07

CHAPTER

# 큐

학습목표

- 큐의 개념을 파악한다.
- 큐에 데이터를 넣거나 추출하는 원리를 이해한다.
- 파이썬으로 큐를 조작하는 코드를 작성한다.
- 큐로 활용되는 다양한 응용 프로그램을 작성한다.



# Section 00 생활 속 자료구조와 알고리즘

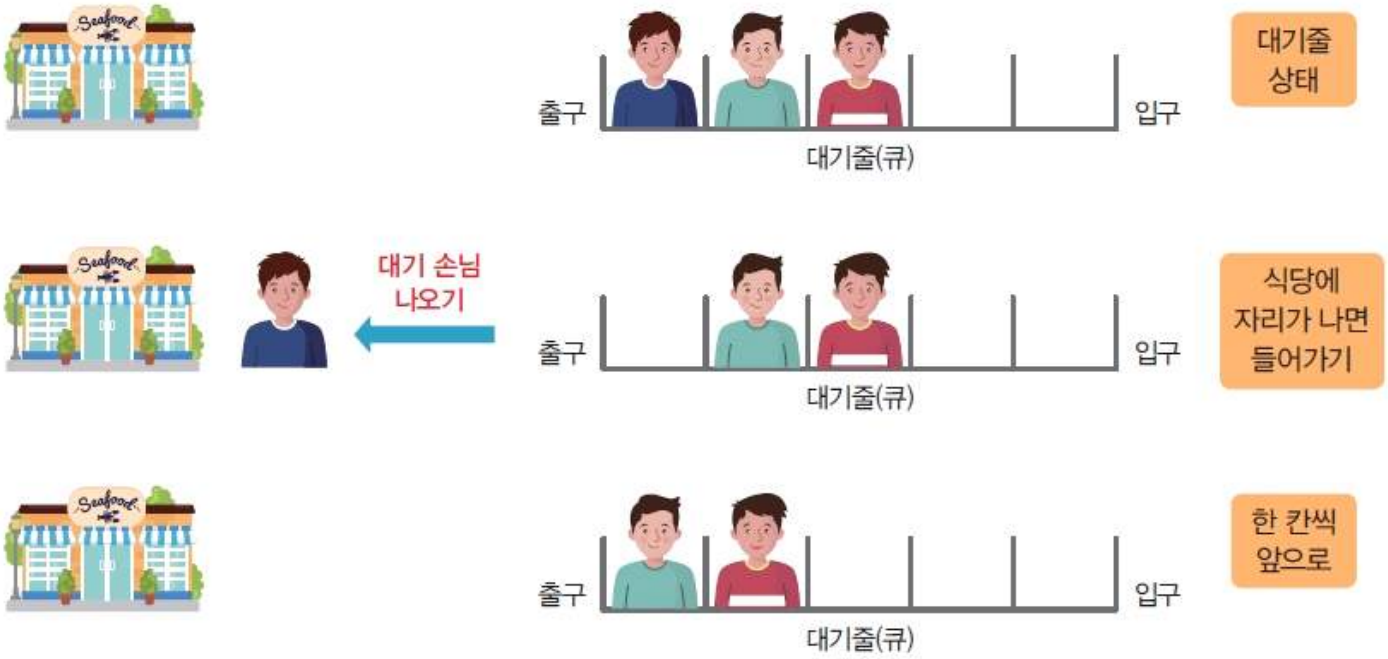
## 큐란?

- 기차가 터널에 들어가는 순서대로 터널을 빠져나오고, ATM기에서 줄을 선 순서대로 예금을 인출하는 것처럼 큐는 먼저 들어간 것이 먼저 나오는 구조를 의미

# 응용예제 01 유명 맛집의 대기줄 구현하기

난이도★★☆☆☆

**예제 설명** 유명 맛집의 대기줄에는 손님들이 들어온 순서대로 줄을 선다. 그리고 대기줄이 꽉 차면 더 이상 손님을 받지 않는다. 이제 대기줄 손님들은 자리가 생기면 1명씩 식당으로 들어간다. 맨 앞쪽 손님이 대기줄에서 식당으로 들어갈 때마다 대기줄 뒤쪽 손님들은 한 칸씩 이동해서 줄을 다시 서도록 한다.



- 큐(Queue)의 기본

# Section 01 큐의 기본

## 큐의 개념

- 큐(Queue) 자료구조는 입구와 출구가 따로 있는 원통 형태
  - 스택과 큐의 차이


그림 7-1 스택과 큐 차이점

# Section 01 큐의 기본

- 대기줄에 차례대로 줄을 서는 예

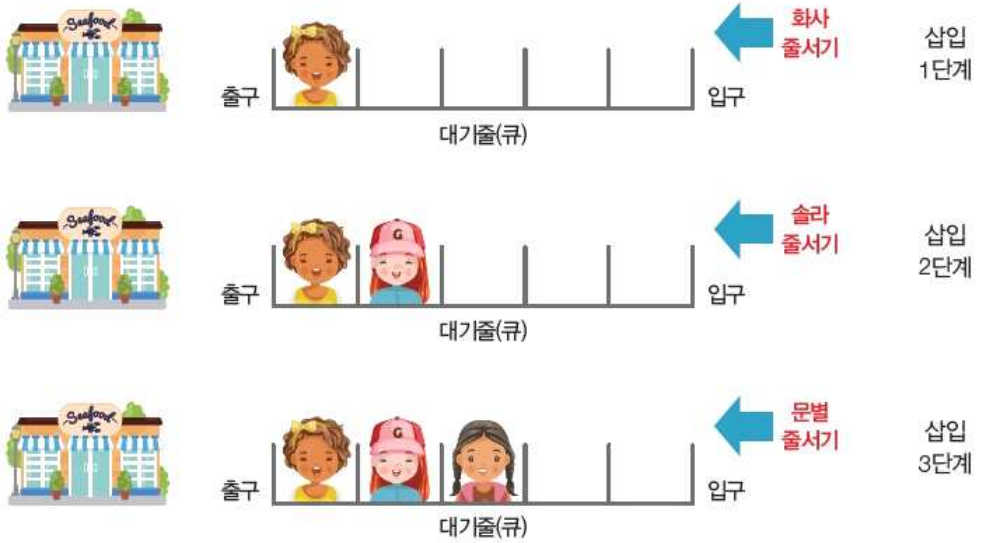



그림 7-2 대기줄(큐)의 삽입 작동 원리

- 대기줄에서 차례대로 나오는 예

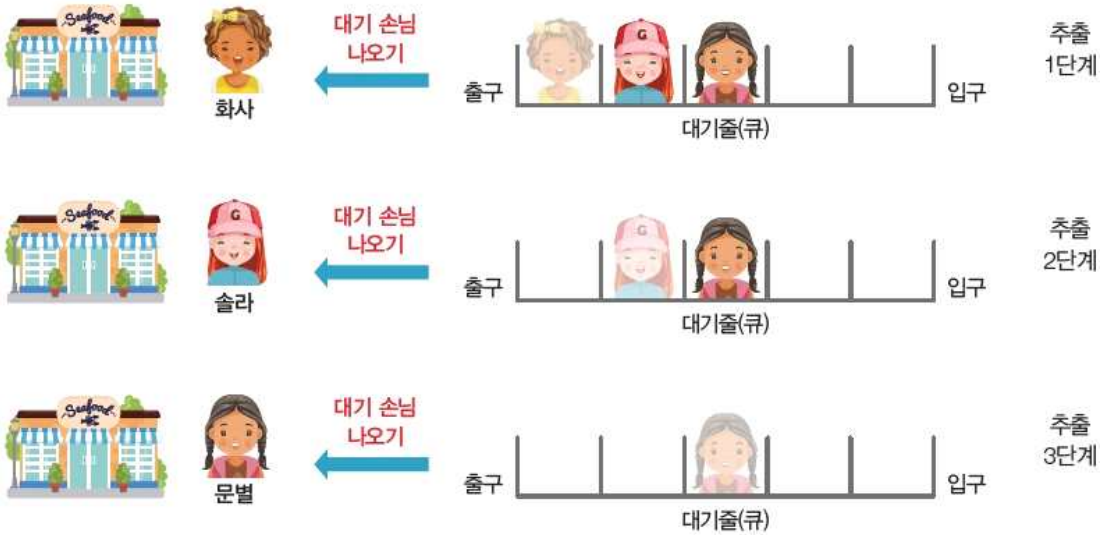



그림 7-3 대기줄(큐)의 추출 작동 원리

# Section 01 큐의 기본

## 큐 원리

### 구조와 용어

- 큐에 데이터를 삽입하는 작동 : enQueue(인큐)
- 데이터를 추출하는 작동 : deQueue(데큐)
- 저장된 데이터 중 첫 번째 데이터 : front(머리)
- 저장된 데이터 중 마지막 데이터 : rear(꼬리)

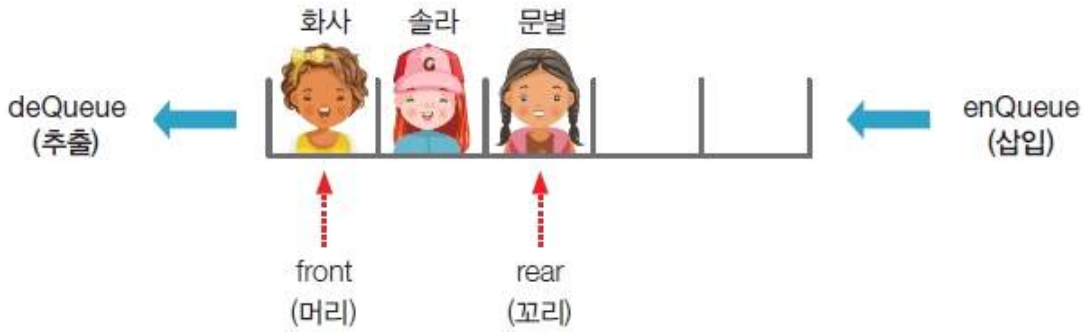



그림 7-4 **큐의 구조와 용어**

# Section 01 큐의 기본

- 데이터 삽입 : enQueue (Rear 이동)

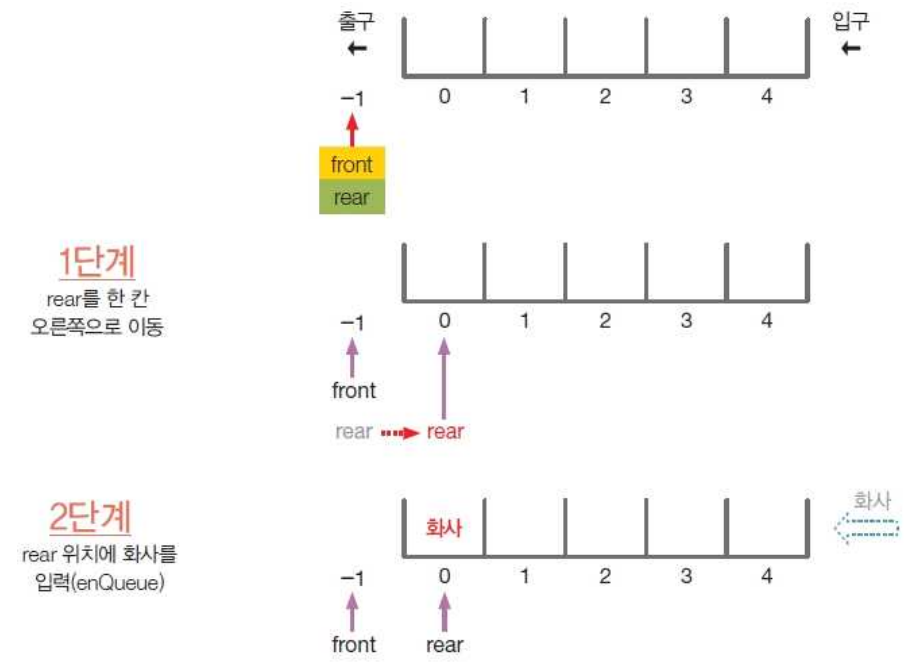



그림 7-5 큐에서 데이터를 삽입(enQueue)하는 과정

- 데이터 추출 : deQueue (Front 이동)

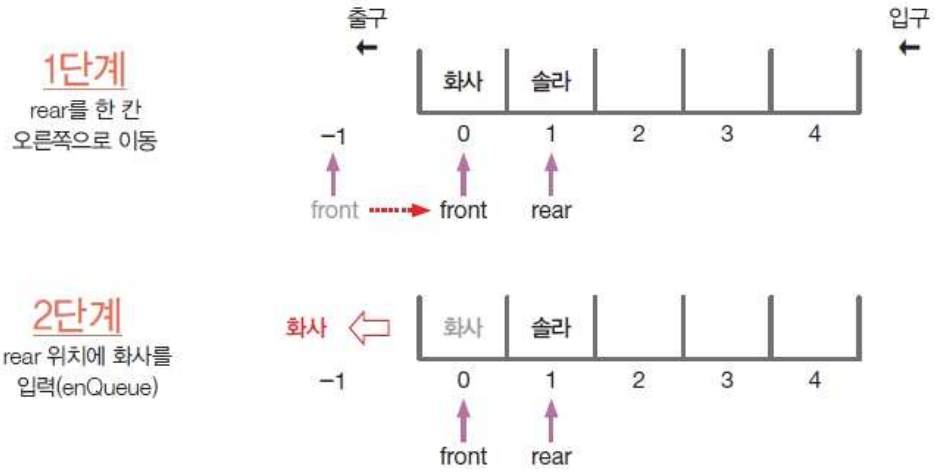



그림 7-6 큐에서 데이터를 추출하는 과정

# Section 02 큐의 간단 구현

## 큐 생성

- 배열 크기를 지정한 후 해당 크기의 빈 큐 생성

```
queue = [None, None, None, None, None]
front = rear = -1
```

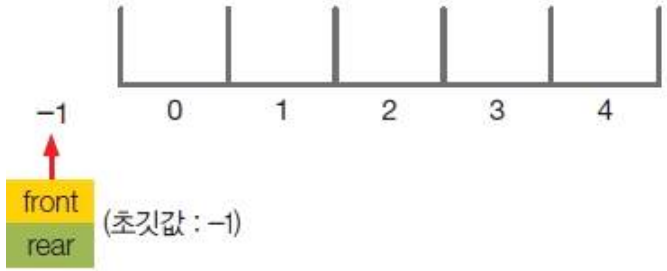



그림 7-7 크기가 5칸인 큐의 초기 상태

# 큐(Queue)의 간단 구현

# Section 02 큐의 간단 구현

## 데이터 삽입 : enQueue

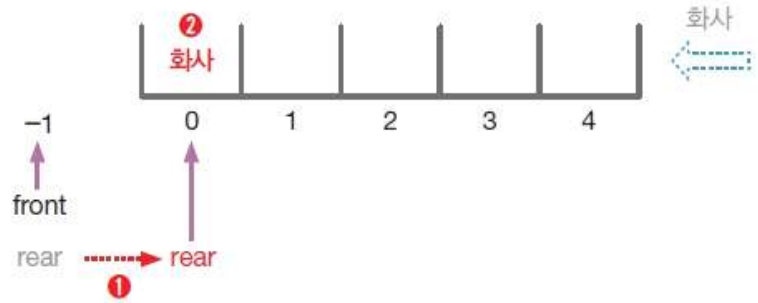



그림 7-8 큐의 데이터 삽입 과정

Code07-01.py 크기가 5칸인 큐의 생성과 데이터 3개 입력

```
1  queue = [None, None, None, None, None]
2  front = rear = -1
3
4  rear += 1 ❶
5  queue[rear] = "화사" ❷
6  rear += 1
7  queue[rear] = "솔라"
8  rear += 1
9  queue[rear] = "문별"
10
11 print("----- 큐 상태 -----")
12 print('[출구] <-- ', end = ' ')
13 for i in range(0, len(queue), 1) :
14     print(queue[i], end = ' ')
15 print('<-- [입구]')
```

실행 결과

```
----- 큐 상태 -----
[출구] <-- 화사 솔라 문별 None None <-- [입구]
```

# Section 02 큐의 간단 구현

## 데이터 추출 : deQueue


그림 7-9 큐의 데이터 추출 과정

Code07-02.py 큐에서 데이터 3개 추출

```
1  queue = ["화사", "솔라", "문별", None, None]
2  front = -1
3  rear = 2
4
5  print("----- 큐 상태 -----")
6  print('[출구] <-- ', end = ' ')
7  for i in range(0, len(queue), 1) :
8      print(queue[i], end = ' ')
9  print('<-- [입구]')
10 print("--------------------")
11
12 front += 1
13 data = queue[front]
14 queue[front] = None
15 print('deQueue --> ', data)
16
```

# Section 02 큐의 간단 구현

```
17 front += 1
18 data = queue[front]
19 queue[front] = None
20 print('deQueue --> ', data)
21
22 front += 1
23 data = queue[front]
24 queue[front] = None
25 print('deQueue --> ', data)
26 print("------------------")
27
28 print("----- 큐 상태 -----")
29 print('[출구] <-- ', end = ' ')
30 for i in range(0, len(queue), 1) :
31     print(queue[i], end = ' ')
32 print('<-- [입구]')
```

실행 결과

```
----- 큐 상태 -----
[출구] <-- 화사 솔라 문별 None None <-- [입구]
------------------
deQueue --> 화사
deQueue --> 솔라
deQueue --> 문별
------------------
----- 큐 상태 -----
[출구] <-- None None None None None <-- [입구]
```

- 큐(Queue)의 일반 구현

# Section 03 큐의 일반 구현

## 큐 초기화

- 5개짜리 빈 큐를 생성하는 코드

```
queue = [None, None, None, None, None]
```

- SIZE 값만 변경하면 원하는 크기의 빈 큐 생성(큐 초기화)

```
SIZE = 5    # 큐 크기
queue = [None for _ in range(SIZE)]
front = rear = -1
```

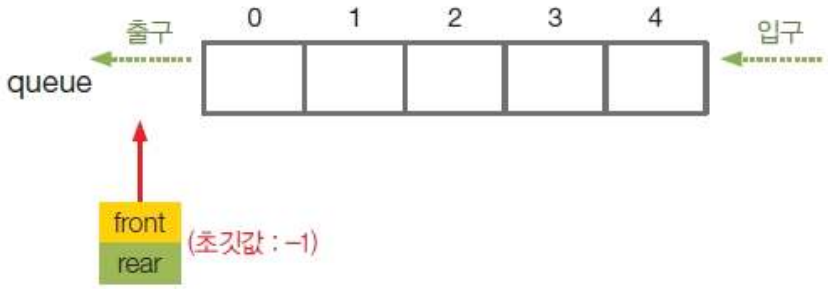



그림 7-10 초기화된 큐

## 데이터 삽입 과정

### 큐가 꽉 찼는지 확인하는 함수

- rear 값이 '큐 크기-1'과 같다면 큐가 꽉 찬 상태


그림 7-11 큐가 꽉 찬 상태

```
if (rear값 == 큐크기-1) :
    큐가 꽉 찼음
```

Code07-03.py 큐가 꽉 찼는지 확인하는 함수

```
1  def isQueueFull() :
2      global SIZE, queue, front, rear
3      if (rear == SIZE-1) :
4          return True
5      else :
6          return False
7
8  SIZE = 5
9  queue = ["화사", "솔라", "문별", "휘인", "선미"]
10 front = -1
11 rear = 4
12
13 print("큐가 꽉 찼는지 여부 ==>", isQueueFull())
```

실행 결과

```
큐가 꽉 찼는지 여부 ==> True
```

# Section 03 큐의 일반 구현

- 큐에 데이터를 삽입하는 함수

Code07-04.py 큐에 데이터를 삽입하는 함수

```
1  def isQueueFull() :
…  # 생략(Code07-03.py의 2~6행과 동일)
7
8  def enQueue(data) :
9      global SIZE, queue, front, rear
10     if (isQueueFull()) :
11         print("큐가 꽉 찼습니다.")
12         return
13     rear += 1
14     queue[rear] = data
15
16 SIZE = 5
17 queue = ["화사", "솔라", "문별", "휘인", None]
18 front = -1
19 rear = 3
20
21 print(queue)
22 enQueue("선미")
23 print(queue)
24 enQueue("재남")
```

실행 결과

```
['화사', '솔라', '문별', '휘인', None]
['화사', '솔라', '문별', '휘인', '선미']
큐가 꽉 찼습니다.
```

# Section 03 큐의 일반 구현

**SELF STUDY 7-1**

Code07-04.py의 isQueueFull() 함수를 없애고, 대신에 enQueue() 함수 안으로 그 기능을 모두 구현하자. 실행 결과는 Code07-04.py와 같다.

실행 결과

```
['화사', '솔라', '문별', '휘인', None]
['화사', '솔라', '문별', '휘인', '선미']
큐가 꽉 찼습니다.
```

## 데이터 추출 과정

### 큐가 비었는지 확인하는 함수

- front와 rear의 값이 같다면 큐가 비어 있는 상태

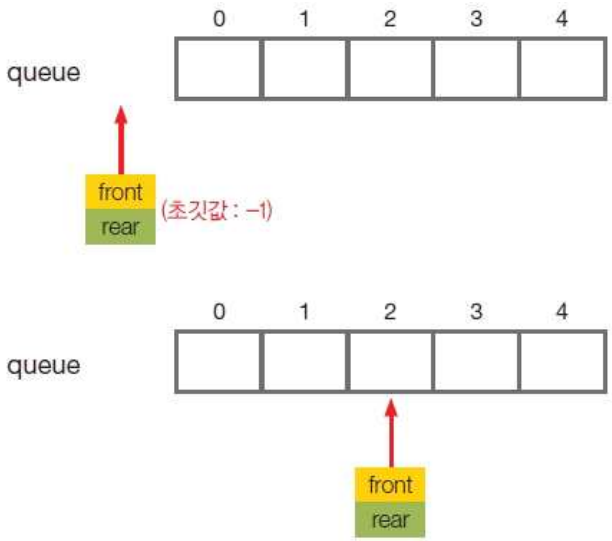



그림 7-12 두 가지 큐가 비어 있는 상태

```
if (front값 == rear값) :
    큐가 비었음
```

# Section 03 큐의 일반 구현

Code07-05.py 큐가 비었는지 체크하는 함수

```
1  def isQueueEmpty() :
2      global SIZE, queue, front, rear
3      if (front == rear) :
4          return True
5      else :
6          return False
7
8  SIZE = 5
9  queue = [None for _ in range(SIZE)]
10 front = rear = -1
11
12 print("큐가 비었는지 여부 ==>", isQueueEmpty())
```

실행 결과

```
큐가 비었는지 여부 ==> True
```

# Section 03 큐의 일반 구현

## 큐에서 데이터를 추출하는 함수

Code07-06.py 큐에서 데이터를 추출하는 함수

```
1  def isQueueEmpty() :
…  # 생략(Code07-05.py의 2~6행과 동일)
7
8  def deQueue() :
9      global SIZE, queue, front, rear
10     if (isQueueEmpty()) :
11         print("큐가 비었습니다.")
12         return None
13     front += 1
14     data = queue[front]
15     queue[front] = None
16     return data
17
18 SIZE = 5
19 queue = ["화사", None, None, None, None]
20 front = -1
21 rear = 0
22
23 print(queue)
24 retData = deQueue()
25 print("추출한 데이터 -->", retData)
26 print(queue)
27 retData = deQueue()
```

실행 결과

```
['화사', None, None, None, None]
추출한 데이터 --> 화사
[None, None, None, None, None]
큐가 비었습니다.
```

# Section 03 큐의 일반 구현

**SELF STUDY 7-2**

Code07-06.py의 isQueueEmpty() 함수를 없애고, 대신에 deQueue() 함수 안으로 그 기능을 모두 구현하자. 실행 결과는 Code07-06.py와 같다.

실행 결과

```
['화사', None, None, None, None]
추출한 데이터 --> 화사
[None, None, None, None, None]
큐가 비었습니다.
```

# Section 03 큐의 일반 구현

## 데이터 확인

추출될 데이터를 큐에 그대로 두고 확인만 하는 것 : peek(픽)

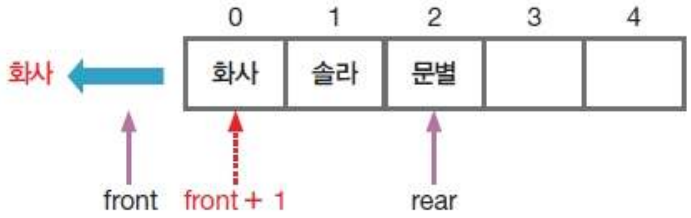



그림 7-13 데이터를 확인하는 peek 작동

Code07-07.py 큐에서 front+1 위치의 데이터를 확인하는 함수

```
1  def isQueueEmpty() :
…  # 생략(Code07-05.py의 2~6행과 동일)
7
8  def peek() :
9      global SIZE, queue, front, rear
10     if (isQueueEmpty()) :
11         print("큐가 비었습니다.")
12         return None
13     return queue[front+1]
14
15 SIZE = 5
16 queue = ["화사", "솔라", "문별", None, None]
17 front = -1
18 rear = 2
```

# Section 03 큐의 일반 구현

```
19
20 print(queue)
21 retData = peek()
22 print("다음에 추출될 데이터 확인 -->", retData)
23 print(queue)
```

실행 결과

```
['화사', '솔라', '문별', None, None]
다음에 추출될 데이터 확인 --> 화사
['화사', '솔라', '문별', None, None]
```

## 큐 완성

### 기능 통합 버전

Code07-08.py 큐 작동을 위한 통합 코드

```
## 함수 선언 부분 ##
def isQueueFull() :                     # 큐가 꽉 찼는지 확인하는 함수
    global SIZE, queue, front, rear
    if (rear == SIZE-1) :
        return True
    else :
        return False

def isQueueEmpty() :                    # 큐가 비었는지 확인하는 함수
    global SIZE, queue, front, rear
    if (front == rear) :
        return True
    else :
        return False

```

# Section 03 큐의 일반 구현

```
16 def enQueue(data) :              # 큐에 데이터를 삽입하는 함수
17     global SIZE, queue, front, rear
18     if (isQueueFull()) :
19         print("큐가 꽉 찼습니다.")
20         return
21     rear += 1
22     queue[rear] = data
23
24 def deQueue() :                   # 큐에서 데이터를 추출하는 함수
25     global SIZE, queue, front, rear
26     if (isQueueEmpty()) :
27         print("큐가 비었습니다.")
28         return None
29     front += 1
30     data = queue[front]
31     queue[front] = None
32     return data
33
34 def peek() :                      # 큐에서 front+1 위치의 데이터를 확인하는 함수
35     global SIZE, queue, front, rear
36     if (isQueueEmpty()) :
37         print("큐가 비었습니다.")
38         return None
39     return queue[front+1]
40
```

# Section 03 큐의 일반 구현

```
41 ## 전역 변수 선언 부분 ##
42 SIZE = int(input("큐 크기를 입력하세요 ==> "))
43 queue = [None for _ in range(SIZE)]
44 front = rear = -1
45
46 ## 메인 코드 부분 ##
47 if __name__ == "__main__" :
48     select = input("삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> ")
49
50     while (select != 'X' and select != 'x') :
51         if select == 'I' or select == 'i' :
52             data = input("입력할 데이터 ==> ")
53             enQueue(data)
54             print("큐 상태 : ", queue)
55         elif select == 'E' or select == 'e' :
56             data = deQueue()
57             print("추출된 데이터 ==> ", data)
58             print("큐 상태 : ", queue)
59         elif select == 'V' or select == 'v' :
60             data = peek()
61             print("확인된 데이터 ==> ", data)
62             print("큐 상태 : ", queue)
63         else :
64             print("입력이 잘못됨")
65
66         select = input("삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> ")
67
68     print("프로그램 종료!")
```

# Section 03 큐의 일반 구현

실행 결과

```
큐 크기를 입력하세요 ==> 5
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 화사
큐 상태 : ['화사', None, None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 솔라
큐 상태 : ['화사', '솔라', None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 문별
큐 상태 : ['화사', '솔라', '문별', None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 화사
큐 상태 : [None, '솔라', '문별', None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 솔라
큐 상태 : [None, None, '문별', None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 문별
큐 상태 : [None, None, None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
큐가 비었습니다.
추출된 데이터 ==> None
큐 상태 : [None, None, None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> X
프로그램 종료!
```

# Section 03 큐의 일반 구현

## 기능 통합 버전 개선

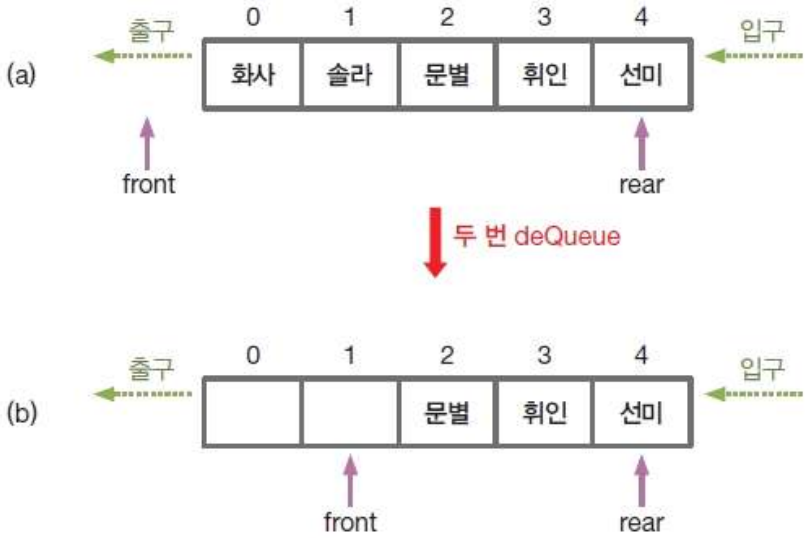



그림 7-14 큐가 꽉 찼는지 확인하는 함수의 문제점

```
if (rear값 == 큐크기-1) :
    큐가 꽉 찼음
```

```
❶ if (rear값 != 큐크기-1) :
        큐가 꽉 차지 않았음
❷ elif (rear값 == 큐크기-1) and (front == -1) :
        큐가 꽉 찼음
❸ else :
        [그림 7-14]의 (b)와 같은 상태로, 데이터를 앞으로 당기면 큐가 꽉 차지 않음
```

# Section 03 큐의 일반 구현

❶ rear ≠ 큐 크기-1인 경우


그림 7-15 rear 뒤에 여유가 있는 상태

❷ rear = 큐 크기-1, front = -1인 경우


그림 7-16 큐가 꽉 찬 상태

❸ rear = 큐 크기-1, front ≠ -1인 경우

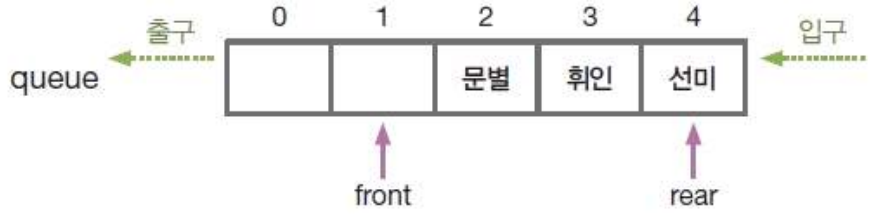
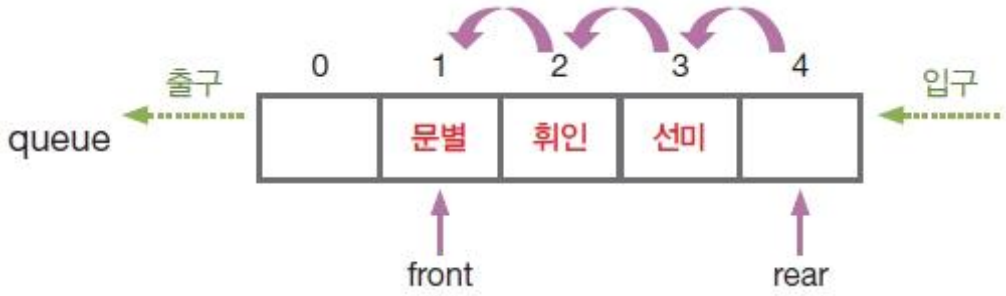



그림 7-17 rear는 끝이지만 앞쪽에 여유가 있는 경우

# Section 03 큐의 일반 구현

1. front+1 위치부터 rear 위치까지 왼쪽으로 한 칸씩 이동시킨다.
2. front 값에서 1을 뺀다.
3. rear 값에서 1을 뺀다.
4. 큐가 꽉 차지 않았다는 의미의 False를 반환한다.

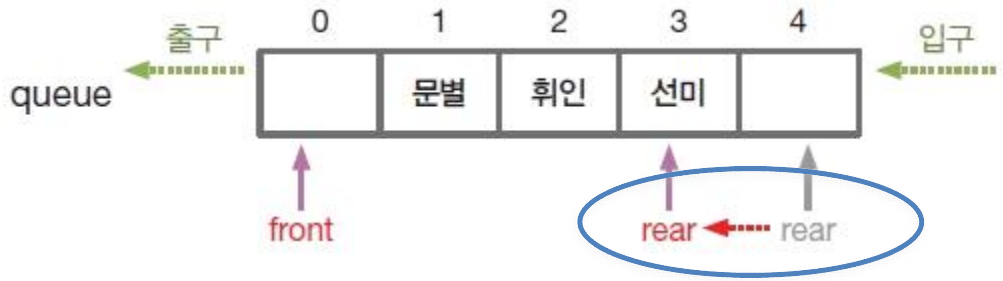
1 front+1 위치부터 마지막 칸까지 왼쪽으로 한 칸씩 이동시킨다.




2 front 값에서 1을 뺀다. 즉, front를 왼쪽으로 한 칸 이동한다.

# Section 03 큐의 일반 구현

3 rear 값에서 1을 뺀다. 즉, rear를 왼쪽으로 한 칸 이동한다.




4 큐가 꽉 차지 않았다는 의미의 False를 반환한다. 그러면 큐가 꽉 차지 않은 것이므로 마지막 칸에 데이터를 추가할 것이다.

# Section 03 큐의 일반 구현

Code07-09.py 큐가 꽉 찼는지 확인하는 함수 개선 버전

```
1  def isQueueFull() :
2      global SIZE, queue, front, rear
3      if (rear != SIZE-1) :
4          return False
5      elif (rear == SIZE-1) and (front == -1) :
6          return True
7      else :
8          for i in range(front+1, SIZE) :
9  (1)         queue[i-1] = queue[i]
10             queue[i] = None
11     (2) front -= 1
12     (3) rear -= 1
13     (4) return False
14
15 SIZE = 5
16 queue = [None, None, "문별", "휘인", "선미"]
17 front = 1
18 rear = 4
19
20 print("큐가 꽉 찼는지 여부 ==>", isQueueFull())
21 print("큐 상태 ==> ", queue)
```

실행 결과

```
큐가 꽉 찼는지 여부 ==> False
큐 상태 ==>  [None, '문별', '휘인', '선미', None]
```

# Section 03 큐의 일반 구현

- 개선된 큐 완성

Code07-10.py 큐 작동을 위한 통합 코드 수정

```
## 함수 선언 부분 ##
def isQueueFull() :
    global SIZE, queue, front, rear
    if (rear != SIZE-1) :
        return False
    elif (rear == SIZE-1) and (front == -1) :
        return True
    else :
        for i in range(front+1, SIZE) :
            queue[i-1] = queue[i]
            queue[i] = None
        front -= 1
        rear -= 1
        return False

... # 생략(Code07-08.py의 9~67행과 동일)
    print("프로그램 종료!")
```

# Section 03 큐의 일반 구현

실행 결과

```
큐 크기를 입력하세요 ==> 5
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 화사
큐 상태 : ['화사', None, None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 솔라
큐 상태 : ['화사', '솔라', None, None, None]
… → 솔라, 문별, 휘인, 선미를 차례로 입력
큐 상태 : ['화사', '솔라', '문별', '휘인', '선미']
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 화사
큐 상태 : [None, '솔라', '문별', '휘인', '선미']
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 솔라
큐 상태 : [None, None, '문별', '휘인', '선미']
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 재남
큐 상태 : [None, '문별', '휘인', '선미', '재남']
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> V
확인된 데이터 ==> 문별
큐 상태 : [None, '문별', '휘인', '선미', '재남']
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> X
프로그램 종료!
```

## 원형 큐의 개념 : 큐의 처음과 끝이 연결된 구조

- 크기가 10만 개인 순차 큐의 앞쪽 일부를 제외하고 데이터가 꽉 찬 상태


그림 7-18 순차 큐에서 앞쪽 일부를 제외하고 데이터가 꽉 찬 상태

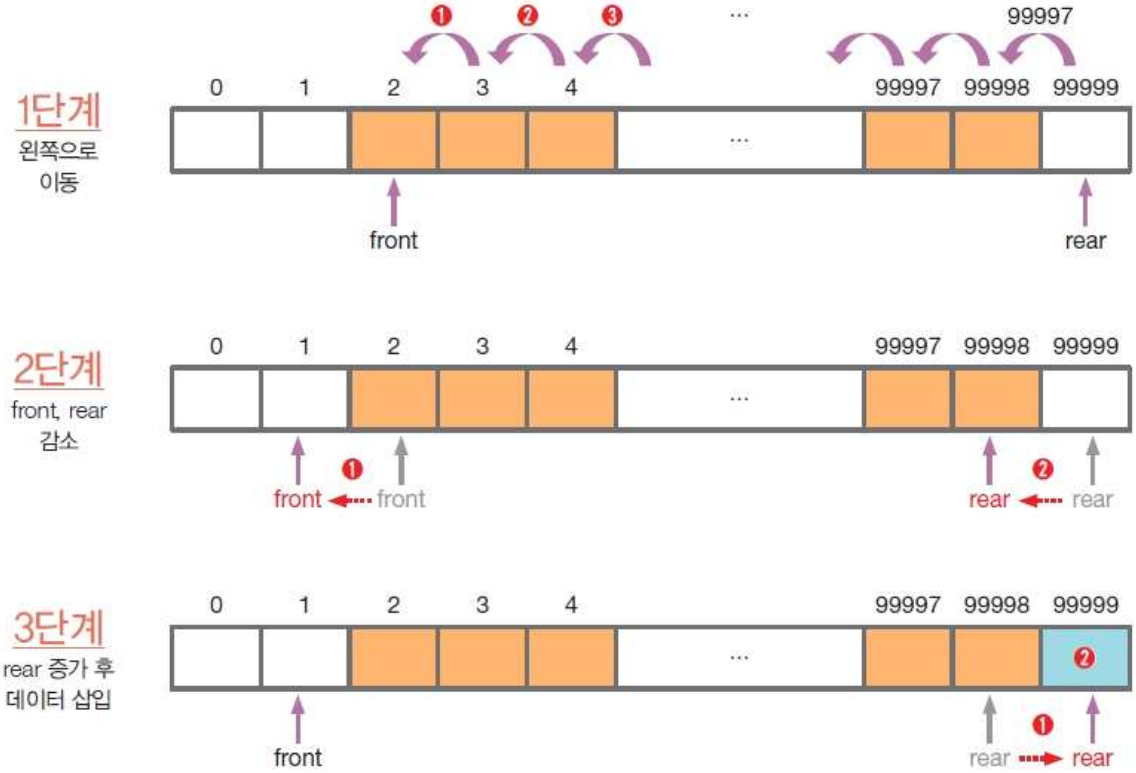



그림 7-19 앞쪽 일부를 제외하고 데이터가 꽉 찬 순차 큐에서 데이터 삽입

← 순차 큐에서는
1단계에서 오버헤드 발생

# Section 04 큐의 응용

- 순차 큐를 구부려서 끝을 이은 원형 큐
  - 오버헤드가 발생하지 않음

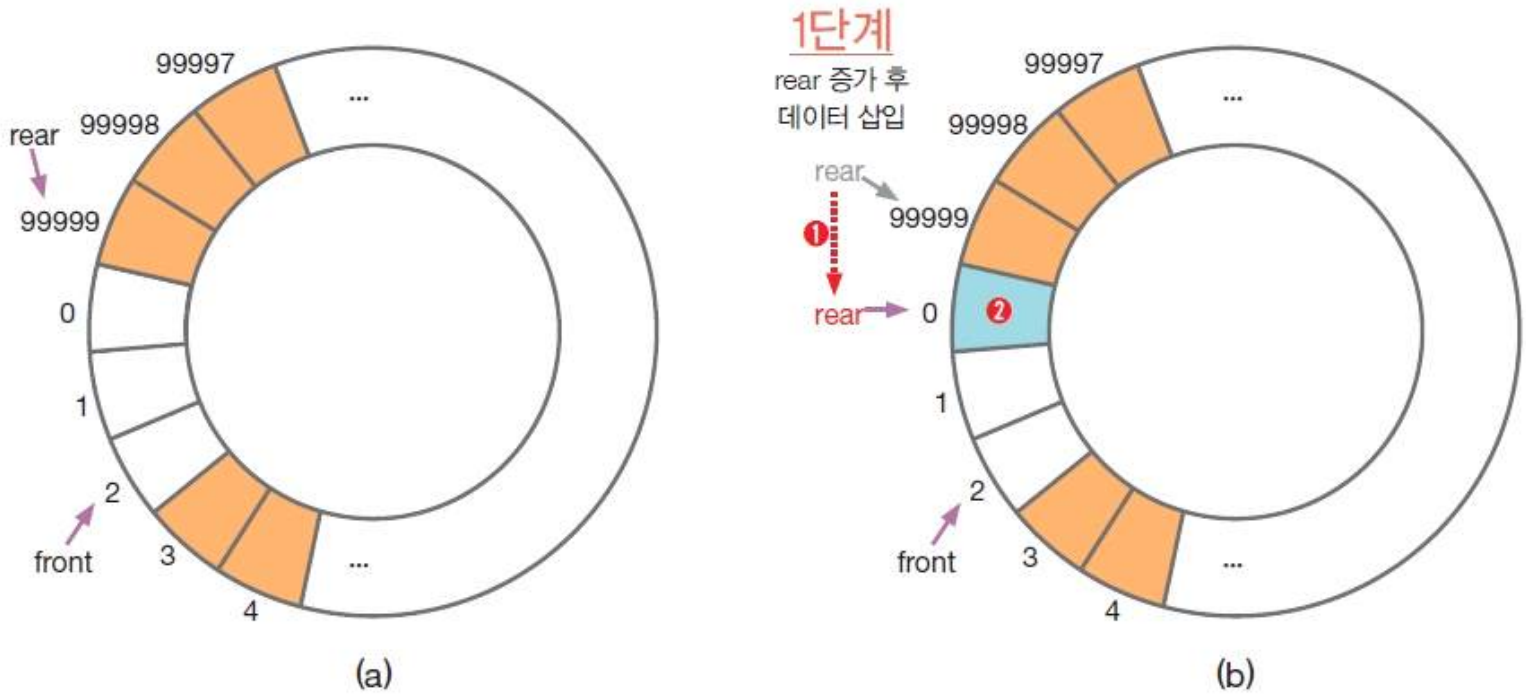



그림 7-20 앞쪽 일부를 제외하고 데이터가 꽉 찬 원형 큐에서 데이터 삽입

## 원형 큐 원리

### 원형 큐 초기화

```
원형큐 = [None, None, None, None, None]
front = rear = 0
```

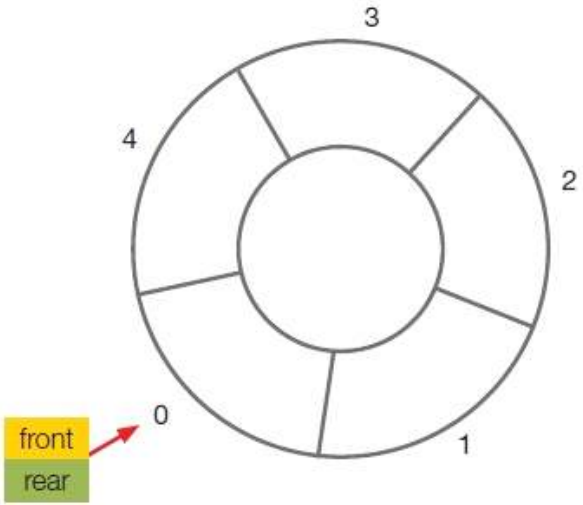
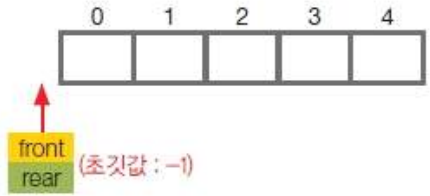



그림 7-21 원형 큐 초기화

**TIP** • 순차 큐 초기화

순차큐 = [None, None, None, None, None]
front = rear = –1



# Section 04 큐의 응용

- 원형 큐가 빈 경우와 꽉 찬 경우
  - front 와 rear가 동일하면 비어 있다는 의미

```
if (front값 == rear값) :
    큐가 비었음
```

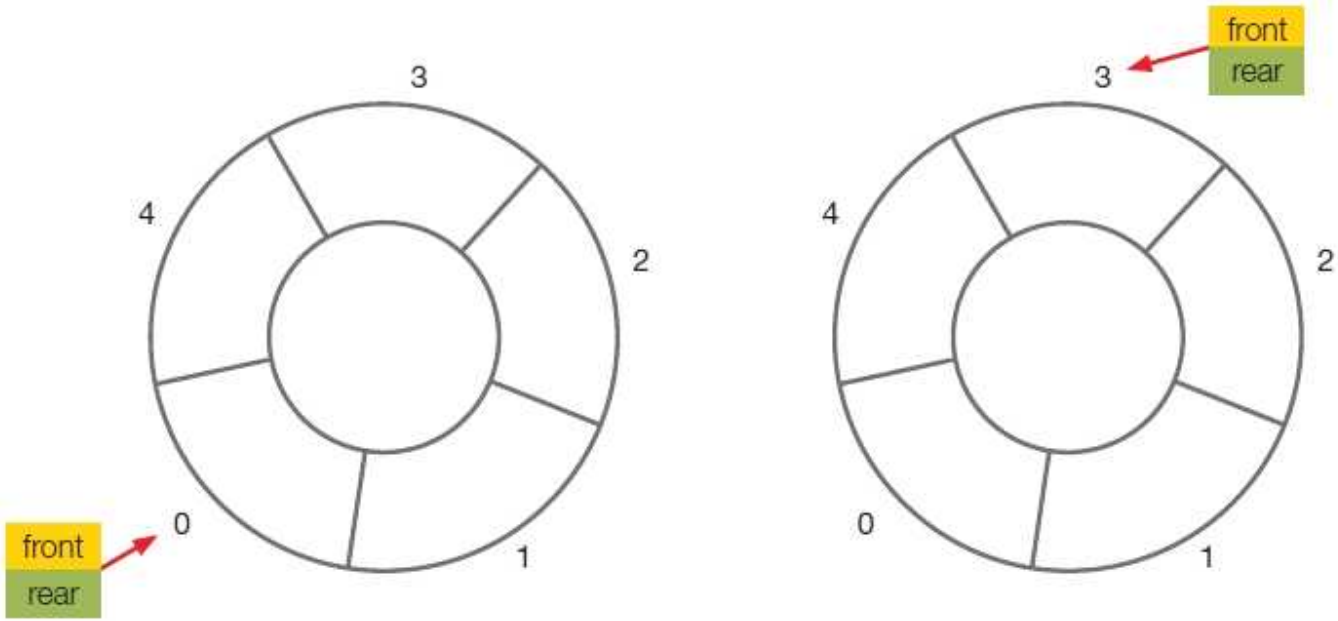



그림 7-22 원형 큐가 비어 있는 상태

# Section 04 큐의 응용

- 원형 큐가 빈 경우와 꽉 찬 경우
  - rear+1과 front가 같은 경우에 원형 큐가 꽉 찬 것으로 처리

```
if ((rear값+1) % 5 == front값) :
    큐가 꽉 찼음
```

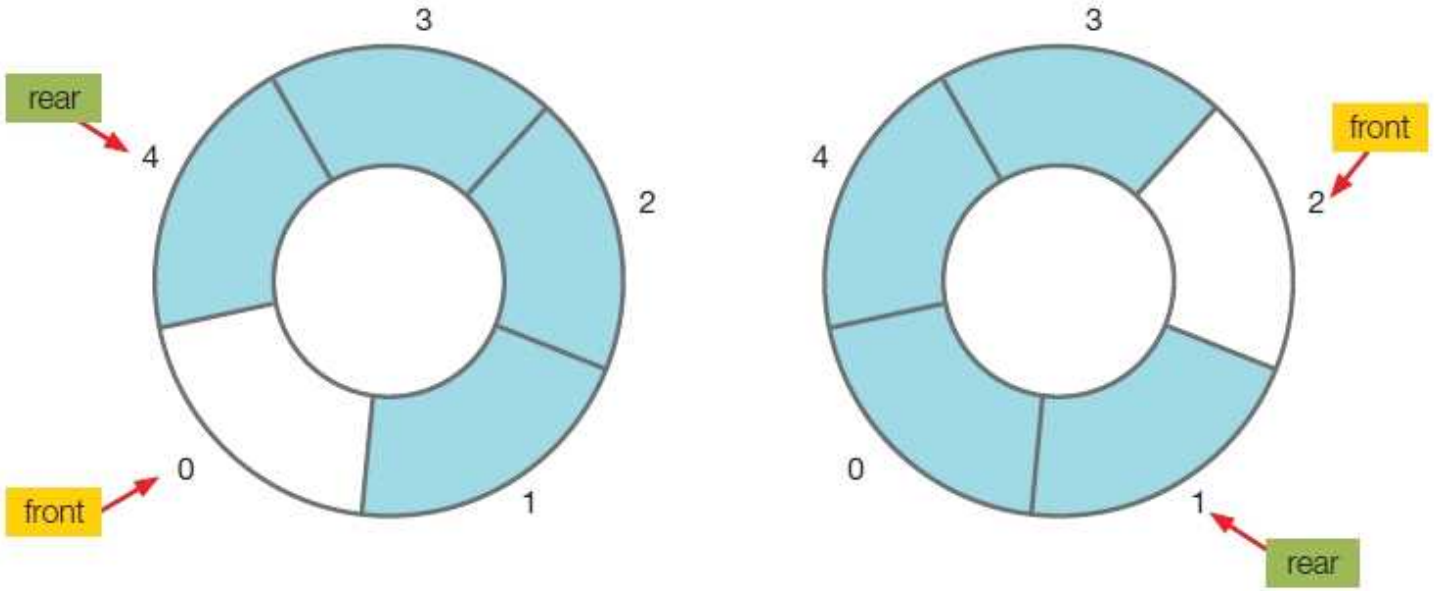



그림 7-23 원형 큐가 꽉 찬 상태

# Section 04 큐의 응용

- 원형 큐가 꽉 찼지만 큐가 비어 있다는 의미로 해석(잘못된 예)

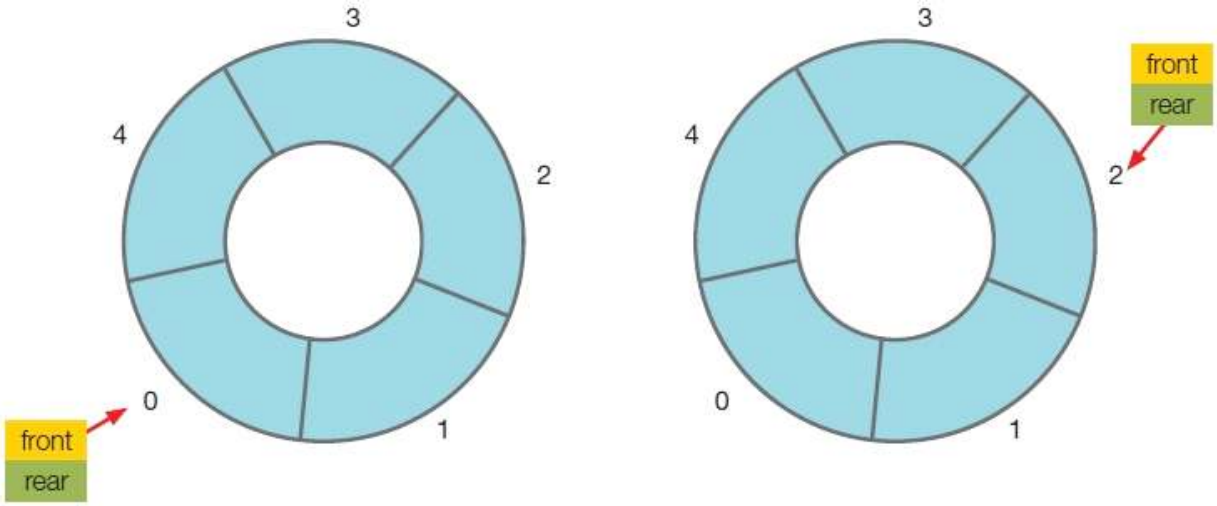


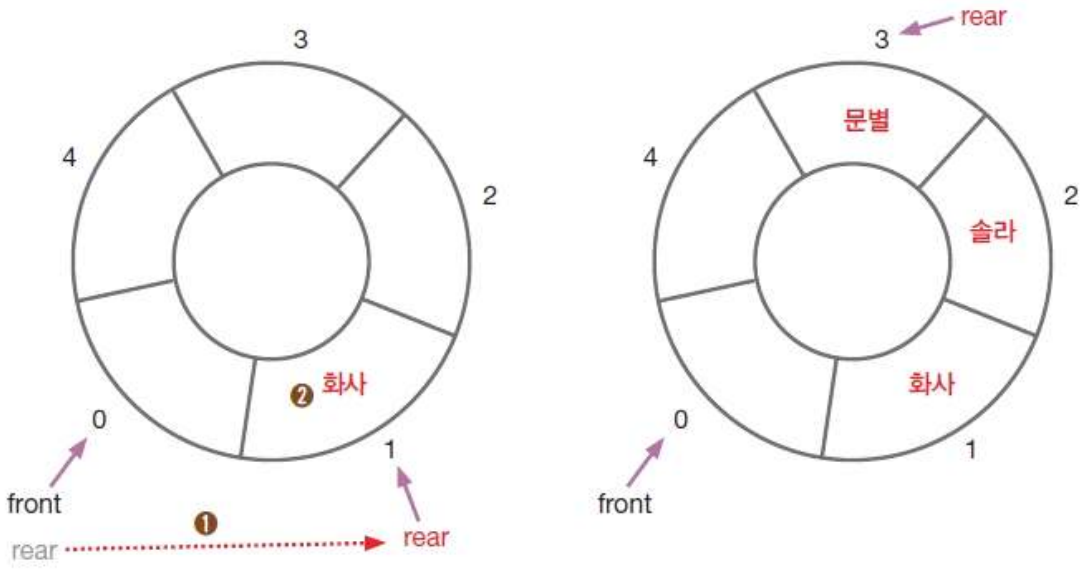

그림 7-24 원형 큐가 꽉 찼지만 빈 것으로 처리되는 잘못된 예

# Section 04 큐의 응용

## 원형 큐의 데이터 삽입과 추출

### 원형 큐에서 데이터를 삽입하는 예

```
if (큐가 꽉 찼음) :
    return
❶ rear = (rear+1) % 큐크기
❷ queue[rear] = "화사"
```




(a) 화사 입력 (b) 솔라와 문별 입력

그림 7-25 원형 큐에 데이터 입력

# Section 04 큐의 응용

## 원형 큐의 데이터 삽입과 추출

### 원형 큐에서 데이터를 추출하는 예

```
if (큐가 비었음) :
    return
❶ front = (front+1) % 큐크기
❷ 데이터 = queue[front]
❸ queue[front] = None
```

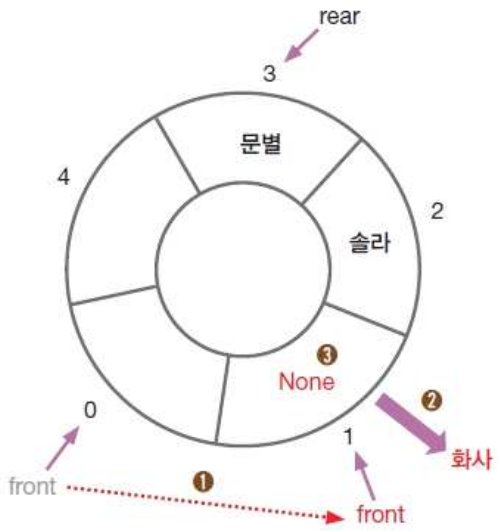



그림 7-26 원형 큐에서 데이터 추출

## 원형 큐 구현

Code07-11.py 원형 큐 작동을 위한 통합 코드

```
## 함수 선언 부분 ##
def isQueueFull() :
    global SIZE, queue, front, rear
    if ((rear+1) % SIZE == front) :
        return True
    else :
        return False

def isQueueEmpty() :
    global SIZE, queue, front, rear
    if (front == rear) :
        return True
    else :
        return False

def enQueue(data) :
    global SIZE, queue, front, rear
    if (isQueueFull()) :
        print("큐가 꽉 찼습니다.")
        return
    rear = (rear+1) % SIZE
    queue[rear] = data

```

# Section 04 큐의 응용

```
24 def deQueue() :
25     global SIZE, queue, front, rear
26     if (isQueueEmpty()) :
27         print("큐가 비었습니다.")
28         return None
29     front = (front+1) % SIZE
30     data = queue[front]
31     queue[front] = None
32     return data
33
34 def peek() :
35     global SIZE, queue, front, rear
36     if (isQueueEmpty()) :
37         print("큐가 비었습니다.")
38         return None
39     return queue[(front+1) % SIZE]
40
41 ## 전역 변수 선언 부분 ##
42 SIZE = int(input("큐 크기를 입력하세요 ==> "))
43 queue = [None for _ in range(SIZE)]
44 front = rear = 0
45
```

```
46 ## 메인 코드 부분 ##
47 if __name__ == "__main__" :
48     select = input("삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> ")
49
50     while (select != 'X' and select != 'x') :
51         if select == 'I' or select == 'i' :
52             data = input("입력할 데이터 ==> ")
53             enQueue(data)
54             print("큐 상태 : ", queue)
55             print("front : ", front, ", rear : ", rear)
56         elif select == 'E' or select == 'e' :
57             data = deQueue()
58             print("추출된 데이터 ==> ", data)
59             print("큐 상태 : ", queue)
60             print("front : ", front, ", rear : ", rear)
61         elif select == 'V' or select == 'v' :
62             data = peek()
63             print("확인된 데이터 ==> ", data)
64             print("큐 상태 : ", queue)
65             print("front : ", front, ", rear : ", rear)
66         else :
67             print("입력이 잘못됨")
68
69         select = input("삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> ")
70
71     print("프로그램 종료!")
```

# Section 04 큐의 응용

실행 결과

```
큐 크기를 입력하세요 ==> 5
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 화사
큐 상태 : [None, '화사', None, None, None]
front : 0 , rear : 1
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 솔라
큐 상태 : [None, '화사', '솔라', None, None]
front : 0 , rear : 2
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 문별
큐 상태 : [None, '화사', '솔라', '문별', None]
front : 0 , rear : 3
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 휘인
큐 상태 : [None, '화사', '솔라', '문별', '휘인']
front : 0 , rear : 4
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 재남
큐가 꽉 찼습니다.
큐 상태 : [None, '화사', '솔라', '문별', '휘인']
```

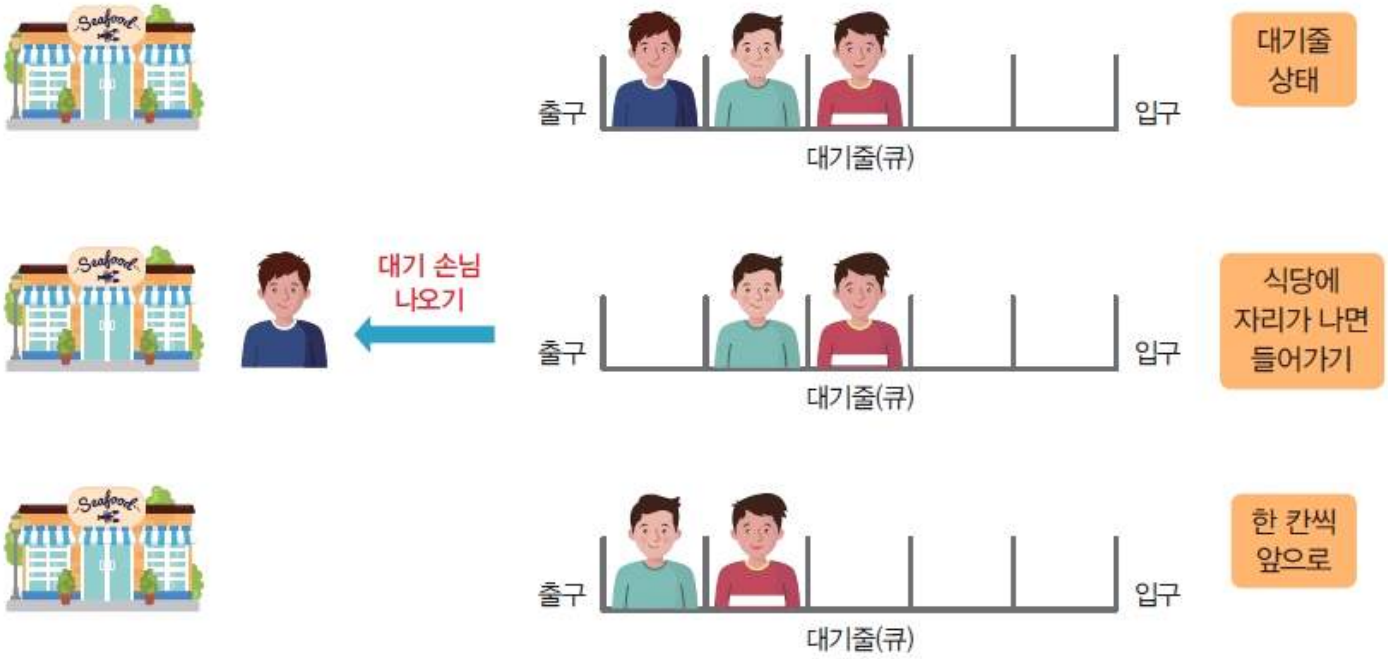
# Section 04 큐의 응용

```
front : 0 , rear : 4
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 화사
큐 상태 : [None, None, '솔라', '문별', '휘인']
front : 1 , rear : 4
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 재남
큐 상태 : ['재남', None, '솔라', '문별', '휘인']
front : 1 , rear : 0
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> V
확인된 데이터 ==> 솔라
큐 상태 : ['재남', None, '솔라', '문별', '휘인']
front : 1 , rear : 0
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==>  솔라
큐 상태 : ['재남', None, None, '문별', '휘인']
front : 2 , rear : 0
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> X
프로그램 종료!
```

# 응용예제 01 유명 맛집의 대기줄 구현하기

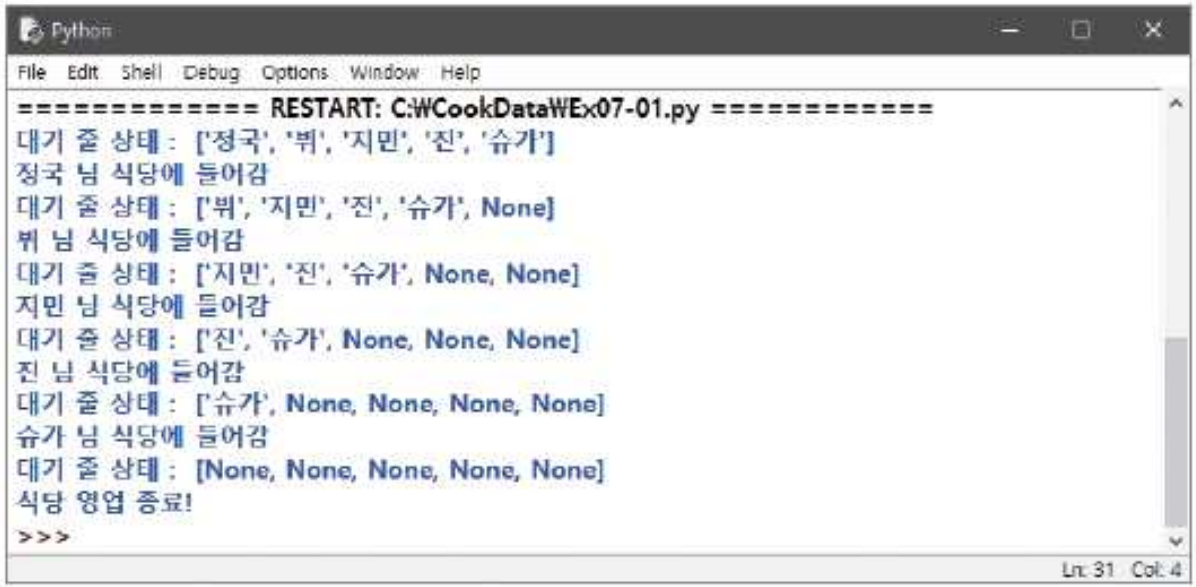
난이도★★☆☆☆

**예제 설명** 유명 맛집의 대기줄에는 손님들이 들어온 순서대로 줄을 선다. 그리고 대기줄이 꽉 차면 더 이상 손님을 받지 않는다. 이제 대기줄 손님들은 자리가 생기면 1명씩 식당으로 들어간다. 맨 앞쪽 손님이 대기줄에서 식당으로 들어갈 때마다 대기줄 뒤쪽 손님들은 한 칸씩 이동해서 줄을 다시 서도록 한다.



# 응용예제 01 유명 맛집의 대기줄 구현하기

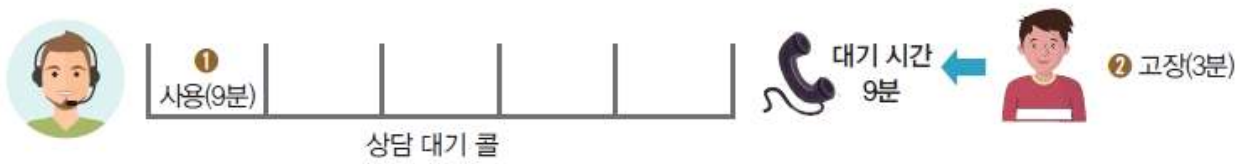
실행 결과



# 응용예제 02 콜센터의 응답 대기 시간 계산하기

난이도★★☆☆

**예제 설명** 한빛전자서비스 콜센터는 9시에 영업을 시작한다. 9시 전부터 전화 문의가 여러 건 대기하고 있다. 전화 문의는 주제에 따라 통화 시간이 다를 것으로 예상한다. 예로 고장 수리는 3분, 사용 문의는 9분, 환불 문의는 4분, 기타 문의는 1분으로 통화 예상 시간이 설정되어 있다. 9시 이전에 고객이 전화를 하면 9시에 업무를 개시한 후 자신이 어느 정도 대기해야 하는지 시간을 알려 준다(단 원형 큐를 이용하여 구현한다).

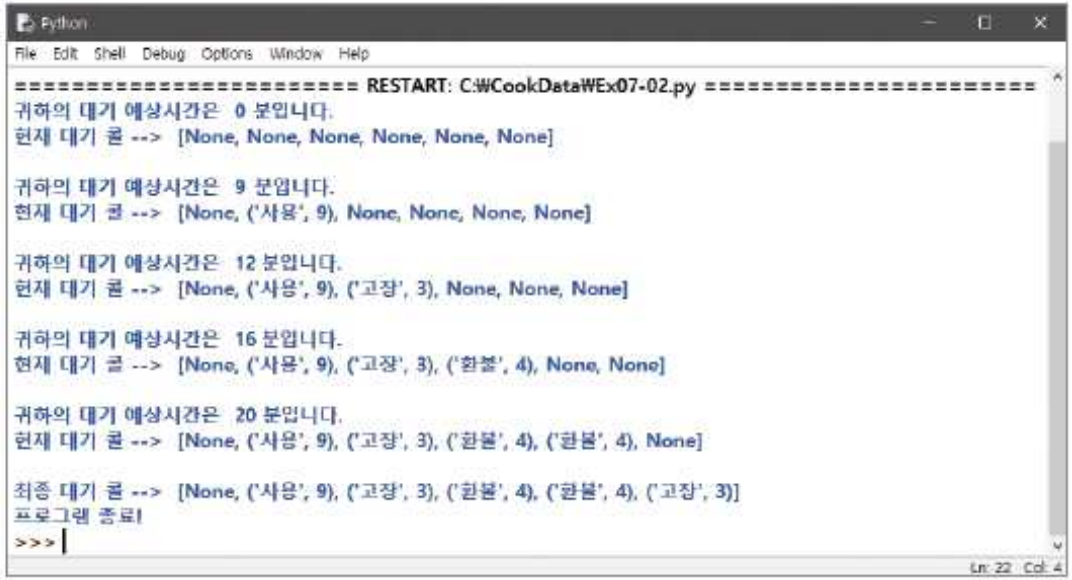





❹ 환불(4분)
❺ 고장(3분)

# 응용예제 02 콜센터의 응답 대기 시간 계산하기

실행 결과



Thank You