mendefinisikan fungsi print_list yang akan digunakan untuk mencetak linked list.
- current = head : Membuat variabel current yang akan digunakan untuk menunjuk ke node saat ini dalam linked list.
- while current: : Melakukan loop sampai node terakhir dalam linked list.
- print(current.data, end=" <-> ") : Mencetak data dari node saat ini.
- current = current.next : Mengatur current ke node berikutnya.
- print("None") : Mencetak None untuk menandakan akhir linked list.

6. Menghapus node akhir: proses menghapus node terakhir dalam linked list.
Ketika node akhir dihapus, maka node sebelumnya menjadi node terakhir yang baru.

- if head is not None: : Mengecek apakah linked list tidak kosong.
- if head.next is None: : Mengecek apakah linked list hanya memiliki satu node.
- head = None : Jika linked list hanya memiliki satu node, maka head diatur ke None.
- else: : Jika linked list memiliki lebih dari satu node.
- current = head : Membuat variabel current yang akan digunakan untuk menunjuk ke node saat ini dalam linked list.
- while current.next: : Melakukan loop sampai node terakhir dalam linked list.
- current = current.next : Mengatur current ke node berikutnya.
- if current.prev: : Mengecek apakah node terakhir memiliki node sebelumnya.
- current.prev.next = None : Mengatur next dari node sebelumnya menjadi None.

7. Menghapus node berdasarkan nilai: proses menghapus node yang memiliki nilai tertentu dalam linked list.

Ketika node dengan nilai tertentu dihapus, maka node sebelumnya dan node berikutnya dihubungkan kembali.

- value = 2 : Membuat variabel value yang akan digunakan untuk mencari node dengan nilai tertentu.
- current = head : Membuat variabel current yang akan digunakan untuk menunjuk ke node saat ini dalam linked list.
- while current: : Melakukan loop sampai node terakhir dalam linked list.
- if current.data == value: : Mengecek apakah data dari node saat ini sama dengan nilai yang dicari.
- if current == head: : Mengecek apakah node yang dihapus adalah head.
- head = current.next : Mengatur head ke node berikutnya.
- if head: : Mengecek apakah head tidak None.
- head.prev = None : Mengatur prev dari node baru menjadi None.
- else: : Jika node yang dihapus bukan head.
- if current.prev: : Mengecek apakah node yang dihapus memiliki node sebelumnya.
- current.prev.next = current.next : Mengatur next dari node sebelumnya ke node berikutnya.

d. Penjelasan output:

1. List sebelum penghapusan: 1 <-> 2 <-> 3 <-> 4 <-> None:

Ini adalah kondisi awal linked list sebelum penghapusan.

Linked list memiliki 4 node dengan nilai 1, 2, 3, dan 4.

Node terakhir memiliki next yang bernilai None, menandakan akhir linked list.

2. List setelah menghapus node pertama: 2 <-> 3 <-> 4 <-> None:

Node pertama dengan nilai 1 telah dihapus dari linked list.

Node kedua dengan nilai 2 menjadi node pertama yang baru.

Linked list sekarang memiliki 3 node dengan nilai 2, 3, dan 4.

3. List setelah menghapus node terakhir: 2 <-> 3 <-> None

Node terakhir dengan nilai 4 telah dihapus dari linked list.

Node dengan nilai 3 menjadi node terakhir yang baru.