LANJUTAN PRAKTEK

21 SAMPAI 29

Praktek 22

```python
# Praktek 22 : Membuat Single Linked-list
# function untuk membuat node
def buat_node(data):
    return {'data': data, 'next': None}

# menambahkan node di akhir list
def tambah_node(head, data):
    new_node = buat_node(data)
    if head is None:
        return new_node
    current = head
    while current['next'] is not None:
        current = current['next']
    current['next'] = new_node
    return head

# menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")

# Contoh Penerapan
# Head awal dari linked-list
head = None
```

```python
# Tambah node
head = tambah_node(head, 10)
head = tambah_node(head, 11)
head = tambah_node(head, 12)

# cetak linked-list
print('Linked-List : ')
cetak_linked_list(head)
```

Hasil ouputnya

```
PS E:\algoritma\Tugas-semester-2\Modul 2> & C:/Users/USER/AppData
Modul 2/Percobaan 4 Single Linked-List/Single Linked-List.py"
Linked-List :
Head → 10 → 11 → 12 → NULL
PS E:\algoritma\Tugas-semester-2\Modul 2>
```

Penjelasannya :

Baris 1:
Komentar judul "Praktek 22: Membuat Single Linked List", menunjukkan fokus praktik adalah implementasi struktur data linked list.

Baris 3–4:
Mendefinisikan fungsi buat_node(data) yang membuat satu node dalam bentuk dictionary dengan kunci 'data' dan 'next'.

Baris 6:
Mendefinisikan fungsi tambah_node(head, data) untuk menambahkan node baru ke akhir linked list.

Baris 7:
Memanggil fungsi buat_node untuk membuat node baru.

Baris 8–9:
Jika head bernilai None, maka linked list masih kosong, dan node baru akan langsung menjadi head.

Baris 10:
Jika tidak kosong, proses iterasi dilakukan mulai dari head.

Baris 11–13:
Melakukan perulangan hingga node terakhir ditemukan (yaitu node dengan 'next' bernilai None).

Baris 14:
Node terakhir diarahkan ke node baru sebagai sambungan.

Baris 15:
Mengembalikan head sebagai referensi awal linked list.

Baris 17:
Mendefinisikan fungsi cetak_linked_list(head) untuk menampilkan isi linked list.

Baris 18:
Memulai iterasi dari head.

Baris 19:
Mencetak label “Head → ” untuk memulai tampilan.

Baris 20–22:
Selama current tidak None, mencetak data tiap node, lalu berpindah ke node berikutnya.

Baris 23:
Mencetak “NULL” untuk menunjukkan akhir dari linked list.

Baris 25:
Inisialisasi linked list dengan head = None, artinya list masih kosong.

Baris 27–29:
Menambahkan tiga node ke dalam linked list dengan nilai 10, 11, dan 12.

Baris 31:
Mencetak label “Linked List: ” sebagai keterangan hasil akhir.

Baris 32:
Memanggil fungsi cetak_linked_list untuk menampilkan seluruh isi linked list.

Praktek 23

```
# Praktek 23 : Traversal Linked-list
# function untuk membuat node
def buat_node(data):
    return {'data': data, 'next': None}

# menambahkan node di akhir list
def tambah_node(head, data):
    new_node = buat_node(data)
    if head is None:
        return new_node
    current = head
    while current['next'] is not None:
        current = current['next']
    current['next'] = new_node
    return head

# traversal untuk cetak isi linked-list
def traversal_to_display(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")
```

```
# traversal untuk menghitung jumlah elemen dalam linked-list
def traversal_to_count_nodes(head):
    count = 0
    current = head
    while current is not None:
        count += 1
        current = current['next']
    return count

# traversal untuk mencari dimana tail (node terakhir)
def traversal_to_get_tail(head):
    if head is None:
        return None
    current = head
    while current['next'] is not None:
        current = current['next']
    return current

# Penerapan
head = None
head = tambah_node(head, 10)
head = tambah_node(head, 15)
head = tambah_node(head, 117)
head = tambah_node(head, 19)

# cetak isi linked-list
print("Isi Linked-List")
traversal_to_display(head)
```

```
# cetak isi linked-list
print("Isi Linked-List")
traversal_to_display(head)

# cetak jumlah node
print("Jumlah Nodes = ", traversal_to_count_nodes(head))

# cetak HEAD node
print("HEAD Node : ", head['data'])

# cetak TAIL NODE
print("TAIL Node : ", traversal_to_get_tail(head)['data'])
```

Hasil ouputnya

```
Isi Linked-List
Head → 10 → 15 → 117 → 19 → NULL
Jumlah Nodes =  4
HEAD Node :  10
TAIL Node :  19
PS E:\algoritma\Tugas-semester-2\Modul 2>
```

Penjelasannya :

Baris 1:
Judul praktik ditulis sebagai “Praktek 23: Traversal Linked-List” — membahas penelusuran node dalam struktur data linked list.

Baris 2–4:
Mendefinisikan fungsi buat_node(data) yang membuat node dengan nilai data dan referensi ke node berikutnya (next) diatur ke None.

Baris 6–13:
Mendefinisikan fungsi tambah_node(head, data) untuk menambahkan node baru ke akhir linked list.

- **Baris 7: Membuat node baru dari data.**
- **Baris 8–9: Jika head kosong, node baru jadi head.**
- **Baris 10–12: Jika tidak kosong, lakukan iterasi sampai node terakhir, lalu sambungkan node baru.**

Baris 15–21:
Mendefinisikan fungsi traversal_to_display(head) untuk menampilkan isi linked list:

- **Baris 16: Menampilkan "Head →".**
- **Baris 17–20: Iterasi setiap node dan cetak nilainya sampai mencapai NULL.**

Baris 23–28:
Mendefinisikan fungsi traversal_to_count_nodes(head) untuk menghitung jumlah node:

- **Baris 24: Variabel count diset awal 0.**
- **Baris 25–27: Iterasi tiap node dan tambahkan 1 ke count.**
- **Baris 28: Mengembalikan jumlah node.**

Baris 30–35:
Mendefinisikan fungsi traversal_to_get_tail(head) untuk mendapatkan nilai node terakhir (tail):

- **Baris 31: Jika head kosong, kembali None.**
- **Baris 32–34: Iterasi sampai node terakhir ditemukan.**
- **Baris 35: Mengembalikan node terakhir.**

Baris 37:
Komentar bahwa bagian ini adalah contoh penerapan fungsi-fungsi sebelumnya.

Baris 38:
Inisialisasi awal head sebagai None.

Baris 40–43:
Menambahkan empat node ke linked list dengan nilai 10, 15, 17, dan 19.

Baris 45:
Mencetak label "Isi Linked-List".

Baris 46:
Memanggil fungsi traversal_to_display(head) untuk menampilkan semua node.

Baris 48:
Mencetak jumlah node menggunakan traversal_to_count_nodes(head).

Baris 50–51:
Menampilkan nilai dari HEAD node (node pertama).

Baris 53–54:
Menampilkan nilai dari TAIL node (node terakhir) menggunakan traversal_to_get_tail(head).

Praktek 24

```
# Praktek 24 : Menyisipkan Node di awal Linked-List
# membuat node baru
def sisip_depan(head, data):
    new_node = {'data': data, 'next': head}
    return new_node

# menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")

# Penerapan membuat linked-list awal
head = None
head = sisip_depan(head, 30)
head = sisip_depan(head, 20)
head = sisip_depan(head, 10)

# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penyisipan di Depan")
cetak = cetak_linked_list(head)

# Penyisipan node
data = 99
head = sisip_depan(head, data)
```

```
print("\nData Yang Disispkan : ", data)

# cetak isi setelah penyisipan node baru di awal
print("\nIsi Linked-List Setelah Penyisipan di Depan")
cetak_linked_list(head)
```

Hasil ouputnya

```
Isi Linked-List Sebelum Penyisipan di Depan
Head → 10 → 20 → 30 → NULL

Data Yang Disispkan :  99

Isi Linked-List Setelah Penyisipan di Depan
Head → 99 → 10 → 20 → 30 → NULL
PS E:\algoritma\Tugas-semester-2\Modul 2>
```

Penjelasannya :

Baris 1:
Judul praktik “Praktek 24: Menyisipkan Node di awal Linked-List”.

Baris 2–5:
Mendefinisikan fungsi sisip_depan(head, data) untuk menyisipkan node baru di awal:

- **Baris 3: Membuat node baru dengan data, dan next menunjuk ke head (node sebelumnya).**
- **Baris 4: Node baru dikembalikan sebagai head yang baru.**

Baris 7–13:
Mendefinisikan fungsi cetak_linked_list(head) untuk menampilkan isi linked list:

- **Baris 8: current diset ke head.**
- **Baris 9: Mencetak label "Head →".**
- **Baris 10–12: Menelusuri node satu per satu dan mencetak nilainya hingga akhir (NULL).**

Baris 15:
Komentar bahwa bagian ini adalah pembuatan linked list awal.

Baris 16:
Inisialisasi head dengan None.

Baris 17–19:
Menambahkan tiga node ke awal linked list secara bertahap:

- **Menyisipkan 30 → 20 → 10**
- **Karena disisipkan di depan, urutan akhir: 10 → 20 → 30**

Baris 21:
Mencetak informasi bahwa isi linked list akan ditampilkan sebelum penyisipan baru.

Baris 22:
Menampilkan isi linked list awal.

Baris 24:
Menentukan data 99 yang akan disisipkan di depan.

Baris 25:
Menyisipkan data 99 ke awal linked list.

Baris 27:
Menampilkan informasi nilai yang disisipkan.

Baris 29:
Menampilkan informasi bahwa isi linked list akan ditampilkan setelah penyisipan.

Baris 30:
Menampilkan isi linked list terbaru setelah penambahan node di depan.

Praktek 25

```
# Praktek 25 : Menyisipkan diposisi manapun
# membuat node baru
def sisip_depan(head, data):
    new_node = {'data': data, 'next': head}
    return new_node

# sisip node diposisi mana saja
def sisip_dimana_aja(head, data, position):
    new_node = {'data': data, 'next': None}

    # cek jika posisi di awal pakai fungsi sisip_depan()
    if position == 0:
        return sisip_depan(head, data)

    current = head
    index = 0

    # traversal menuju posisi yang diinginkan dan bukan posisi 0
    while current is not None and index < position - 1:
        current = current['next']
        index += 1

    if current is None:
        print("Posisi melebihi panjang linked list!")
        return head

    # ubah next dari node sebelumnya menjadi node baru
    new_node['next'] = current['next']
    current['next'] = new_node
    return head
```

```python
## menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")

# Penerapan
# membuat linked-list awal
head = None
head = sisip_depan(head, 30)
head = sisip_depan(head, 20)
head = sisip_depan(head, 10)
head = sisip_depan(head, 50)
head = sisip_depan(head, 70)

# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penyisipan")
cetak = cetak_linked_list(head)

# Penyisipan node
data = 99
pos = 3
head = sisip_dimana_aja(head, data, pos)

print("\nData Yang Disispkan : ", data)
print("Pada posisi : ", pos, "")
```

```python
# cetak isi setelah penyisipan node baru di awal
print("\nIsi Linked-List Setelah Penyisipan di tengah")
cetak_linked_list(head)
```

Hasil ouputnya

```
Isi Linked-List Sebelum Penyisipan
Head → 70 → 50 → 10 → 20 → 30 → NULL

Data Yang Disispkan :  99
Pada posisi :  3

Isi Linked-List Setelah Penyisipan di tengah
Head → 70 → 50 → 10 → 99 → 20 → 30 → NULL
PS E:\algoritma\Tugas-semester-2\Modul 2>
```

Penjelasannya :

Bagian 1 – Fungsi untuk membuat node baru

Baris 1:
Judul praktik “Praktek 25: Menyisipkan node di posisi tertentu”.

Baris 2–5:
Fungsi sisip_depan(head, data) membuat node baru dan meletakkannya di depan (awalan).

- **Baris 3: Node baru dibuat dengan data, dan next menunjuk ke head.**
- **Baris 4: Mengembalikan node baru sebagai head.**

Baris 7–8:
Fungsi sisip_tengah(head, data, posisi) untuk menyisipkan node di tengah.

- **Membuat node baru.**

Baris 9–10:
Jika posisi 0, maka gunakan sisip_depan() karena dianggap sebagai awal.

Baris 11–14:
Traversal untuk menuju ke posisi sebelum lokasi penyisipan.

- **index digunakan sebagai penghitung langkah.**
- **Melintasi hingga mencapai posisi − 1.**

Baris 15–16:
Jika current bernilai None, berarti posisi terlalu jauh atau melebihi panjang linked list → kembalikan head tanpa perubahan.

Baris 17–19:
Sisipkan node baru di antara node saat ini dan node setelahnya.

- **new_node['next'] diarahkan ke node setelah current.**
- **current['next'] diarahkan ke new_node.**

Baris 21–26:
Fungsi cetak_linked_list(head) untuk menampilkan isi linked list dari awal ke akhir hingga NULL.

Baris 28:
Inisialisasi linked list kosong (head = None).

Baris 29–32:
Menambahkan node di depan: 30, 20, 10 → hasil awal: 10 → 20 → 30.

Baris 34–35:
Menampilkan isi linked list sebelum penyisipan di tengah.

Baris 37–39:
Menyisipkan nilai 99 pada posisi ke-2 (setelah node ke-1, sebelum node ke-2).

Baris 41:
Menampilkan nilai dan posisi yang disisipkan.

Baris 43:
Menampilkan isi linked list setelah penyisipan node di tengah.

Praktek 26

```python
# Praktek 26 : Penghapusan node di awal linked-list
# membuat node baru
def
    (variable) new_node: dict[str, Any]
    new_node = {'data': data, 'next': head}
    return new_node

# sisip node diposisi mana saja
def sisip_dimana_aja(head, data, position):
    new_node = {'data': data, 'next': None}

    # cek jika posisi di awal pakai fungsi sisip_depan()
    if position == 0:
        return sisip_depan(head, data)

    current = head
    index = 0

    # traversal menuju posisi yang diinginkan dan bukan posisi 0
    while current is not None and index < position - 1:
        current = current['next']
        index += 1

    if current is None:
        print("Posisi melebihi panjang linked list!")
        return head

    # ubah next dari node sebelumnya menjadi node baru
    new_node['next'] = current['next']
    current['next'] = new_node
    return head
```

```python
# menghapus head node dan mengembalikan head baru
def hapus_head(head):
    # cek apakah list kosong
    if head is None:
        print("Linked-List kosong, tidak ada yang bisa")
        return None
    print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-list")
    return head['next']

## menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")

# Penerapan
# membuat linked-list awal
head = None
head = sisip_depan(head, 30) # tail
head = sisip_depan(head, 20)
head = sisip_depan(head, 10)
head = sisip_depan(head, 50)
head = sisip_depan(head, 70) # head

# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penghapusan")
cetak_linked_list(head)
```

```python
# Penghapusan head linked-list
head = hapus_head(head)

# cetak isi setelah hapus head linked-list
print("Isi Linked-List Setelah Penghapusan Head ")
cetak_linked_list(head)
```

Hasil ouputnya

```
Isi Linked-List Sebelum Penghapusan
Head → 70 → 50 → 10 → 20 → 30 → NULL

Node dengan data '70' dihapus dari head linked-list
Isi Linked-List Setelah Penghapusan Head
Head → 50 → 10 → 20 → 30 → NULL
PS E:\algoritma\Tugas-semester-2\Modul 2>
```

Penjelasannya :

Baris 1:
Judul praktik: “Praktek 26: Penghapusan node di awal Linked-List”.

Baris 2–5:
Fungsi sisip_depan(head, data) untuk membuat node baru dan menyisipkannya di awal.

- **Baris 3: Node baru dibuat dengan data dan menunjuk ke head.**
- **Baris 4: Mengembalikan node baru.**

Baris 6–7:
Fungsi sisip_diimana(head, data, posisi) untuk menyisipkan node pada posisi tertentu.

- **Node dibuat dengan menunjuk ke None.**

Baris 8–10:
Jika posisi 0, maka langsung gunakan fungsi sisip_depan().

Baris 11–16:
Traversal untuk mencari posisi sebelum tempat penyisipan.

- **Gunakan variabel index sebagai penghitung.**
- **Berjalan hingga position - 1.**

Baris 17–18:
Jika node tujuan tidak ditemukan (posisi lebih panjang dari panjang linked-list), tampilkan peringatan dan kembalikan head tanpa perubahan.

Baris 19–21:
Proses penyisipan node ke posisi tengah.

- **Node baru disisipkan setelah node yang ditemukan.**

Baris 23–28:
Fungsi hapus_head(head) untuk menghapus node pertama.

- **Jika linked-list kosong, tampilkan pesan.**
- **Jika tidak kosong, tampilkan data node yang dihapus dan kembalikan head berikutnya.**

Bagian 3 – Fungsi menampilkan isi linked-list

Baris 30–35:
Fungsi cetak_linked_list(head) untuk menampilkan semua node dalam linked-list.

- **Dimulai dari head, ditampilkan satu per satu hingga NULL.**

Baris 37:
Inisialisasi linked-list kosong.

Baris 38–42:
Menambahkan beberapa node ke dalam linked list menggunakan sisip_depan().

- **Hasil akhir: 20 → 30 → 40 → 50 (karena sisip depan).**

Baris 44–45:
Menampilkan isi linked-list sebelum penghapusan.

Baris 47:
Menghapus head dengan hapus_head(head).

Baris 49–50:
Menampilkan isi linked-list setelah node pertama (head) dihapus.

Praktek 27

```
# Praktek 27 : Menghapus node Tail
# membuat node baru
def sisip_depan(head, data):
    new_node = {'data': data, 'next': head}
    return new_node

# menghapus head node dan mengembalikan head baru
def hapus_tail(head):
    # cek apakah head node == None
    if head is None:
        print('Linked-List Kosong, tidak ada yang bisa dihapus!')
        return None

    # cek node hanya 1
    if head['next'] is None:
        print(f"Node dengan data '{head['data']}' dihapus. Linked list sekarang kosong.")
        return None

    current = head
    while current['next']['next'] is not None:
        current = current['next']

    print(f"\nNode dengan data '{current['next']['data']}' dihapus dari akhir.")
    current['next'] = None
    return head
```

```python
## menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")

# Penerapan
# membuat linked-list awal
head = None
head = sisip_depan(head, 30) # tail
head = sisip_depan(head, 20)
head = sisip_depan(head, 10)
head = sisip_depan(head, 50)
head = sisip_depan(head, 70) # head

# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penghapusan")
cetak_linked_list(head)

# Penghapusan tail linked-list
head = hapus_tail(head)

# cetak isi setelah hapus Tail linked-list
print("Isi Linked-List Setelah Penghapusan Tail ")
cetak_linked_list(head)
```

Hasil ouputnya

```
Isi Linked-List Sebelum Penghapusan
Head → 70 → 50 → 10 → 20 → 30 → NULL

Node dengan data '30' dihapus dari akhir.
Isi Linked-List Setelah Penghapusan Tail
Head → 70 → 50 → 10 → 20 → NULL
PS E:\algoritma\Tugas-semester-2\Modul 2>
```

Penjelasannya :

Bagian 1 – Fungsi membuat node baru

Baris 2–5:
Fungsi sisip_depan(head, data) untuk menyisipkan node baru di depan.

- **new_node menyimpan data dan menunjuk ke head.**
- **Node baru dikembalikan.**

Baris 7–25:
Fungsi hapus_tail(head) untuk menghapus node terakhir:

- **Baris 8–10: Jika linked list kosong (head == None), tampilkan pesan bahwa tidak ada yang bisa dihapus.**
- **Baris 12–14: Jika hanya ada satu node (tail = head), tampilkan pesan bahwa node dihapus dan linked list kosong.**
- **Baris 16–21:**
  Traversal mencari node sebelum tail.
  - **Saat node berikutnya (current['next']['next']) sudah None, berarti node saat ini adalah sebelum tail.**

    - **Tampilkan data node tail yang dihapus dan setel pointer ke None (menghapus tail).**
- **Baris 22: Kembalikan head yang sudah diperbarui.**

Baris 27–33:
Fungsi cetak_linked_list(head) untuk mencetak isi linked list:

- **Mulai dari head, cetak setiap data hingga node terakhir, lalu tampilkan NULL.**

Baris 35:
Inisialisasi linked list kosong (head = None).

Baris 36–40:
Menyisipkan node secara berurutan ke depan:

- **Head akhir: 70 → 60 → 50 → 20 → 30 (30 adalah tail karena dimasukkan pertama).**

Baris 42–43:
Menampilkan isi linked list sebelum penghapusan tail.

Baris 45:
Memanggil hapus_tail(head) untuk menghapus node terakhir (30).

Baris 47–48:
Menampilkan isi linked list setelah tail dihapus.

Praktek 28

```
# Praktek 28 : Menghapus node di posisi manapun (tengah)
# membuat node baru
def sisip_depan(head, data):
    new_node = {'data': data, 'next': head}
    return new_node

# menghapus head node dan mengembalikan head baru
def hapus_head(head):
    # cek apakah list kosong
    if head is None:
        print("Linked-List kosong, tidak ada yang bisa")
        return None
    print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-list")
    return head['next']

# menghapus node pada posisi manapun (tengah)
def hapus_tengah(head, position):
    # cek apakah head node == None
    if head is None:
        print('\nLinked-List Kosong, tidak ada yang bisa dihapus!')
        return None

    # cek apakah posisi < 0
    if position < 0:
        print('\nPosisi Tidak Valid')
        return head
```

```
    # Cek apakah posisi = 0
    if position == 0:
        print(f"Node dengan data '{head['data']}' dihapus dari posisi 0.")
        hapus_head(head)
        return head['next']

    current = head
    index = 0

    # cari node sebelum posisi target
    while current is not None and index < position -1:
        current = current['next']
        index += 1

    # Jika posisi yang diinputkan lebih besar dari panjang list
    if current is None or current['next'] is None:
        print("\nPosisi melebih panjang dari linked-list")
        return head

    print(f"\nNode dengan data '{current['next']['data']}' dihapus dari posisi {position}.")
    current['next'] = current['next']['next']
    return head

## menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")
```

Hasil ouputnya

```
Isi Linked-List Sebelum Penghapusan
Head → 70 → 50 → 10 → 20 → 30 → NULL

Node dengan data '10' dihapus dari posisi 2.

Isi Linked-List Setelah Penghapusan Tengah
Head → 70 → 50 → 20 → 30 → NULL
PS E:\algoritma\Tugas-semester-2\Modul 2>
```

Penjelasannya :

Baris 1
Komentar bahwa ini adalah praktek menghapus node di posisi tengah pada linked list.

Baris 2
Mendefinisikan fungsi sisip_depanh() untuk menyisipkan elemen di depan linked list.

Baris 3
Membuat node baru berupa dictionary yang berisi data dan next, lalu mengembalikan node tersebut sebagai head baru.

Baris 5
Mendefinisikan fungsi hapus_tengah() untuk menghapus node dari linked list di posisi tertentu.

Baris 6
Memeriksa apakah head kosong. Jika ya, tidak ada yang bisa dihapus.

Baris 7
Menampilkan pesan bahwa linked list kosong.

Baris 8
Mengembalikan nilai head karena tidak ada perubahan.

Baris 10
Jika position adalah 0, maka node paling depan (head) akan dihapus.

Baris 11
Menampilkan pesan bahwa node pertama dihapus.

Baris 12
Mengembalikan node setelah head sebagai head baru.

Baris 14
Menginisialisasi variabel current dengan head.

Baris 15
Jika node awal kosong (seharusnya redundant karena sudah dicek sebelumnya), tampilkan pesan dan return.

Baris 18
Memeriksa apakah posisi < 0. Jika ya, berarti tidak valid.

Baris 19
Menampilkan pesan bahwa posisi tidak valid.

Baris 20
Mengembalikan head tanpa perubahan.

Baris 22
Memeriksa lagi apakah posisi = 0. (Redundant karena sudah ditangani sebelumnya.)

Baris 23
Menampilkan pesan bahwa node dengan data tertentu dihapus dari posisi 0.

Baris 24
Menghapus node pertama dengan menggeser head ke node berikutnya.

Baris 26
Menginisialisasi pointer current kembali ke head.

Baris 27
Inisialisasi indeks untuk tracking posisi.

Baris 29
Melakukan loop untuk mencari node sebelum posisi target (position - 1).

Baris 30
Memindahkan pointer ke node berikutnya.

Baris 31
Menambah nilai indeks.

Baris 33
Memeriksa apakah posisi melebihi panjang list (yaitu current atau current['next'] kosong).

Baris 34
Jika ya, tampilkan pesan bahwa posisi melebihi panjang linked list.

Baris 35
Kembalikan head karena tidak ada yang dihapus.

Baris 37
Menampilkan pesan bahwa node pada posisi tertentu akan dihapus.

Baris 38
Menghapus node dengan mengubah referensi next dari node sebelumnya.

Baris 39
Mengembalikan head setelah penghapusan.

Baris 41
Mendefinisikan fungsi cetak_linked_list() untuk mencetak isi linked list.

Baris 42
Mengatur pointer current ke head list.

Baris 43
Melakukan perulangan untuk mencetak isi node sampai None.

Baris 44
Mencetak data dari node dan panah ke node berikutnya.

Baris 45
Pindah ke node berikutnya.

Baris 46
Mencetak "None" sebagai penutup linked list.

Baris 48
Mulai penerapan: membuat linked list awal.

Baris 49–51
Menyisipkan data 30, 20, dan 10 ke linked list di depan (maka list = 10 → 20 → 30).

Baris 53
Mencetak isi linked list sebelum penghapusan.

Baris 54
Memanggil fungsi cetak_linked_list().

Baris 55
Mencetak baris kosong untuk jeda.

Baris 57
Memanggil fungsi hapus_tengah() untuk menghapus node di posisi ke-1.

Baris 59
Mencetak isi linked list setelah node di posisi tengah dihapus.

Baris 60
Memanggil kembali fungsi cetak_linked_list() untuk melihat hasil akhir.

Praktek 29

```python
# Praktek 29 : Membuat Double Linked-List di Awal
# membuat node baru
def buat_node_double(data):
    return {'data': data, 'prev': head, 'next': None}

# Menambahkan node baru di awal double linked-list
def tambah_node_depan(head, data):
    new_node = buat_node_double(data)
    new_node['next'] = head
    new_node['prev'] = None

    if head is not None:
        head['prev'] = new_node

    return new_node

# Mencetak double linked-list dengan traversal maju
def cetak_dll(head):
    current = head
    print('HEAD', end=' <-> ')
    while current:
        print(current['data'], end=' <-> ')
        current = current['next']
    print('NULL')

# Penerapannya
# Head awal dari linked-list
head = None
```

```python
# Tambah Node
head = tambah_node_depan(head, 16) # 16
head = tambah_node_depan(head, 19) # 16 <-> 19

# Cetak double linked-list sebelum penyisipan di awal node
print("Double Linked-list Awal Sebelum Penyisipan : \n", end='')
cetak_dll(head)

# Tambah Node didepan double linked-list
head = tambah_node_depan(head, 22) # 16 <-> 19 <-> 22
head = tambah_node_depan(head, 99) # 16 <-> 19 <-> 22 <-> 99

# Cetak double linked-list setelah penyisipan di awal node
print("\nDouble Linked-list Awal Setelah Penyisipan: \n", end='')
cetak_dll(head)
```

Hasil ouputnya

```
PS E:\algoritma\Tugas-semester-2\Modul 2> & C:/Users/USER/AppDa
Modul 2/Percobaan 5 Double Linked-List/Double Linked-List.py"
Double Linked-list Awal Sebelum Penyisipan :
HEAD <-> 19 <-> 16 <-> NULL

Double Linked-list Awal Setelah Penyisipan:
HEAD <-> 99 <-> 22 <-> 19 <-> 16 <-> NULL
PS E:\algoritma\Tugas-semester-2\Modul 2>
```

Penjelasannya :

Baris 1
Komentar bahwa ini adalah Praktek 29 untuk membuat double linked list dari depan (awal).

Baris 2
Komentar bahwa akan dibuat node baru.

Baris 3
Mendefinisikan fungsi buat_node_double(data) untuk membuat node double linked-list.

Baris 4
Fungsi mengembalikan dictionary berisi data, prev = None, dan next = None.

Baris 6
Komentar bahwa fungsi selanjutnya akan menambahkan node baru di awal double linked list.

Baris 7
Mendefinisikan fungsi tambah_node_depan(head, data).

Baris 8
Membuat node baru dengan buat_node_double(data).

Baris 9
Menetapkan new_node['next'] menunjuk ke head (karena ditambahkan di awal).

Baris 10
Menetapkan new_node['prev'] tetap None (karena jadi node pertama).

Baris 11
Jika head tidak kosong, maka node lama harus menunjuk balik ke new_node.

Baris 12
Mengatur head['prev'] = new_node untuk menghubungkan balik.

Baris 14
Mengembalikan node baru sebagai head yang baru.

Baris 16
Komentar bahwa fungsi berikut akan mencetak isi double linked list dengan traversal maju (next).

Baris 17
Mendefinisikan fungsi cetak_dl(head).

Baris 18
Mencetak “HEAD” sebagai awal pencetakan.

Baris 19
Menginisialisasi pointer current ke head.

Baris 20
Melakukan loop selama current tidak kosong.

Baris 21
Mencetak data node sekarang dan panah ganda (<>).

Baris 22
Memindahkan current ke node berikutnya (next).

Baris 23
Setelah loop selesai, mencetak NULL sebagai akhir dari list.

Baris 25
Komentar bahwa bagian ini adalah penerapan fungsi.

Baris 26
Inisialisasi head menjadi kosong (None).

Baris 28
Menambahkan node bernilai 16 di depan. Sekarang list: 16

Baris 29
Menambahkan node bernilai 19 di depan. Sekarang list: 19 → 16

Baris 30
Komentar bahwa double linked list akan dicetak sebelum penyisipan di awal.

Baris 31
Mencetak judul tampilan.

Baris 32
Memanggil cetak_dl(head) untuk menampilkan isi list.

Baris 34
Komentar bahwa akan ditambahkan lagi node di depan dengan data 22 dan 99.

Baris 35
Menambahkan node 22 di depan. Sekarang list: 22 → 19 → 16

Baris 36
Menambahkan node 99 di depan. Sekarang list: 99 → 22 → 19 → 16

Baris 37
Mencetak judul list setelah penyisipan awal.

Baris 38
Memanggil kembali cetak_dl(head) untuk menampilkan isi double linked list yang sudah diperbarui.