

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 > ...
1  # Praktek 22 : Membuat Single Linked-list
2
3  # function untuk membuat node
4  def buat_node(data):
5      return {'data': data, 'next': None}
6
7  # menambahkan node di akhir list
8  def tambah_node(head, data):
9      new_node = buat_node(data)
10     if head is None:
11         return new_node
12     current = head
13     while current['next'] is not None:
14         current = current['next']
15     current['next'] = new_node
16     return head
17
18 # menampilkan linked-list
19 def cetak_linked_list(head):
20     current = head
21     print('Head', end=' → ')
22     while current is not None:
23         print(current['data'], end=' → ')
24         current = current['next']
25     print("NULL")
26
27 # Contoh Penerapan
28 # Head awal dari linked-list
29 head = None
30
31 # Tambah node
32 head = tambah_node(head, 10)
33 head = tambah_node(head, 11)
34 head = tambah_node(head, 12)
35
36 # cetak linked-list
37 print('Linked-List : ')
38 cetak_linked_list(head)
39
```

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 > ...
40 # Praktek 23 : Traversal Linked-list
41 # function untuk membuat node
42 def buat_node(data):
43     return {'data': data, 'next': None}
44
45 # menambahkan node di akhir list
46 def tambah_node(head, data):
47     new_node = buat_node(data)
48     if head is None:
49         return new_node
50     current = head
51     while current['next'] is not None:
52         current = current['next']
53     current['next'] = new_node
54     return head
55
56 # traversal untuk cetak isi linked-list
57 def traversal_to_display(head):
58     current = head
59     print('Head', end=' → ')
60     while current is not None:
61         print(current['data'], end=' → ')
62         current = current['next']
63     print("NULL")
```

```
Modul 2 > Latihan_Linked-list.py > ...
82
83 # Penerapan
84 head = None
85 head = tambah_node(head, 10)
86 head = tambah_node(head, 15)
87 head = tambah_node(head, 117)
88 head = tambah_node(head, 19)
89
90 # cetak isi linked-list
91 print("Isi Linked-List")
92 traversal_to_display(head)
93
94 # cetak jumlah node
95 print("Jumlah Nodes = ", traversal_to_count_nodes(head))
96
97 # cetak HEAD node
98 print("HEAD Node : ", head['data'])
99
100 # cetak TAIL NODE
101 print("TAIL Node : ", traversal_to_get_tail(head)['data'])
```

```
Modul 2 > Latihan_Linked-list.py > ...
64
65 # traversal untuk menghitung jumlah elemen dalam linked-list
66 def traversal_to_count_nodes(head):
67     count = 0
68     current = head
69     while current is not None:
70         count += 1
71         current = current['next']
72     return count
73
74 # traversal untuk mencari dimana tail (node terakhir)
75 def traversal_to_get_tail(head):
76     if head is None:
77         return None
78     current = head
79     while current['next'] is not None:
80         current = current['next']
81     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 Disisipkan : ", 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
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")

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 Disisipkan : ", 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)
```

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):
211
212     # ubah next dari node sebelumnya menjadi node baru
213     new_node['next'] = current['next']
214     current['next'] = new_node
215     return head
216
217 # menghapus head node dan mengembalikan head baru
218 def hapus_head(head):
219     # cek apakah list kosong
220     if head is None:
221         print("Linked-List kosong, tidak ada yang bisa")
222         return None
223     print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-list")
224     return head['next']
225
226 ## menampilkan linked-list
227 def cetak_linked_list(head):
228     current = head
229     print('Head', end=' → ')
230     while current is not None:
231         print(current['data'], end=' → ')
232         current = current['next']
233     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)
326
```

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.

Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja

```
327 # Praktek 28 : Menghapus node di posisi manapun (tengah)
328 # membuat node baru
329 def sisip_depan(head, data):
330     new_node = {'data': data, 'next': head}
331     return new_node
332
333 # menghapus head node dan mengembalikan head baru
334 def hapus_head(head):
335     # cek apakah list kosong
336     if head is None:
337         print("Linked-List kosong, tidak ada yang bisa")
338         return None
339     print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-list")
340     return head['next']
341
342 # menghapus node pada posisi manapun (tengah)
343 def hapus_tengah(head, position):
344     # cek apakah head node == None
345     if head is None:
346         print('\nLinked-List Kosong, tidak ada yang bisa dihapus!')
347         return None
348
349     # cek apakah posisi < 0
350     if position < 0:
351         print('\nPosisi Tidak Valid')
352     return head
```

Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja

```
343 def hapus_tengah(head, position):
354     # Cek apakah posisi = 0
355     if position == 0:
356         print(f"Node dengan data '{head['data']}' dihapus dari posisi 0.")
357         hapus_head(head)
358         return head['next']
359
360     current = head
361     index = 0
362
363     # cari node sebelum posisi target
364     while current is not None and index < position - 1:
365         current = current['next']
366         index += 1
367
368     # Jika posisi yang diinputkan lebih besar dari panjang list
369     if current is None or current['next'] is None:
370         print("\nPosisi melebihi panjang dari linked-list")
371         return head
372
373     print(f"\nNode dengan data '{current['next']['data']}' dihapus dari posisi {position}.")
374     current['next'] = current['next']['next']
375     return head
```

Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja

```
376
377 ## menampilkan linked-list
378 def cetak_linked_list(head):
379     current = head
380     print('Head', end=' → ')
381     while current is not None:
382         print(current['data'], end=' → ')
383         current = current['next']
384     print("NULL")
385
386 # Penerapan
387 # membuat linked-list awal
388 head = None
389 head = sisip_depan(head, 30) # tail
390 head = sisip_depan(head, 20)
391 head = sisip_depan(head, 10)
392 head = sisip_depan(head, 50)
393 head = sisip_depan(head, 70) # head
394
395 # cetak isi linked-list awal
396 print("Isi Linked-List Sebelum Penghapusan")
397 cetak_linked_list(head)
398
399 # Penghapusan ditengah linked-list
400 head = hapus_tengah(head, 2)
401
402 # cetak isi setelah hapus tengah linked-list
403 print("\nIsi Linked-List Setelah Penghapusan Tengah ")
404 cetak_linked_list(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
406 # Praktek 29 : Membuat Double Linked-List di Awal
407 # membuat node baru
408 def buat_node_double(data):
409     return {'data': data, 'prev': head, 'next': None}
410
411 # Menambahkan node baru di awal double linked-list
412 def tambah_node_depan(head, data):
413     new_node = buat_node_double(data)
414     new_node['next'] = head
415     new_node['prev'] = None
416
417     if head is not None:
418         head['prev'] = new_node
419
420     return new_node
421
422 # Mencetak double linked-list dengan traversal maju
423 def cetak_dll(head):
424     current = head
425     print('HEAD', end=' <-> ')
426     while current:
427         print(current['data'], end=' <-> ')
428         current = current['next']
429     print('NULL')
430
431 # Penerapannya
432 # Head awal dari linked-list
433 head = None
434
435 # Tambah Node
436 head = tambah_node_depan(head, 16) # 16
437 head = tambah_node_depan(head, 19) # 16 <-> 19
438
439 # Cetak double linked-list sebelum penyisipan di awal node
440 print("\nDouble Linked-list Awal Sebelum Penyisipan : \n", end='')
441 cetak_dll(head)
442
443 # Tambah Node didepan double linked-list
444 head = tambah_node_depan(head, 22) # 16 <-> 19 <-> 22
445 head = tambah_node_depan(head, 99) # 16 <-> 19 <-> 22 <-> 99
446
447 # Cetak double linked-list setelah penyisipan di awal node
448 print("\nDouble Linked-list Awal Setelah Penyisipan: \n", end='')
449 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
451 # Praktek 30 : Membuat Double Linked-List di Akhir
452 # membuat node baru
453 def buat_node_double(data):
454     return {'data': data, 'prev': head, 'next': None}
455
456 # Menambahkan node baru di akhir double linked-list
457 def tambah_node_akhir(head, data):
458     new_node = buat_node_double(data)
459
460     # Jika list kosong
461     if head is None:
462         return new_node
463
464     # Jika list tidak kosong, cari node terakhir
465     current = head
466     while current['next'] is not None:
467         current = current['next']
468
469     # Sambungkan node terakhir ke node baru
470     current['next'] = new_node
471     new_node['prev'] = current
472
473     return head
```

```
Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja
475
476 # Mencetak double linked-list dengan traversal maju
477 def cetak_dll(head):
478     current = head
479     print('HEAD', end=' <-> ')
480     while current:
481         print(current['data'], end=' <-> ')
482         current = current['next']
483     print('NULL')
484
485 # Penerapannya
486 # Head awal dari linked-list
487 head = None
488
489 # Tambah Node
490 head = tambah_node_depan(head, 16) # 16
491 head = tambah_node_depan(head, 19) # 19 <-> 16
492
493 # Cetak double linked-list sebelum penyisipan di akhir node
494 print("Double Linked-list Sebelum Penyisipan diakhir: \n", end='')
495 cetak_dll(head)
496
497 # Tambah Node diakhir double linked-list
498 head = tambah_node_akhir(head, 22) # 19 <-> 16 <-> 22
499 head = tambah_node_akhir(head, 99) # 19 <-> 16 <-> 22 <-> 99
500
501 # Cetak double linked-list setelah penyisipan di akhir node
502 print("\nDouble Linked-list Setelah Penyisipan diakhir: \n", end='')
503 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
505 # Praktek 31 : Menyisipkan Node Baru di Tengah Double Linked-list
506 # membuat node baru
507 def buat_node_double(data):
508     return {'data': data, 'prev': head, 'next': None}
509
510 # Menambahkan node baru di awal double linked-list
511 def tambah_node_depan(head, data):
512     new_node = buat_node_double(data)
513     new_node['next'] = head
514     new_node['prev'] = None
515
516     if head is not None:
517         head['prev'] = new_node
518
519     return new_node
520
521 # sisip node di posisi mana saja
522 def sisip_double_dimana_aja(head, data, position):
523     # membuat node baru
524     new_node = buat_node_double(data)
525
526     # cek jika posisi = 0 gunakan fungsi tambah_node_depan()
527     if position == 0:
528         return tambah_node_depan(head, data)
529
530     # cek jika posisi < 0, input tidak valid
531     if position < 0:
532         print('\nPosisi Tidak Valid')
533         return head
534
535     # deklarasi node pertama
536     current = head
537     index = 1
538
539     # traversal menuju posisi yang diinginkan dan bukan posisi 0
540     while current is not None and index < position - 1:
541         current = current['next']
542         index += 1
543
544     # validasi posisi
545     if current is None:
546         print("Posisi melebihi panjang linked list!")
547         return head
548
549     # sisipkan node diantara current dan current.next
550     next_node = current['next']
551     current['next'] = new_node
552     new_node['prev'] = current
553     new_node['next'] = next_node
554     if next_node is not None:
555         next_node['prev'] = new_node
556
557
558     return head
559
560 # Mencetak double linked-list dengan traversal maju
561 def cetak_dll(head):
562     current = head
563     print('HEAD', end=' <-> ')
564     while current:
565         print(current['data'], end=' <-> ')
566         current = current['next']
567     print('NULL')
```

```
Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja
568
569 # Penerapannya
570 # Head awal dari linked-list
571 head = None
572
573 # Tambah Node
574 head = tambah_node_depan(head, 16) # 16
575 head = tambah_node_depan(head, 19) # 16 <-> 19
576
577 # Cetak double linked-list sebelum penyisipan di awal node
578 print("Double Linked-list Awal Sebelum Penyisipan Tengah: \n", end='')
579 cetak_dll(head)
580
581 # Tambah Node pada posisi mana saja, di double linked-list
582 head = sisip_double_dimana_aja(head, 22, 1) # 19 <-> 22 <-> 16
583 head = sisip_double_dimana_aja(head, 10, 2) # 19 <-> 10 <-> 22 <-> 16
584 head = sisip_double_dimana_aja(head, 30, 3) # 9 <-> 10 <-> 30 <-> 22 <-> 16
585
586 # Cetak double linked-list setelah penyisipan di awal node
587 print("\nDouble Linked-list Awal Setelah Penyisipan Tengah: \n", end='')
588 cetak_dll(head)
```

```
Modul 2 > Latihan_Linked-list.py > sisip_dimana_aja
522 def sisip_double_dimana_aja(head, data, position):
534
535     # deklarasi node pertama
536     current = head
537     index = 1
538
539     # traversal menuju posisi yang diinginkan dan bukan posisi 0
540     while current is not None and index < position - 1:
541         current = current['next']
542         index += 1
543
544     # validasi posisi
545     if current is None:
546         print("Posisi melebihi panjang linked list!")
547         return head
548
549     # sisipkan node diantara current dan current.next
550     next_node = current['next']
551     current['next'] = new_node
552     new_node['prev'] = current
553     new_node['next'] = next_node
554     if next_node is not None:
555         next_node['prev'] = new_node
556
557
558     return head
```