Linked-list

Praktek 22: Membuat Single Linked-list

- **Tujuan**: Membuat struktur data linked list dasar.
- **Fungsi buat_node(data)**: Membuat node baru berisi data dan penunjuk next ke None.
- **Fungsi tambah_node(head, data)**: Menambahkan node di akhir linked list.
- **Fungsi cetak_linked_list(head)**: Traversal untuk mencetak isi list.

Modul 2 > Latihan_Linked-list.py > ...

```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

```

Modul 2 > Latihan_Linked-list.py > ...

```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")

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

# 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)

```

Praktek 23: Traversal Linked-list

- **Menambahkan tiga fungsi traversal:**
    - traversal_to_display: Menampilkan data dalam list.
    - traversal_to_count_nodes: Menghitung jumlah node.
    - traversal_to_get_tail: Mendapatkan node terakhir (tail).
- Berguna untuk mengakses dan memverifikasi isi linked list.

Modul 2 > Latihan_Linked-list.py > ...

```python
# 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")
```

Modul 2 > Latihan_Linked-list.py > ...

```python

# 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 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'])
```

Modul 2 > Latihan_Linked-list.py > ...

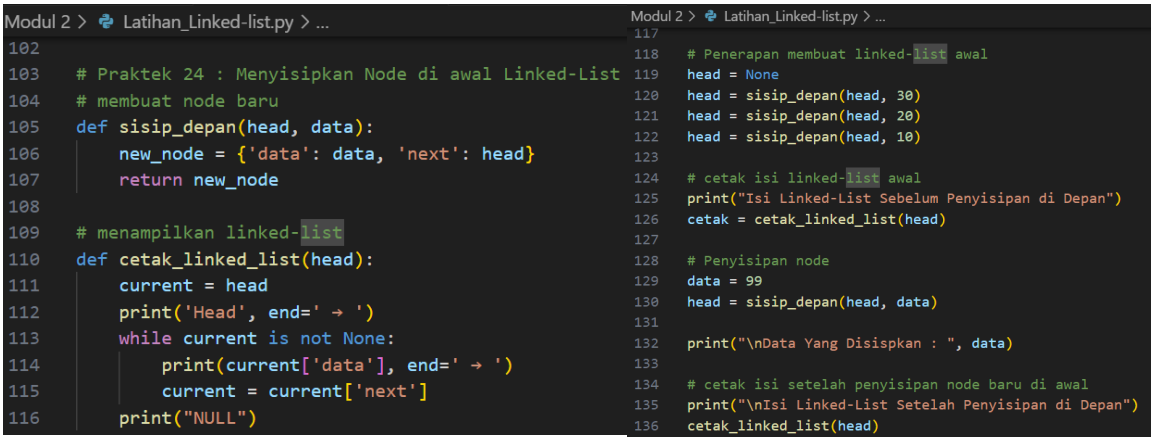
```python

# 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
```

Praktek 24: Menyisipkan Node di Awal

- **Fungsi sisip_depan(head, data)**: Menambahkan node di awal list.
- **Kelebihan**: Operasi O(1), sangat cepat.
- **Hasil**: Node baru langsung menjadi head.

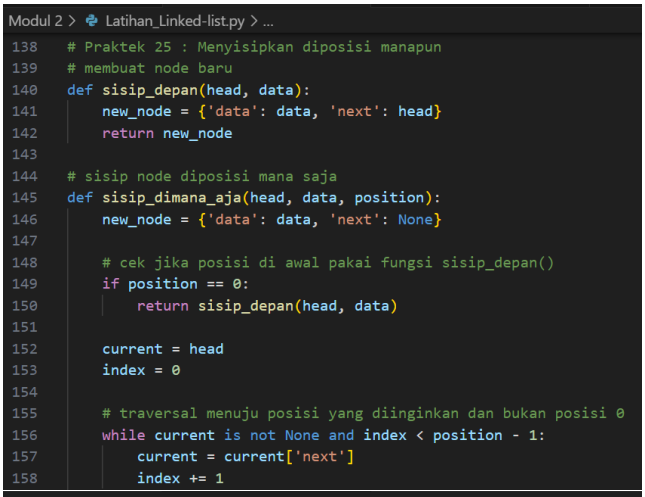


```
Modul 2 > Latihan_Linked-list.py > ...
102
103   # Praktek 24 : Menyisipkan Node di awal Linked-List
104   # membuat node baru
105   def sisip_depan(head, data):
106       new_node = {'data': data, 'next': head}
107       return new_node
108
109   # menampilkan linked-list
110   def cetak_linked_list(head):
111       current = head
112       print('Head', end=' → ')
113       while current is not None:
114           print(current['data'], end=' → ')
115           current = current['next']
116       print("NULL")
```

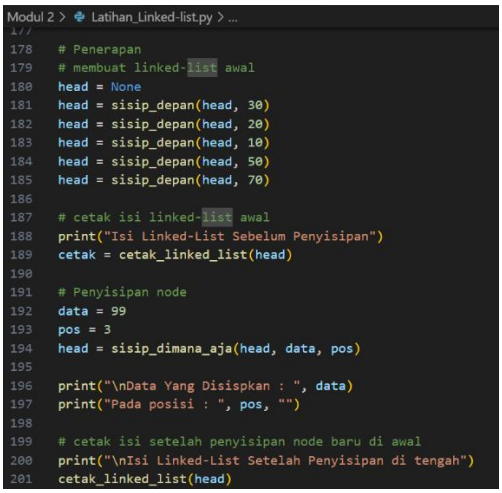
```
Modul 2 > Latihan_Linked-list.py > ...
117
118   # Penerapan membuat linked-list awal
119   head = None
120   head = sisip_depan(head, 30)
121   head = sisip_depan(head, 20)
122   head = sisip_depan(head, 10)
123
124   # cetak isi linked-list awal
125   print("Isi Linked-List Sebelum Penyisipan di Depan")
126   cetak = cetak_linked_list(head)
127
128   # Penyisipan node
129   data = 99
130   head = sisip_depan(head, data)
131
132   print("\nData Yang Disispkan : ", data)
133
134   # cetak isi setelah penyisipan node baru di awal
135   print("\nIsi Linked-List Setelah Penyisipan di Depan")
136   cetak_linked_list(head)
```

Praktek 25: Menyisipkan di Posisi Manapun

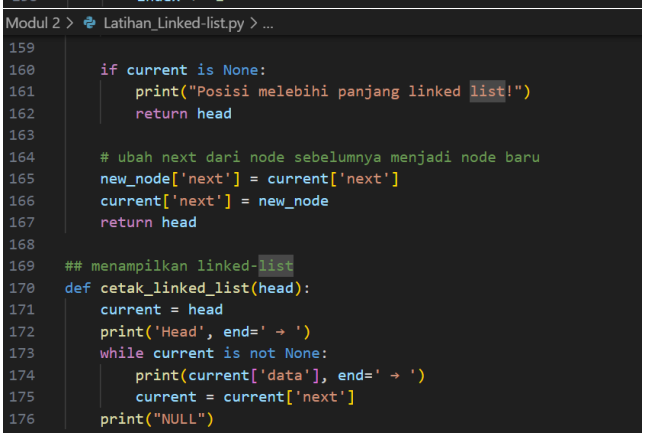
- **Fungsi sisip_dimana_aja(head, data, position)**:
    - Jika position == 0, pakai sisip_depan.
    - Jika tidak, traversal ke posisi position - 1, lalu sambungkan node baru.
- Berguna untuk menyisipkan di tengah list.



```
Modul 2 > Latihan_Linked-list.py > ...
138  # Praktek 25 : Menyisipkan diposisi manapun
139  # membuat node baru
140  def sisip_depan(head, data):
141      new_node = {'data': data, 'next': head}
142      return new_node
143
144  # sisip node diposisi mana saja
145  def sisip_dimana_aja(head, data, position):
146      new_node = {'data': data, 'next': None}
147
148      # cek jika posisi di awal pakai fungsi sisip_depan()
149      if position == 0:
150          return sisip_depan(head, data)
151
152      current = head
153      index = 0
154
155      # traversal menuju posisi yang diinginkan dan bukan posisi 0
156      while current is not None and index < position - 1:
157          current = current['next']
158          index += 1
```



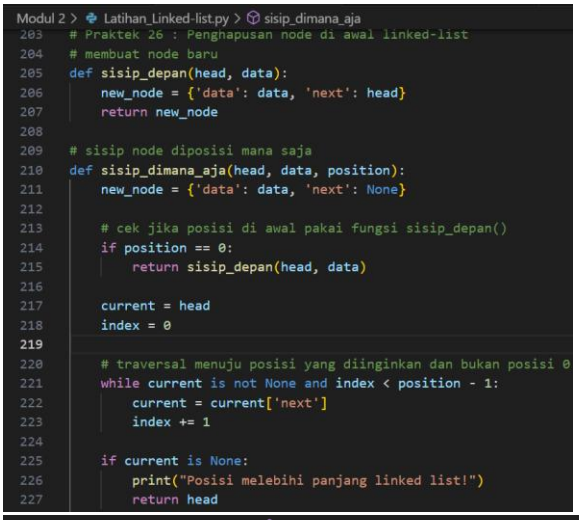
```
Modul 2 > Latihan_Linked-list.py > ...
178  # Penerapan
179  # membuat linked-list awal
180  head = None
181  head = sisip_depan(head, 30)
182  head = sisip_depan(head, 20)
183  head = sisip_depan(head, 10)
184  head = sisip_depan(head, 50)
185  head = sisip_depan(head, 70)
186
187  # cetak isi linked-list awal
188  print("Isi Linked-List Sebelum Penyisipan")
189  cetak = cetak_linked_list(head)
190
191  # Penyisipan node
192  data = 99
193  pos = 3
194  head = sisip_dimana_aja(head, data, pos)
195
196  print("\nData Yang Disispkan : ", data)
197  print("Pada posisi : ", pos, "")
198
199  # cetak isi setelah penyisipan node baru di awal
200  print("\nIsi Linked-List Setelah Penyisipan di tengah")
201  cetak_linked_list(head)
```



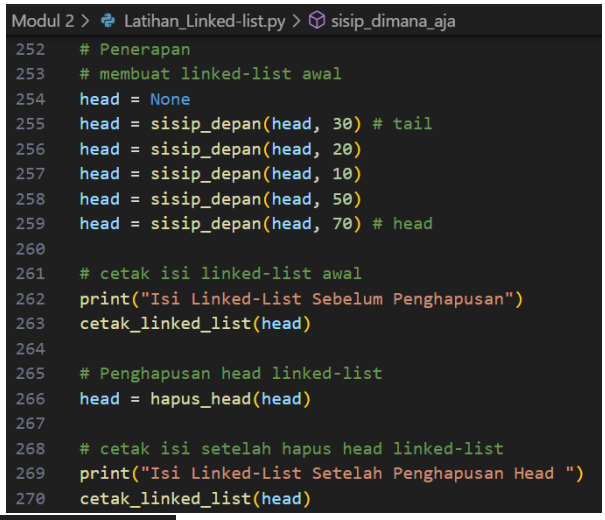
```
Modul 2 > Latihan_Linked-list.py > ...
159
160      if current is None:
161          print("Posisi melebihi panjang linked list!")
162          return head
163
164      # ubah next dari node sebelumnya menjadi node baru
165      new_node['next'] = current['next']
166      current['next'] = new_node
167      return head
168
169  ## menampilkan linked-list
170  def cetak_linked_list(head):
171      current = head
172      print('Head', end=' → ')
173      while current is not None:
174          print(current['data'], end=' → ')
175          current = current['next']
176      print("NULL")
```

Praktek 26: Menghapus Node di Awal (Head)

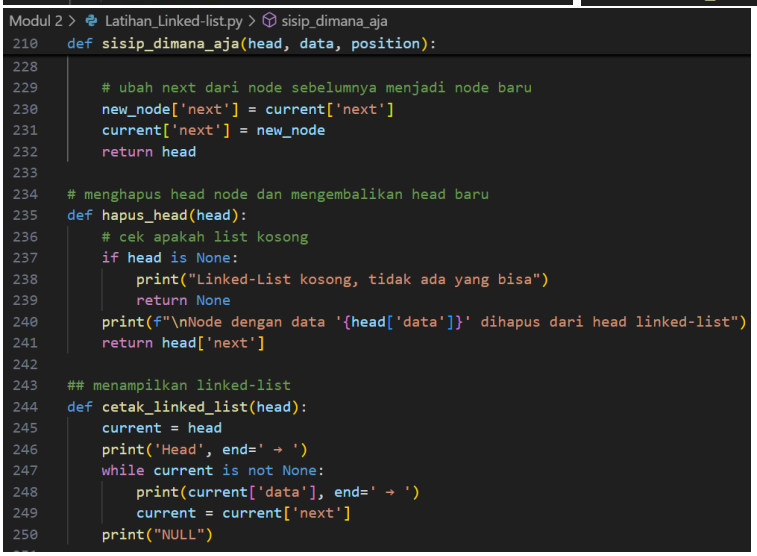
- **Fungsi hapus_head(head)**:
    - Hapus node pertama.
    - Return node setelahnya sebagai head baru.
- **Kelebihan**: Operasi sangat cepat, O(1).



```
Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja
203  # Praktek 26 : Penghapusan node di awal linked-list
204  # membuat node baru
205  def sisip_depan(head, data):
206      new_node = {'data': data, 'next': head}
207      return new_node
208
209  # sisip node diposisi mana saja
210  def sisip_dimana_aja(head, data, position):
211      new_node = {'data': data, 'next': None}
212
213      # cek jika posisi di awal pakai fungsi sisip_depan()
214      if position == 0:
215          return sisip_depan(head, data)
216
217      current = head
218      index = 0
219
220      # traversal menuju posisi yang diinginkan dan bukan posisi 0
221      while current is not None and index < position - 1:
222          current = current['next']
223          index += 1
224
225      if current is None:
226          print("Posisi melebihi panjang linked list!")
227          return head
```



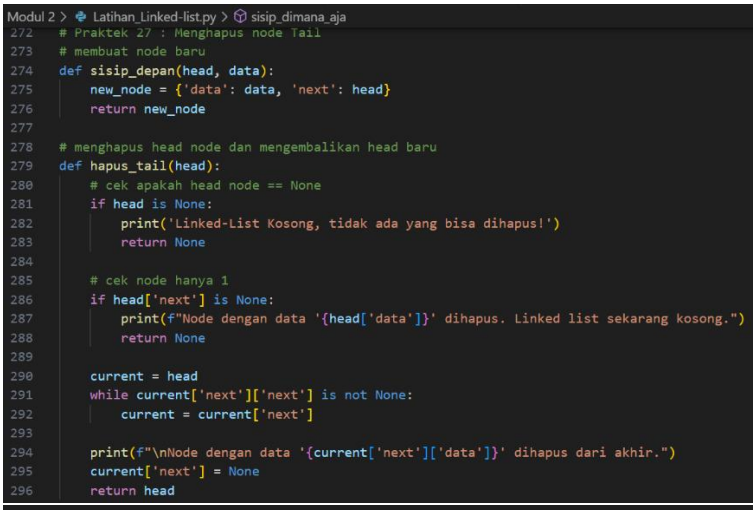
```
Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja
252  # Penerapan
253  # membuat linked-list awal
254  head = None
255  head = sisip_depan(head, 30) # tail
256  head = sisip_depan(head, 20)
257  head = sisip_depan(head, 10)
258  head = sisip_depan(head, 50)
259  head = sisip_depan(head, 70) # head
260
261  # cetak isi linked-list awal
262  print("Isi Linked-List Sebelum Penghapusan")
263  cetak_linked_list(head)
264
265  # Penghapusan head linked-list
266  head = hapus_head(head)
267
268  # cetak isi setelah hapus head linked-list
269  print("Isi Linked-List Setelah Penghapusan Head ")
270  cetak_linked_list(head)
```



```
Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja
210  def sisip_dimana_aja(head, data, position):
228
229      # ubah next dari node sebelumnya menjadi node baru
230      new_node['next'] = current['next']
231      current['next'] = new_node
232      return head
233
234  # menghapus head node dan mengembalikan head baru
235  def hapus_head(head):
236      # cek apakah list kosong
237      if head is None:
238          print("Linked-List kosong, tidak ada yang bisa")
239          return None
240      print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-list")
241      return head['next']
242
243  ## menampilkan linked-list
244  def cetak_linked_list(head):
245      current = head
246      print('Head', end=' → ')
247      while current is not None:
248          print(current['data'], end=' → ')
249          current = current['next']
250      print("NULL")
```

Praktek 27: Menghapus Node di Akhir (Tail)

- **Fungsi hapus_tail(head)**:
    - Traverse hingga node sebelum tail.
    - Set next-nya menjadi None, memutus tail.
- **Cacat**: O(n), karena butuh traversal sampai akhir.

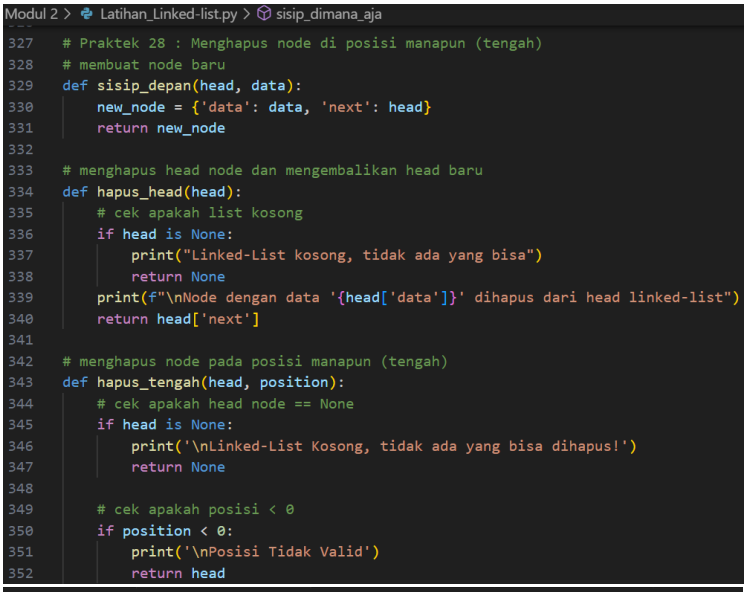


```
Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja
272  # Praktek 27 : Menghapus node Tail
273  # membuat node baru
274  def sisip_depan(head, data):
275      new_node = {'data': data, 'next': head}
276      return new_node
277
278  # menghapus head node dan mengembalikan head baru
279  def hapus_tail(head):
280      # cek apakah head node == None
281      if head is None:
282          print('Linked-List Kosong, tidak ada yang bisa dihapus!')
283          return None
284
285      # cek node hanya 1
286      if head['next'] is None:
287          print(f"Node dengan data '{head['data']}' dihapus. Linked list sekarang kosong.")
288          return None
289
290      current = head
291      while current['next']['next'] is not None:
292          current = current['next']
293
294      print(f"\nNode dengan data '{current['next']['data']}' dihapus dari akhir.")
295      current['next'] = None
296      return head
```

```
Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja
298  ## menampilkan linked-list
299  def cetak_linked_list(head):
300      current = head
301      print('Head', end=' → ')
302      while current is not None:
303          print(current['data'], end=' → ')
304          current = current['next']
305      print("NULL")
306
307  # Penerapan
308  # membuat linked-list awal
309  head = None
310  head = sisip_depan(head, 30) # tail
311  head = sisip_depan(head, 20)
312  head = sisip_depan(head, 10)
313  head = sisip_depan(head, 50)
314  head = sisip_depan(head, 70) # head
315
316  # cetak isi linked-list awal
317  print("Isi Linked-List Sebelum Penghapusan")
318  cetak_linked_list(head)
319
320  # Penghapusan tail linked-list
321  head = hapus_tail(head)
322
323  # cetak isi setelah hapus Tail linked-list
324  print("Isi Linked-List Setelah Penghapusan Tail ")
325  cetak_linked_list(head)
```

Praktek 28: Menghapus Node Tengah

- **Fungsi hapus_tengah(head, position)**:
    - Jika position == 0, sama seperti hapus_head.
    - Jika position valid di tengah list, hapus node target dengan memotong pointer.
- **Validasi**: Menangani posisi tidak valid.

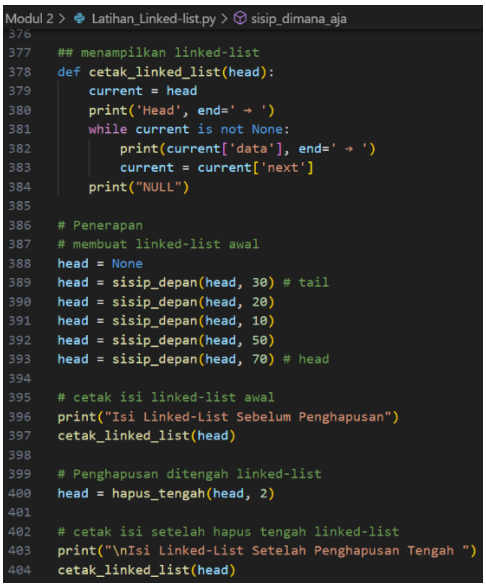


```python
# Praktek 28 : Menghapus node di posisi manapun (tengah)
# membuat node baru
def sisip_depan(head, data):
    new_node = {'data': data, 'next': head}
    return new_node

# 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
```



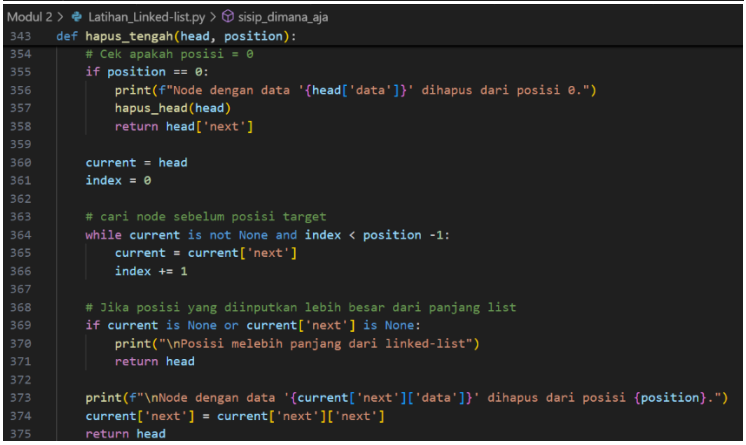
```python
## menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")

# 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 ditengah linked-list
head = hapus_tengah(head, 2)

# cetak isi setelah hapus tengah linked-list
print("\nIsi Linked-List Setelah Penghapusan Tengah ")
cetak_linked_list(head)
```



```python
def hapus_tengah(head, position):
    # 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
```

Praktek 29: Double Linked-List di Awal

- **Struktur**: Tiap node punya prev dan next.
- **Fungsi tambah_node_depan(head, data)**:
    - Sisip node di awal dengan mengatur prev dan next.
- **Traversal dua arah memungkinkan**.

Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja

```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
```

Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja

```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)
```

Praktek 30: Double Linked-List di Akhir

- **Fungsi tambah_node_akhir(head, data)**:
    - Traverse ke akhir list.
    - Tambahkan node baru dengan prev menunjuk ke node terakhir.
- Cocok untuk membuat daftar berurutan dari depan ke belakang.

Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja

```python
# Praktek 30 : Membuat Double Linked-List di Akhir
# membuat node baru
def buat_node_double(data):
    return {'data': data, 'prev': head, 'next': None}

# Menambahkan node baru di akhir double linked-list
def tambah_node_akhir(head, data):
    new_node = buat_node_double(data)

    # Jika list kosong
    if head is None:
        return new_node

    # Jika list tidak kosong, cari node terakhir
    current = head
    while current['next'] is not None:
        current = current['next']

    # Sambungkan node terakhir ke node baru
    current['next'] = new_node
    new_node['prev'] = current

    return head
```

Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja

```python

# 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

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

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

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

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

Praktek 31: Menyisipkan Node Tengah Double Linked-list

- **Fungsi sisip_double_dimana_aja(head, data, position)**:
    - Sisip node di posisi tertentu.
    - Atur prev dan next dengan benar agar tetap konsisten secara dua arah.
- **Validasi posisi sangat penting** agar tidak terjadi kesalahan referensi node.

Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja

```python
# Praktek 31 : Menyisipakan Node Baru di Tengah Double Linked-list
# 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

# sisip node diposisi mana saja
def sisip_double_dimana_aja(head, data, position):
    # membuat node baru
    new_node = buat_node_double(data)

    # cek jika posisi = 0 gunakan fungsi tambah_node_depan()
    if position == 0:
        return tambah_node_depan(head, data)

    # cek jika posisi < 0, input tidak valid
    if position < 0:
        print('\nPosisi Tidak Valid')
        return head
```

Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja

```python
# 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

# 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 Tengah: \n", end='')
cetak_dll(head)

# Tambah Node pada posisi mana saja, di double linked-list
head = sisip_double_dimana_aja(head, 22, 1) # 19 <-> 22 <-> 16
head = sisip_double_dimana_aja(head, 10, 2) # 19 <-> 10 <-> 22 <-> 16
head = sisip_double_dimana_aja(head, 30, 3) # 9 <-> 10 <-> 30 <-> 22 <-> 16

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

Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja

```python
def sisip_double_dimana_aja(head, data, position):

    # deklarasi node pertama
    current = head
    index = 1

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

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

    # sisipkan node diantara current dan current.next
    next_node = current['next']
    current['next'] = new_node
    new_node['prev'] = current
    new_node['next'] = next_node
    if next_node is not None:
        next_node['prev'] = new_node


    return head
```