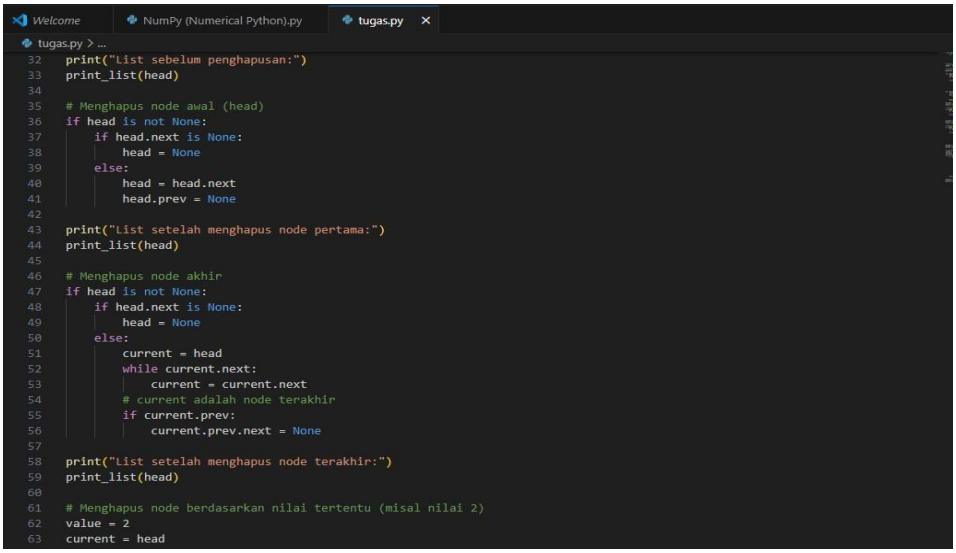
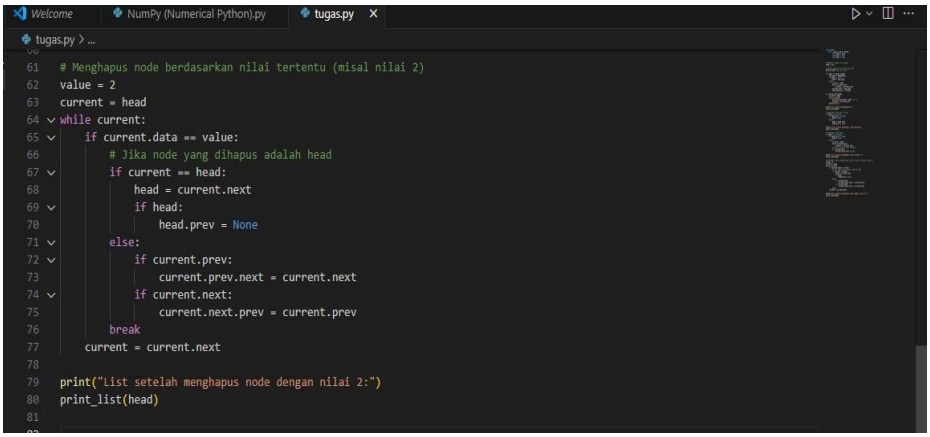
Tugas Modul 2:

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


# Membuat linked list kosong
head = None

# Fungsi append di-inline tanpa def
data_to_add = [1, 2, 3, 4]

for data in data_to_add:
    new_node = Node(data)
    if head is None:
        head = new_node
    else:
        current = head
        while current.next:
            current = current.next
        current.next = new_node
        new_node.prev = current

def print_list(head):
    current = head
    while current:
        print(current.data, end=" <-> ")
        current = current.next
    print("None")
```





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. Menghaous 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:**

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. Menghaous 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.