7장. 큐

Contents

주요 내용

- 01 큐란
- 02 리스트를 이용한 큐
- 03 연결 리스트를 이용한 큐
- 04 큐 응용: 좌우동형 문자열 체크

학습목표

- 큐의 의미를 이해한다.
- 큐의 추상적 구조를 이해한다.
- 리스트를 이용한 큐의 원리와 구현을 이해한다.
- 연결 리스트를 이용한 큐의 원리와 구현을 이해한다.
- 큐가 다른 클래스를 어떻게 재사용하는지 관찰한다.
- 간단한 큐의 파이썬 구현을 연습하고 이를 응용할 수 있도록 한다.

01 큐란

큐 개념의 일상 예

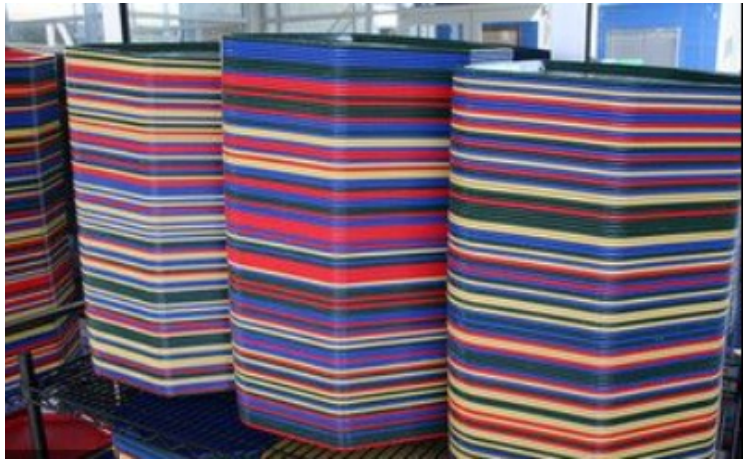
그림 7-1 큐 개념의 일상 예

- 오래된 우유부터 먹기
- 식당에서 기다리는 사람들의 열
- Lotto 복권을 사려는 사람들의 열

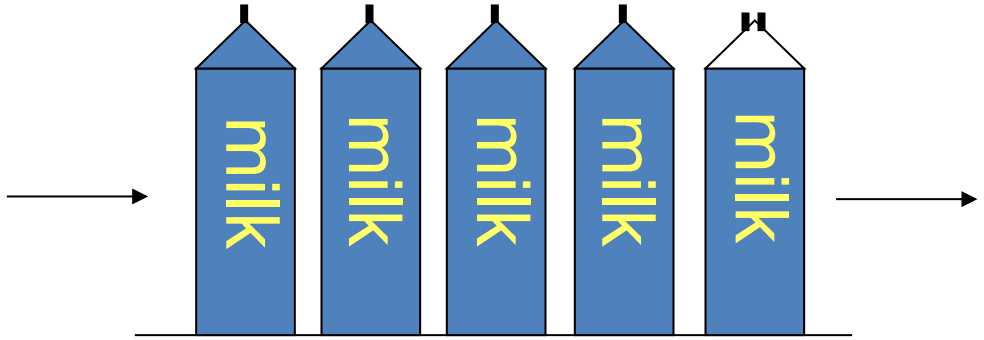
Stack vs. Queue

Stack: 최근에 삽입된 원소를 제거한다
Queue: 가장 먼저 삽입된 원소를 제거한다

Stack과 Queue는
쌍으로 생각하면서 공부할 것



stack



queue

✓ Stack: LIFO (Last-In-First-Out)

✓ Queue: FIFO (First-In-First-Out)

큐의 개념과 원리

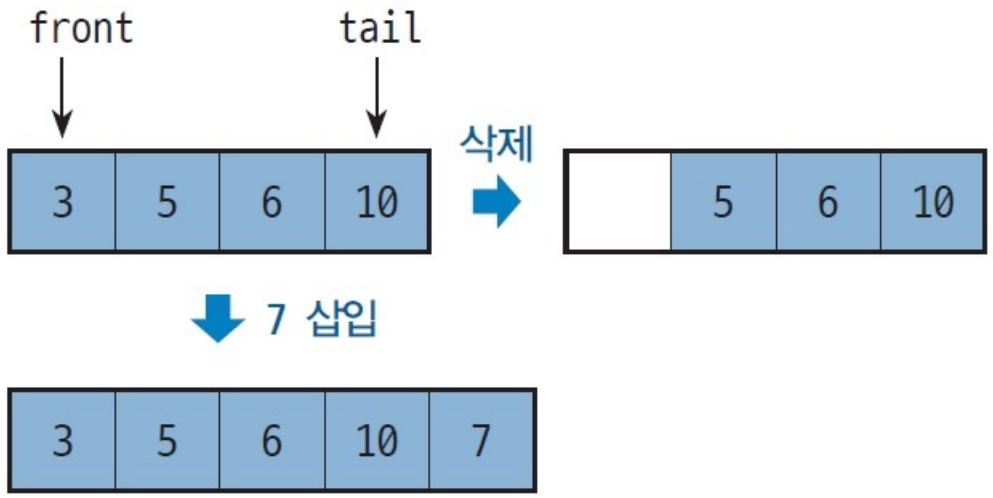


그림 7-2 큐의 개념과 원리

추상 데이터 타입 큐

맨 끝에 원소를 추가한다
맨 앞의 원소를 삭제하면서 알려준다
맨 앞의 원소를 알려준다
큐가 비어 있는지 확인한다
큐를 깨끗이 비운다

그림 7-3 ADT 큐

유일하게 접근 가능한 원소는
가장 먼저 삽입한 원소

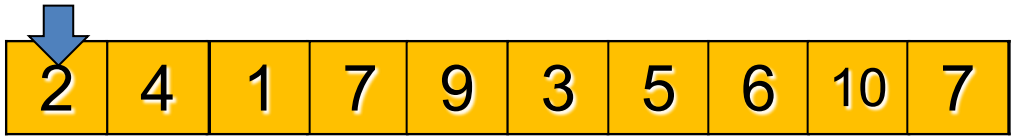
FIFO(First-In-First-Out)란 별칭을 갖고 있다

02 리스트 큐

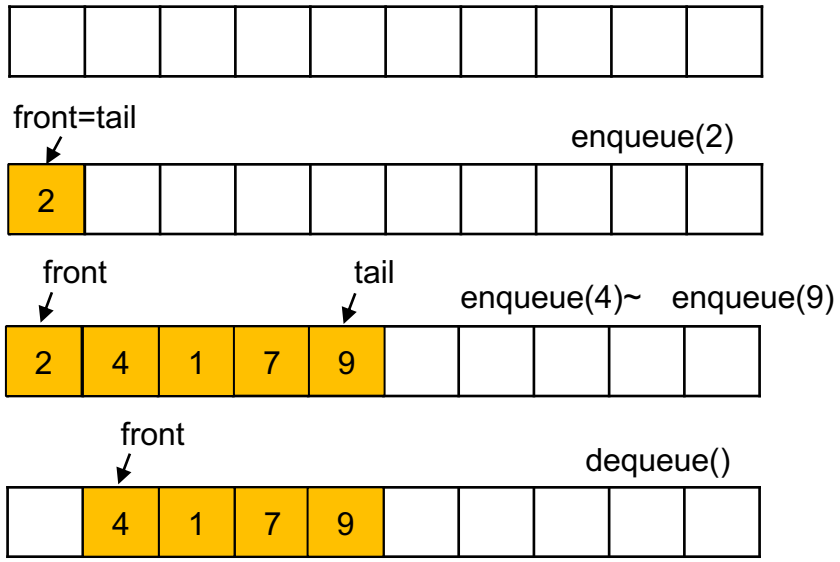
A Naive Version

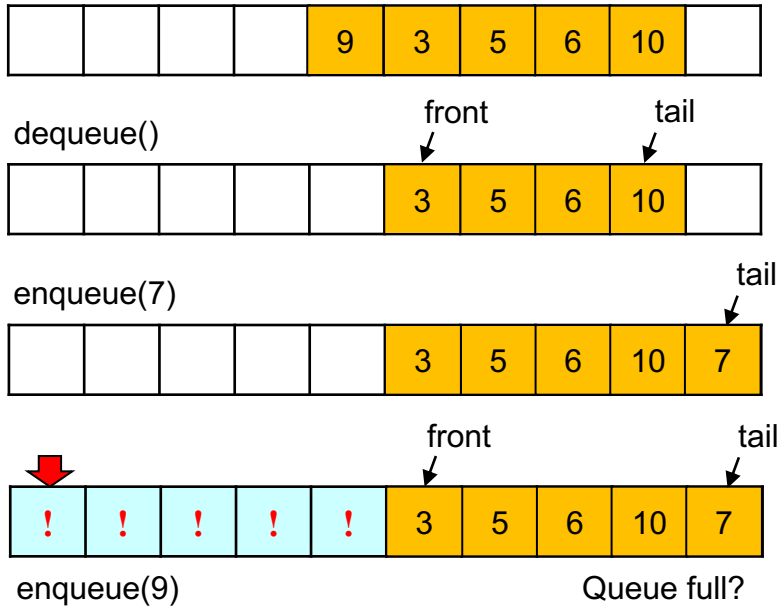
← Animation

queue.enqueue(5)
queue.enqueue(2)
queue.enqueue(7)
queueFront ← queue.front()
queueFront ← queue.dequeue()
queueFront ← queue.dequeue()



Animation





단순히 구현하면 공간이 있는데도 큐가 꽉 찬 것처럼 보인다

원형 큐circular queue로 구현하면 이 문제는 해결된다

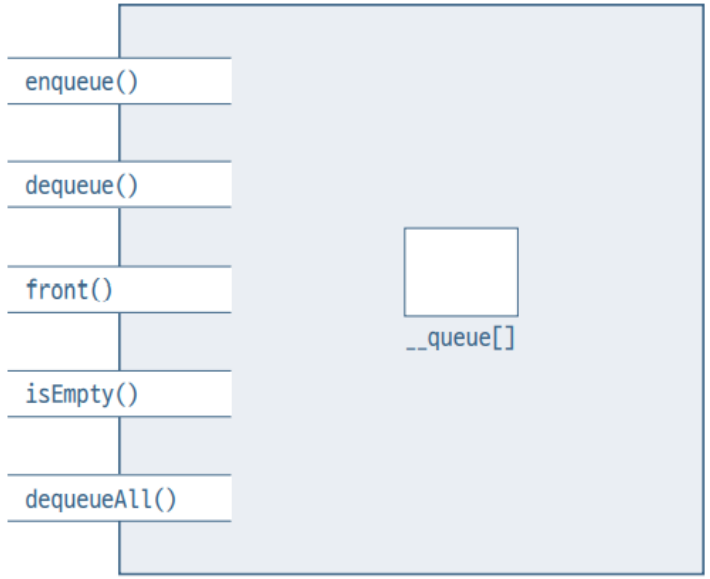
리스트 큐 객체 구조

필드:

__queue[] ◀ 큐 원소들이 저장되는 연결 리스트

작업:

enqueue() ◀ 큐의 맨 뒤에 원소를 삽입한다
dequeue() ◀ 큐의 맨 앞에 있는 원소를 알려주고 삭제한다
front() ◀ 큐의 맨 앞에 있는 원소를 알려준다
isEmpty() ◀ 큐가 빈 큐인지를 알려준다
dequeueAll() ◀ 큐를 깨끗이 청소한다



삽입 Insertion

코드 7-1 큐에 원소 x 삽입하기

```
def enqueue(self, x):
    self.__queue.append(x)
```

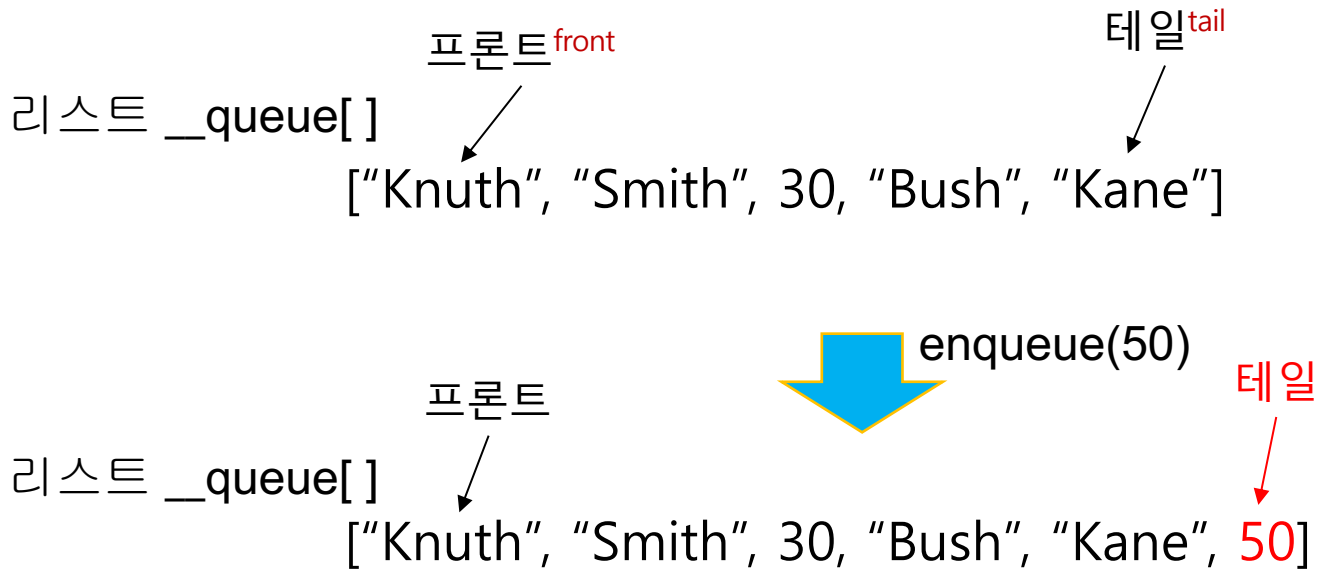
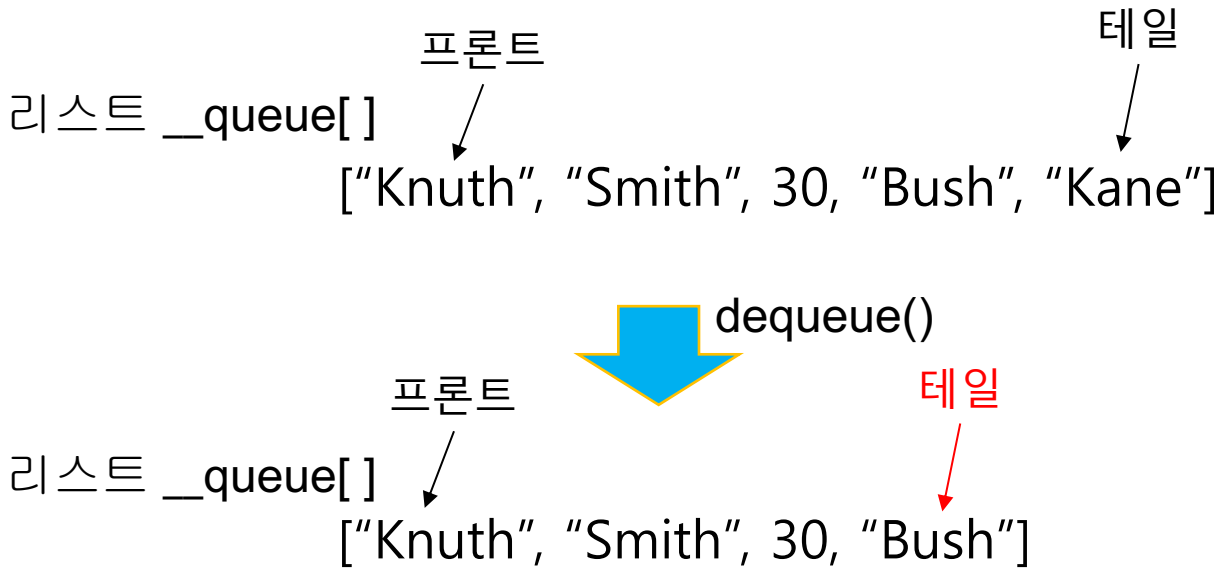


그림 7-6 리스트로 만든 큐에 원소를 삽입하는 예

삭제 Deletion

코드 7-2 큐에서 원소 삭제하기

```
def dequeue(self):
    return self.__queue.pop(0) # .pop(0): 리스트의 첫 원소를 삭제한 후 원소 리턴
```



기타 작업

코드 7-3 큐의 맨 앞 원소 알려주기

```
def front(self):
    return self.__queue[0]
```

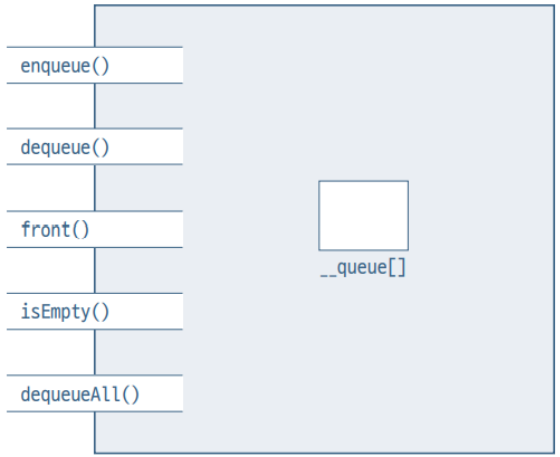
코드 7-4 큐가 비었는지 확인하기

```
def isEmpty(self) -> bool:
    return (len(self.__queue) == 0);
```

코드 7-5 큐 비우기

```
def dequeueAll(self):
    self.__queue = []    # 또는 self.__queue.clear()
```

파이썬 구현




```
class ListQueue:
        def __init__(self):
                self.__queue = []

        def enqueue(self, x):
                self.__queue.append(x)

        def dequeue(self):
                return self.__queue.pop(0)   # pop(0): 리스트의 첫 원소 삭제 후 리턴

        def front(self):
                if  self.isEmpty():
                        return None
                else:
                        return self.__queue[0]

        def isEmpty(self) -> bool:
                return (len(self.__queue) == 0)

        def dequeueAll(self):
                self.__queue.clear()

        def printQueue(self):
                print("Queue from front:", end=' ')
                for i in range(len(self.__queue)):
                        print()
```

클래스 ListQueue 사용 예

리스트 큐 사용 예

```
q1 = ListQueue()
q1.enqueue("Mon")
q1.enqueue("Tue")
q1.enqueue(1234)
q1.enqueue("Wed")
q1.dequeue()
q1.enqueue('aaa')
q1.printQueue()
```

수행 결과

```
Queue from front: Tue 1234 wed aaa
```

03 연결 리스트 큐

연결 리스트 큐 객체 구조

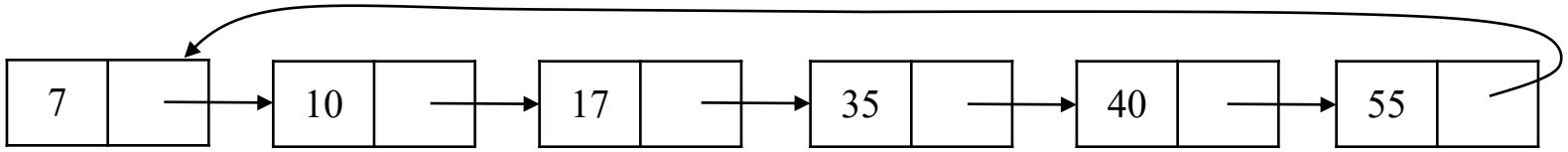


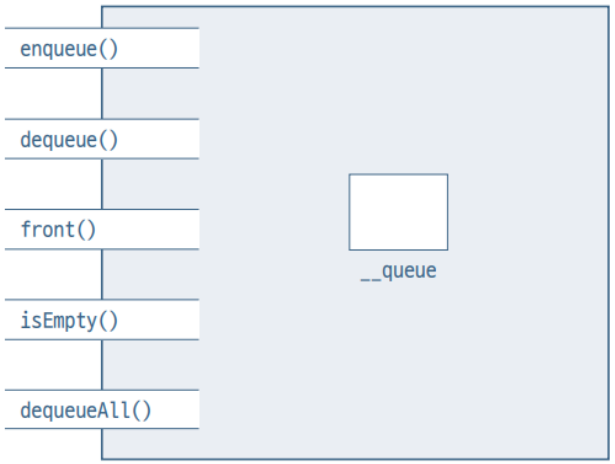
그림 **7-8** 연결 리스트를 이용한 큐의 예

필드:

__queue ◀ 큐 원소들이 저장되는 리스트

작업:

enqueue() ◀ 큐의 맨 뒤에 원소를 삽입한다
dequeue() ◀ 큐의 맨 앞에 있는 원소를 알려주고 삭제한다
front() ◀ 큐의 맨 앞에 있는 원소를 알려준다
isEmpty() ◀ 큐가 빈 큐인지를 알려준다
dequeueAll() ◀ 큐를 깨끗이 청소한다



삽입 Insertion

코드 7-8 큐에 원소 삽입하기

```
def enqueue(self, x):
    self.__queue.append(x)
```

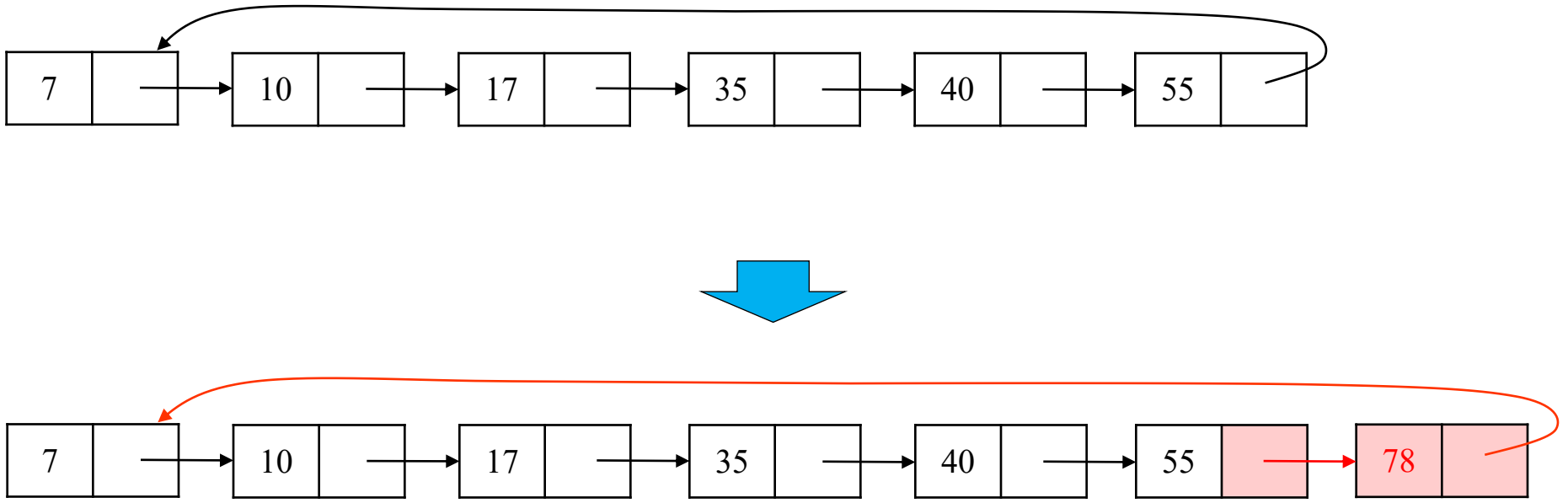


그림 7-10 원형 연결 리스트로 구현된 큐에 원소가 삽입되는 예

삭제 Deletion

코드 7-9 큐에서 원소 삭제하기

```
def dequeue(self):
    return self.__queue.pop(0) # .pop(0): 리스트의 첫 원소를 삭제한 후 원소 리턴
```

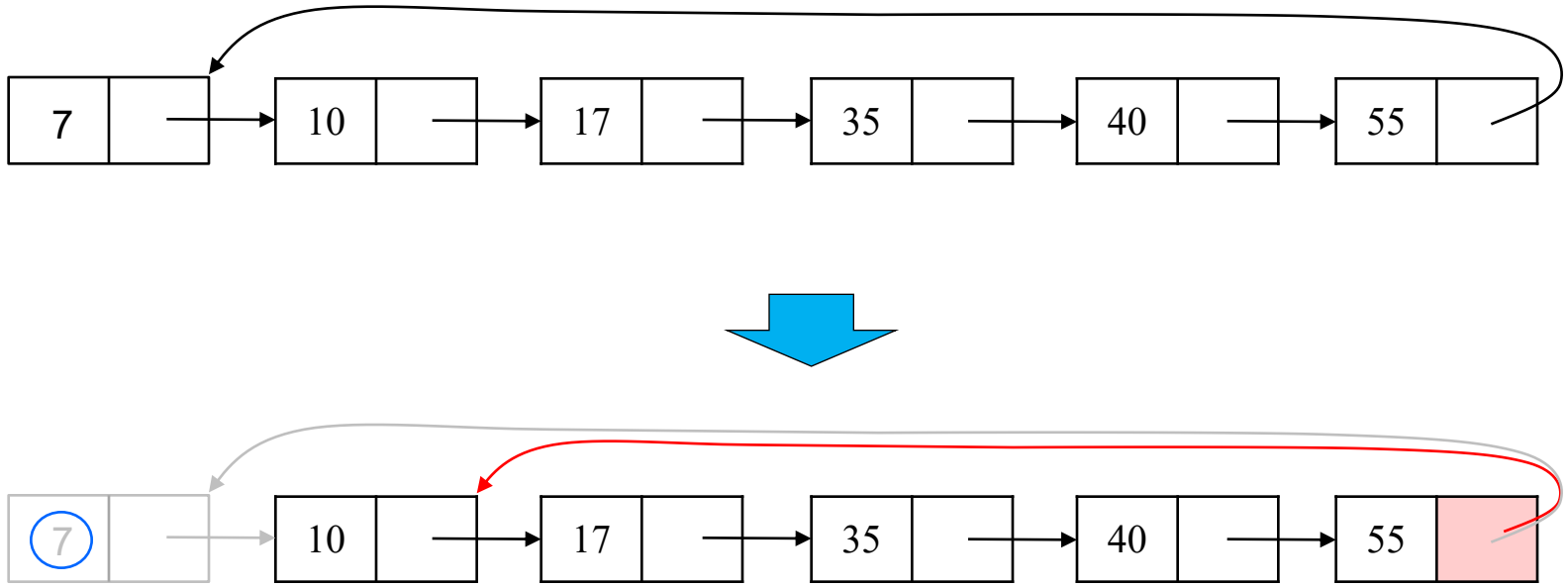
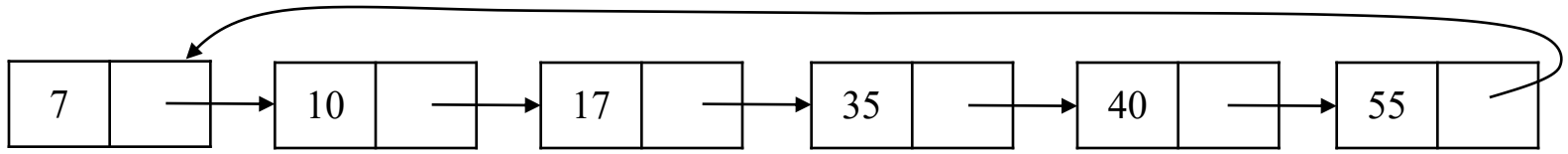


그림 7-11 원형 연결 리스트로 구현된 큐에서 삭제하는 예

기타 작업



코드 7-10 큐의 맨 앞 원소 알려주기

```
def front(self):
    return self.__queue.get(0)    # .get(0): 리스트의 첫 원소 리턴
```

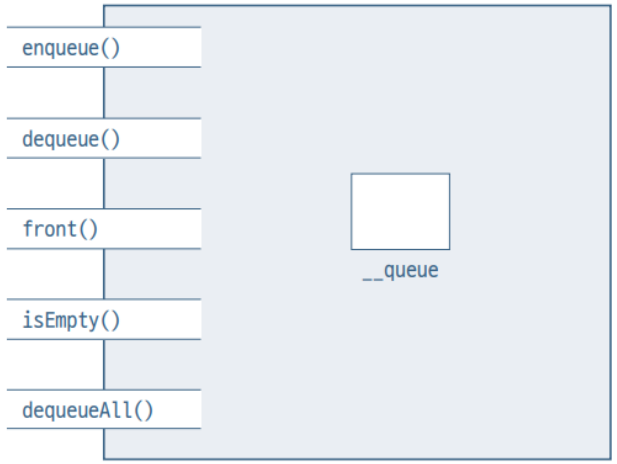
코드 7-11 큐가 비었는지 확인하기

```
def isEmpty(self) -> bool:
    return self.__queue.isEmpty()
```

코드 7-12 큐 비우기

```
def dequeueAll(self):
    self.__queue.clear()
```

파이썬 구현




```
class LinkedQueue:
        def __init__(self):
                self.__queue = CircularLinkedList()

        def enqueue(self, x):
                self.__queue.append(x)

        def dequeue(self):
                return self.__queue.pop(0)  # pop(0): 리스트의 첫 원소 삭제 후 리턴

        def front(self):
                return self.__queue.get(0)     # .get(0): 리스트의 첫 원소 리턴

        def isEmpty(self) -> bool:
                return (self.__queue.isEmpty())

        def dequeueAll(self):
                self.__queue.clear()

        def printQueue(self):
                print("Queue from front:", end=' ')
                for i in range(self.__queue.size()):
                        print(self.__queue.get(i), end=' ')
                print()
```

클래스 ListQueue 사용 예

연결 리스트 큐 사용 예

```
q1 = LinkedQueue()
q1.enqueue("Mon")
q1.enqueue("Tue")
q1.enqueue(1234)
q1.enqueue("Wed")
q1.dequeue()
q1.enqueue('aaa')
q1.printQueue()
```

수행 결과

```
Queue from front: Tue 1234 wed aaa
```

04 큐 활용 예

Palindrome 체크

```
def isPalindrome(A) -> bool:
        s = ListStack(); q = ListQueue()
        for i in range(len(A)):
                s.push(A[i]); q.enqueue(A[i])
        while (not q.isEmpty()) and s.pop() == q.dequeue():
                {}
        if q.isEmpty():
                return True
        else:
                return False

def main():
        print("Palindrome Check!")
        str = 'lioninoil'   # 테스트 문자열
        t = isPalindrome(str)
        print(str, " is Palindrome?: ", t)

if __name__ == "__main__":
        main()
```

- palindrome(좌우동형) 문자열 : 앞에서부터 읽으나 뒤에서부터 읽으나 같은 문자열

- 예
  abbcbba : 좌우동형 o
  abb : 좌우동형 x

출력

```
Palindrome Check!
Is lioninoil Palindrome?: true
```