

# Data Structure

Dummy Linked List

# Dummy LinkedList

이번 시간에 다뤄볼 자료구조는  
Linked List 중에서도 Dummy Linked List 입니다.

## 1. 노드란?



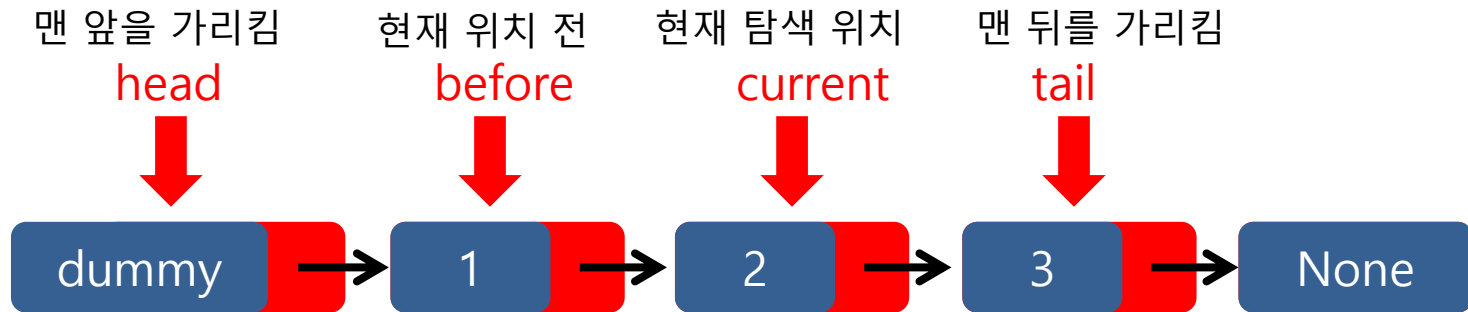
# Dummy LinkedList

## 2. 더미란?

더미란 실제 데이터를 담고 있는 노드가 아니라  
구현의 편의를 위해 맨 앞에 두는 무의미한 노드입니다.



## Dummy LinkedList의 구현



실제로 링크드 리스트 클래스 내에는 노드를 가리키는

1. 4개의 참조와
2. 데이터의 개수를 담고 있는 데이터 개수

만 존재함.

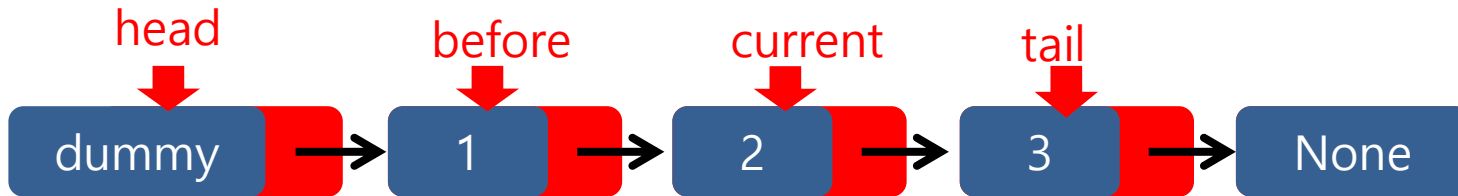
## Dummy LinkedList의 구현



노드를 구현한 클래스

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

## Dummy LinkedList의 구현



1. 4개의 참조
2. 데이터의 개수를 담고 있는 데이터 개수

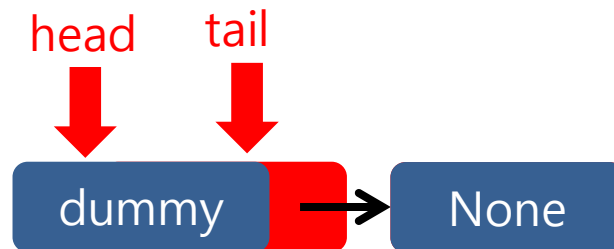
```
def __init__(self):  
    dummy = Node("dummy")  
  
    self.head = dummy  
    self.tail = dummy  
  
    self.current = None  
    self.before = None  
  
    self.num_of_data = 0
```

# Dummy LinkedList의 구현

## 메소드 구현

### 1. 생성자

```
def __init__(self):  
    dummy = Node("dummy")  
  
    self.head = dummy  
    self.tail = dummy  
  
    self.current = None  
    self.before = None  
  
    self.num_of_data = 0
```

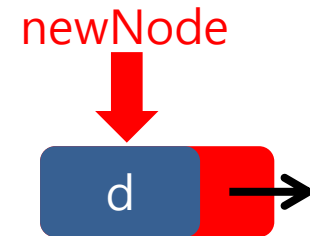
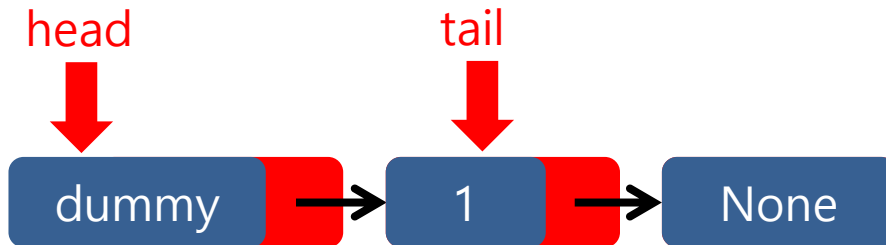


## Dummy LinkedList의 구현

### 메소드 구현

2. append() : data의 Insert 1

```
def append(self, data):  
    new_node = Node(data)  
    self.tail.next = new_node  
    self.tail = new_node  
    self.num_of_data += 1
```



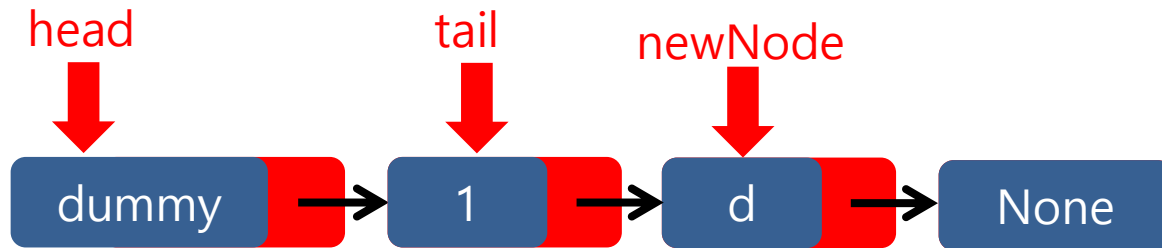


## Dummy LinkedList의 구현

### 메소드 구현

#### 3. append() : data의 Insert 2

```
def append(self, data):  
    new_node = Node(data)  
    self.tail.next = new_node  
    self.tail = new_node  
    self.num_of_data += 1
```

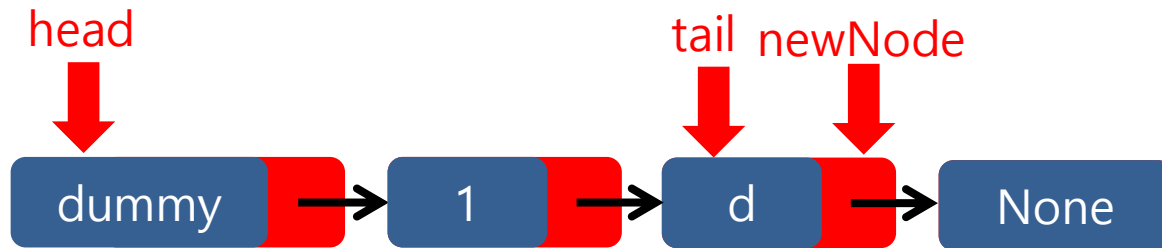


## Dummy LinkedList의 구현

### 메소드 구현

#### 4. append() : data의 Insert 3

```
def append(self, data):  
    new_node = Node(data)  
    self.tail.next = new_node  
    self.tail = new_node  
    self.num_of_data += 1
```

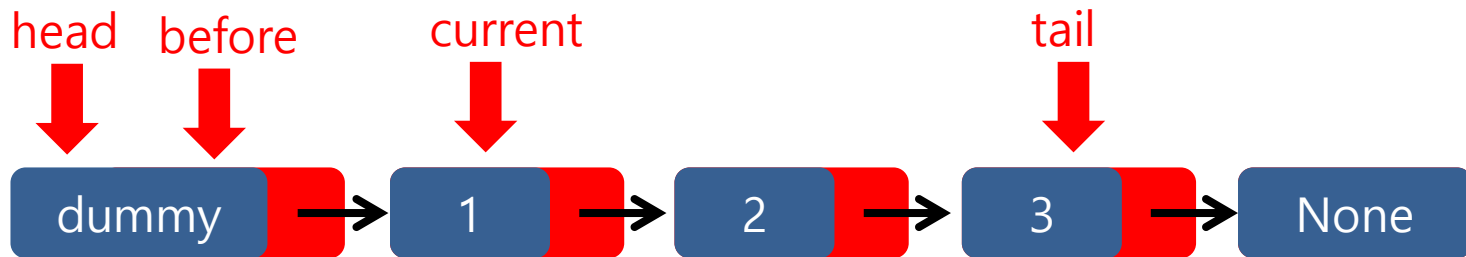


## Dummy LinkedList의 구현

### 메소드 구현

5. first() : data의 search 1-1

```
def first(self):  
    self.before = self.head  
    self.current = self.head.next  
  
    if self.current:  
        return self.current.data  
  
    return None
```

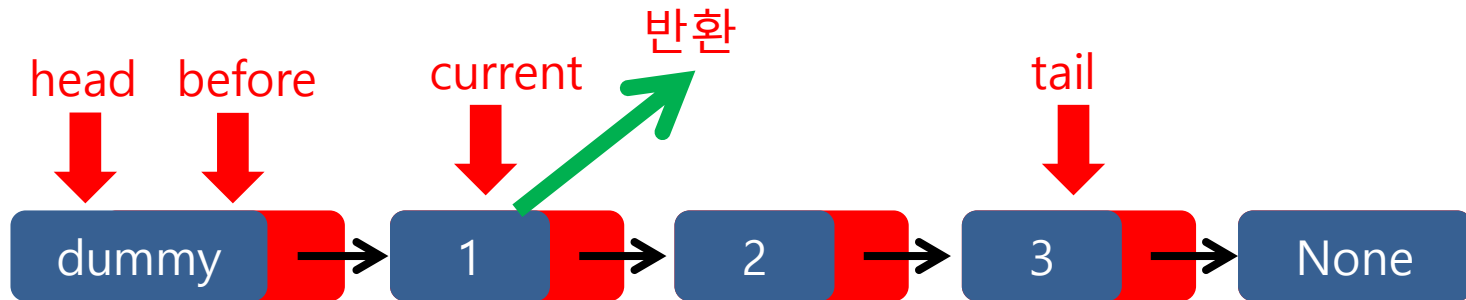


## Dummy LinkedList의 구현

### 메소드 구현

6. first() : data의 search 1-2

```
def first(self):  
    self.before = self.head  
    self.current = self.head.next  
  
    if self.current:  
        return self.current.data  
  
    return None
```

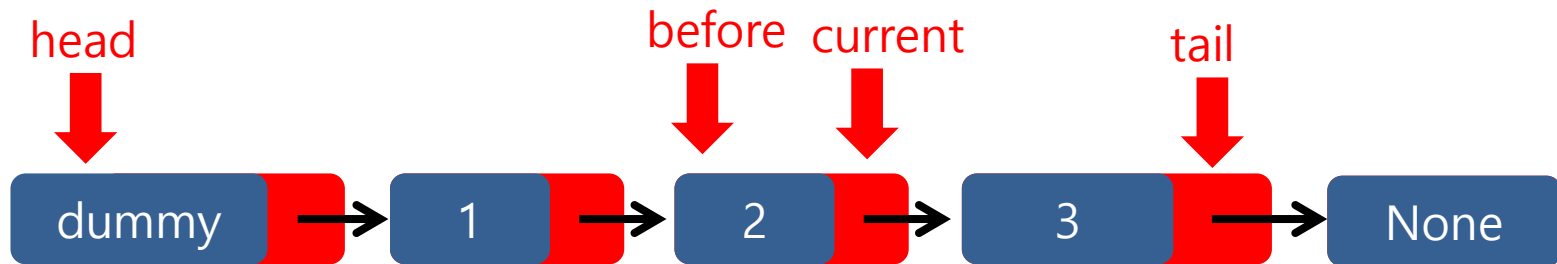


## Dummy LinkedList의 구현

### 메소드 구현

7. next() : data의 search 2-1

```
def next(self):  
    if not self.current.next:  
        return None  
  
    self.before = self.current  
    self.current = self.current.next  
  
    return self.current.data
```

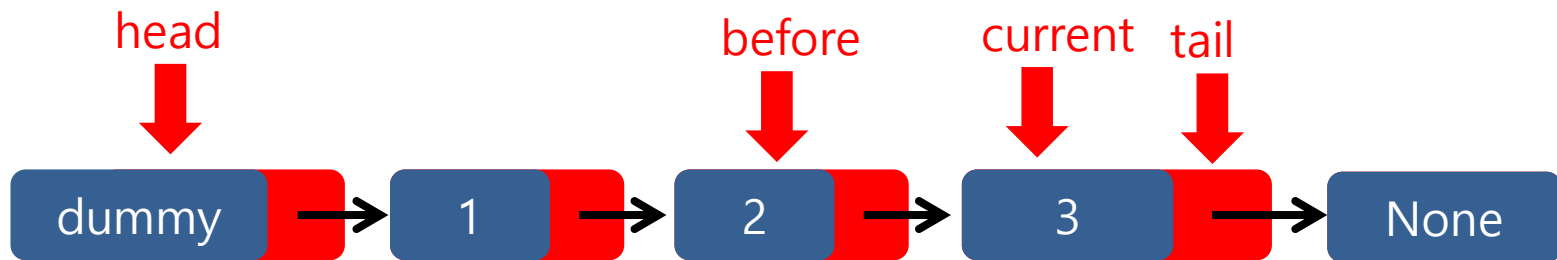


## Dummy LinkedList의 구현

### 메소드 구현

8. next() : data의 search 2-2

```
def next(self):  
    if not self.current.next:  
        return None  
  
    self.before = self.current  
    self.current = self.current.next  
  
    return self.current.data
```

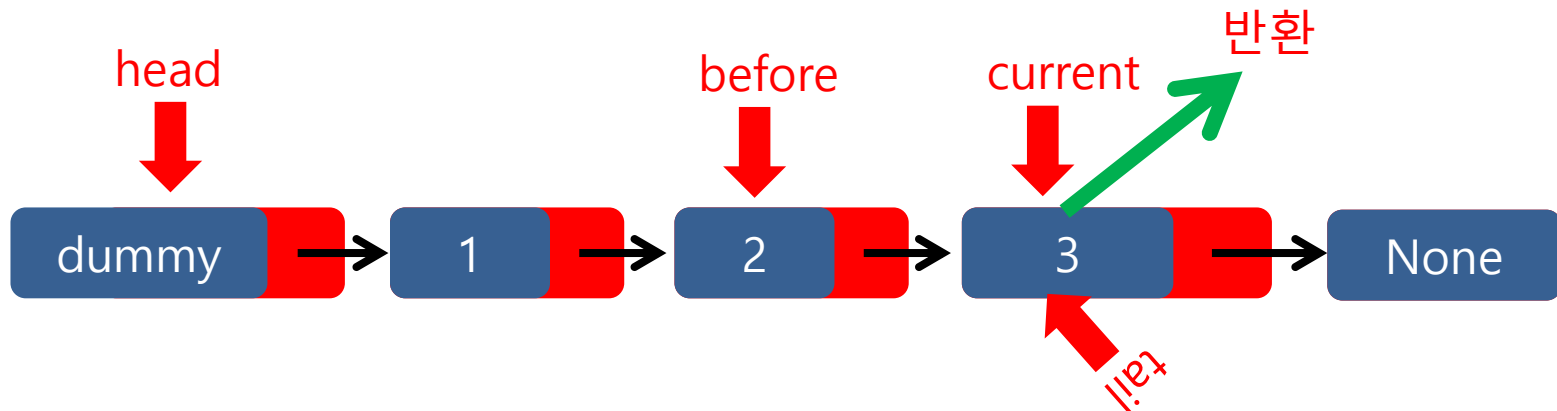


## Dummy LinkedList의 구현

### 메소드 구현

9. next() : data의 search 2-3

```
def next(self):  
    if not self.current.next:  
        return None  
  
    self.before = self.current  
    self.current = self.current.next  
  
    return self.current.data
```



## Dummy LinkedList의 구현

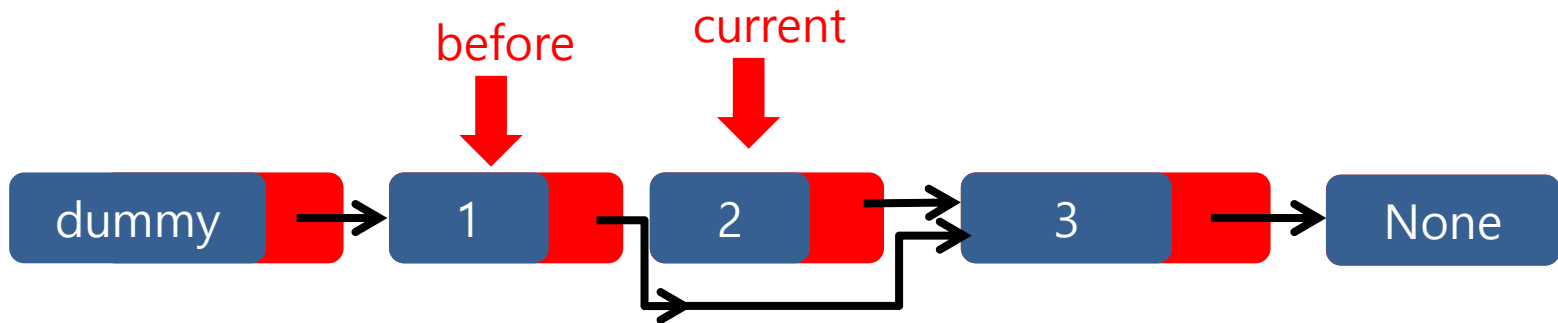
### 메소드 구현

10. delete() : data의 삭제 1

```
def delete(self):  
    ret_data = self.current.data  
  
    if self.current == self.tail:  
        self.tail = self.head
```

```
    self.before.next = self.current.next  
    self.current = self.before  
    self.num_of_data -= 1
```

```
    return ret_data
```



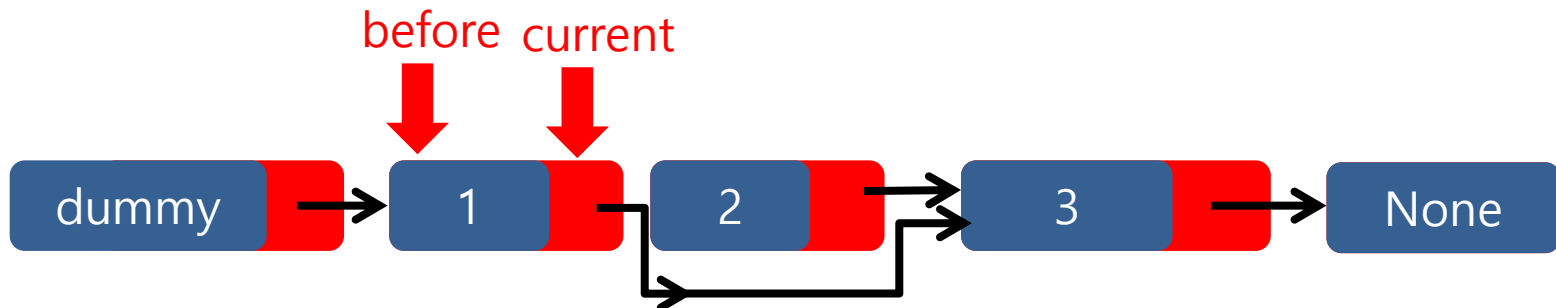


## Dummy LinkedList의 구현

### 메소드 구현

10. delete() : data의 삭제 2

```
def delete(self):  
    ret_data = self.current.data  
  
    if self.current == self.tail:  
        self.tail = self.head  
  
    self.before.next = self.current.next  
    self.current = self.before  
    self.num_of_data -= 1  
  
    return ret_data
```



## Dummy LinkedList의 구현

### 메소드 구현

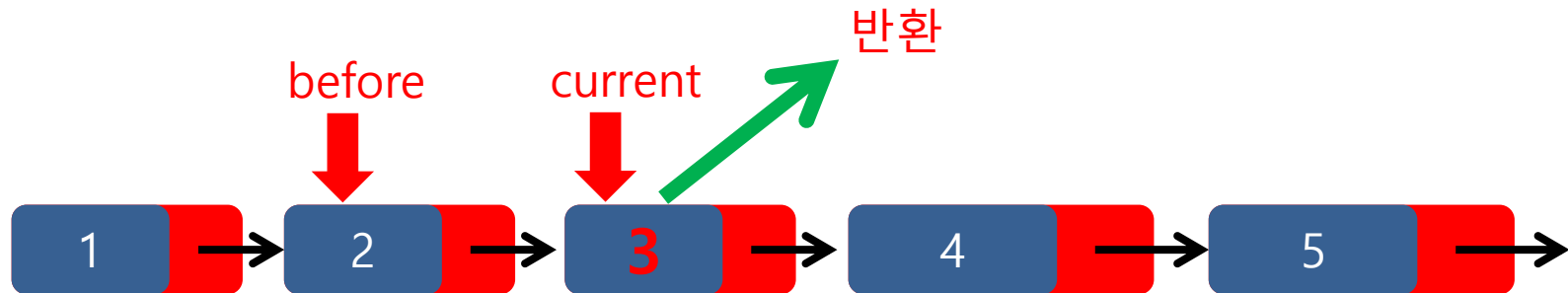
11. delete 함수에서 current가 before로 옮기는 이유

```
def next(self):  
    if not self.current.next:  
        return None
```

```
    self.before = self.current  
    self.current = self.current.next
```

```
    return self.current.data
```

next() 함수가 끝난다는 것은  
Current가 마지막에 가리킨 노드의  
Data를 이미 반환했다는 것

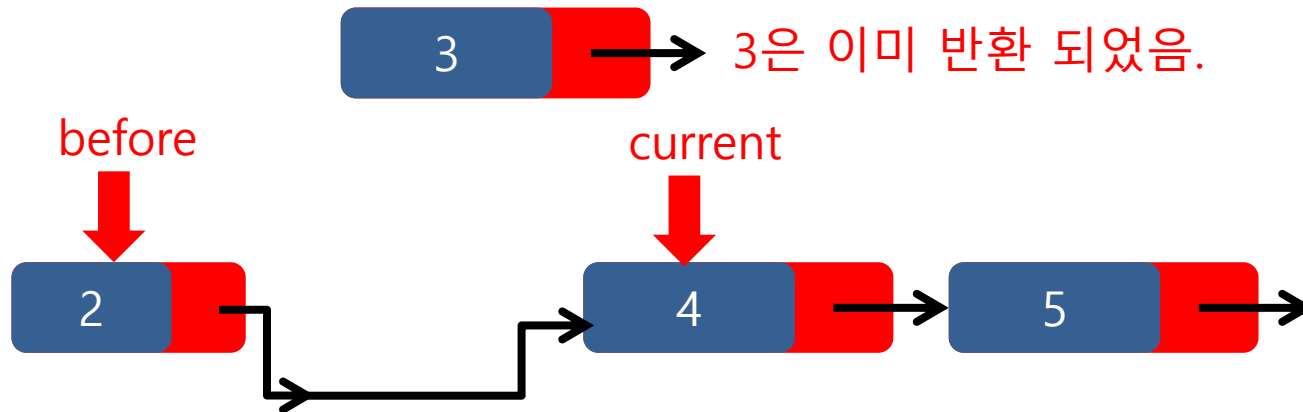


## Dummy LinkedList의 구현

### 메소드 구현

#### 11. delete 함수에서 current가 before로 옮기는 이유

current가 3이 저장된 데이터를 삭제 후 뒤 노드로 가버린다면

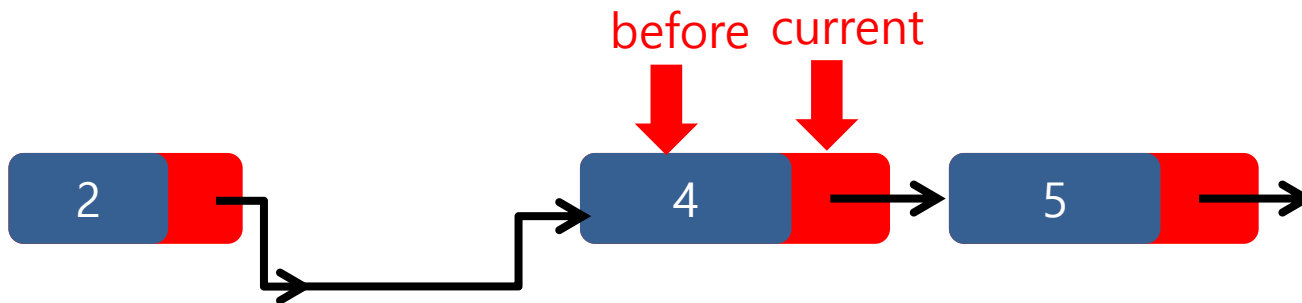


## Dummy LinkedList의 구현

### 메소드 구현

11. delete 함수에서 current가 before로 옮기는 이유

```
def next(self):  
    if not self.current.next:  
        return None  
  
    self.before = self.current  
    self.current = self.current.next  
  
    return self.current.data
```

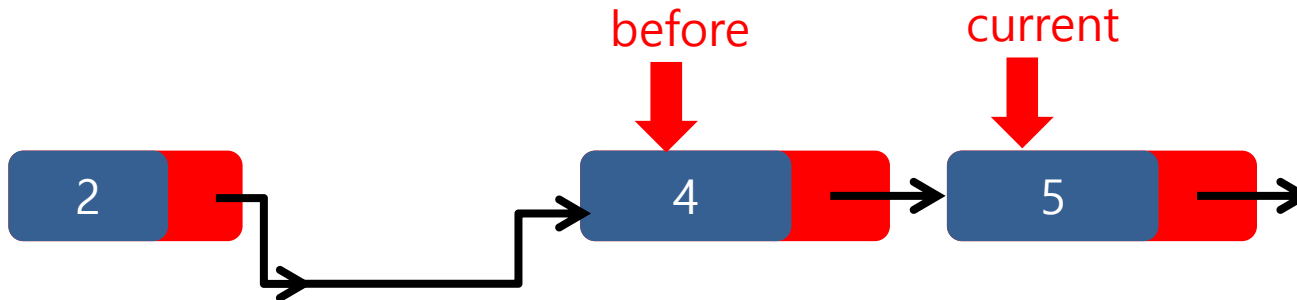


## Dummy LinkedList의 구현

### 멤버함수 구현

11. delete 함수에서 current가 before로 옮기는 이유

```
def next(self):  
    if not self.current.next:  
        return None  
  
    self.before = self.current  
    self.current = self.current.next  
  
    return self.current.data
```



## Dummy LinkedList의 구현

### 멤버함수 구현

11. delete 함수에서 current가 before로 옮기는 이유

```
def next(self):  
    if not self.current.next:  
        return None
```

4는 반환되지 않는다!!!

```
self.before = self.current  
self.current = self.current.next
```

```
return self.current.data
```

