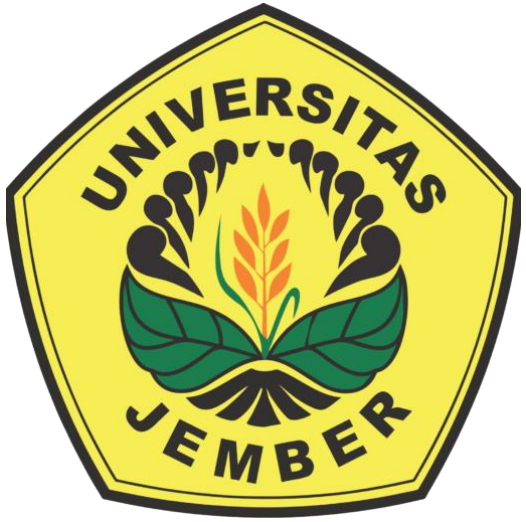


Struktur Data

Linked List

MUHAMMAD ‘ARIFUL FURQON S.PD., M.KOM.

FAKULTAS ILMU KOMPUTER

UNIVERSITAS JEMBER

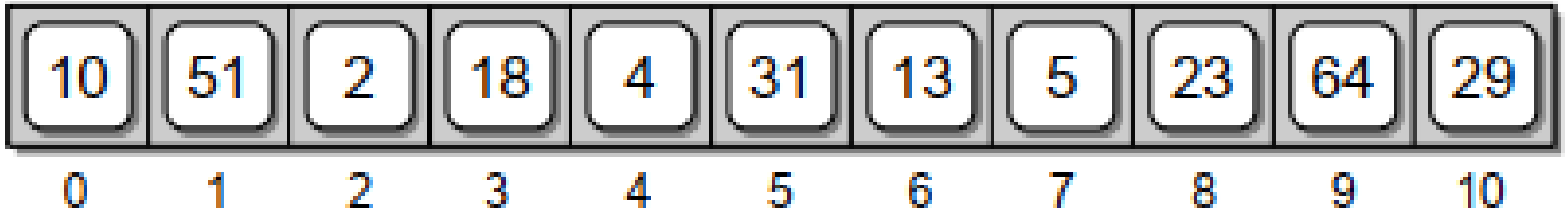
Important Links

Link PPT : https://unej.id/SD06materi

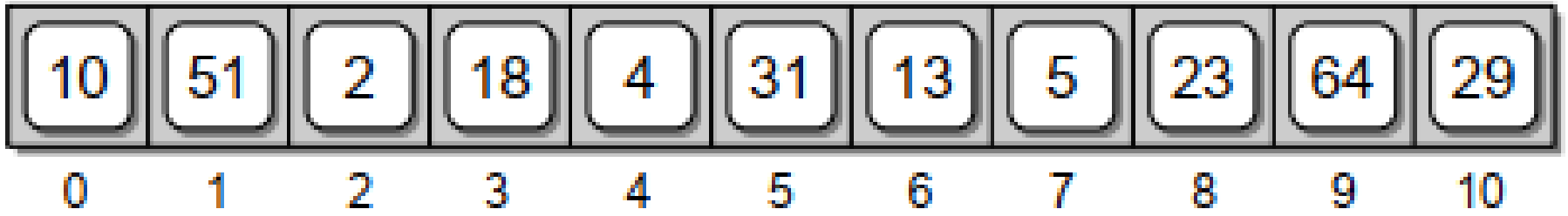
Link Code: https://unej.id/SD06code

Link Record: https://unej.id/zoomSD

Preview – Array



Preview – Array

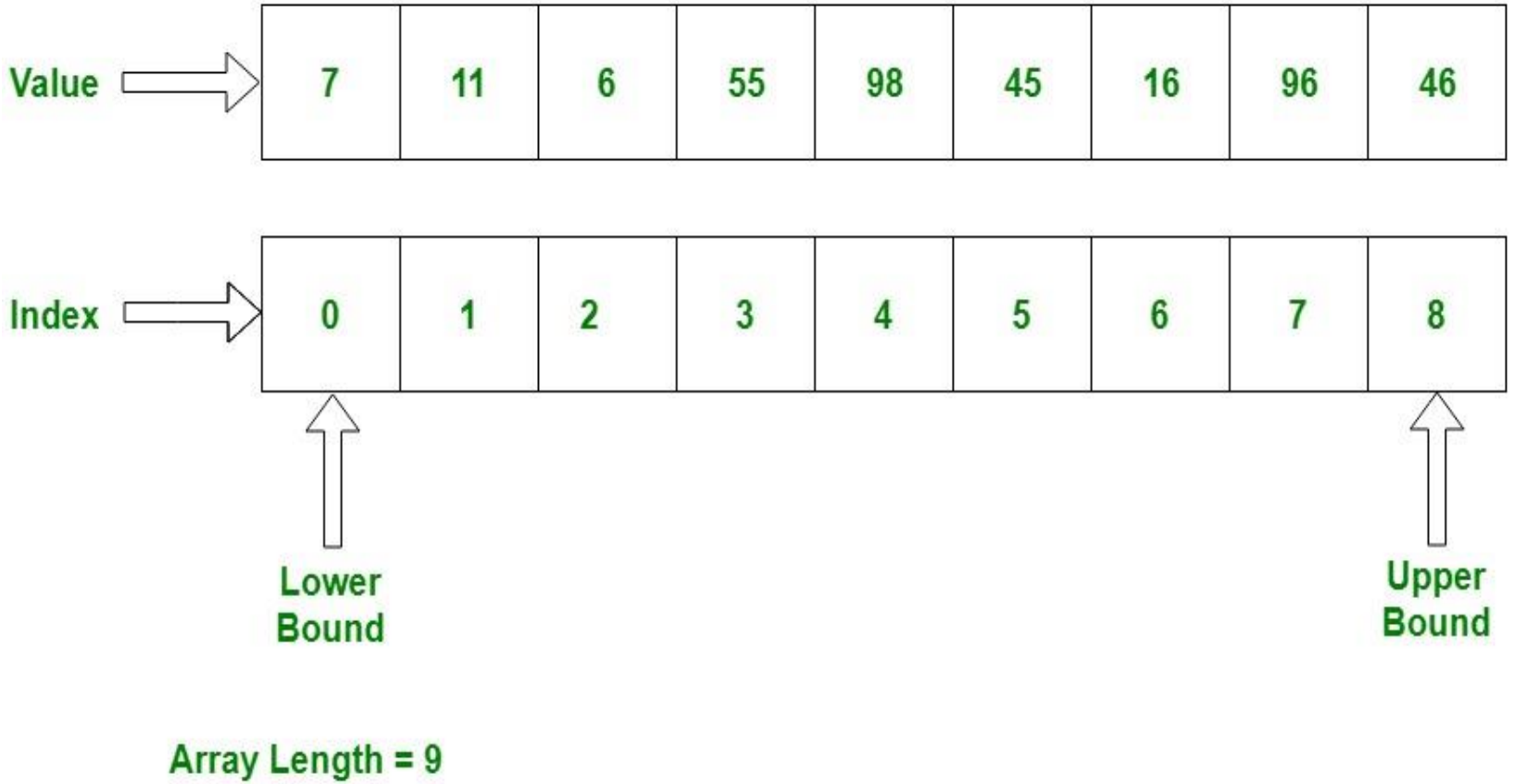


- Sebuah variabel yang bisa menyimpan banyak data.
- Data yang disimpan tipenya sejenis.
- Setiap data diidentifikasi dari indeks : A[0], A[1], …, A[n - 1]. Ada bahasa yang memulai indeksnya dari 0 ada yang mulai dari 1.

Preview – Array

- Pada saat menggunakan array, perlu ditentukan banyaknya anggota array atau elemen array
- Di memori, elemen elemen array disimpan di alamat memory yang secara fisik berurutan.
- Pengakses elemen array membutuhkan waktu yang sama cepat. Waktu untuk mengakses A[0] sama cepat dengan mengakses A[1000]

Preview – Array



Array - NumPy

List cocok digunakan untuk menyimpan data dimensi satu yang kecil.

```
>>> a = [1,3,5,7,9]
>>> print(a[2:4])
[5, 7]
>>> b = [[1, 3, 5, 7, 9], [2, 4, 6, 8, 10]]
>>> print(b[0])
[1, 3, 5, 7, 9]
>>> print(b[1][2:4])
[6, 8]
```

```
>>> a = [1,3,5,7,9]
>>> b = [3,5,6,7,9]
>>> c = a + b
>>> print (c)
[1, 3, 5, 7, 9, 3, 5, 6, 7, 9]
```

- Tetapi tidak bisa digunakan secara langsung dengan operator aritmatika (+, -, *, /, …)
- Membutuhkan array yang efisien dengan aritmatik dan multidimensi.
- **Numpy**

```
>>> import numpy
```

- Mirip dengan lists, tetapi lebih baik untuk handle array

Array - NumPy

```
>>> l = [[1, 2, 3], [3, 6, 9], [2, 4, 6]]  # create a list
>>> a = numpy.array(l)  # convert a list to an array
>>>print(a)
[[1 2 3]
 [3 6 9]
 [2 4 6]]
>>> a.shape
(3, 3)
>>> print(a.dtype)  # get type of an array
int64
# or directly as matrix
>>> M = array([[1, 2], [3, 4]])
>>> M.shape
(2,2)
>>> M.dtype
dtype('int64')
```

Array - NumPy


```
>>> print(a)
[[1 2 3]
 [3 6 9]
 [2 4 6]]
>>> print(a[0])  # this is just like a list of lists
?
>>> print(a[1, 2])  # arrays can be given comma separated indices
?
>>> print(a[1, 1:3])  # and slices
?
>>> print(a[:,1])
?
```

Array - NumPy

```
>>> print(a)
[[1 2 3]
 [3 6 9]
 [2 4 6]]
>>> print(a[0])  # this is just like a list of lists
[1 2 3]
>>> print(a[1, 2])  # arrays can be given comma separated indices
9
>>> print(a[1, 1:3])  # and slices
[6 9]
>>> print(a[:,1])
[2 6 4]
```

Keterbatasan Array

Ukurannya tetap.

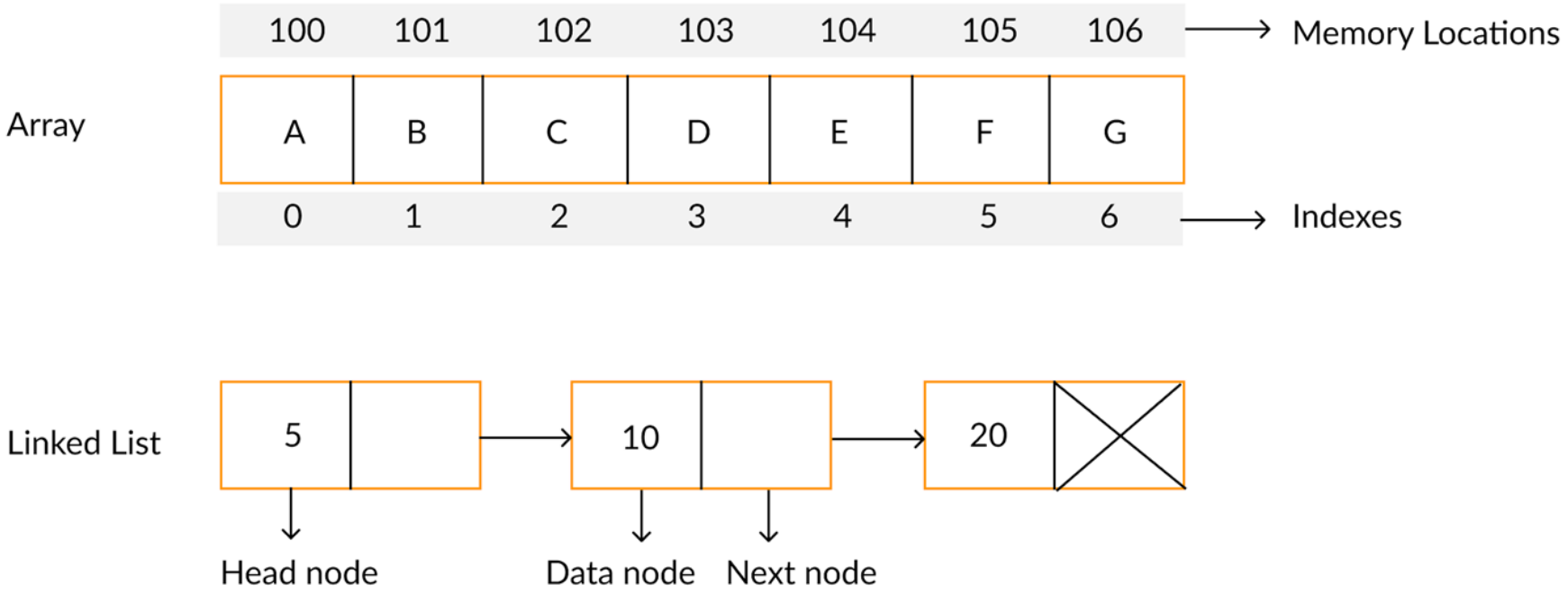
- Tidak bisa menambah elemen jika sudah penuh.
- Jika elemen array banyak yang tidak terpakai, penggunaan memory tidak efisien.

Untuk menambah atau menghapus satu elemen di awal / di tengah harus menggeser banyak elemen array.

Linked List

- Struktur data yang bisa menyimpan banyak elemen.
- Setiap elemen TIDAK dinomori.
- Setiap elemen “menunjuk” elemen selanjutnya.

Array vs. Linked List



Kelebihan Linked Lists

Elemen elemen linked list tidak harus disimpan berurutan di memory

- Bisa menambah atau menghapus elemen di awal / di tengah list tanpa menggeser elemen yang lain

Ukuran dari linked list (banyaknya anggota linked list) bisa berubah secara dinamis.

- Banyaknya elemen list bisa disesuaikan dengan kebutuhan
- Node bisa ditambah / dihapus ketika run time
- Penggunaan memory lebih fleksibel

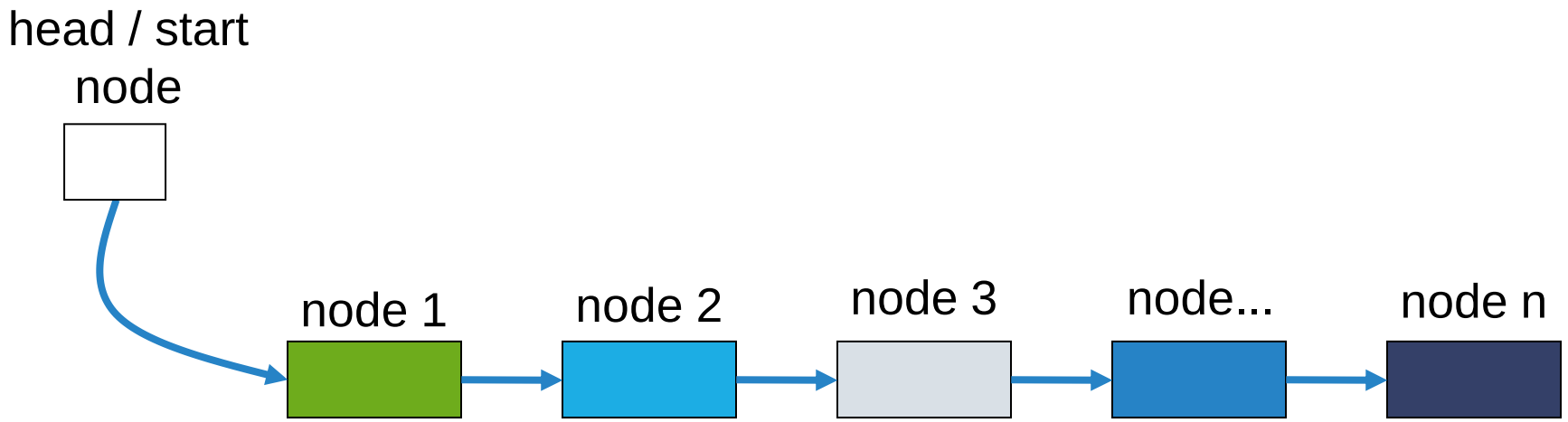
Node

Linked list merupakan rangkaian dari elemen yang disebut **node**.

Sebuah node pada linked list komponen datanya terdiri atas 2 bagian:

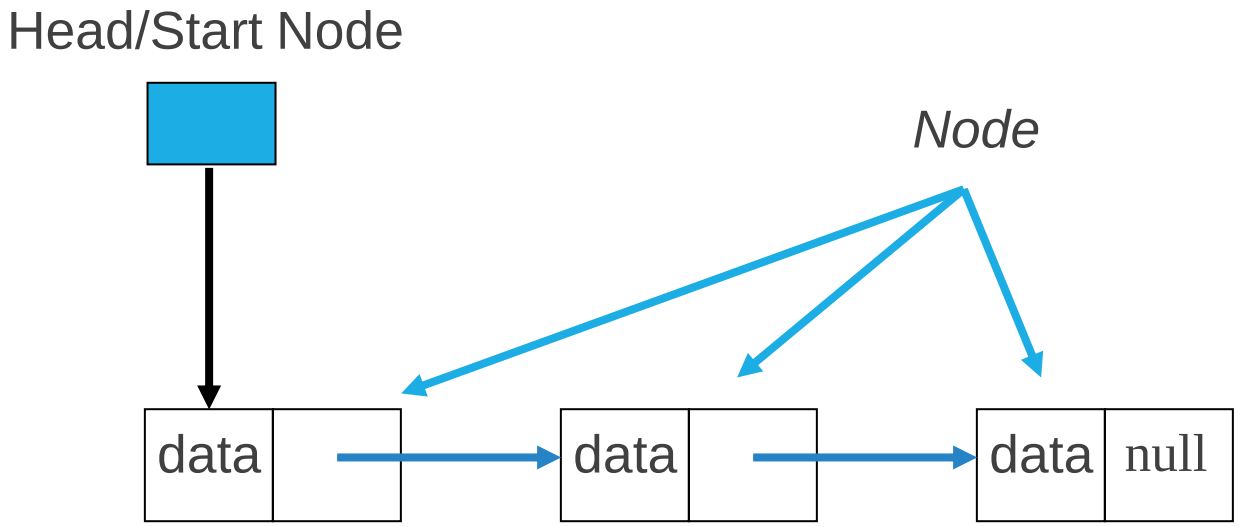
- Komponen untuk **menyimpan data**.
- Komponen yang **menunjuk node** berikutnya pada linked list (next/ref).

Diagram Konseptual Single-Linked List



Single Linked List

Head / start node adalah pointer yang menunjuk node pertama



Single Linked List

- Node yang ada di linked list bisa **ditambah** atau **dihapus** sesuai kebutuhan pada saat run time.
- Node pertama linked list disebut **start node** atau **head**.
- Ketika list kosong start node bernilai **null**.
- Setiap node bagian ref-nya menunjuk ke node selanjutnya.
- Node terakhir linked list nextnya bernilai null.

Node

```
1  class Node:
2      def __init__(self, data):
3          self.item = data
4          self.ref = None
5
6  class LinkedList:
7      def __init__(self):
8          self.start_node = None
```

untuk menyimpan data

menyimpan node selanjutnya

Mula mula bernilai null karena start node tidak menunjuk node manapun

Jika linked list sudah memiliki anggota. Maka start node SELALU menunjuk node pertama

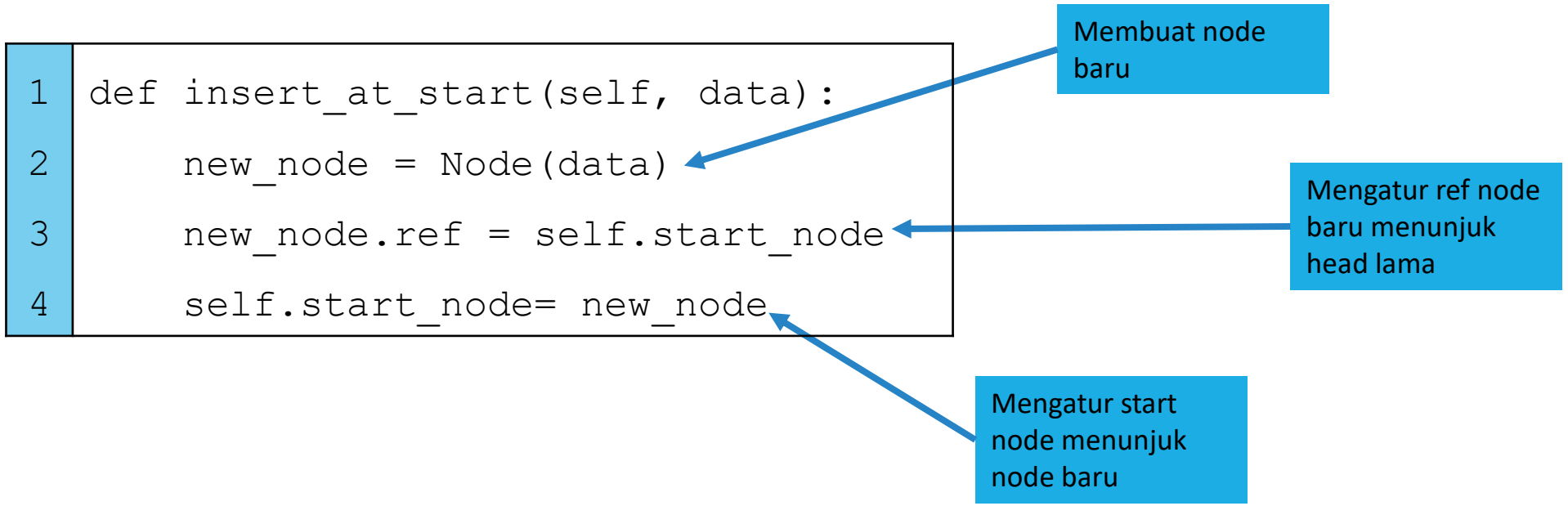
Operasi pada linked list

Operasi pada list

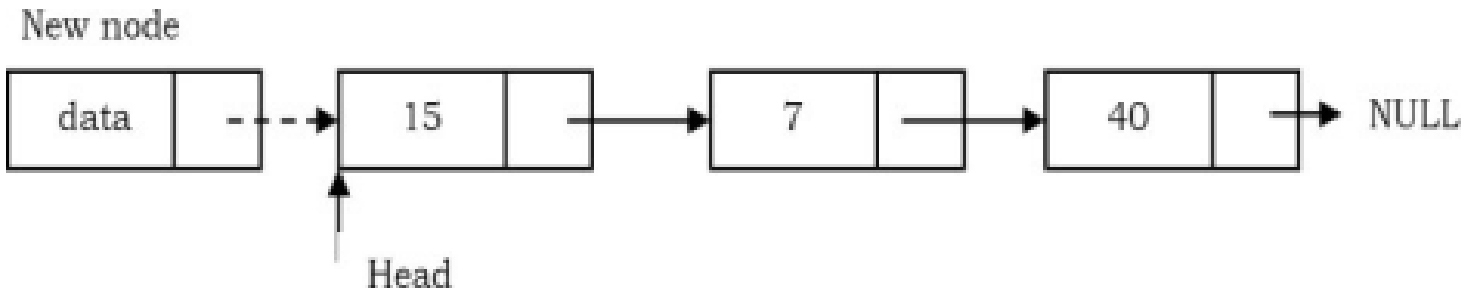
- Menambah node baru di akhir linked list
- Menambah node baru di awal linked list
- Menyisipkan node sebelum node tertentu
- Menyisipkan node setelah node tertentu
- Menyisipkan node di index tertentu
- Menghapus node di awal
- Menghapus node di akhir
- Menghapus node yang memiliki nilai tertentu

Operasi penambahan / penghapusan node **jangan** sampai menyebabkan link / rantai ref pada linked list **terputus**

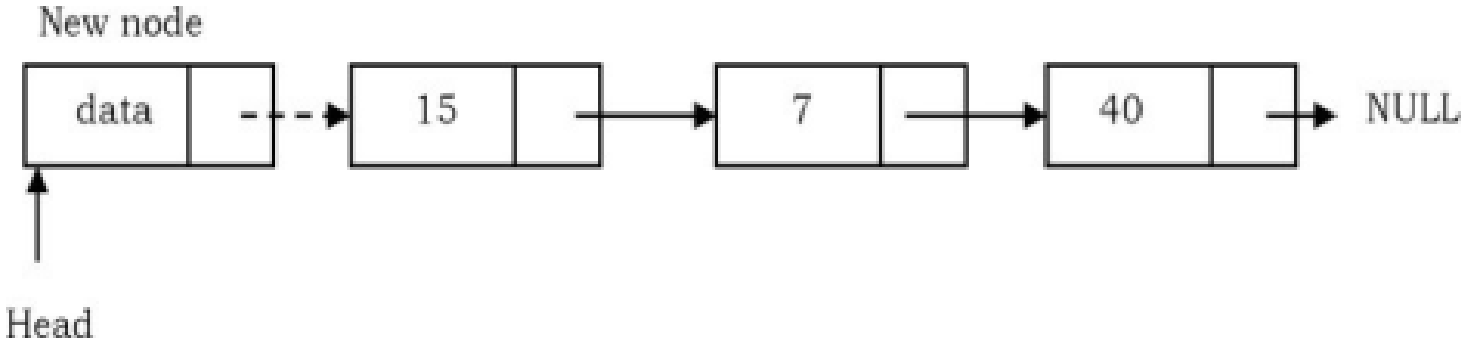
Menambah Node Baru di Awal Linked List



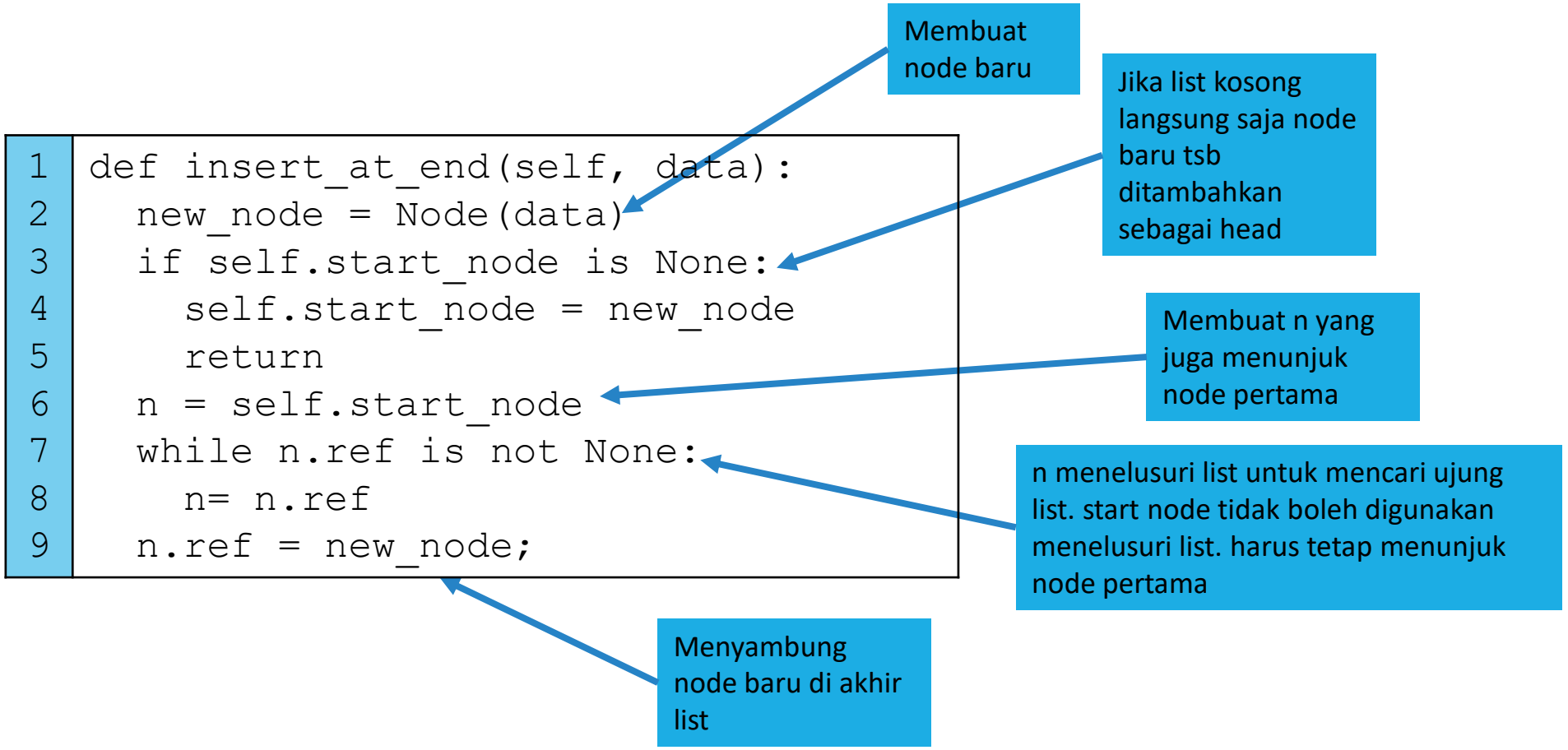
Menambah Node Baru di Awal Linked List



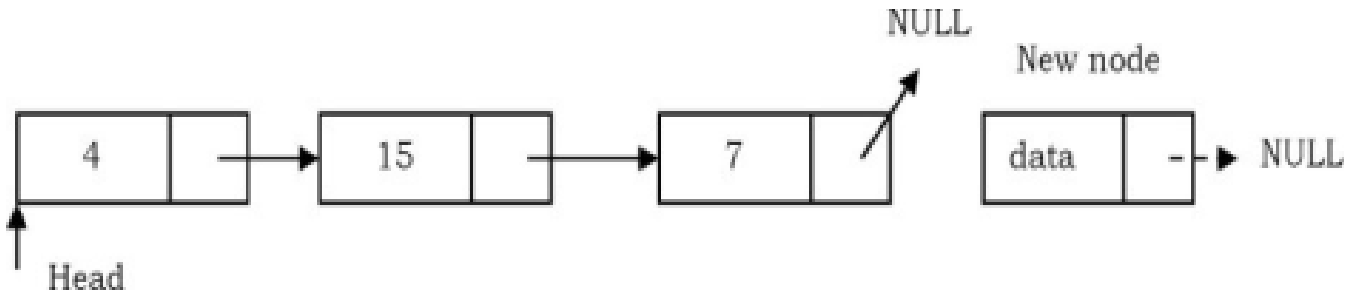
- Update head pointer to point to the new node.



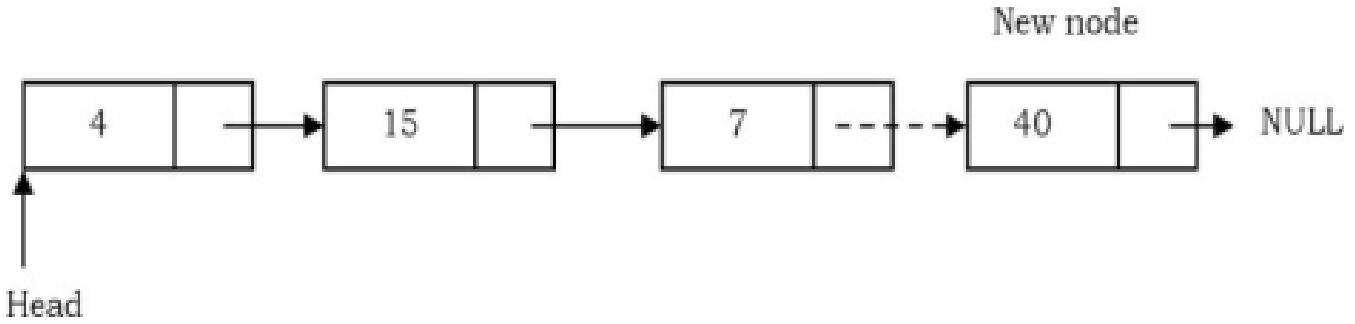
Menambah Node Baru di Akhir Linked List



Menambah Node Baru di Akhir Linked List



- Last nodes next pointer points to the new node.



Mencetak Seluruh Isi List

```
1  def traverse_list(self):
2    if self.start_node is None:
3      print("List has no element")
4      return
5    else:
6      n = self.start_node
7      while n is not None:
8        print(n.item , " ")
9          n = n.ref
```

Memeriksa apakah list kosong

Membuat n yang juga menunjuk node pertama

n menelusuri setiap node di list. mulai dari start node ke node2 selanjutnya

Berpindah ke node selanjutnya

Menambah node baru di akhir linked

```
1   def insert_at_index (self, index, data):
2     if index == 1:
3       new_node = Node(data)
4       new_node.ref = self.start_node
5       self.start_node = new_node
6     i = 1
7     n = self.start_node
8     while i < index-1 and n is not None:
9       n = n.ref
10      i = i+1
11    if n is None:
12      print("Index out of bound")
13    else:
14      new_node = Node(data)
15      new_node.ref = n.ref
16      n.ref = new_node
```

Memasukkan node di index pertama

Jika node diinsertkan di index yang lebih dari 1

Membuat n yang juga menunjuk node pertama

n menelusuri list untuk mencari posisi index

Menyisipkan node di index

Latihan

Buatlah sebuah list kosong (kode kelas linkedlist bisa dilihat di drive)

1. Masukkan data 9 menggunakan operasi tambah di awal. Gambarkan isi list saat ini
2. Masukkan data 2 menggunakan operasi tambah di awal. Gambarkan isi list saat ini
3. Masukkan data 6 menggunakan operasi tambah di akhir. Gambarkan isi list saat ini
4. Masukkan data 11 menggunakan operasi tambah di akhir. Gambarkan isi list saat ini
5. Masukkan data 2 di indeks pertama. Gambarkan isi list saat ini
6. Masukkan data 12 di indeks 4. Gambarkan isi list saat ini
7. Cetaklah isi list

Terima Kasih!