

NAMA : Ruhasdi
NIM : 24241016
MATKUL : STRUKTUR DATA

Praktek 1

main.py

+

```
1 # impor library numpy
2 import numpy as np
3
4 # membuat array dengan numpy
5 nilai_siswa = np.array([85, 55, 40, 90])
6
7 # akses data pada array
8 print(nilai_siswa[3])
```

Outputnya :

90

```
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Penjelasannya :

Baris 1:

bahwa baris berikutnya akan mengimpor library NumPy, yang digunakan untuk operasi dan array.

Baris 2:

Mengimpor library NumPy dan memberi alias np supaya lebih ringkas saat digunakan. Setelah Bisa menggunakan np.array() untuk membuat array, bukan menulis numpy.array().

Baris 3:

menjelaskan bahwa baris di bawah akan membuat array menggunakan NumPy.

Baris 4:

Membuat sebuah array NumPy berisi nilai siswa: 85, 55, 40, dan 90. Array ini disimpan dalam Variable.

Penjelasan outputnya :

Program mencetak 90 ke layar karena itu adalah nilai pada indeks ke-3 dari array cara perhitunganya dari nol mulai dari indeks tersebut maka hasil akhirnya 90.

Praktek 2

```
# impor library numpy
import numpy as np

# membuat array dengan numpy
nilai_siswa_1 = np.array([75, 65, 45, 80])
nilai_siswa_2 = np.array([[85, 55, 40], [50, 40, 99]])

# cara akses elemen array
print(nilai_siswa_1[0])
print(nilai_siswa_2[1][1])

# mengubah nilai elemen array
nilai_siswa_1[0] = 88
# mengubah nilai elemen array
nilai_siswa_1[0] = 88
nilai_siswa_2[1][1] = 70

# cek perubahannya dengan akses elemen array
print(nilai_siswa_1[0])
print(nilai_siswa_2[1][1])

# Cek ukuran dan dimensi array
print("Ukuran Array : ", nilai_siswa_1.shape)
print("Ukuran Array : ", nilai_siswa_2.shape)
print("Dimensi Array : ", nilai_siswa_2.ndim)
```

Outputnya :

```
75
40
88
70
Ukuran Array : (4,)
Ukuran Array : (2, 3)
Dimensi Array : 2
```

Penjelasannya :

Baris 1

yang menunjukkan bahwa baris selanjutnya akan mengimpor library NumPy

Baris 2

Mengimpor library NumPy dan memberinya alias np agar lebih ringkas saat digunakan dalam Kode.

Baris 3

bahwa kita akan membuat array menggunakan NumPy.

Baris 4

Membuat array 1 dimensi dengan 4 elemen, lalu disimpan ke variabel nilai_siswa_1.

Baris 5

Membuat array 2 dimensi (seperti matriks 2x3) yang disimpan dalam nilai_siswa_2.

Baris 6

Yang menandai bahwa kita akan mengakses nilai-nilai dalam array.

Baris 7

Menampilkan elemen pertama dari nilai_siswa_1, yaitu 75.

Baris 8

Menampilkan baris ke-2, kolom ke-2 dari nilai_siswa_2, yaitu 40.

Baris 9

bahwa kita akan mengubah isi array.

Baris 10

Mengubah elemen pertama dari nilai_siswa_1 menjadi 88

Baris 11

Mengubah elemen baris ke-2, kolom ke-2 dari nilai_siswa_2 menjadi 70.

Baris 12

bahwa kita akan melihat apakah perubahan berhasil.

Baris 13

Menampilkan elemen pertama dari nilai_siswa_1 yang sekarang sudah diubah menjadi 88.

Baris 14

Menampilkan nilai pada nilai_siswa_2[1][1] yang sekarang menjadi 70.

Baris 15

bahwa kita akan mengecek bentuk dan dimensi array.

Baris 16

Menampilkan ukuran (jumlah elemen per dimensi) dari nilai_siswa_1.

Hasil: (4,) → array 1 dimensi dengan 4 elemen.

Baris 17

Menampilkan ukuran dari nilai_siswa_2.

Hasil: (2, 3) → array 2 dimensi, 2 baris dan 3 kolom.

Baris 18

Menampilkan jumlah dimensi dari nilai_siswa_2, yaitu 2 (karena bentuknya seperti tabel/baris-Kolom).

Praktek 3

```
# impor library numpy
import numpy as np

# membuat array
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

# menggunakan operasi penjumlahan pada 2 array
print(a + b)      # array([5, 7, 9])

# Indexing dan Slicing pada Array
arr = np.array([10, 20, 30, 40])
print(arr[1:3])   # array([20, 30])
```

Outputnya :

```
[5 7 9]
[20 30]
10
20
30
40
```

Penjelasannya:**Baris 1**

Komentar yang menjelaskan bahwa library NumPy akan diimpor.

Baris 2

Mengimpor library NumPy dan memberi alias np supaya lebih singkat saat digunakan.

Baris 3

bahwa kita akan membuat array NumPy.

Baris 4

Membuat array a dengan elemen [1, 2, 3].

Baris 5

Membuat array b = np.array([4, 5, 6])

Baris 6

bahwa kita akan melakukan penjumlahan antar array

Baris 7

Menambahkan array a dan b secara elemen (element-wise):

[1+4, 2+5, 3+6] → [5, 7, 9].

Baris 8

bahwa baris berikut akan menunjukkan teknik mengambil sebagian isi array.

Baris 9

Membuat array arr dengan 4 elemen: [10, 20, 30, 40]

Baris 10

Mengambil elemen dari indeks ke-1 hingga sebelum ke-3 (slicing):

arr[1:3] → [20, 30].

Baris 11

bahwa kita akan melakukan iterasi (perulangan) pada elemen array

Baris 12–13

for x in arr:

 print(x)

Melakukan loop untuk mencetak setiap elemen dalam array arr.

Praktek 4**Metode Traversal**

```
# membuat array
```

```
arr = [1, 2, 3, 4, 5]
```

```
# Linear Traversal ke tiap elemen arr
```

```
print("Linear Traversal: ", end=" ")
```

```
for i in arr:
```

```
    print(i, end=" ")
```

```
print()
```

Outputnya :

```
Linear Traversal:  1 2 3 4 5
```

```
** Process exited - Return Code: 0 **
```

```
Press Enter to exit terminal
```

Penjelasannya :**Baris 1**

Komentar yang menjelaskan bahwa kamu akan membuat array (dalam bentuk list di Python, Bukan numpyarray).

Baris 2

Membuat list bernama arr yang berisi lima elemen: [1, 2, 3, 4, 5].

Baris 3

Komentar bahwa kamu akan melakukan traversal linear, yaitu mengunjungi dan memproses Elemen satu persatu dari ke kiri ke kanan.

Baris 4

Mencetak teks "Linear Traversal: " tanpa pindah ke baris baru (karena end=" " membuat kursor Tetap dibaris yang sama dan menambahkan spasi).

Baris 5-6

for i in arr:

print(i, end="

print(i, end=" ") mencetak setiap elemen diikuti oleh spasi, bukan pindah baris.

Baris 7

Mencetak baris kosong untuk pindah ke baris baru setelah selesai

Mencetak semua elemen baru.

Linear Traversal: 1 2 3 4 5

Teks "Linear Traversal: " dicetak terlebih dahulu.

Kemudian setiap elemen 1 2 3 4 5 dicetak di baris yang sama, dipisahkan oleh spasi.

Setelah selesai, baris kosong ditambahkan dengan print() untuk menjaga format tampilan.

Praktek 5

```
# membuat array
arr = [1, 2, 3, 4, 5]

# Reverse Traversal dari elemen akhir
print("Reverse Traversal: ", end="")
for i in range(len(arr) - 1, -1, -1):
    print(arr[i], end=" ")
print()
```

Outputnya :

```
Reverse Traversal: 5 4 3 2 1
```

```
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Penjelasannya :

Baris 1

Komentar bahwa kamu akan membuat array (list) di baris berikutnya.

Baris 2

Membuat list arr yang berisi lima elemen dari 1 sampai 5.

Baris 3

Komentar bahwa kamu akan mencetak elemen dari list arr secara terbalik (dari belakang Ke depan.

Baris 4

Mencetak teks "Reverse Traversal: " tanpa pindah baris karena end="" menjaga agar output Selanjutnya dicetak dibaris yang sama.

Baris 5

len(arr) - 1 = 4 → indeks terakhir (karena jumlah elemen 5 dan indeks mulai dari 0).

-1 adalah batas akhir (exclusive) → berarti iterasi akan berhenti sebelum mencapai indeks -1,

Alias berhenti di 0

-1 adalah langkah (step) → artinya mundur satu per satu.

Jadi, range(4, -1, -1) menghasilkan urutan indeks: 4, 3, 2, 1, 0

Baris 6

Untuk setiap indeks i, ambil elemen arr[i] lalu cetak di baris yang sama, dipisahkan dengan spasi.

Baris7

Pindah ke baris baru setelah mencetak semua elemen, agar output rapi.

Penjelasan outputnya :

Reverse Traversal: 5 4 3 2 1

Program mencetak elemen dari indeks terakhir (arr[4] = 5) sampai indeks pertama (arr[0] = 1)

Secara mundur

Semuanya dicetak dalam satu baris setelah teks "Reverse Traversal: ".

Praktek 7

```
# membuat array
arr = [1, 2, 3, 4, 5]

# mendeklarasikan nilai awal
n = len(arr)
i = 0

print("Linear Traversal using while loop: ", end=" ")
# Linear Traversal dengan while
while i < n:
    print(arr[i], end=" ")
    i += 1
print()
```

Outputnya :

Linear Traversal using while loop: 1 2 3 4 5

```
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Penjelasannya :

Baris 1

bahwakamuakanmembuat array (list).

Baris 2

Membuat list arrberisi 5 elemen: [1, 2, 3, 4, 5].

Baris 3

bahwavariabel-variabelawalakandidefinisikan.

Baris 4

n menyimpanpanjang (jumlahelemen) dari array arr, yaitu 5.

Baris 5

Variabelidigunakansebagaiindeksawaluntukperulangan. Dimulaidari 0 (indekspertama Array.

Baris 6

Mencetaktekspembuka, tanpapindah baris, karena end=" " menjaga agar output berikutnya

Tetep di baris yang sama

Baris 7

akan menggunakan perulangan while untuk traversal.

Baris 8–10

Perulangan akan berjalan selama kurang dari n (panjang array).

arr[i] mencetak elemen ke-i dari array.

end=" " agar semua elemen dicetak dalam satu baris, dipisahkan spasi.

i += 1 menaikkan indeks agar pindah ke elemen berikutnya.

Loop ini mencetak 1 2 3 4 5

Baris 11

Pindah ke baris baru setelah selesai mencetak elemen array.

Penjelasan outputnya :

Linear Traversal using while loop: 1 2 3 4 5

Program menelusuri list dari elemen pertama hingga terakhir menggunakan while, dan

Mencetak semua elemen secara berurutan.

Praktek 8

```
# membuat array
arr = [1, 2, 3, 4, 5]

# mendeklarasikan nilai awal
start = 0
end = len(arr) - 1

print("Reverse Traversal using while loop: ", end=" ")
# Reverse Traversal dengan while
while start < end:

    arr[start], arr[end] = arr[end], arr[start]
    start += 1
    end -= 1
print(arr)
```

Outputnya :

```
Reverse Traversal using while loop: [5, 4, 3, 2, 1]
```

```
** Process exited - Return Code: 0 **
```

```
Press Enter to exit terminal
```

Penjelasannya :

Baris 1

membuat sebuah array (list).

Baris 2

Membuat list bernama arr berisi elemen [1, 2, 3, 4, 5].

Baris 3

menetapkan variabel awal untuk indeks traversal.

Baris 4–5

start diset ke indeks pertama (0).

end diset ke indeks terakhir ($\text{len}(\text{arr}) - 1 = 4$).

Variabel ini akan digunakan untuk menukar elemen dari ujung ke tengah.

Baris 6

Mencetak teks sebagai keterangan, tanpa pindah ke baris baru ($\text{end}=""$).

Baris 7

melakukan pembalikan isi array dengan perulangan while.

Baris 8–11

Loop akan terus berjalan selama $\text{start} < \text{end}$.

Di dalam loop:

Elemen pada posisi start dan end ditukar (swap).

Kemudian start maju ke kanan (+1) dan end mundur ke kiri (-1).

Proses ini membalik urutan elemen dari luar ke dalam.

1. $\text{start}=0, \text{end}=4$: tukar 1 dan 5 $\rightarrow [5, 2, 3, 4, 1]$

2. $\text{start}=1, \text{end}=3$: tukar 2 dan 4 $\rightarrow [5, 4, 3, 2, 1]$

3. $\text{start}=2, \text{end}=2$: kondisi $\text{start} < \text{end}$ sudah tidak terpenuhi, loop berhenti.

Baris 12

Mencetak isi array setelah dibalik. Hasil akhirnya:

[5, 4, 3, 2, 1]

Penjelasan outputnya :

Reverse Traversal using while loop: [5, 4, 3, 2, 1]

Array awal [1, 2, 3, 4, 5] dibalik urutannya menjadi [5, 4, 3, 2, 1].

Proses ini disebut reverse in-place karena dilakukan langsung di array yang sama tanpa

Membuat array baru.

Praktek 9

```
# membuat array
arr = [12, 16, 20, 40, 50, 70]

# cetak arr sebelum penyisipan
print("Array Sebelum Insertion : ", arr)

# cetak panjang array sebelum penyisipan
print("Panjang Array : ", len(arr))

# menyisipkan array di akhir elemen menggunakan .append()
arr.append(26)

# cetak arr setelah penyisipan
print("Array Setelah Insertion : ", arr)

# cetak panjang array setelah penyisipan
print("Panjang Array : ", len(arr))
```

Outputnya :

Array Sebelum Insertion : [12, 16, 20, 40, 50, 70]

Panjang Array : 6

Array Setelah Insertion : [12, 16, 20, 40, 50, 70, 26]

Panjang Array : 7

Penjelasannya :**Baris 1**

membuat array (dalam Python disebut list).

Baris 2

Membuat list arr dengan 6 elemen angka: [12, 16, 20, 40, 50, 70]

Baris 3

mencetak array sebelum elemen baru disisipkan.

Baris 4

Mencetak list arr sebelum ada perubahan:

Output:

Array Sebelum Insertion : [12, 16, 20, 40, 50, 70]

Baris 5

mencetak jumlah elemen list sebelum penambahan.

Baris 6

Menggunakan len(arr) untuk menghitung jumlah elemen, yaitu 6.

Output:

Panjang Array : 6

Baris 7

menambahkan elemen di akhir list dengan fungsi .append().

Baris 8

Menambahkan angka 26 ke akhir list arr.

List berubah menjadi: [12, 16, 20, 40, 50, 70, 26]

Baris 9

mencetak array setelah penambahan elemen.

Baris 10

Mencetak array setelah penambahan output: array setelah insertion : [12, 16, 20, 40, 50, 70, 26]

Baris 11

mencetak jumlah elemen setelah penambahan.

Baris 12

Mencetak jumlah elemen saat ini, yaitu 7.

Output:

Panjang Array : 7

Penjelasan outputnya :

Array sebelum insertion: [12, 16, 20, 40, 50, 70]

Panjang Array : 6

Array Setelah Insertion : [12, 16, 20, 40, 50, 70, 26]

Panjang Array : 7

Praktek 10

```
# membuat array
arr = [12, 16, 20, 40, 50, 70]

# cetak arr sebelum penyisipan
print("Array Sebelum Insertion : ", arr)

# cetak panjang array sebelum penyisipan
print("Panjang Array : ", len(arr))

# menyisipkan array pada tengah elemen menggunakan .insert(pos, x)
arr.insert(4, 5)

# cetak arr setelah penyisipan
print("Array Setelah Insertion : ", arr)

# cetak panjang array setelah penyisipan
print("Panjang Array : ", len(arr))
```

Outputnya :

```
Array Sebelum Insertion : [12, 16, 20, 40, 50, 70]
Panjang Array : 6
Array Setelah Insertion : [12, 16, 20, 40, 5, 50, 70]
Panjang Array : 7
```

Penjelasannya :

Baris 1

membuat array (dalam Python disebut list).

Baris 2

Membuat list arr dengan 6 elemen angka: [12, 16, 20, 40, 50, 70]

Baris 3

mencetak isi array sebelum elemen baru disisipkan.

Baris 4

Mencetak isi list arr sebelum ada perubahan:

Output:

```
Array Sebelum Insertion : [12, 16, 20, 40, 50, 70]
```

Baris 5

mencetak jumlah elemen list sebelum penambahan.

Baris 6

Menggunakan `len(arr)` untuk menghitung jumlah elemen, yaitu 6.

Output:

```
Panjang Array : 6
```

Baris 7

menambahkan elemen di akhir list dengan fungsi `.append()`.

Baris 8

Menambahkan angka 26 ke akhir list arr.

List berubah menjadi: [12, 16, 20, 40, 50, 70, 26]

Baris 9

mencetak array setelah penambahan elemen.

Baris 10

Mencetak isi array setelah penambahan output: array setelah insertion : [12, 16, 20, 40, 50, 70, 26]

Baris 11

mencetak jumlah elemen setelah penambahan.

Baris 12

Mencetak jumlah elemen saat ini, yaitu 7.

Output:

Panjang Array : 7

Penjelasan outputnya :

Array sebelum insertion: [12, 16, 20, 40, 50, 70]

Panjang Array : 6

Array Setelah Insertion : [12, 16, 20, 40, 50, 70, 26]

Panjang Array : 7

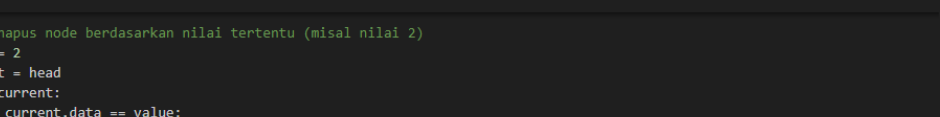
B. Tugas Modul 2:

```
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

```



The screenshot shows a Jupyter Notebook with a code cell containing Python code to delete a node from a doubly linked list. The code is as follows:

```

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

```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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>

```

a. Penjelasan 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. 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.
Linked list sekarang memiliki 2 node dengan nilai 2 dan 3.

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

Node dengan nilai 2 telah dihapus dari linked list.
Linked list sekarang hanya memiliki 1 node dengan nilai 3.
Node dengan nilai 3 menjadi node pertama dan terakhir dalam linked list.