

LINKED LIST

OLEH : ANDRI HERYANDI, M.T.



03

MATERI

01158 - ALGORITMA DAN STRUKTUR DATA 2

- **Pengertian Linked List**
- **Jenis-jenis Linked List**
- **Perbandingan Array dengan Linked List**
- **Struktur Node**
 - Pendefinisian Struktur Node
 - Membuat sebuah Node
 - Mengakses Node
- **Struktur Linked List**
 - Pendeklarasian Linked List
 - Operasi-operasi dalam Linked List

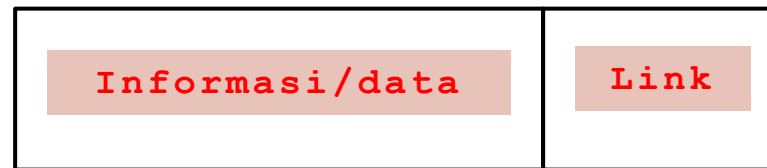


PENGERTIAN LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Linked list adalah sekumpulan data yang tersusun secara linier, berurutan (sekuensial), dan dinamis.
- Setiap data disebut sebagai Node (dalam Bahasa Indonesia disebut Simpul).
- Setiap node memiliki data (informasi) dan link yang menghubungkan dengan node lain.

Struktur Sebuah Node



- Field **informasi** menyimpan data yang dimiliki oleh sebuah node. Tipe data untuk informasi boleh berbentuk tipe data dasar atau pun tipe data bentukan.
- Field **link** akan menyimpan pointer/alamat node yang berhubungan (bisa menunjukan data selanjutnya atau pun data sebelumnya).

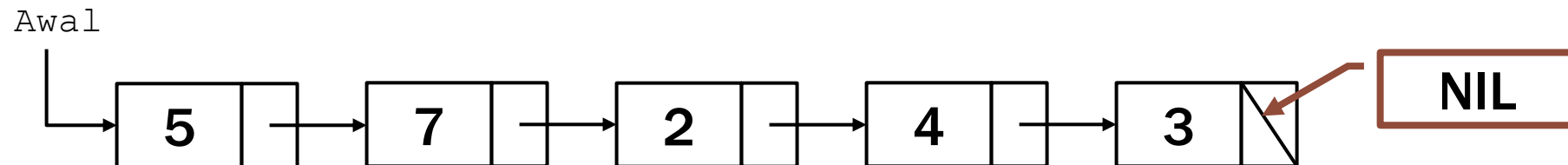


PENGERTIAN LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Sebuah linked list cukup memiliki sebuah link yang menunjuk ke node pertama dari linked list (biasanya disebut head, first, atau dalam Bahasa Indonesia adalah kepala atau awal). Anda juga boleh memiliki linked list yang menggunakan 2 link penunjuk yaitu link awal (head/first) dan link akhir (tail/last)
- Node terakhir adalah node yang tidak lagi mempunyai hubungan ke node lain (ditandai dengan nilai **NIL**(Not In List) atau **NULL** atau **None** dalam python).

Contoh Linked List dengan 5 buah data

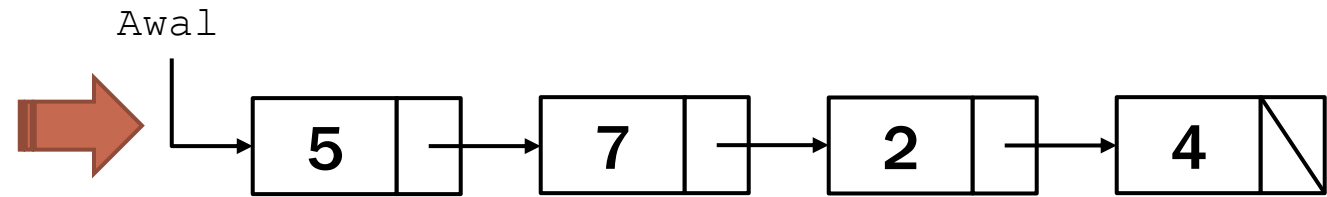


JENIS-JENIS LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

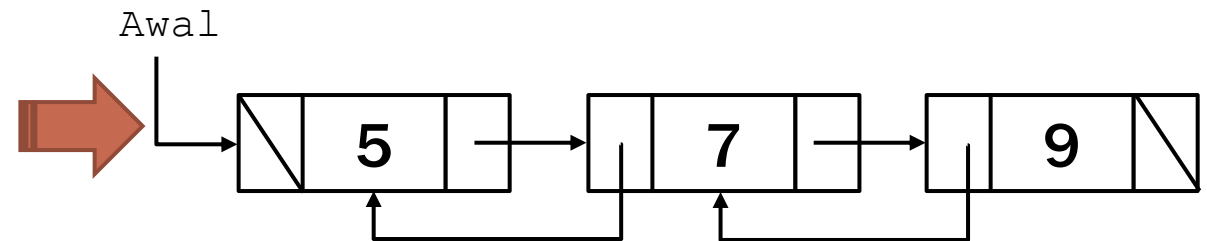
■ Single linked list

- Linked list yang setiap nodenya hanya memiliki 1 link ke node lain (paling umum menunjuk ke node berikutnya).



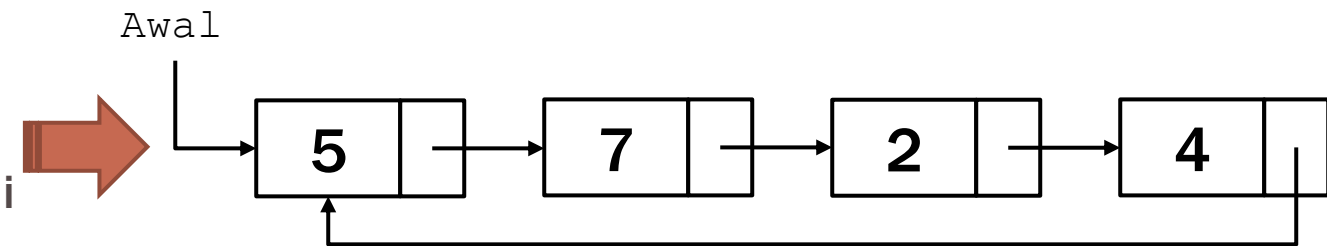
■ Double linked list

- Linked list yang setiap nodenya memiliki 2 link ke node lain (umumnya, 1 node menunjuk ke node berikutnya, dan 1 node lagi menunjuk ke node sebelumnya)



■ Circular linked list

- Bisa berbentuk single linked list atau pun double linked list.
- Linked list yang node terakhirnya memiliki field next tidak menunjuk ke nilai NIL tetapi menunjuk ke awal linked list atau pointer previous dari awal selalu menunjuk ke node akhir.



ARRAY VS LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

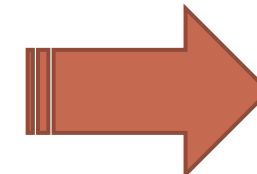
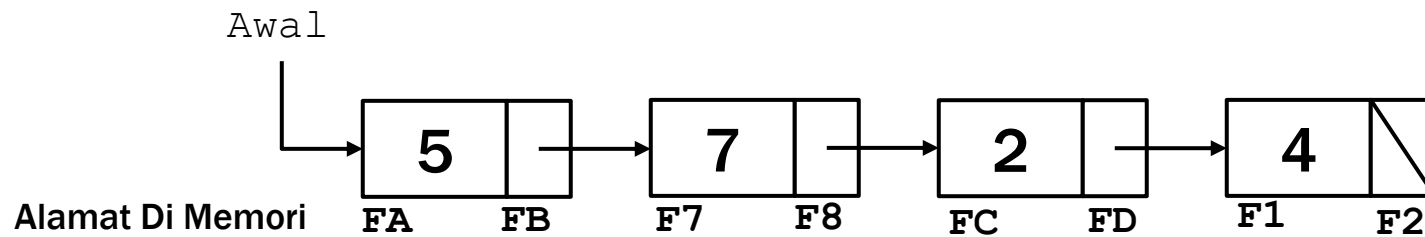
Array	Linked List
Statis Data dialokasikan sebanyak maksimum data yang ditentukan bahkan sejak program dirancang yang digunakan untuk menampung data yang akan dioleh oleh program. Data yang dialokasikan bisa terisi semua atau pun tidak terisi (bisa efektif atau pun boros).	Dinamis Data dialokasikan sesuai kebutuhan ketika program dijalankan. Alokasi data dilakukan ketika program membutuhkan penyimpanan.
Akses Random Bisa pindah ke data berapa pun dengan mudah. Contoh : Ketika anda berada di data ke-1 anda bisa langsung pindah ke data ke-100 dengan mudah.	Akses Sekuensial Jika ingin pindah ke node tertentu, maka harus selalu menelusuri linked list dari awal linked list (tidak bisa langsung pindah ke node posisi tertentu). Contoh : Ketika anda berada di data ke-1 maka anda harus menelusuri linked list satu persatu untuk pindah ke data ke-100.
Data terbatas Proses penambahan data tidak bisa dilakukan jika maksimum data telah tercapai. Penghapusan data dalam array tidak mengembalikan memori yang telah digunakan (karena proses penghapusan data dalam array umumnya hanya pergeseran data saja)	Data tidak terbatas Proses penambahan data bisa terus dilakukan sampai komputer tidak lagi memberikan alokasi memori untuk menyimpan data tersebut. Penghapusan data benar-benar mengembalikan lagi memori yang sudah digunakan.
Oleh : Andri Heryandi, M.T.	6



LINKED LIST DI DALAM MEMORI

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Walaupun node-node yang ada dalam linked list berhubungan secara berurutan, tetapi di memori tidak selalu seperti itu.
- Berikut adalah ilustrasinya



Linked List Di Memori Komputer

Address	Isi	
F0		Node 4
F1	4	
F2	NIL	
F3		
F4		Node 2
F5		
F6		
F7	7	
F8	FC	Node 1
F9		
FA	5	
FB	F7	
FC	2	Node 3
FD	F1	
FE		
FF		

Awal

STRUKTUR NODE

OLEH : ANDRI HERYANDI, M.T.



PENDEFINISIAN NODE (PYTHON)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Node/simpul adalah elemen dari linked list yang berisi data (info) dan link yang menunjuk ke node lain.
- Berikut adalah pendefinisian class Node dalam Bahasa Python

```
class Node:  
    def __init__(self, info):  
        self.info = info  
        self.next = None
```

- Keterangan :
 - Sebuah node disusun dalam bentuk class bernama Node
 - Class Node mempunyai constructor dengan 2 parameter (parameter self sifatnya wajib [aturan python], dan parameter info yang berisi data yang akan disimpan dalam node)
 - Class Node mempunyai 2 buah attribute yaitu :
 - Attribute/property/field **info** yang akan berisi data yang disimpan dalam node. Nilai defaultnya diisi sesuai parameter info.
 - Attribute/property/field **next** yang akan berisi alamat Node selanjutnya. Nilai defaultnya selalu NIL/None



MEMBUAT SEBUAH NODE

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Setelah membuat class Node, maka jika anda ingin membuat variable/object yang berjenis Node, maka lakukan dengan cara instansiasi objek (menciptakan objek) dengan memanggil constructor dari class Node.
- Berikut contoh membuat objek Node

```
class Node:
    def __init__(self, info):
        self.info = info
        self.next = None

Node1 = Node(10) # membuat objek Node1, diisi info 10
Node2 = Node(5)  # membuat objek Node2, diisi info 5

print("Isi Node 1 : ", Node1.info, " - ", Node1.next)
print("Isi Node 2 : ", Node2.info, " - ", Node2.next)
```

Hasil Eksekusi :

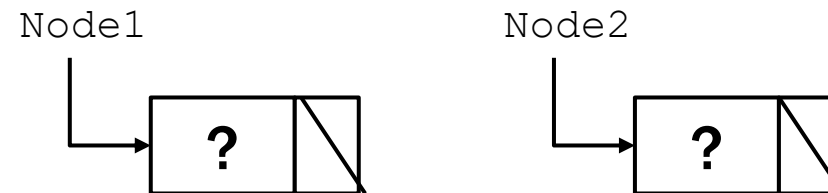
```
Isi Node 1 : 10 - None
Isi Node 2 : 5 - None
```



MENGAKSES SEBUAH NODE

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Asumsikan anda mempunyai 2 buah node yang diberi nama Node1 dan Node2.

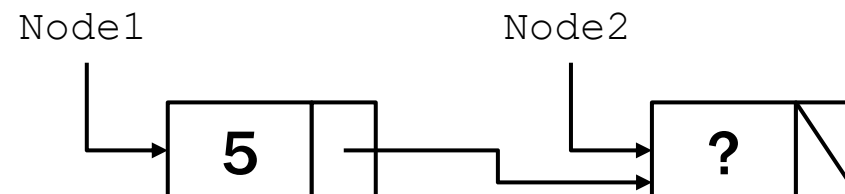


- Cara aksesnya (dalam Bahasa Python) adalah :

- Untuk mengakses field informasi pada Node1 maka menggunakan sintak **Node1.info** (diasumsikan nama field yang menampung data menggunakan nama **info**)
- Untuk mengakses field link pada Node1 maka menggunakan sintak **Node1.next** (diasumsikan nama field untuk pointer sambungan menggunakan nama **next**)

- Contoh :

- **Node1.info = 5** Perintah ini berguna untuk mengubah informasi milik **Node1** menjadi 5.
- **Node2.next = None** Perintah ini digunakan untuk mengubah field **next** dari **Node2** menjadi **nil**.
- **Node1.next = Node2** Perintah ini digunakan untuk mengisi **next** dari **Node1** agar menunjuk ke **Node2**.



SINGLE LINKED LIST

OLEH : ANDRI HERYANDI, M.T.



OPERASI-OPERASI LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Pendefinisian struktur data linked list.
- Inisialisasi linked list
- Pembuatan Node Baru
- Pemeriksaan kondisi kosong linked list.
- Traversal Linked List
 - Traversal untuk menampilkan linked list
 - Traversal untuk menghitung banyaknya elemen linked list
- Pencarian node berdasarkan posisi
 - Mencari node di posisi pertama
 - Mencari node di posisi akhir.
 - Mencari node di posisi tertentu



OPERASI-OPERASI LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Penambahan Data

- Penambahan di awal linked list
- Penambahan di tengah (penyisipan) linked list
- Penambahan di akhir linked list

■ Update Data

■ Penghapusan Data

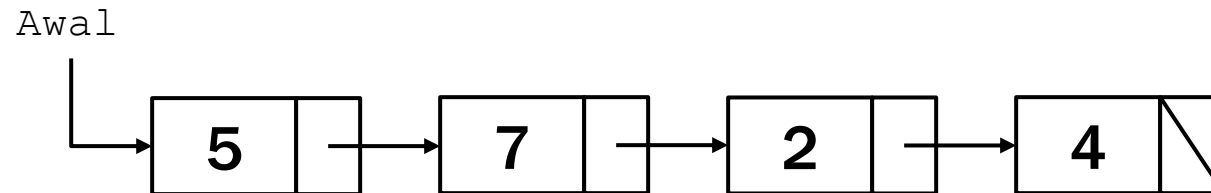
- Penghapusan di awal linked list
- Penghapusan di tengah linked list
- Penghapusan di akhir linked list
- Penghapusan semua elemen



PENDEKLARASIAN STRUKTUR DATA

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Perhatikan gambar linked list berikut :



- Dari gambar di atas ada beberapa hal yang bisa disimpulkan, yaitu :
 - Setiap node mempunyai 2 bagian yaitu bagian **info** dan bagian **next**. Oleh karena itu sebuah node haruslah merupakan sebuah tipe bentukan/class. Asumsikan namanya adalah **Node** (Tipe/class Node).
 - Bagian **info** akan berisi data yang akan disimpan. Tipe datanya sesuai dengan tipe data yang akan ditampung. Bisa tipe data dasar atau pun tipe data bentukan (record/class).
 - Bagian **next** akan berisi data berupa pointer yang menunjuk ke alamat node berikutnya. Node berikutnya juga pastinya bertipe Node.

PENDEFINISIAN LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Berikut adalah pendefinisian linked list dalam bahasa python:

```
class LinkedList:  
    def __init__(self):  
        self.awal = None
```

- Keterangan :
 - Sebuah linked list adalah sebuah class yang memiliki sebuah property bernama **awal**.
 - Nilai awal property ini adalah **NIL/Null/None** (yang artinya pointer awalnya belum memiliki elemen/node).



INISIALISASI LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

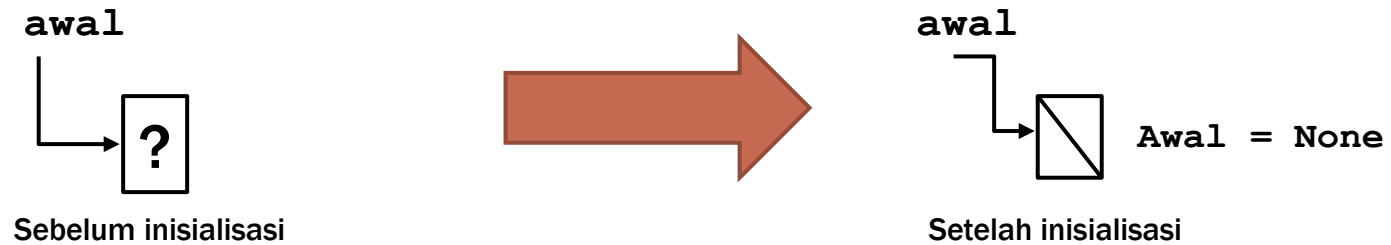
- Inisialisasi linked list adalah proses yang berguna untuk mempersiapkan linked list untuk dipakai.
- Operasi ini biasanya dilakukan dengan memberikan nilai nil terhadap pointer yang menunjuk ke data pertama.
- Operasi ini biasanya dilakukan ketika sebuah linklist pertama kali dibuat/digunakan



INISIALISASI LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Ilustrasi inisialisasi linked list



INISIALISASI LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Method/Constructor inisialisasi linked list

```
def __init__(self):  
    self.awal = None
```

- Keterangan :

- Method/Constructor ini harus ditulis dalam pendefinisian class Linkedlist.

- Source code lengkap :

```
class LinkedList:  
    def __init__(self):  
        self.awal = None
```



INISIALISASI LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Contoh pemanggilan operasi inisialisasi linked list

```
list1 = LinkedList() # pembuatan objek  
print("Awal Linked List1 : ", list1.awal) </>
```

Awal Linked List1 : None



■ Source Code lengkap :

```
class LinkedList: </>  
    def __init__(self):  
        self.awal = None  
  
list1 = LinkedList()  
print("Awal Linked List1 : ", list1.awal)
```



PEMERIKSAAN KONDISI KOSONG LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Proses pemeriksaan kondisi kosong linked list digunakan untuk memeriksa apakah linked list telah memiliki data atau tidak.
- Pemeriksaan ini biasanya digunakan ketika :
 - Ketika mau penambahan data
 - Melakukan penghapusan node
 - Menampilkan isi linked list
- Sebuah linked list dinyatakan kosong jika pointer yang menunjuk ke awal data bernilai NIL/NULL/None.



PEMERIKSAAN KONDISI KOSONG LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Function **isEmpty**

```
def isEmpty(self):  
    return self.awal is None
```

■ Keterangan :

- Method/Function ini harus ditulis dalam pendefinisian class LinkedList.

■ Source Code lengkap class LinkedList:

```
class LinkedList:  
    def __init__(self):  
        self.awal = None  
    def isEmpty(self):  
        return self.awal is None
```




PEMERIKSAAN KONDISI KOSONG LINKED LIST

01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Contoh pemanggilan function `isEmpty`

```
list1 = LinkedList()
if list1.isEmpty():
    print("Linked List Kosong")
else:
    print("Linked List tidak Kosong")

node1 = Node(50)
list1.awal = node1 # sambungkan awal ke node1
print("Setelah penyambungan node ke linked list")
if list1.isEmpty():
    print("Linked List Kosong")
else:
    print("Linked List tidak Kosong")
```

A black icon representing a terminal window, showing a white background with a black border and a small white cursor icon.

Linked List Kosong
Setelah penyambungan node ke linked
list
Linked List tidak Kosong

TRAVERSAL DI LINKED LIST

(MENAMPILKAN DATA DI DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

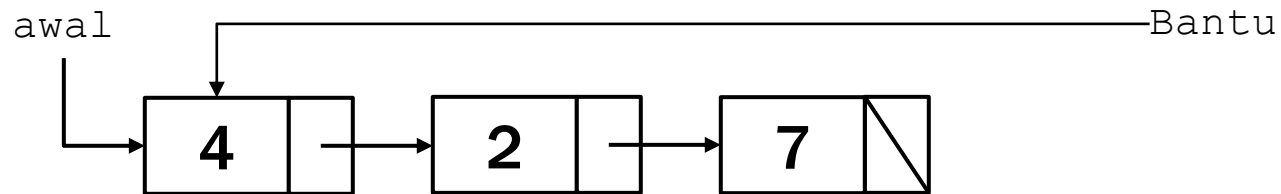
- Menampilkan data adalah proses yang sangat penting ketika membuat sebuah aplikasi.
- Langkah-langkah untuk menampilkan data dalam linked list adalah
 1. Periksa apakah linked listnya kosong
 2. Jika linked list dalam kondisi kosong, maka tampilkan pesan bahwa “Data Kosong”.
 3. Jika linked list dalam kondisi tidak kosong, maka lakukan traversal linked list untuk menampilkan data.
 - A. Buat sebuah pointer yang akan digunakan untuk menelusuri linked list (sebut sebagai variable **Bantu**).
 - B. **Bantu** dimulai dari awal linked list.
 - C. Ulangi langkah D dan E selama variable **Bantu** tidak nil (ujung linked list)
 - D. Tampilkan data (info) pada node ditunjuk oleh **Bantu**.
 - E. Pindahkan **Bantu** ke node selanjutnya (**Bantu.next**).



TRAVERSAL DI LINKED LIST (MENAMPILKAN DATA DI DALAM LINKED LIST)

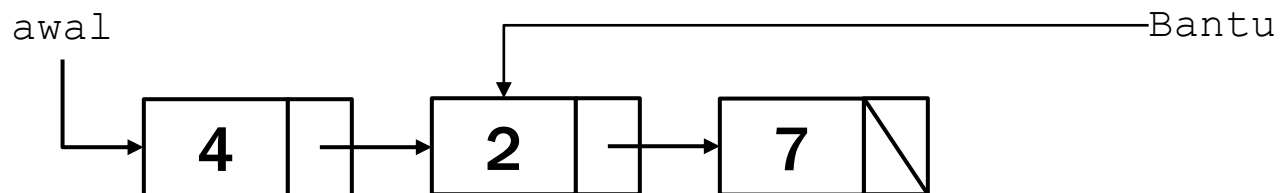
01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Ilustrasi menampilkan data di dalam linked list



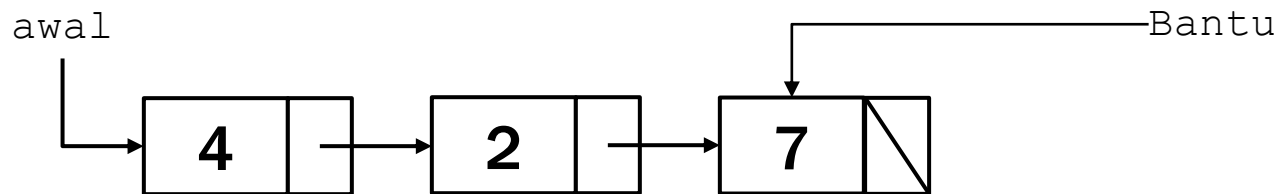
```
Bantu = awal
```

```
print(Bantu.info) # Tulis 4
```



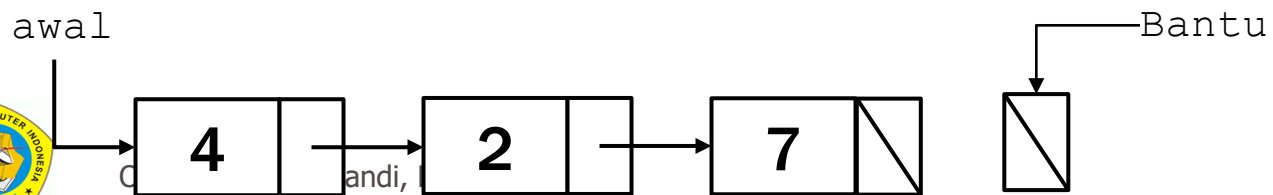
```
Bantu = Bantu.next
```

```
print(Bantu.info) # Tulis 2
```



```
Bantu = Bantu.next
```

```
print(Bantu.info) # Tulis 7
```



```
Bantu = Bantu.next
```

```
// Pengulangan selesai karena  
// Bantu sudah nil/None
```



TRAVERSAL DI LINKED LIST

(MENAMPILKAN DATA DI DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Function menampilkan data dalam linked list (**display**)

```
def display(self):  
    print("Isi Linked List : ",end="")  
    if self.isEmpty():  
        print("[Data Kosong]")  
    else:  
        bantu = self.awal  
        while bantu is not None:  
            print(bantu.info," ",end="")  
            bantu = bantu.next  
        print() # pindah baris
```

- Keterangan :

- Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



TRAVERSAL DI LINKED LIST

(MENAMPILKAN DATA DI DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Contoh pemanggilan operasi display

```
list1 = LinkedList()
list1.display()
node1 = Node(4)
node2 = Node(2)
node3 = Node(7)
list1.awal = node1 # sambungkan node1 ke awal linked list
node1.next = node2 # sambungkan node2 ke next dari node1
node2.next = node3 # sambungkan node3 ke next dari node2
list1.display()
```



```
Isi Linked List : [Data Kosong]
Isi Linked List : 4  2  7
```



TRAVERSAL DI LINKED LIST

(MENGHITUNG BANYAK DATA DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

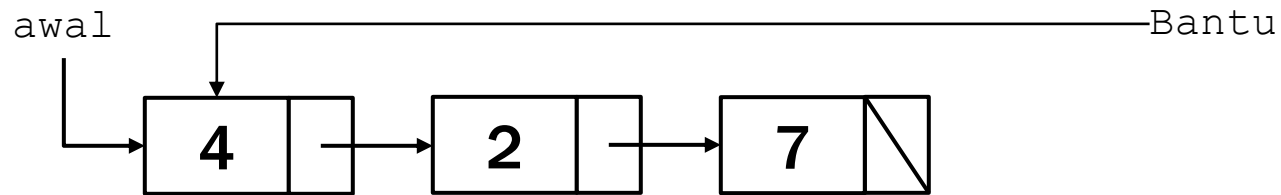
- Proses ini digunakan untuk menghitung berapa banyak data/node yang berada dalam linked list
- Proses ini sangat berguna ketika membutuhkan informasi banyaknya data (misalnya ketika akan melakukan penghapusan data, maka harus diperiksa dulu apakah datanya ada atau tidak).
- Langkah-langkah untuk menghitung banyak data dalam linked list adalah
 1. Periksa apakah linked listnya kosong
 2. Jika linked list dalam kondisi kosong, maka **banyakNode = 0**.
 3. Jika linked list dalam kondisi tidak kosong, maka lakukan traversal linked list untuk menghitung banyaknya data.
 - A. Buat sebuah pointer yang akan digunakan untuk menelusuri linked list (sebut sebagai variable **Bantu**) dan variable penampung banyaknya data (sebut sebagai variable **banyakNode**).
 - B. **Bantu** dimulai dari awal linked list, dan banyakNode diinisialisasi dengan nilai 1.
 - C. Ulangi langkah D dan E selama field **Next** dari variable **Bantu** tidak nil (ujung linked list)
 - D. Tambahkan nilai 1 ke **banyakNode**..
 - E. Pindahkan **Bantu** ke node selanjutnya (**Bantu.next**).
 - F. Setelah selesai pengulangan, maka banyak data dapat dilihat pada variable **banyakNode**.
 4. Returnkan isi **banyakNode**



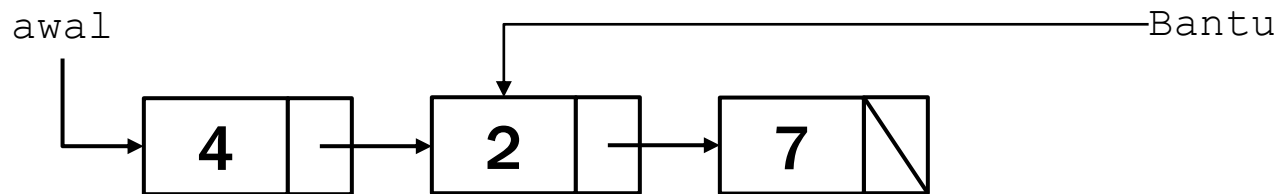
TRAVERSAL DI LINKED LIST (MENGHITUNG BANYAK DATA DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

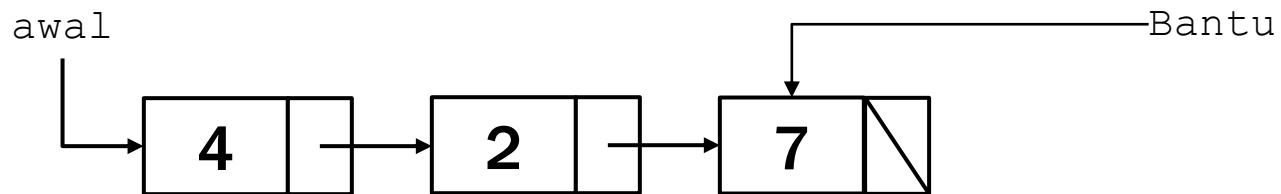
- Ilustrasi menghitung banyaknya data dalam linked list (ketika linked list tidak kosong)



```
Bantu = awal  
banyakNode = 1
```



```
Bantu = Bantu.next;  
banyakNode = banyakNode+1 # 2
```



```
Bantu = Bantu.next;  
banyakNode = banyakNode+1 # 3  
  
// Pengulangan selesai karena  
// Bantu.next sudah nil/None
```

TRAVERSAL DI LINKED LIST

(MENGHITUNG BANYAK DATA DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Function menghitung banyaknya node/data dalam linked list (**size**)

```
def size(self):  
    if self.isEmpty():  
        banyakNode = 0  
    else:  
        bantu = self.awal  
        banyakNode = 1  
        while bantu.next is not None:  
            bantu = bantu.next  
            banyakNode = banyakNode + 1  
    return banyakNode
```

- Keterangan :

- Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



TRAVERSAL DI LINKED LIST

(MENGHITUNG BANYAK DATA DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Contoh pemanggilan function **size**

```
list1 = LinkedList()
print("Banyak data : ", list1.size())
node1 = Node(5)
node2 = Node(7)
node3 = Node(9)
list1.awal = node1
node1.next = node2
node2.next = node3
print("Banyak data : ", list1.size())
```



```
Banyak data : 0
Banyak data : 3
```



PENCARIAN NODE BERDASAR POSISI

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Proses pencarian node digunakan untuk mendapatkan alamat dari sebuah node pada posisi tertentu di dalam linked list.
- Proses ini dibuat untuk mempermudah proses pencarian node ketika akan melakukan proses seperti penambahan di akhir, penambahan di awal, penghapusan di tengah dan lain-lain.
- Ada tiga pencarian node yang bisa dilakukan
 - Mencari node yang berada di awal linked list (`getFirst`)
 - Mencari node yang berada di akhir linked list (`getLast`).
 - Mencari node yang berada di posisi tertentu di dalam linked list (`get`)



PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Pencarian node di awal linked list digunakan untuk mendapatkan pointer yang menunjuk ke awal linked list (tentu saja).
- Proses ini sebenarnya sangat sederhana yaitu cukup dengan mereturnkan nilai awal linked list saja. Tetapi untuk mempermudah keterbacaan source code, maka sebaiknya dibuat dalam bentuk function.

Temp = list.awal

Temp = list.getFirst()

Kedua statement tersebut sama-sama berguna untuk mengisi variable Temp dengan alamat dari node pertama suatu linked list.

Mana yang lebih mudah dimengerti?



PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Function pencarian node awal linked list (**getFirst**)

```
def getFirst(self):  
    return self.awal
```

- Keterangan :
 - Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

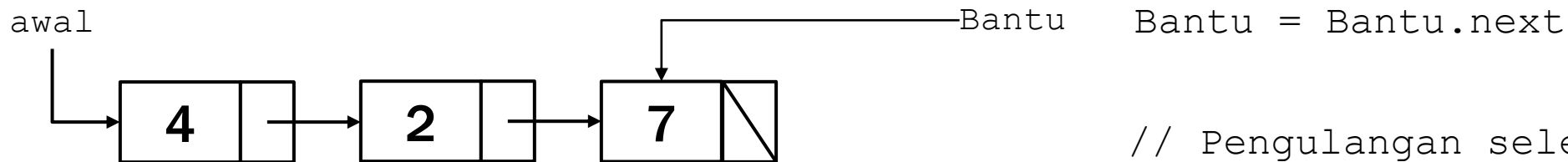
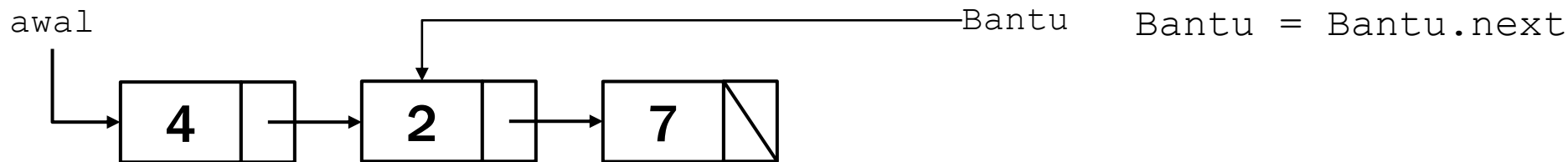
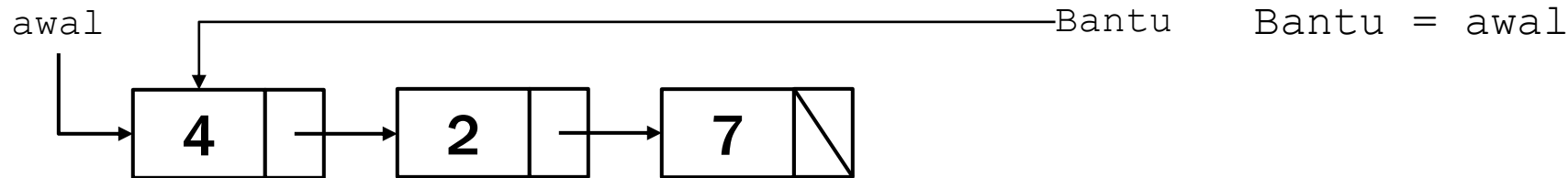
- Pencarian node di akhir linked list digunakan untuk mendapatkan pointer yang menunjuk ke node di akhir linked list.
- Algoritmanya adalah :
 1. Periksa apakah linked list kosong
 2. Jika linked list kosong maka posisi akhir linked list adalah **nil/None**.
 3. Jika linked list tidak kosong maka lakukan langkah berikut :
 - A. Buat variable **Bantu** yang akan digunakan untuk menelusuri linked list
 - B. Variable **Bantu** dimulai dari awal Linked List
 - C. Ulangi proses D selama field **Next** dari **Bantu** belum menunjukan **nil/None** (akhir dari linked list)
 - D. Variable **Bantu** pindah ke node selanjutnya (**Bantu.next**)
 - E. Setelah pengulangan, maka posisi akhir linked list didapatkan pada posisi **Bantu**



PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Ilustrasi pencarian node akhir linked list



```
// Pengulangan selesai karena  
// Bantu.next sudah nil/None
```

PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Function pencarian node akhir linked list (**getLast**)

```
def getLast(self):  
    if self.isEmpty():  
        return None  
    else:  
        bantu = self.awal  
        while bantu.next is not None:  
            bantu = bantu.next  
        return bantu
```

- Keterangan :
 - Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI POSISI TERTENTU LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Pencarian node di posisi tertentu linked list digunakan untuk mendapatkan pointer yang menunjuk ke node yang berada pada posisi tertentu di dalam linked list.
- Algoritmanya adalah :
 1. Periksa apakah linked list dan index posisi yang dicari adalah sah (valid). Suatu posisi disebut tidak valid kalau linked list kosong atau index posisinya kurang dari 1 atau index posisinya melebihi banyaknya data.
 2. Jika posisi tidak valid maka returnkan bahwa nodenya tidak ada (nil)
 3. Jika posisi valid (berada dalam range 1 s.d banyak element linked list) maka lakukan langkah berikut
 - A. Buat variable **Bantu** yang akan digunakan untuk menelusuri linked list, dan variable **posisi** yang digunakan untuk mencatat posisi dari node yang ditunjuk **Bantu**.
 - B. Variable **Bantu** dimulai dari awal Linked List sehingga variable **posisi** bernilai 1 (karena berada di awal linked list)
 - C. Ulangi proses D dan E, selama nilai posisi masih lebih kecil dari index yang dicari
 - D. Variable **Bantu** pindah ke node selanjutnya (**Bantu.next**)
 - E. Variable **posisi** bertambah 1 (karena **Bantu** sudah pindah ke node berikutnya).
 - F. Setelah pengulangan, node yang dicari didapatkan pada posisi **Bantu**. Oleh karena itu returnkan posisi **Bantu**.
 4. Jika linked list kosong maka posisi akhir linked list adalah **nil**.

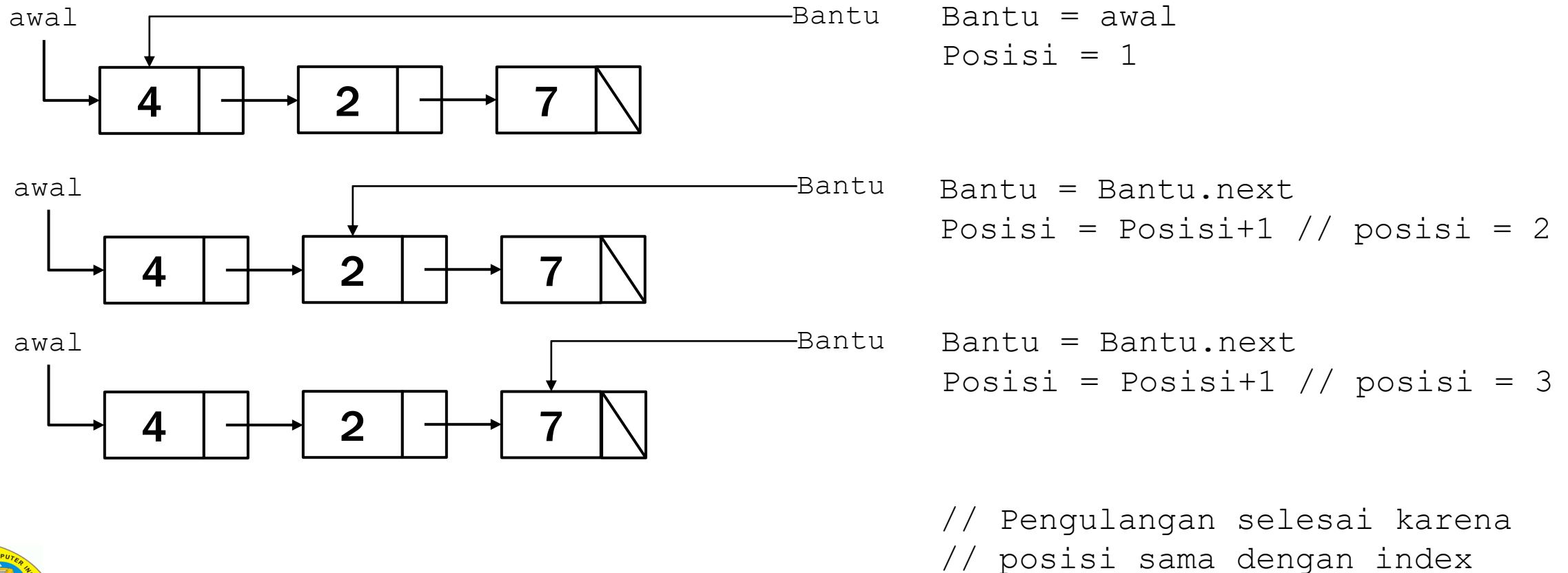


PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Ilustrasi pencarian node di posisi tertentu di dalam linked list

■ Contoh mencari lokasi node pada index ke-3 (Contoh : parameter index = 3)



PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI POSISI TERTENTU LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Function pencarian node di posisi tertentu di dalam linked list (**get**)

```
def get(self, index):  
    if self.isEmpty() or index<1 or index>self.size():  
        return None  
    else:  
        bantu = self_awal  
        pos = 1  
        while posisi<index:  
            bantu = bantu.next  
            pos = pos +1  
        return bantu
```

- Keterangan :

- Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



PENCARIAN NODE BERDASAR POSISI

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Contoh pemanggilan function-function pencarian node (**getFirst**, **getLast**, **get**)

```
node1 = Node(5) # buat node1
print("Node 1 : ", node1)
node2 = Node(7) # buat node2
print("Node 2 : ", node2)
node3 = Node(9) # buat node3
print("Node 3 : ", node3)
```



```
list1 = LinkedList() # buat linked list
list1.awal = node1    # sambungkan node 1
node1.next = node2    # sambungkan node 2
node2.next = node3    # sambungkan node 3

print("Node Pertama : ", list1.getFirst())
print("Node Ke-2 : ", list1.get(2))
print("Node Terakhir : ", list1.getLast())
print("Node Invalid : ", list1.get(10))
```

```
Node 1 : <__main__.Node object at 0x009C1CA0>
Node 2 : <__main__.Node object at 0x009C1430>
Node 3 : <__main__.Node object at 0x009C1400>
Node Pertama : <__main__.Node object at 0x009C1CA0>
Node Ke-2 : <__main__.Node object at 0x009C1430>
Node Terakhir : <__main__.Node object at 0x009C1400>
Node Invalid : None
```



PENAMBAHAN DATA

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Penambahan data dalam linked list berarti melakukan pembuatan sebuah node baru, kemudian menggabungkan node baru tersebut dengan linked list yang sudah ada.
- Ada beberapa cara penambahan data pada linked list sesuai posisi penambahannya, yaitu :
 - Penambahan data di awal linked list
 - Pengambilan data di posisi tertentu (penyisipan)
 - Penambahan data di akhir linked list



PENAMBAHAN DATA (DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Penambahan data di awal dilakukan dengan algoritma sebagai berikut:
 1. Buat node baru (asumsikan namanya **newNode**)
 2. Sambungkan field **next** dari node baru (**newNode**) ke node pertama linked list
 3. Pindahkan node pertama linked list ke node baru (**newNode**)

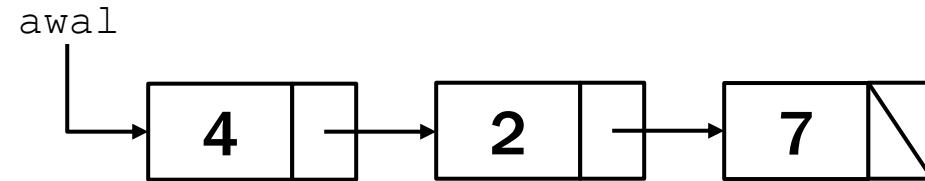


PENAMBAHAN DATA (DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Ilustrasi penambahan data di awal linked list
 - Contoh menambahkan data 9 di awal linked list

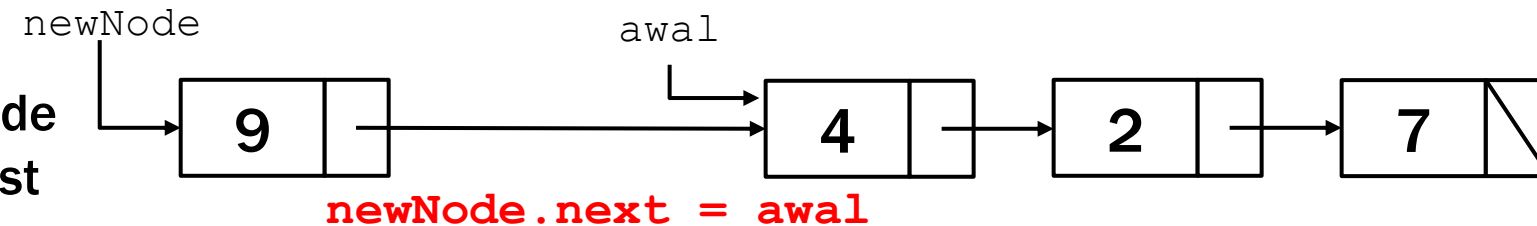
Kondisi awal



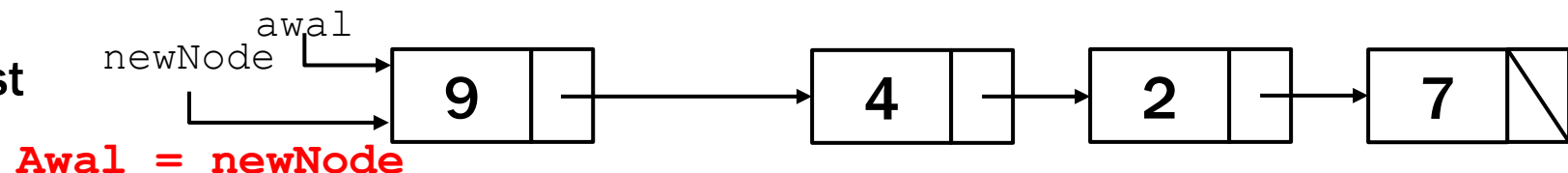
Buat Node Baru



Sambungkan next node baru ke awal linked list



Pindahkan node pertama linked list ke node baru



Oleh : Andri Haryandi, M.T.

PENAMBAHAN DATA (DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Procedure penambahan data di awal linked list (**addFirst**)

```
def addFirst(self, data):  
    newNode = Node(data)  
    newNode.next = self.awal  
    self.awal = newNode
```

- Keterangan :
 - Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



PENAMBAHAN DATA (DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Contoh pemanggilan operasi penambahan data di awal (**addFirst**)

```
list1 = LinkedList() # buat linked list </>
list1.display()
list1.addFirst(7)
list1.display()
list1.addFirst(2)
list1.display()
list1.addFirst(4)
list1.display()
list1.addFirst(9)
list1.display()
```

Isi Linked List : [Data Kosong]

Isi Linked List : 7

Isi Linked List : 2 7

Isi Linked List : 4 2 7

Isi Linked List : 9 4 2 7



PENAMBAHAN DATA (DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Penambahan data di akhir dilakukan dengan algoritma sebagai berikut:
 1. Periksa apakah linked list dalam keadaan kosong
 2. Jika linked list dalam keadaan kosong, maka data baru ditambahkan di awal linked list (**addFirst**)
 3. Jika linked list dalam keadaan tidak kosong maka lakukan langkah 4 sampai 6
 4. Buat node baru (asumsikan namanya **newNode**)
 5. Cari node terakhir (gunakan **getLast()**), dan simpan dalam variable pointer **last**.
 6. Sambungkan field **next** dari node terakhir (**last**) ke node baru (**newNode**)

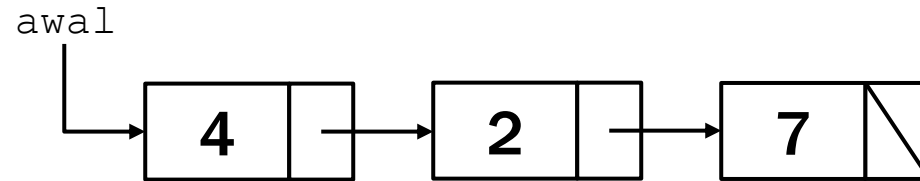


PENAMBAHAN DATA (DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Ilustrasi penambahan data di akhir linked list
 - Contoh menambahkan data 9 di akhir linked list

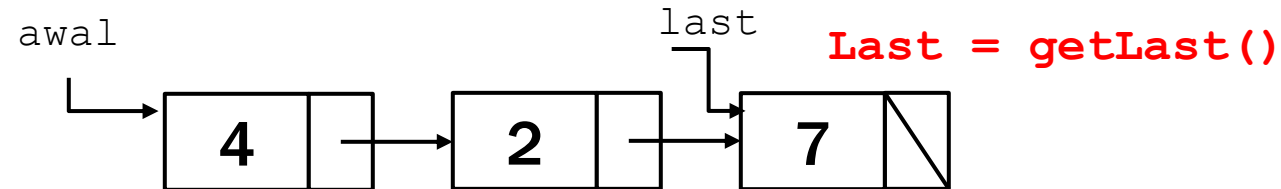
Kondisi awal



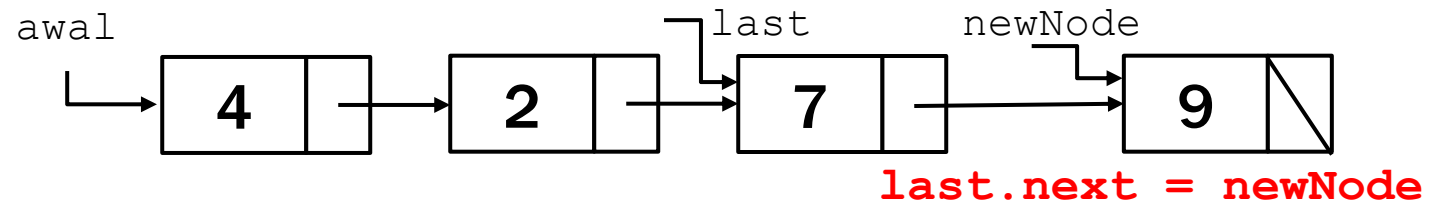
Buat Node Baru



Cari node terakhir



Sambungkan next
dari node terakhir ke
node baru



Oleh : Andri Heryandi, M.T.

PENAMBAHAN DATA (DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Procedure penambahan data di akhir linked list (**addLast**)

```
def addLast(self, data):  
    if self.isEmpty():  
        self.addFirst(data)  
    else:  
        last = self.getLast()  
        newNode = Node(data)  
        last.next = newNode
```

- Keterangan :

- Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



PENAMBAHAN DATA (DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Contoh pemanggilan procedure penambahan data di akhir (**addLast**)

```
list1 = LinkedList() # buat linked list </>
list1.display()
list1.addLast(4)
list1.display()
list1.addLast(2)
list1.display()
list1.addLast(7)
list1.display()
list1.addLast(9)
list1.display()
```

Isi Linked List : [Data Kosong]

Isi Linked List : 4

Isi Linked List : 4 2

Isi Linked List : 4 2 7

Isi Linked List : 4 2 7 9



PENAMBAHAN DATA (DI POSISI TERTENTU)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Penambahan data di posisi tertentu di dalam linked list dilakukan dengan algoritma sebagai berikut:
 1. Periksa apakah posisi index sisip sama dengan 1, jika benar maka lakukan penambahan data di awal (**addFirst**), jika tidak, lanjutkan ke proses 2 dan seterusnya.
 2. Periksa apakah posisi index sisip lebih dari 1 dan kurang dari atau sama dengan banyaknya data dalam linked list (**size(list)**), jika benar maka lakukan langkah 3 s.d 6, jika tidak maka tampilkan pesan bahwa “Posisi sisip tidak sah”
 3. Ambil node sebelum node posisi sisip (posisi **index-1**), asumsikan namanya **prevNode**
 4. Buat node baru dan isi datanya. Asumsikan namanya adalah **newNode**
 5. Sambungkan **next** dari node baru (newNode) ke **next** dari **prevNode**
 6. Sambungkan **next** dari **prevNode** ke node baru (**newnode**)

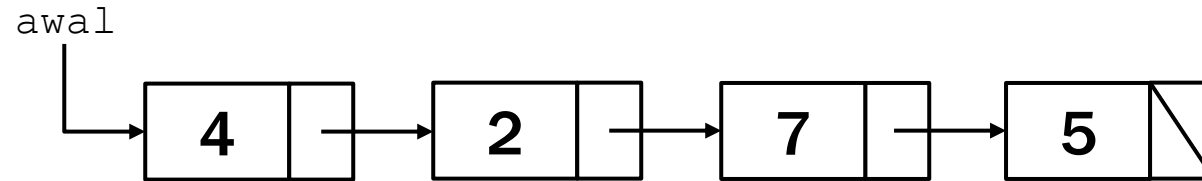


PENAMBAHAN DATA (DI POSISI TERTENTU)

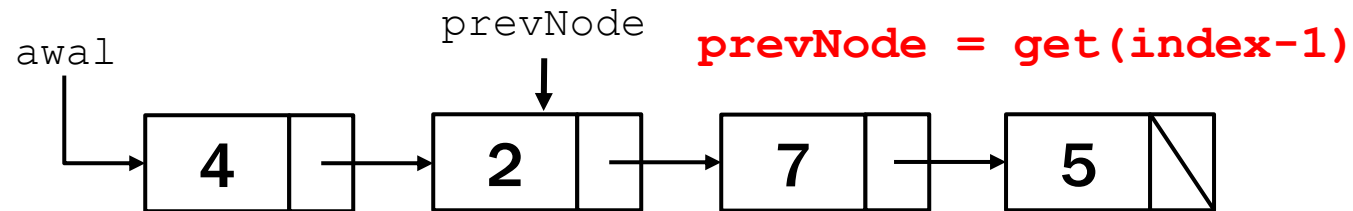
01158 - ALGORITMA DAN STRUKTUR DATA 2

- Ilustrasi penambahan data di posisi tertentu di dalam linked list
 - Contoh menambahkan data 9 di posisi index ke-3 dalam linked list

Kondisi awal



Cari node sebelum
node posisi sisip



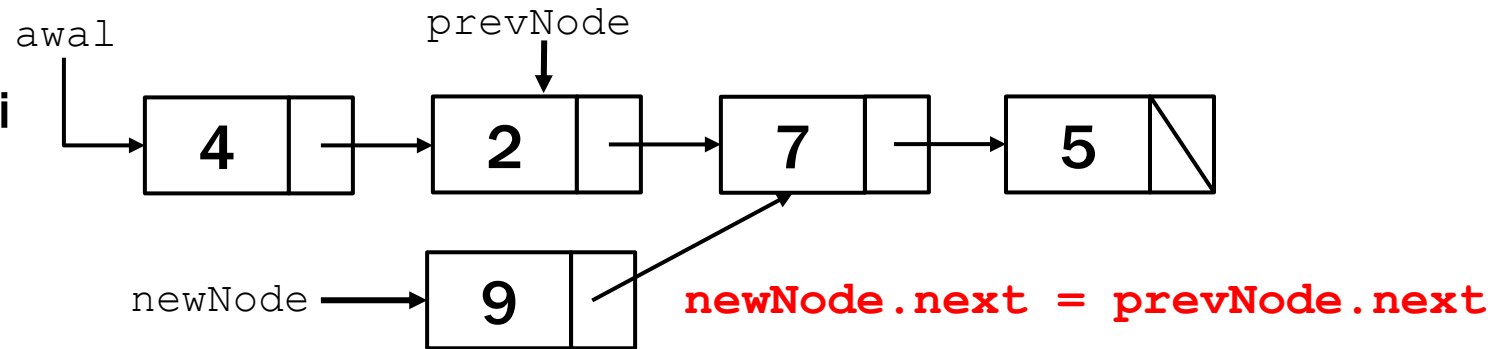
Buat Node baru



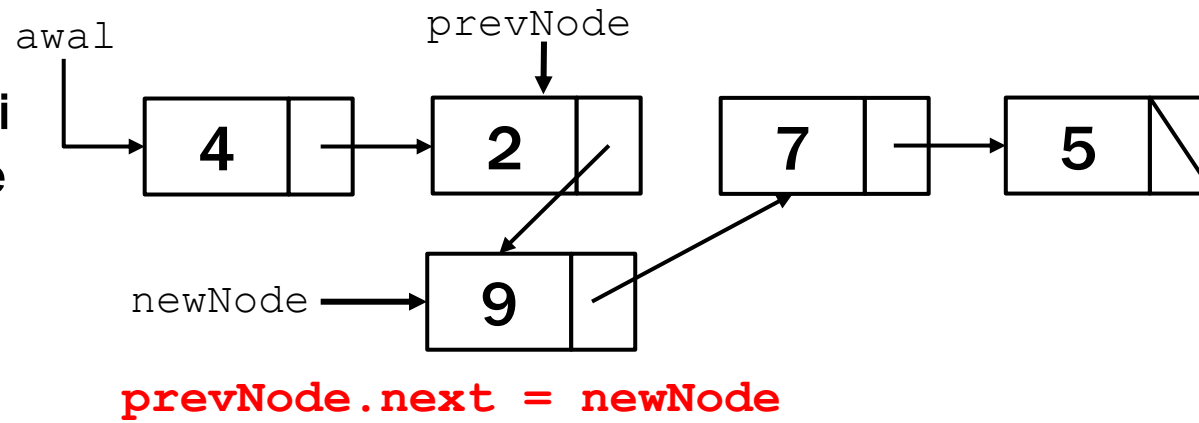
PENAMBAHAN DATA (DI POSISI TERTENTU)

01158 - ALGORITMA DAN STRUKTUR DATA 2

Sambungkan next dari
newNode ke next dari
prevNode



Sambungkan next dari
prevNode ke newNode



PENAMBAHAN DATA (DI POSISI TERTENTU)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Procedure penambahan data di posisi tertentu di dalam linked list (**add**)

```
def add(self, index, data):  
    if index==1:  
        self.addFirst(data)  
    elif index>1 and index<=self.size():  
        prevNode = self.get(index-1)  
        newNode = Node(data)  
        newNode.next = prevNode.next  
        prevNode.next = newNode  
    else:  
        print("Posisi Sisip Invalid")
```

- Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



PENAMBAHAN DATA (DI POSISI TERTENTU)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Contoh pemanggilan operasi penambahan data di posisi tertentu (**add**)

```
list1 = LinkedList() # buat linked list </>
list1.display()
list1.add(1,4) # sisipkan 4 di posisi 1
list1.display()
list1.addLast(2)
list1.addLast(7)
list1.addLast(5)
list1.display()
list1.add(3,9) # sisipkan 9 di posisi 3
list1.display()
list1.add(300,99) # sisipkan 99 di posisi 300
list1.display()
```

Isi Linked List : [Data Kosong]

Isi Linked List : 4

Isi Linked List : 4 2 7 5

Isi Linked List : 4 2 9 7 5

Posisi Sisip Invalid

Isi Linked List : 4 2 9 7 5



UPDATE DATA

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Update data adalah proses mengubah data yang ada pada sebuah node dalam linked list.
- Perubahan data tidak mengubah struktur urutan node.
- Algoritma untuk melakukan update data dalam linked list adalah :
 1. Cari node sesuai index posisi node yang akan diupdate (asumsikan nama variabelnya **nodeUpdate**)
 2. Jika node tidak ditemukan maka tuliskan pesan bahwa “Data tidak ditemukan”
 3. Jika node ditemukan maka updatelah data pada node tersebut dengan data barunya (**nodeUpdate.info = databaru**).

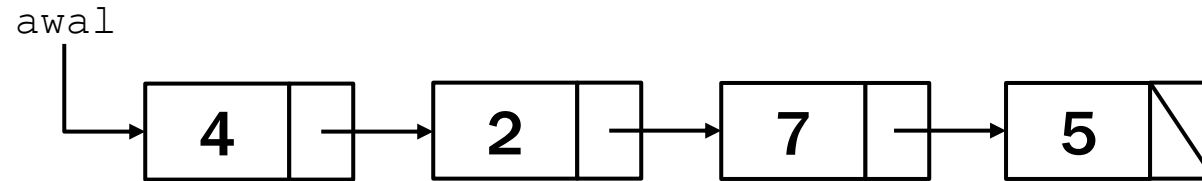


UPDATE DATA

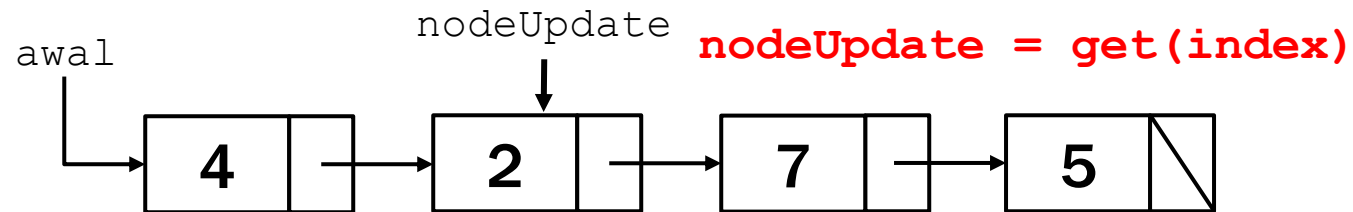
01158 - ALGORITMA DAN STRUKTUR DATA 2

- Ilustrasi pengupdatean data di posisi tertentu di dalam linked list
 - Contoh mengupdate data pada node index ke-2 menjadi 9

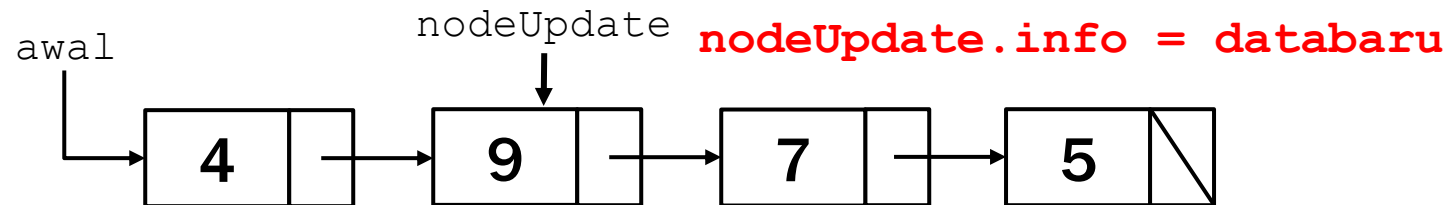
Kondisi awal



Cari node pada node posisi update



Cari node pada node posisi update



UPDATE DATA

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Procedure update data di posisi tertentu di dalam linked list (**update**)

```
def update(self, index, data):  
    nodeUpdate = self.get(index)  
    if nodeUpdate is None:  
        print("Data tidak ditemukan")  
    else:  
        nodeUpdate.info = data
```

- Keterangan :
 - Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



UPDATE DATA

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Contoh pemanggilan operasi update data di posisi tertentu (**update**)

```
list1 = LinkedList()  
list1.addLast(4)  
list1.addLast(2)  
list1.addLast(7)  
list1.addLast(5)  
list1.display()  
list1.update(2, 9)  
list1.display()  
list1.update(200, 99)  
list1.display()
```



Isi Linked List : 4 2 7 5

Isi Linked List : 4 9 7 5

Data tidak ditemukan

Isi Linked List : 4 9 7 5



PENGHAPUSAN DATA

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Penghapusan data dalam linked list adalah proses menghilangkan sebuah node dari linked list dan mengembalikan kembali alokasi memori yang telah digunakan (deallocation).
- Berbeda dengan array statis yang tidak bisa dihapus dari memori, seluruh data dalam linked list bisa benar-benar dihapus dari memori, sehingga penggunaan memori bisa dikembalikan.
- Ada beberapa cara penghapusan data di linked list
 - Penghapusan data di awal linked list
 - Penghapusan data di posisi tertentu di dalam linked list
 - Penghapusan data di akhir linked list
 - Penghapusan seluruh data dalam linked list.



PENGHAPUSAN DATA (DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Penghapusan data di awal linked list berguna untuk menghapus data pada posisi awal linked list dan mengembalikan kembali memori yang telah digunakan.
- Algoritma untuk melakukan penghapusan di posisi awal linked list adalah :
 1. Periksa apakah linked list dalam kondisi kosong atau tidak.
 2. Jika linked list dalam kondisi kosong maka tuliskan bahwa “Penghapusan gagal karena data kosong”.
 3. Jika linked list tidak kosong, maka lakukan langkah 4 s.d 6
 4. Simpan posisi awal linked list ke sebuah pointer baru (diasumsikan bernama **first**).
 5. Pindahkan posisi awal linked list ke node berikutnya.
 6. Hapus node yang ditunjuk oleh pointer **first** dari memori (dealokasi).

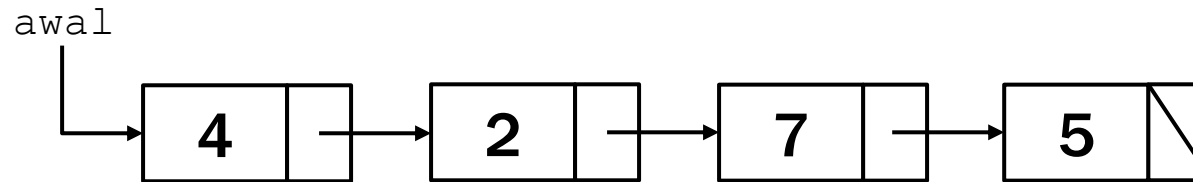


PENGHAPUSAN DATA (DI AWAL LINKED LIST)

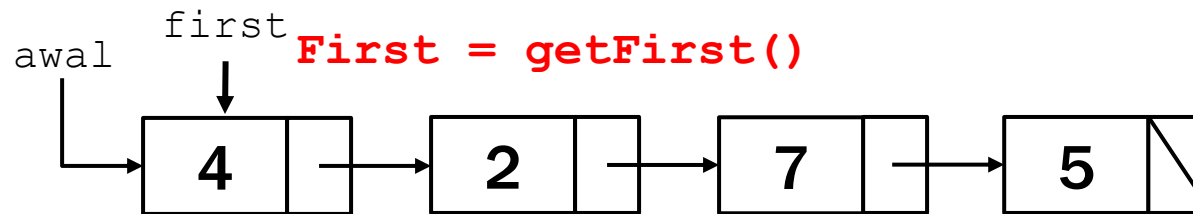
01158 - ALGORITMA DAN STRUKTUR DATA 2

■ Ilustrasi penghapusan data di awal linked list

Kondisi awal



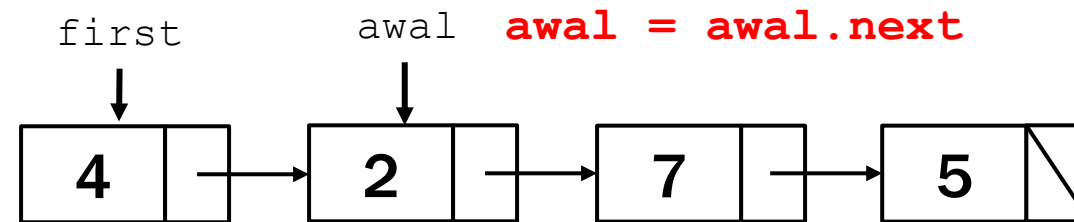
Simpan posisi awal
sekarang ke variable
lain (first)



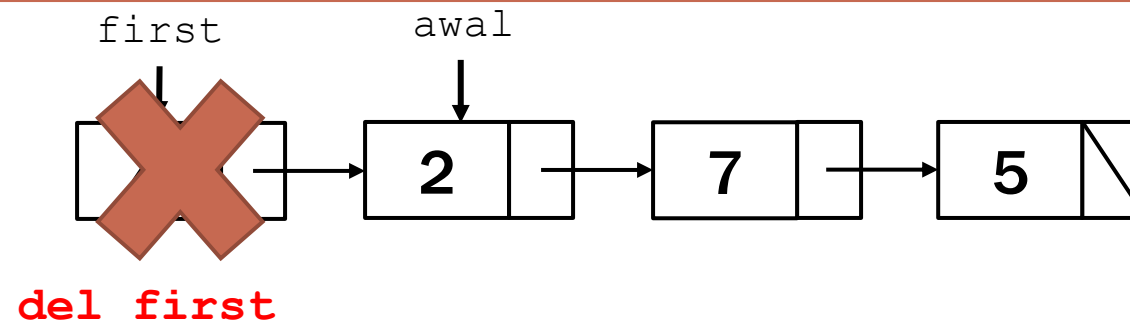
PENGHAPUSAN DATA (DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

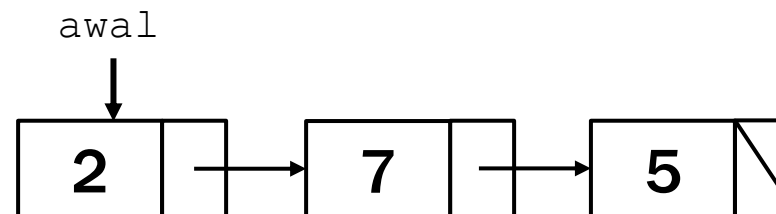
Pindahkan awal linked list ke node berikutnya



Hapus node yang ditunjuk oleh first.



Kondisi setelah penghapusan



PENGHAPUSAN DATA (DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Procedure hapus node di awal linked list (**removeFirst**)

```
def removeFirst(self):  
    if self.isEmpty():  
        print("Penghapusan gagal karena list kosong")  
    else:  
        first = self.getFirst()  
        self.awal = first.next  
        del first
```

- Keterangan :

- Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



PENGHAPUSAN DATA (DI AWAL LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Contoh pemanggilan operasi hapus node di awal linked list (**removeFirst**)

```
list1 = LinkedList()
list1.addLast(4)
list1.addLast(2)
list1.addLast(7)
list1.addLast(5)
list1.display()
list1.removeFirst()
list1.display()
list1.removeFirst()
list1.display()
list1.removeFirst()
list1.display()
list1.removeFirst()
list1.display()
list1.removeFirst()
```



Isi Linked List : 4 2 7 5

Isi Linked List : 2 7 5

Isi Linked List : 7 5

Isi Linked List : 5

Isi Linked List : [Data Kosong]

Penghapusan gagal karena list
kosong



PENGHAPUSAN DATA (DI POSISI TERTENTU DI DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Penghapusan data di posisi tertentu di dalam linked list berguna untuk menghapus data pada posisi tertentu yang ada di dalam linked list dan mengembalikan kembali memori yang telah digunakan.
- Algoritma untuk melakukan penghapusan di posisi posisi tertentu dalam linked list adalah :
 1. Periksa apakah linked list dalam kondisi kosong atau tidak.
 2. Jika linked list dalam kondisi kosong maka tuliskan bahwa “Penghapusan gagal karena data kosong”.
 3. Jika linked list tidak kosong, maka lakukan langkah 4 s.d 9
 4. Periksa apakah index posisi hapus sama dengan 1. Jika benar maka lakukan penghapusan node depan (**removeFirst**). Tetapi jika index posisi hapus tidak sama dengan 1 (setelah 1) maka lakukan langkah 5 s.d 9.
 5. Cari node sebelum node yang akan dihapus (**index-1**). Simpan dalam sebuah pointer bernama **prevNode**.
 6. Jika node posisi sebelum node dihapus tidak ditemukan, maka tampilkan pesan bahwa “Posisi hapus tidak ditemukan”. Tetapi jika posisi sebelum node yang akan dihapus ditemukan maka lakukan langkah 7 s.d 9
 7. Simpan node yang akan dihapus (didapat dari **prevNode^.next**) ke sebuah variable bernama **removedNode**.
 8. Pindahkan **next** dari **prevNode** ke **next** dari **removedNode**. Ini dilakukan agar node yang ditunjuk oleh **removedNode** terlepas dari linked list.
 9. Hapus node yang ditunjuk oleh **removedNode** dari memori.

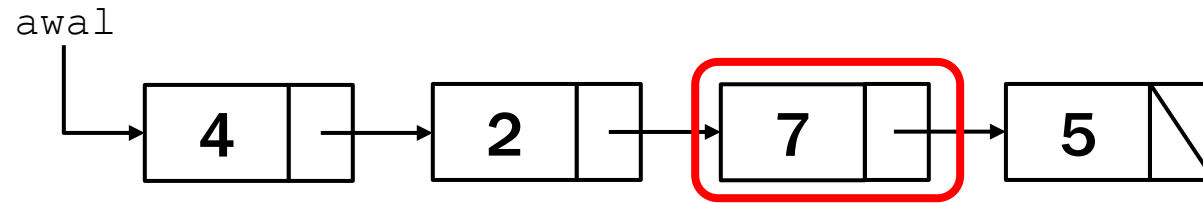


PENGHAPUSAN DATA (DI POSISI TERTENTU DI DALAM LINKED LIST)

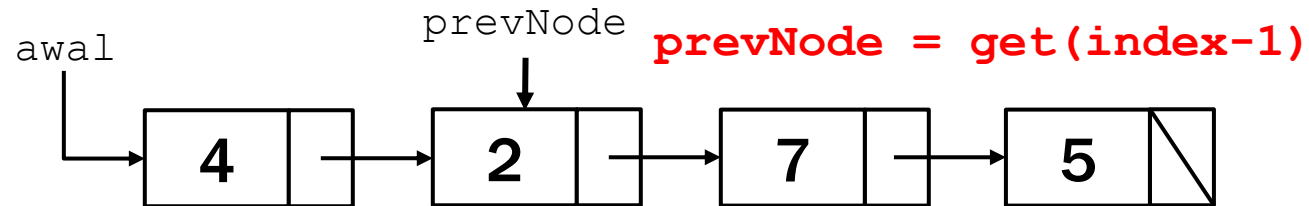
01158 - ALGORITMA DAN STRUKTUR DATA 2

- Ilustrasi penghapusan data di posisi tertentu di dalam linked list
 - Contoh : menghapus data pada index ke-3

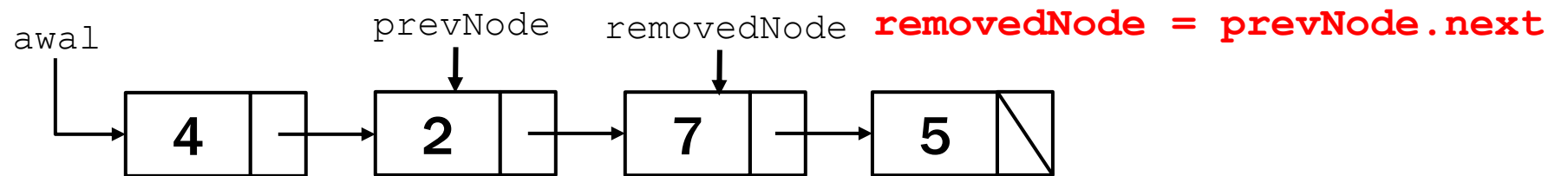
Kondisi awal



Cari node sebelum
node yang dihapus
(prevNode).



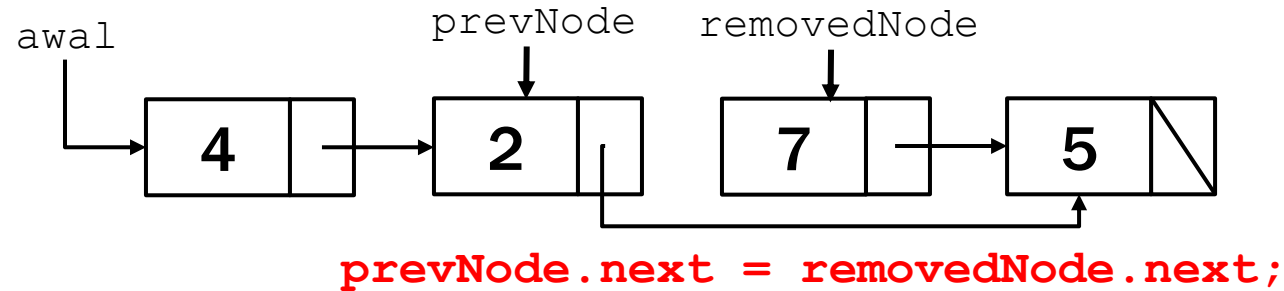
Cari node yang akan
dihapus
(removedNode).



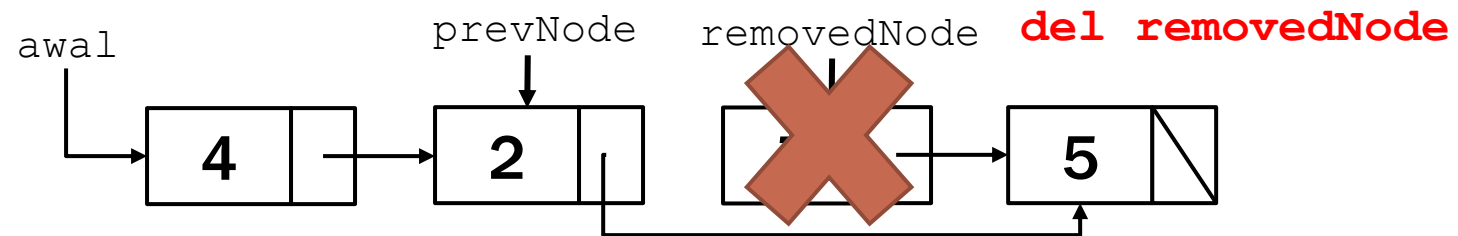
PENGHAPUSAN DATA (DI POSISI TERTENTU DI DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

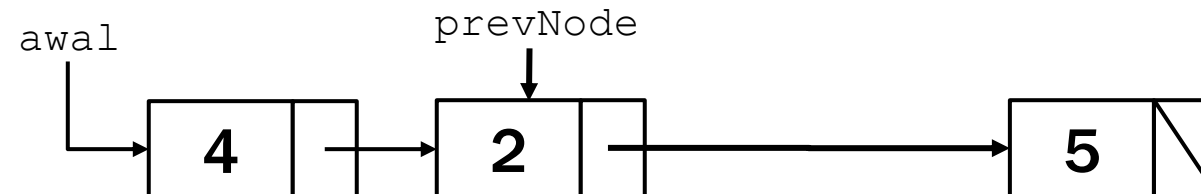
Pindahkan next dari
prevNode ke next dari
removedNode



Hapus node
removedNode



Kondisi setelah
penghapusan



PENGHAPUSAN DATA (DI POSISI TERTENTU DI DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Procedure hapus node di posisi tertentu di dalam linked list (**remove**)

```
def remove(self, index):  
    if self.isEmpty():  
        print("Penghapusan gagal karena list kosong")  
    else:  
        if index==1:  
            self.removeFirst()  
        else:  
            prevNode = self.get(index-1)  
            if prevNode is None:  
                print("Penghapusan gagal karena index tidak ditemukan")  
            else:  
                deletedNode = prevNode.next  
                prevNode.next = deletedNode.next  
                del deletedNode
```

- Keterangan :
 - Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



PENGHAPUSAN DATA (DI POSISI TERTENTU DI DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Contoh pemanggilan procedure hapus node di posisi tertentu (**remove**)

```
list1 = LinkedList()
list1.addLast(4)
list1.addLast(2)
list1.addLast(7)
list1.addLast(5)
list1.display()
list1.remove(3)
list1.display()
list1.remove(2)
list1.display()
list1.remove(1)
list1.display()
list1.remove(3)
list1.display()
```



Isi Linked List : 4 2 7 5

Isi Linked List : 4 2 5

Isi Linked List : 4 5

Isi Linked List : 5

Penghapusan gagal karena index
tidak ditemukan

Isi Linked List : 5



PENGHAPUSAN DATA (DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Penghapusan data di posisi linked list berguna untuk menghapus data pada posisi tertentu yang ada di dalam linked list dan mengembalikan kembali memori yang telah digunakan.
- Langkah-langkah penghapusan node pada posisi akhir linked list sama dengan penghapusan pada posisi tertentu. Perbedaannya adalah posisi hapusnya selalu pada index terakhir (`size(list)`)

`removeLast()`
`=`
`remove(size(list))`

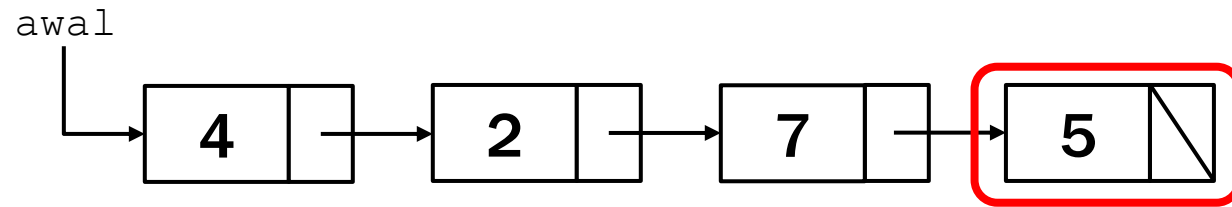


PENGHAPUSAN DATA (DI AKHIR LINKED LIST)

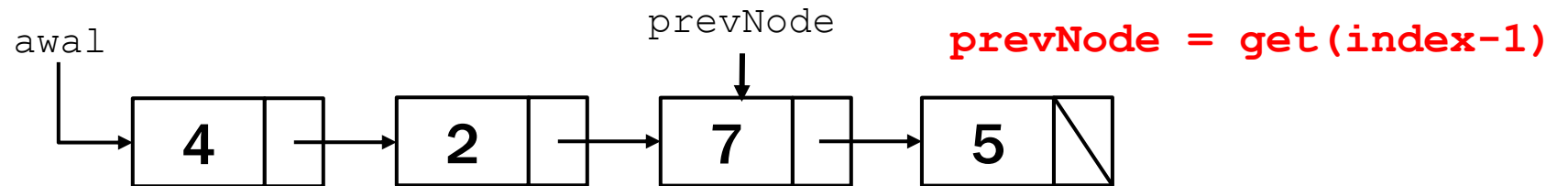
01158 - ALGORITMA DAN STRUKTUR DATA 2

- Ilustrasi penghapusan data di akhir linked list
 - Contoh : menghapus node terakhir (index ke-4)

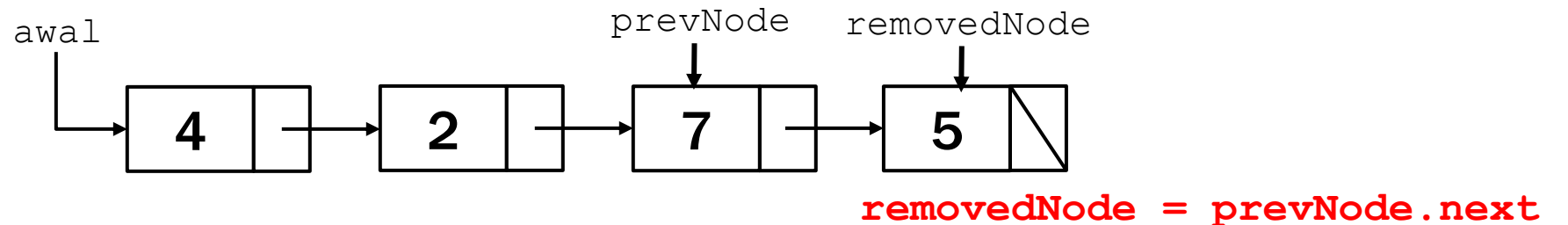
Kondisi awal



Cari node sebelum
node yang dihapus
(prevNode).



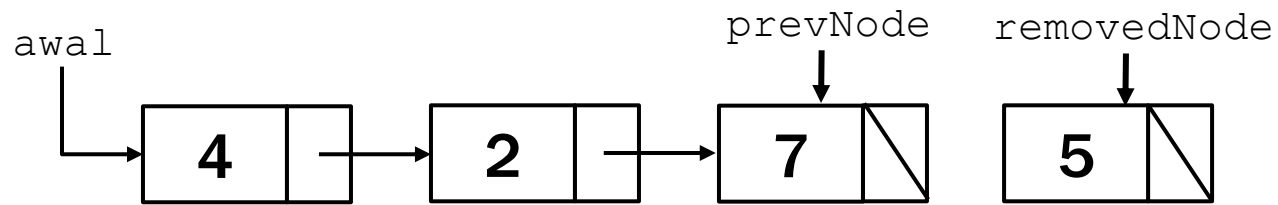
Cari node yang akan
dihapus
(removedNode).



PENGHAPUSAN DATA (DI AKHIR LINKED LIST)

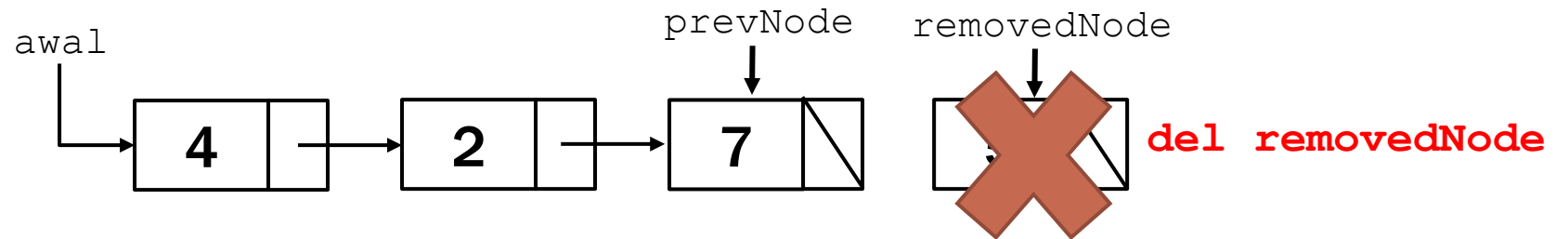
01158 - ALGORITMA DAN STRUKTUR DATA 2

Pindahkan next dari
prevNode ke next dari
removedNode

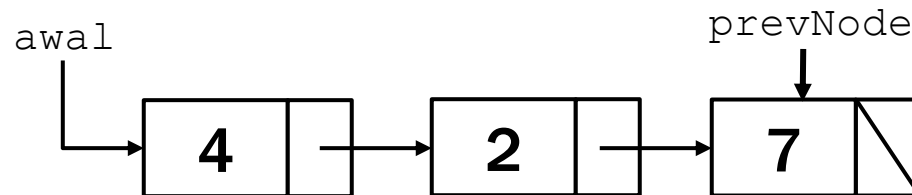


prevNode.next = removedNode.next

Hapus node
removedNode



Kondisi setelah
penghapusan



PENGHAPUSAN DATA (DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Operasi hapus node di posisi terakhir di dalam linked list (**removeLast**)

```
def removeLast(self):  
    self.remove(self.size())
```

- Keterangan :
 - Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



PENGHAPUSAN DATA (DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Contoh pemanggilan procedure hapus node di posisi terakhir (**removeLast**)

```
list1 = LinkedList()  
list1.addLast(4)  
list1.addLast(2)  
list1.addLast(7)  
list1.addLast(5)  
list1.display()  
list1.removeLast()  
list1.display()  
list1.removeLast()  
list1.display()
```



```
Isi Linked List : 4  2  7  5  
Isi Linked List : 4  2  7  
Isi Linked List : 4  2
```



PENGHAPUSAN DATA (SELURUH DATA DALAM LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Penghapusan seluruh data di posisi linked list berguna untuk mengembalikan seluruh memori yang digunakan oleh linked list.
- Proses ini biasanya dilakukan ketika data dalam linked list sudah tidak diperlukan lagi.
- Contoh waktu penghapusan seluruh data dalam linked list adalah ketika program telah selesai dieksekusi, maka tepat sebelum berakhirnya program ada baiknya semua data yang ada dihapus terlebih dahulu.
- Algoritma untuk melakukan penghapusan seluruh data dalam linked list adalah :
 1. Selama linked list masih belum kosong maka lakukan langkah 2
 2. Hapus node pada posisi awal (**removeFirst**).

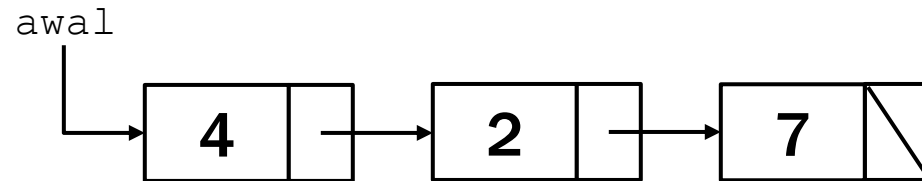


PENGHAPUSAN DATA (SELURUH DATA DALAM LINKED LIST)

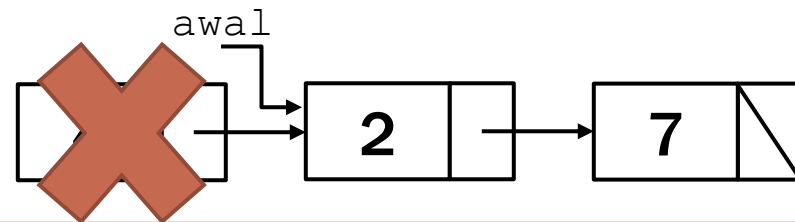
01158 - ALGORITMA DAN STRUKTUR DATA 2

- Ilustrasi penghapusan seluruh data di linked list

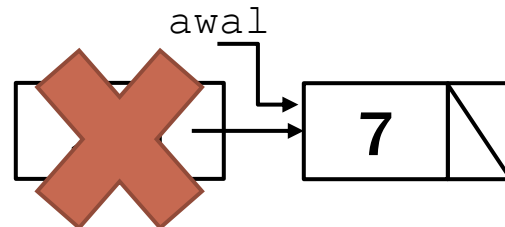
Kondisi awal



Jika linked list belum kosong, maka hapus node depan



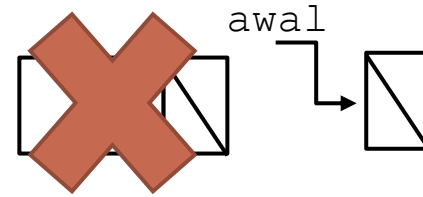
Jika linked list belum kosong, maka hapus node depan



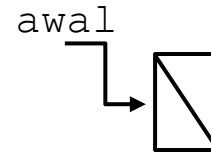
PENGHAPUSAN DATA (DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

Jika linked list belum
kosong, maka hapus
node depan



Kondisi setelah
penghapusan



PENGHAPUSAN DATA (DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Method/Function hapus semua node di dalam linked list (**removeAll**)

```
def removeAll(self):  
    while not self.isEmpty():  
        self.removeFirst();
```

- Keterangan :
 - Method/Function ini harus ditulis dalam pendefinisian class LinkedList.



PENGHAPUSAN DATA (DI AKHIR LINKED LIST)

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Contoh pemanggilan operasi hapus semua node di dalam linked list (**removeAll**)

```
list1 = LinkedList()
list1.addLast(4)
list1.addLast(2)
list1.addLast(7)
list1.addLast(5)
list1.display()
list1.removeAll()
list1.display()
print("Awal list1 : ", list1.awal)
```



Isi Linked List : 4 2 7 5

Isi Linked List : [Data Kosong]

Awal list1 : None



SOURCE CODE

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Source code class Node dan LinkedList dapat didownload di

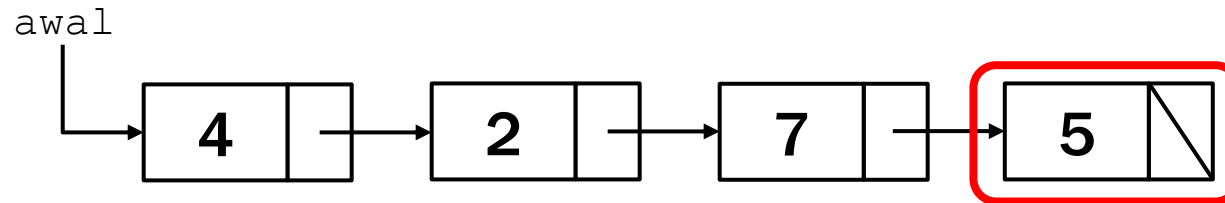
<https://github.com/andri-heryandi/asdat2/blob/main/linkedlist.py>



LATIHAN 1

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Buatlah suatu method/function linked list untuk melakukan pencarian.
- Pencarian dilakukan dengan melakukan penelusuran linked list sampai menemukan data yang dicari.
- Function ini akan mereturnkan index dari data yang dicari.
- Contoh :

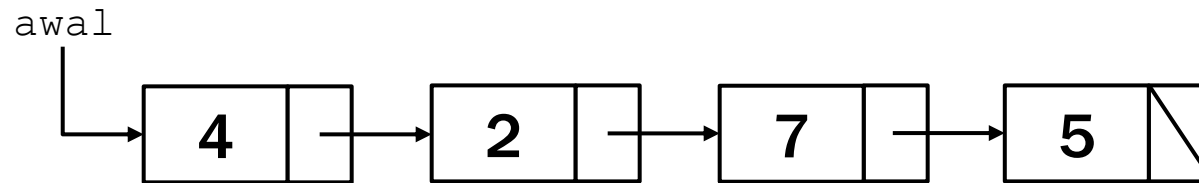


- Kalau anda memanggil “**idx = list1.search(7)**”, maka akan mereturnkan nilai 3
- Kalau anda memanggil “**idx = list1.search(9)**”, maka akan mereturnkan nilai 0 (data tidak ada)

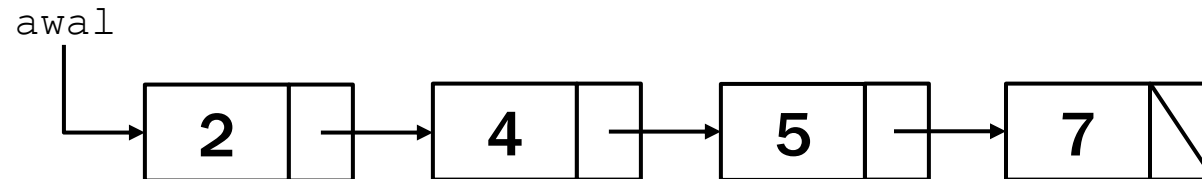
LATIHAN 2

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Buatlah suatu method/function linked list untuk melakukan pengurutan data.
- Function ini akan menghasilkan linked list yang terurut berdasarkan data dalam node secara ascending (menaik).
- Contoh :



- Kalau anda memanggil “**list1.sort()**”, maka akan menghasilkan linked list sebagai berikut:



Next.....

DOUBLE LINKED LIST

OLEH : ANDRI HERYANDI, M.T.



CIRCULAR LINKED LIST

OLEH : ANDRI HERYANDI, M.T.



FORUM DISKUSI

01158 - ALGORITMA DAN STRUKTUR DATA 2



LMS UNIKOM

<https://lms.unikom.ac.id>



**Group Whatsapp
Perkuliahahan**



Youtube Comments



Oleh : Andri Heryandi, M.T.