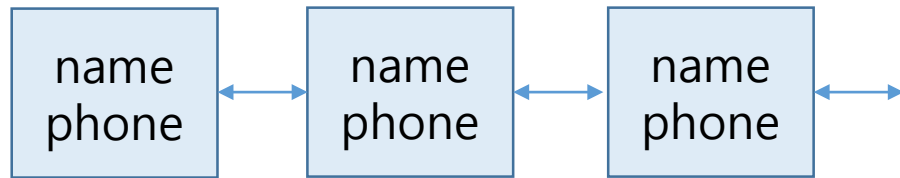


# Singly Linked List

(단방향 연결 리스트)

# Linked List

- *Linked List?*

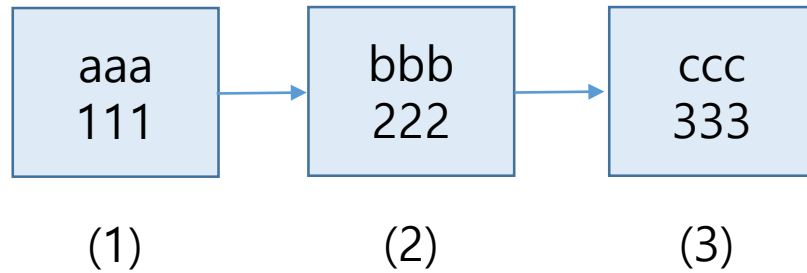


- 종류

- 단방향 연결 리스트 (*singly linked list*)
- 양방향 연결 리스트 (*doubly linked list*)
- 원형 연결 리스트 (*circular linked list*)
- ...

# 단방향 연결 리스트

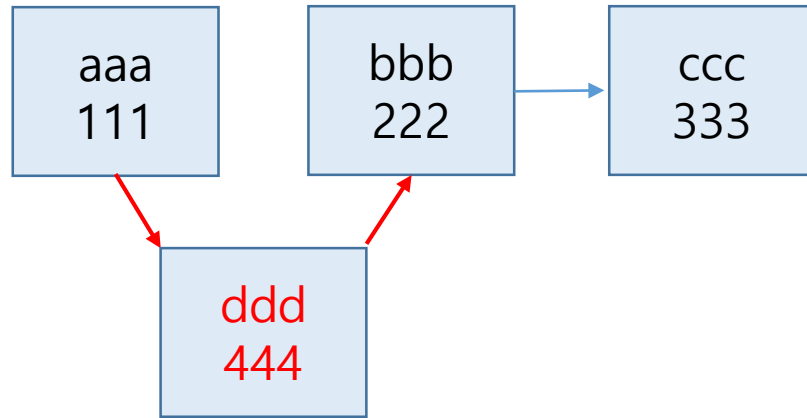
- 검색



- 시간 복잡도 (time complexity)

-  $O(N)$  => 연결 리스트의 노드 개수에 비례하여 선형 증가

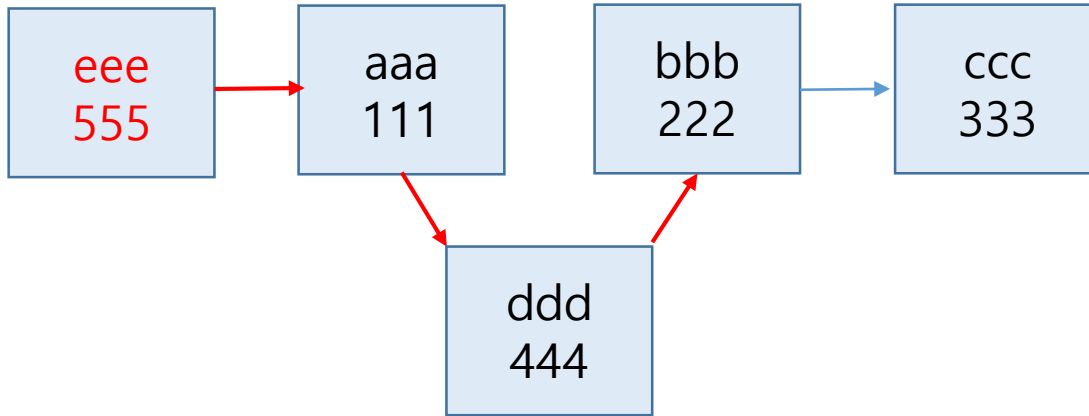
- 삽입



- 시간 복잡도

-  $O(1)$  if the previous node is given

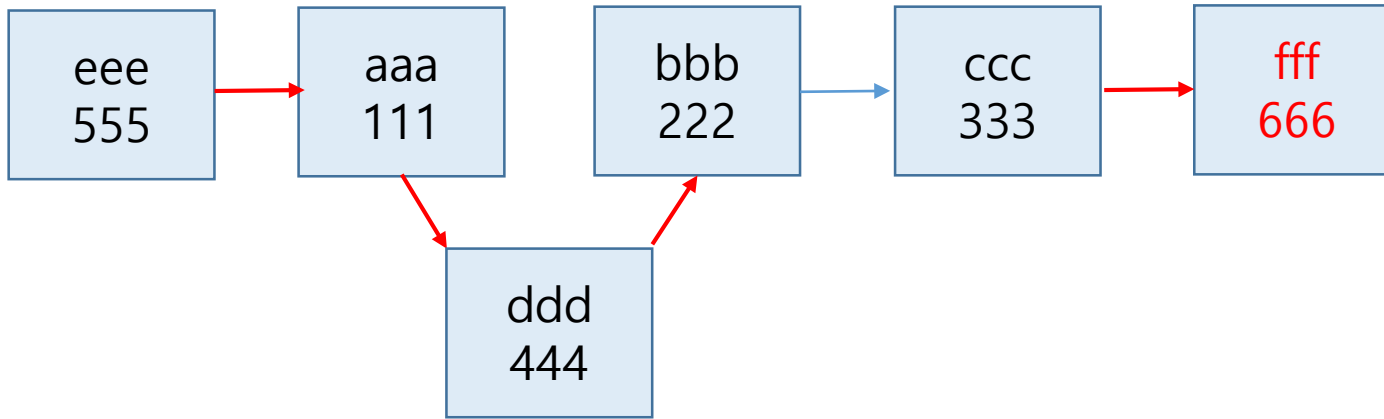
- 전면 삽입



- 시간 복잡도

-  $O(1)$

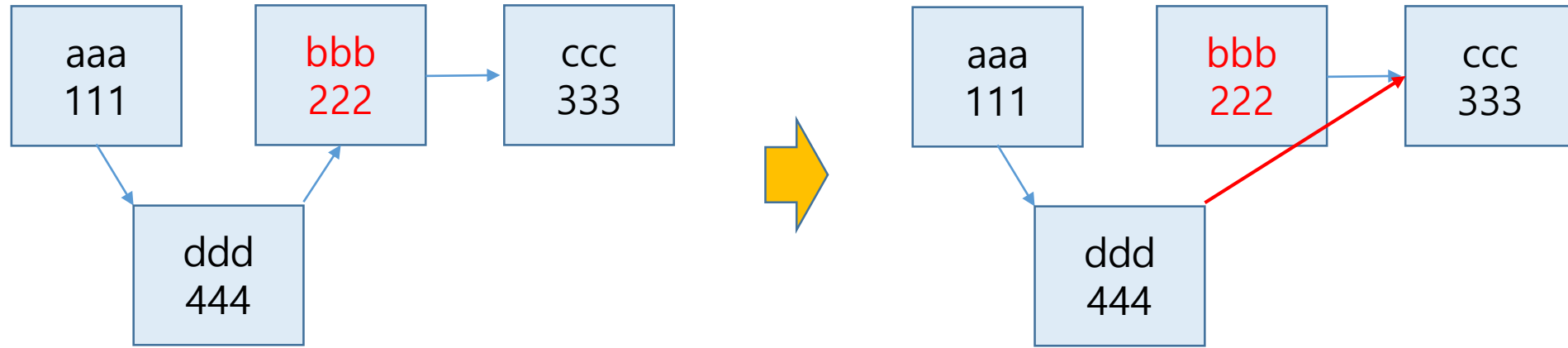
- 후면 삽입



- 시간 복잡도

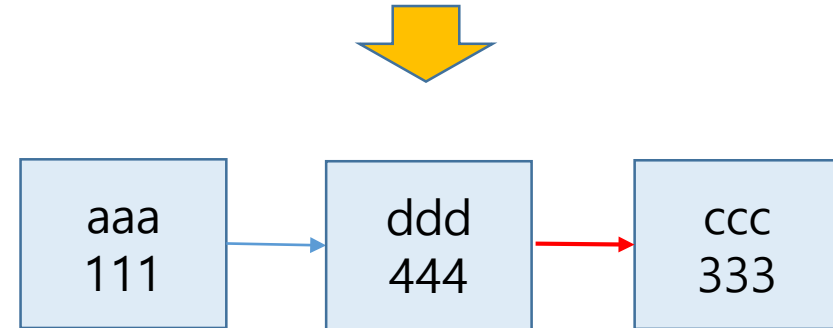
- $O(N)$  if the previous node is not given

- 삭제



- 시간 복잡도

- $O(1)$  if the previous node is given
- $O(N)$  otherwise




# 단방향 연결 리스트 구현 - 1단계

- 클래스 *SNode* 정의

- 리스트를 구성하는 하나의 노드에 해당

- *two member variables*

- *item* (노드의 데이터를 가리킴), *next* (다음 노드를 가리킴)



item  
next

- 하나의 생성자 & *no other methods*

```
class SNode:
```

```
    def __init__(self, item, next=None):
```

```
        self.item = item
```

```
        self.next = next
```



# 단방향 연결 리스트 구현 - 2단계

- 클래스 *SList* 정의

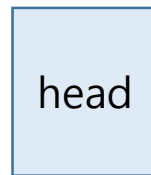
- one member variables

- *head* (리스트의 첫 노드를 가리킴)

- 생성자, *insert\_front()*, *delete\_front()*, *search()*, *print\_list()*

```
class SList:
```

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



# 단방향 연결 리스트 구현 - 2단계

- 클래스 *SList* 정의

- insert\_front()*

```
class SList:
```

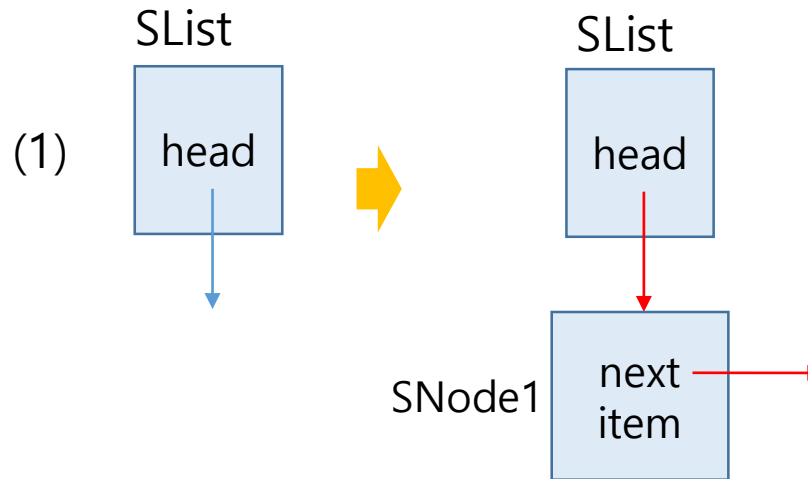
```
...
```

```
def insert_front(self, item):
```

```
    snode = SNode(item, self.head)
```

```
    self.head = snode
```

```
    return
```



# 단방향 연결 리스트 구현 - 2단계

- 클래스 *SList* 정의

- insert\_front()*

```
class SList:
```

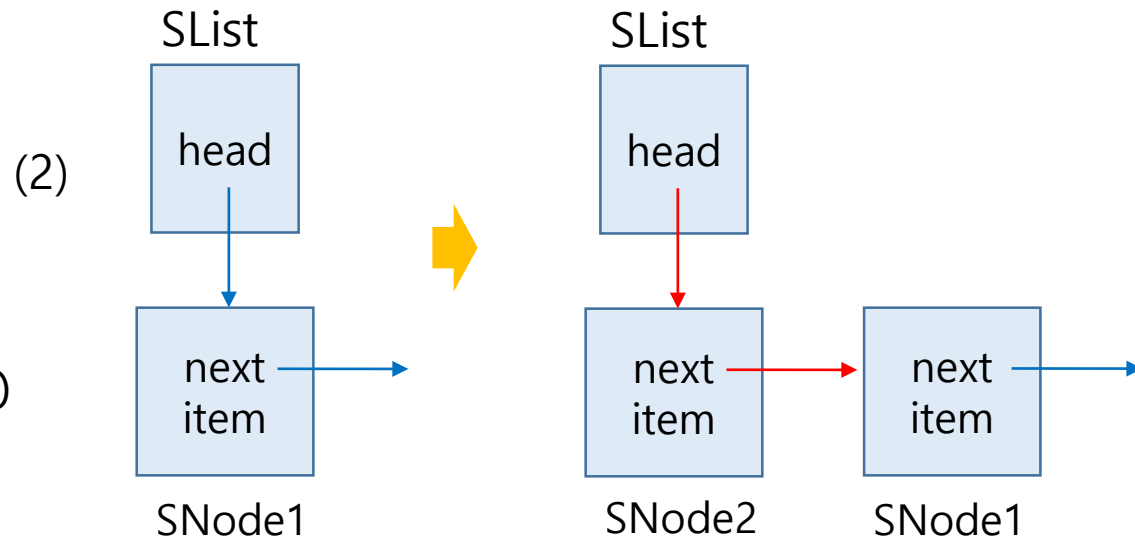
```
...
```

```
def insert_front(self, item):
```

```
    snode = SNode(item, self.head)
```

```
    self.head = snode
```

```
    return
```

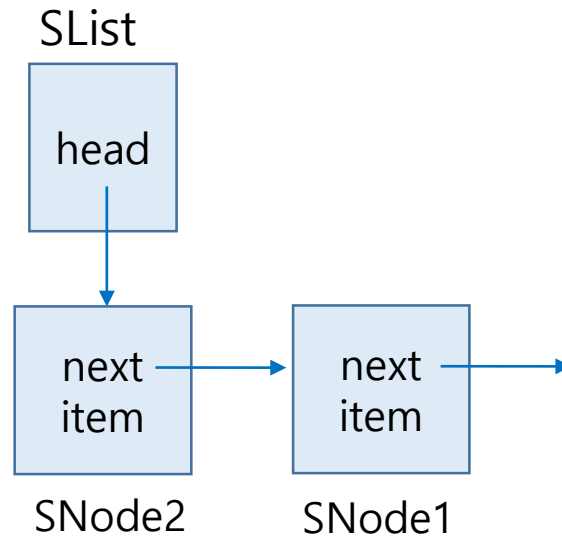


# 단방향 연결 리스트 구현 - 2단계

- 클래스 `SList` 정의

- `print_list()`

```
class SList:
    ...
    def print_list(self):
        p = self.head
        while p:
            if p.next != None:
                print(p.item, ' -> ', end='')
            else:
                print(p.item)
            p = p.next
```



# 단방향 연결 리스트 테스트 프로그램 /

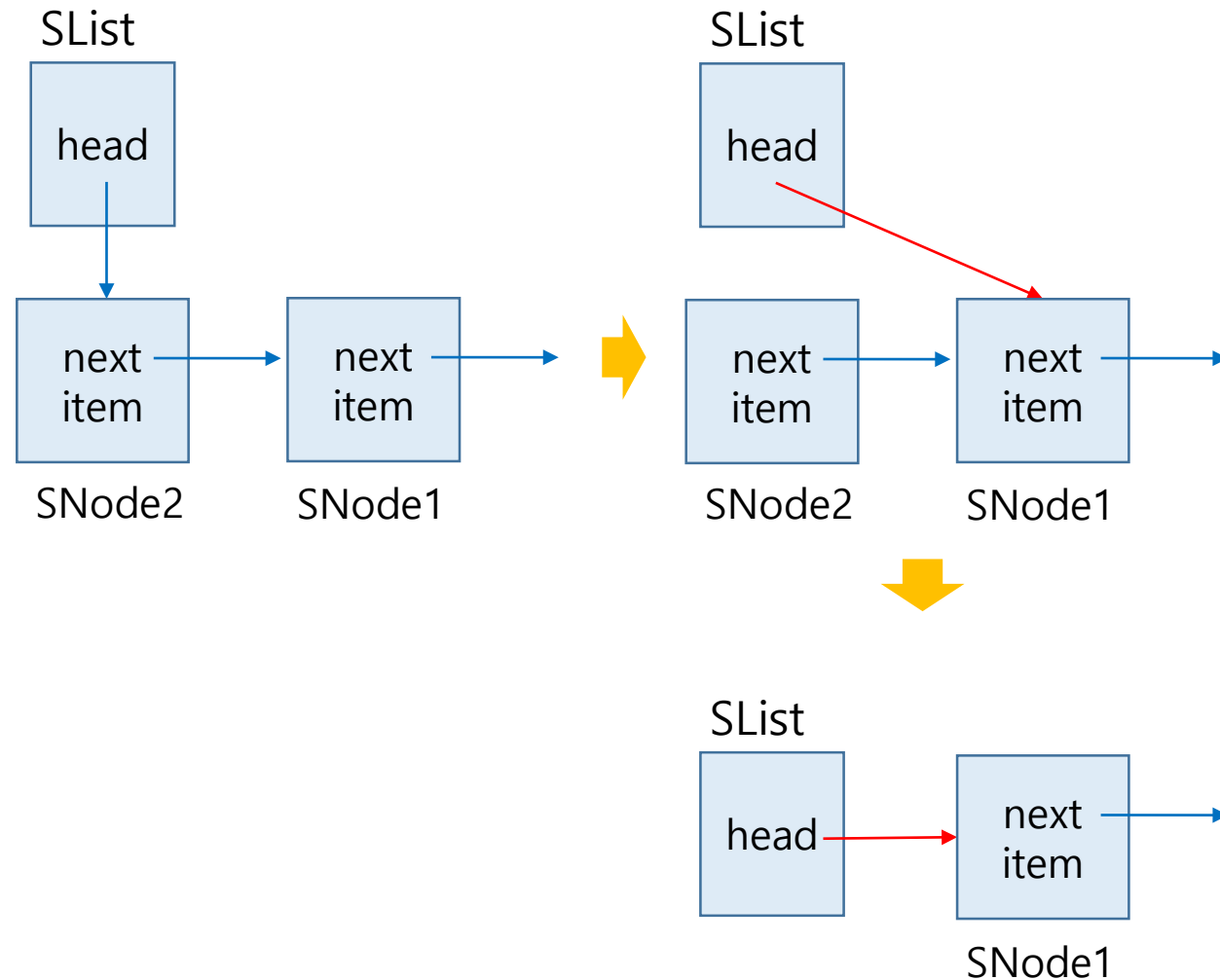
```
s = SList()                                # 빈 연결 리스트 생성
s.insert_front("mango")
s.insert_front("tomato")
s.insert_front("orange")
s.insert_front("apple")
s.print_list()
```

# 단방향 연결 리스트 구현 - 3단계

- 클래스 *SList* 정의

- *delete\_front()*

```
class SList:  
    ...  
    def delete_front(self):
```



# 단방향 연결 리스트 활용 프로그램2

```
s = SList()                                # 빈 연결 리스트 생성
s.insert_front("mango")
s.insert_front("tomato")
s.insert_front("orange")
s.insert_front("apple")
s.print_list()                              # 출력
s.delete_front()                            # 삭제
s.print_list()
```

# 단방향 연결 리스트 구현 - 4단계

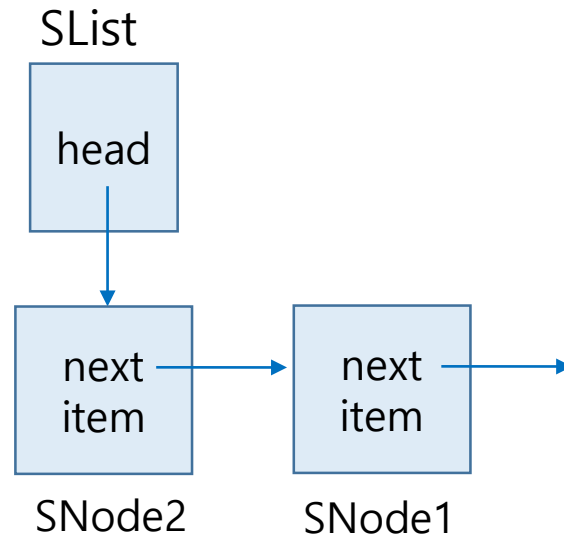
- 클래스 `SList` 정의

- `delete_target(item)`

```
class SList:
```

```
...
```

```
def delete_target(self, item):
```





# 단방향 연결 리스트 활용 프로그램3

```
s = SList()                                # 빈 연결 리스트 생성
s.insert_front("mango")
s.insert_front("tomato")
s.insert_front("orange")
s.insert_front("apple")
s.print_list()
(index, snode) = s.search('apple')          # 검색
print('apple', index)
s.delete_front()                            # 삭제
s.print_list()
s.delete_target("mango")
s.print_list()
```

# Homework candidate

- `insert_back()` 함수 구현하기
  - 리스트 뒤에 추가
- `delete_back()` 함수 구현하기
  - 리스트 마지막 `node` 삭제