Data Structure

Dummy Linked List

Dummy LinkedList

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

1. 노드란?

저장하고 싶은 데이터

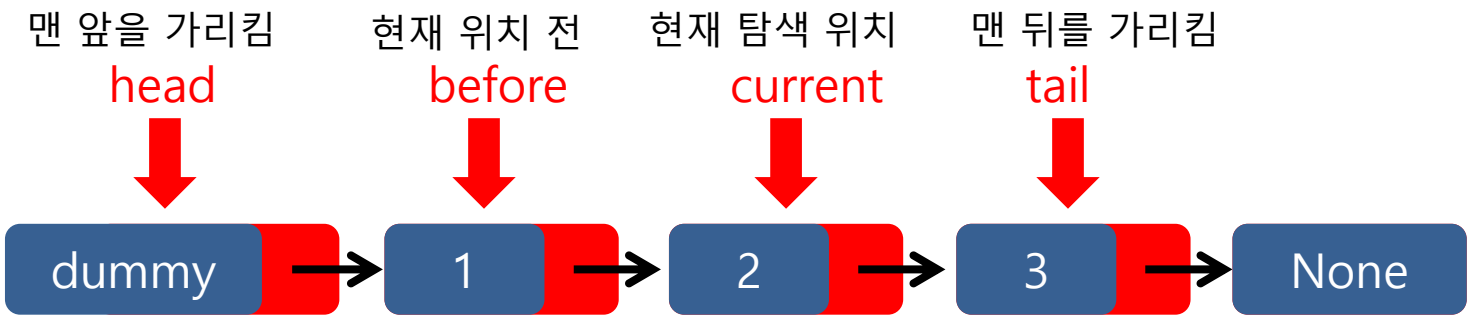
다음 노드를 가리키는 참조

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

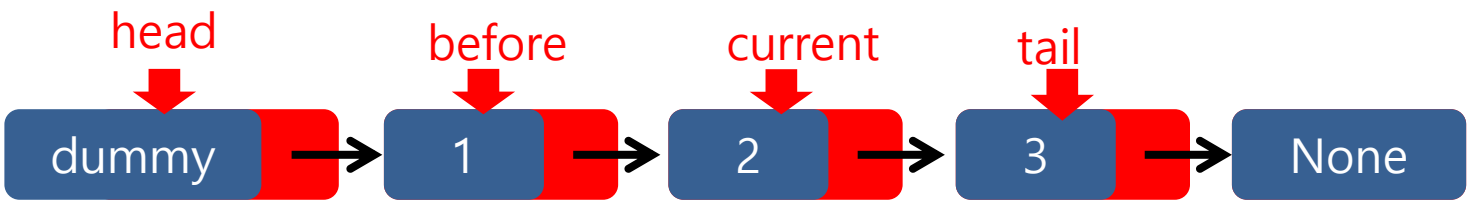
만 존재함.

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

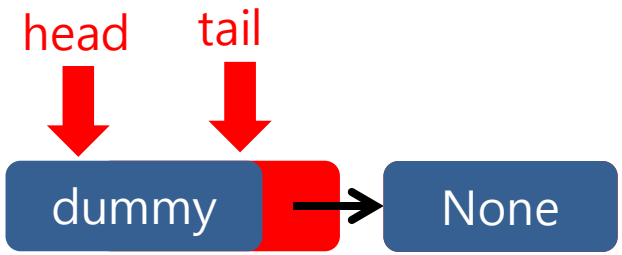
```
def __init__(self):
    dummy = Node("dummy")

    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
```

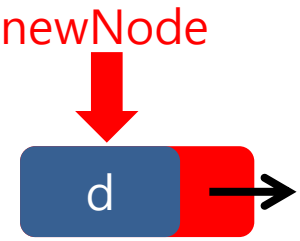


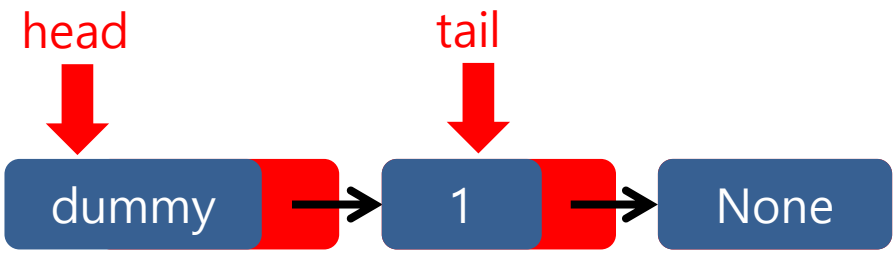
Dummy LinkedList의 구현

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
```



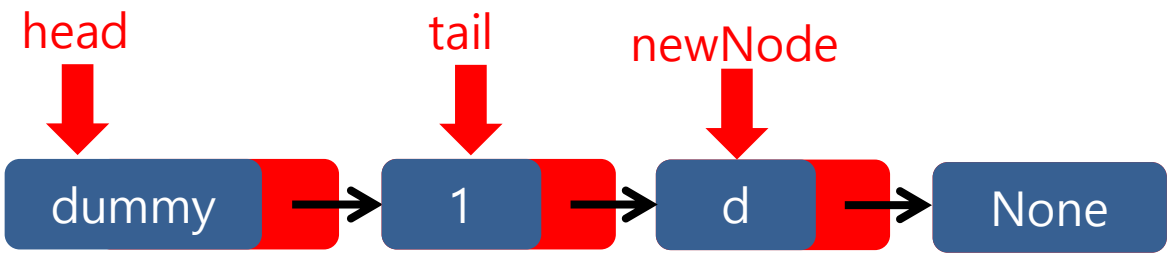


Dummy LinkedList의 구현

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
```

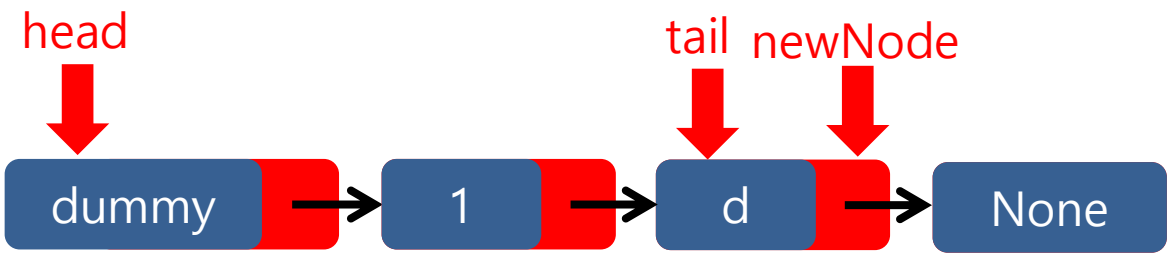


Dummy LinkedList의 구현

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
```

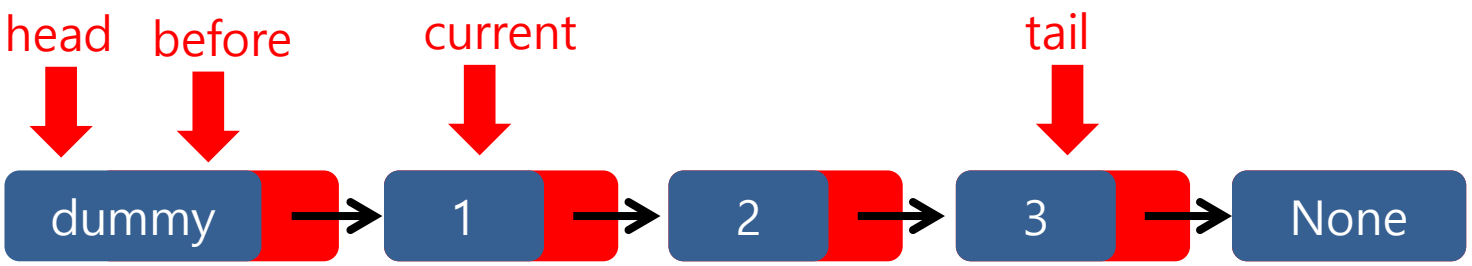


Dummy LinkedList의 구현

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
```

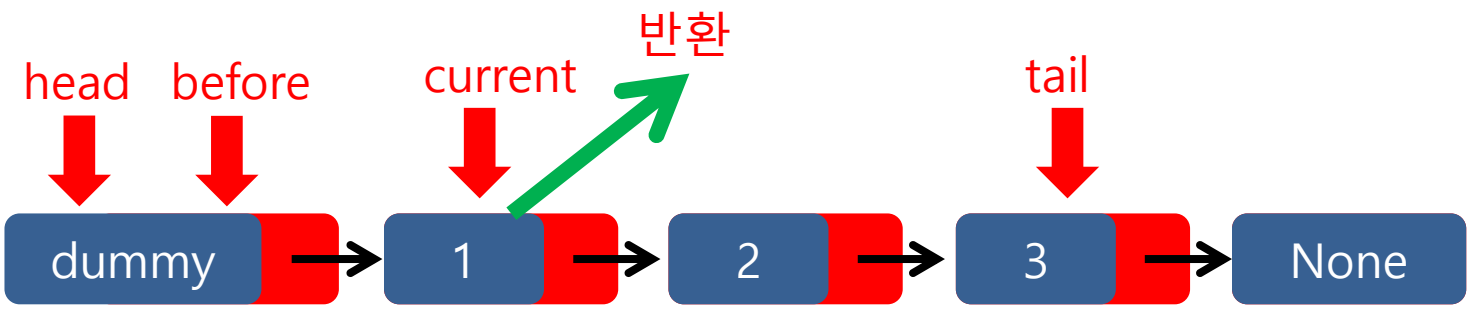


Dummy LinkedList의 구현

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
```

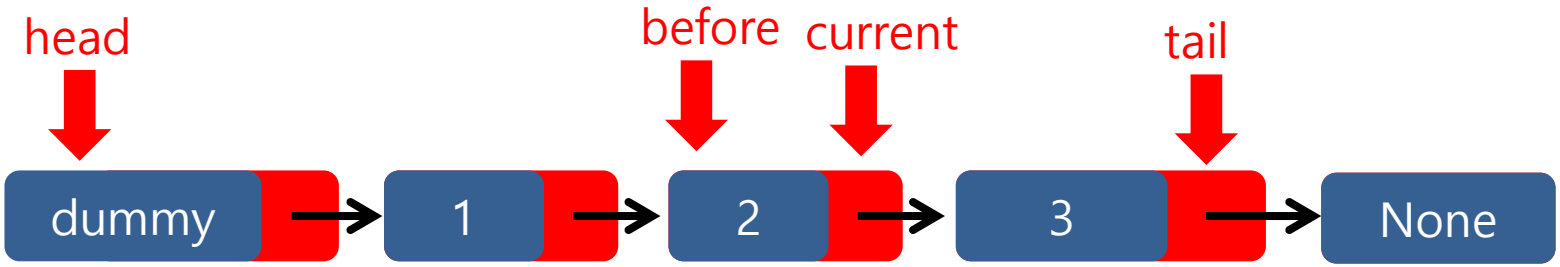


Dummy LinkedList의 구현

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
```

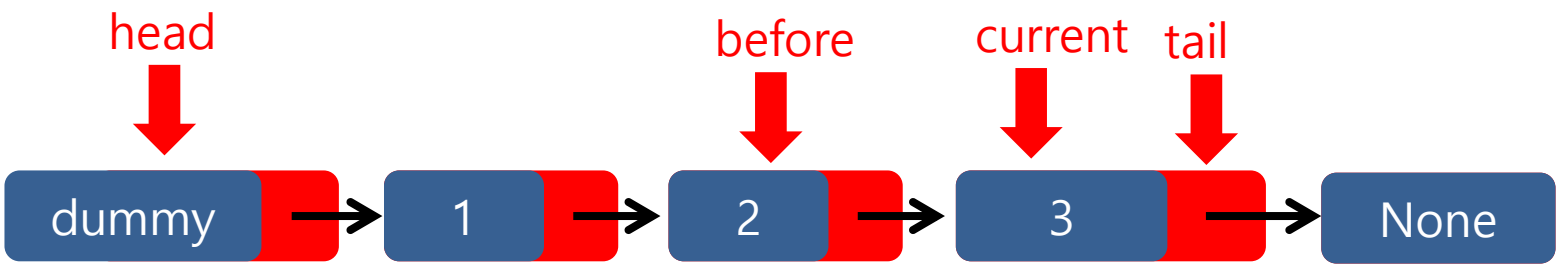


Dummy LinkedList의 구현

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
```

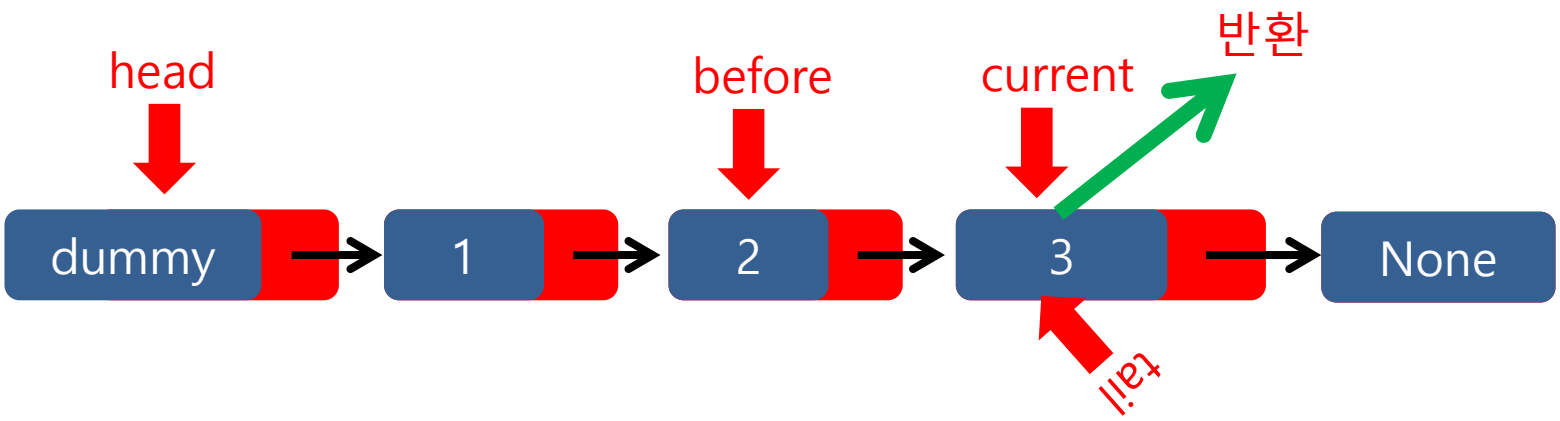


Dummy LinkedList의 구현

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
```

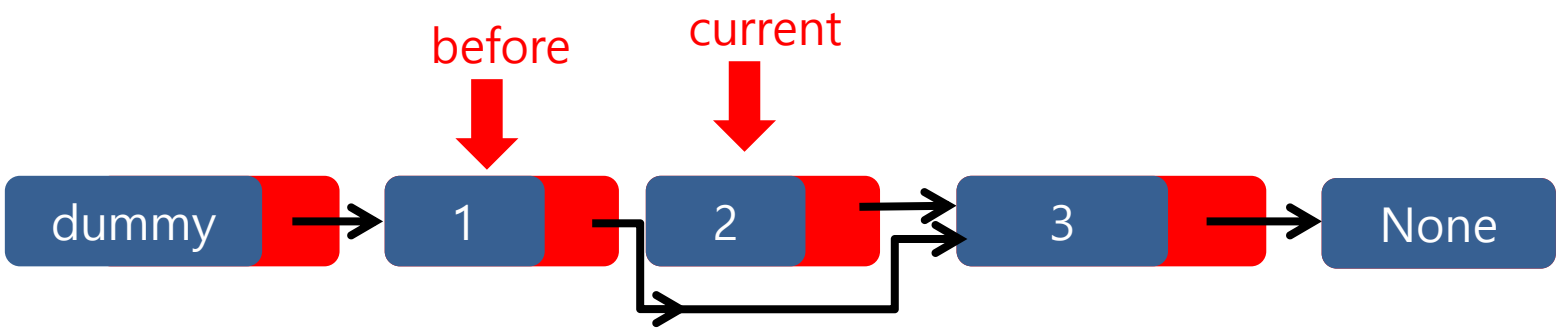


Dummy LinkedList의 구현

메소드 구현

10. delete() : data의 삭제 1

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

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

    return ret_data
```

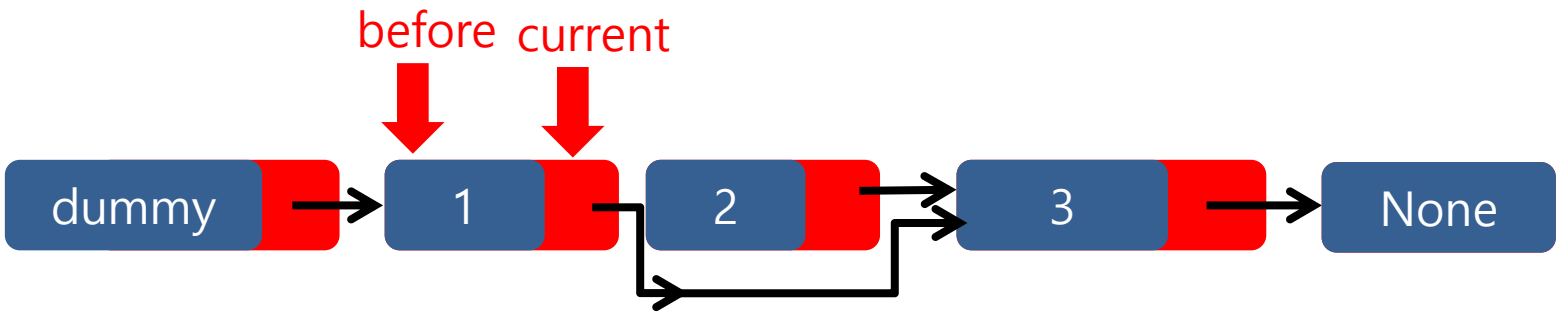


Dummy LinkedList의 구현

메소드 구현

11. delete() : data의 삭제 2

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

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

    return ret_data
```



Dummy LinkedList의 구현

13. 왜 더미가 들어가는가?

우리가 본 TED 영상에서 리누스 토발즈가 단순 싱글 리스트의 단점에 대해서 말하고 있습니다.

물론 토발즈는 자신만의 코딩으로 이 문제를 해결해버리지만....;;;;;;

우리는 토발즈 같은 천재가 아니니 어떤 방법이 있는지 알아봅시다.

Dummy LinkedList의 구현

메소드 구현

13. 왜 더미가 들어가는가?

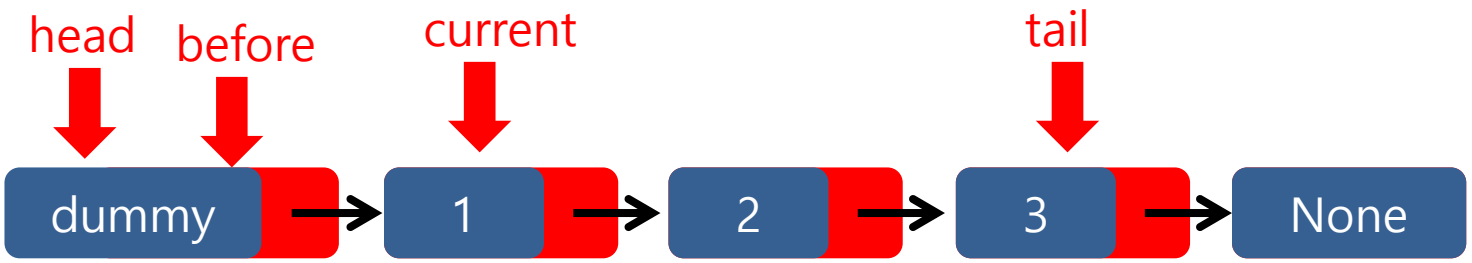
if문 같은 분기문이 보이지 않죠?!

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

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

    return ret_data
```

더미를 넣음으로써
두 경우

1. 가장 첫 번째 데이터를 지우는 경우
2. 두 번째 이후의 데이터를 지우는 경우

를 한번에 모두 처리할 수 있다.



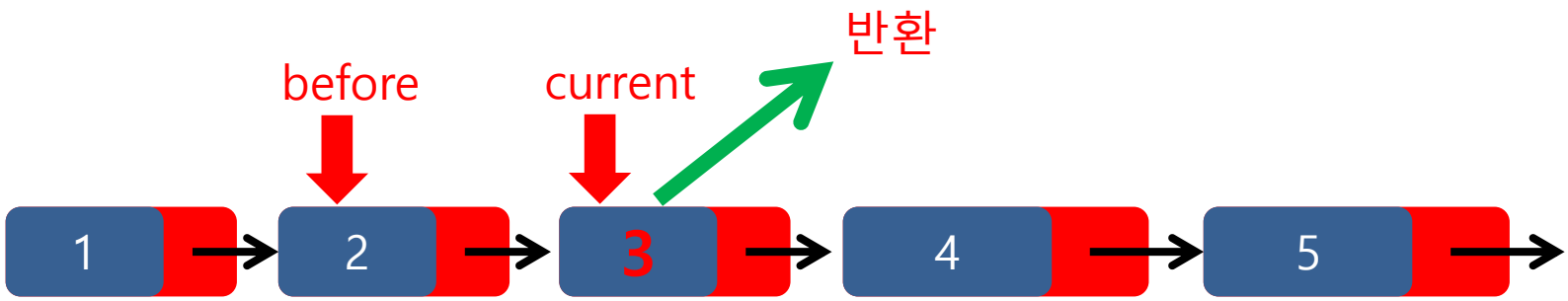
Dummy LinkedList의 구현

메소드 구현

14. before가 필요한 이유

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

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

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

    return self.current.data
```

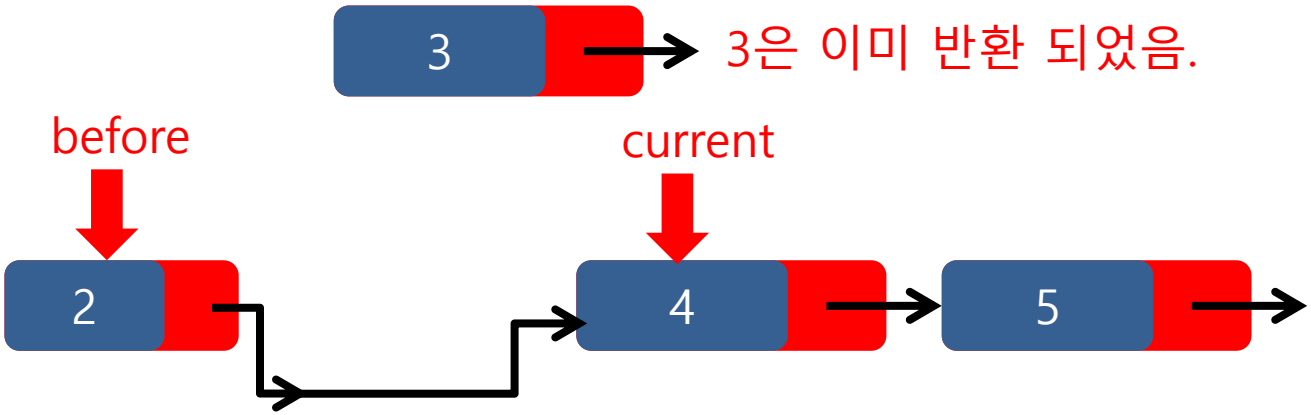


Dummy LinkedList의 구현

메소드 구현

14. before가 필요한 이유

current가 3이 저장된 데이터를 삭제 후 뒤 노드로 가버린다면

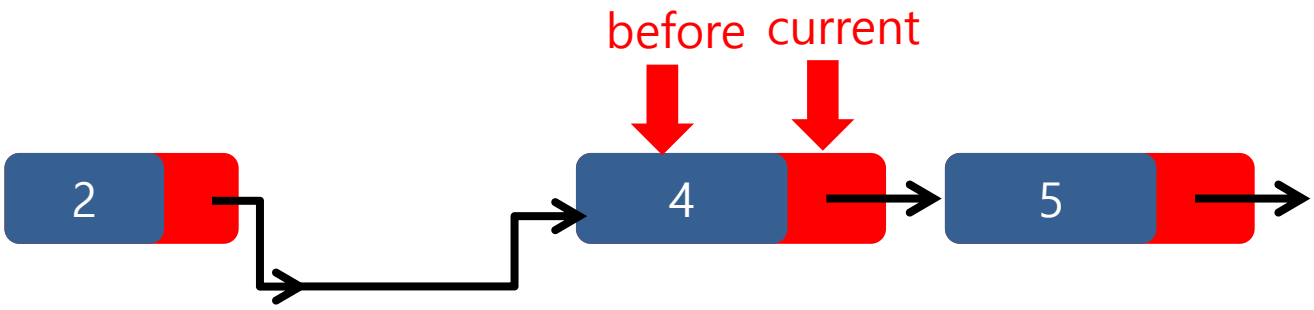


Dummy LinkedList의 구현

메소드 구현

14. before가 필요한 이유

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

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

    return self.current.data
```

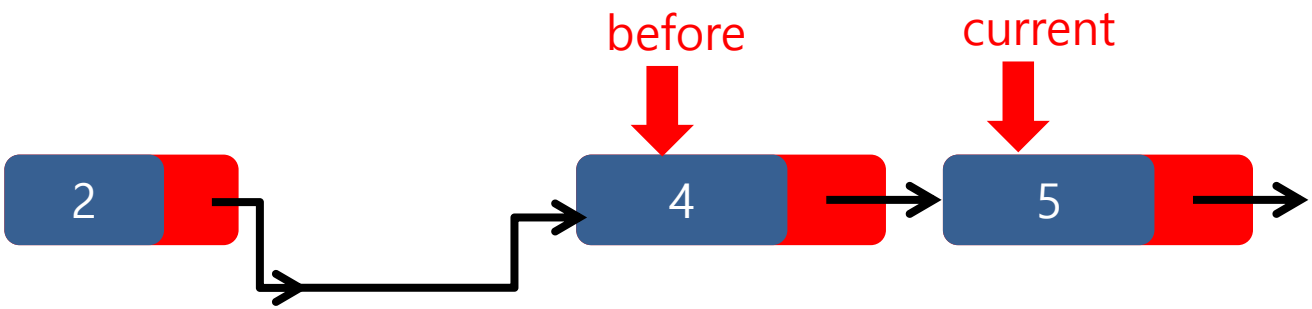


Dummy LinkedList의 구현

멤버함수 구현

14. before가 필요한 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

14. before가 필요한 이유

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

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

    return self.current.data
```

4는 반환되지 않는다!!!