CHAPTER.14

큐의 일반 구현 및 응용

학습 내용

[1] 큐의 일반 구현

[2] 큐의 응용

학습 목표

- 파이썬으로 큐를 조작하는 코드를 작성할 수 있다.
- 큐로 활용되는 다양한 응용 프로그램을 작성할 수 있다.

01 큐의 일반 구현

1) 큐의 초기화
2) 데이터 삽입 과정
3) 데이터 추출 과정
4) 데이터 확인
5) 큐의 완성

1] 큐 초기화

5개짜리 빈 큐를 생성하는 코드

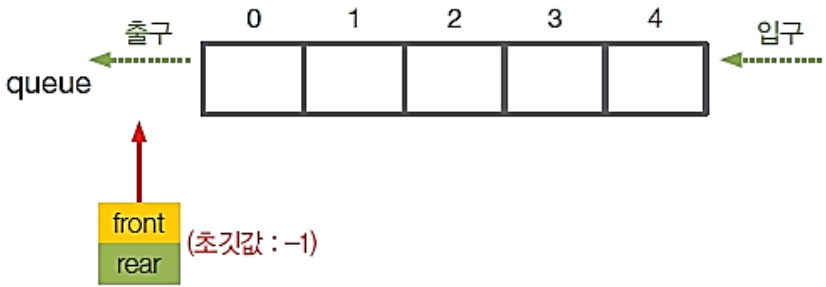
```
queue = [None, None, None, None, None]
```

SIZE 값만 변경하면 원하는 크기의 빈 큐 생성(큐 초기화)

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

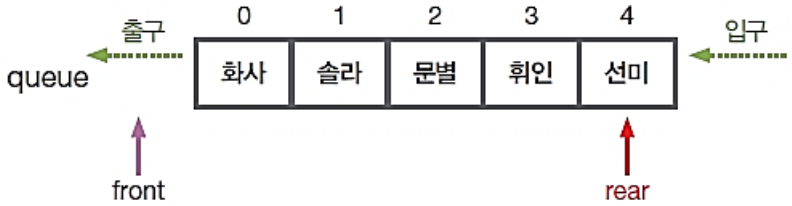
SIZE 값만 변경하면 원하는 크기의 빈 큐 생성(초기화)

- 초기화된 큐



큐가 꽉 찼는지 확인하는 함수

- rear 값이 '큐 크기-1'과 같다면 큐가 꽉 찬 상태




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

2] 데이터 삽입 과정

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
```

2] 데이터 삽입 과정

큐에 데이터를 삽입하는 함수

```
1  def isQueueFull() :
…  # 생략(    앞쪽의    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
```

큐에 데이터를 삽입하는 함수

```
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.
```

2] 데이터 삽입 과정

데이터 삽입 과정 실습

앞쪽의 소스의 isQueueFull() 함수를 없애고,
대신에 enQueue() 함수 안으로 그 기능을 모두 구현하자.
실행 결과는 다음과 같다.

실행 결과

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

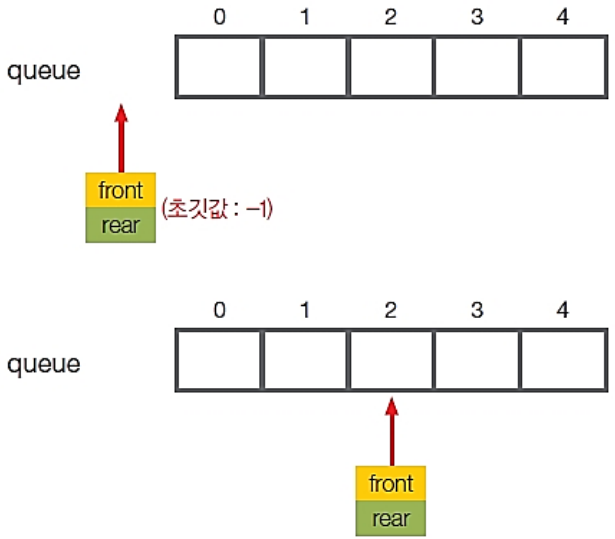
데이터 삽입 과정 실습

3] 데이터 추출 과정

큐가 비었는지 확인하는 함수

- front와 rear의 값이 같다면 큐가 비어 있는 상태
- 두 가지 큐가 비어 있는 상태

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



큐가 비었는지 확인하는 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
```

3] 데이터 추출 과정

큐에서 데이터를 추출하는 함수

```
1  def isQueueEmpty() :
…  # 생략(     앞쪽의     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
```

3] 데이터 추출 과정

큐에서 데이터를 추출하는 함수

```
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.
```

데이터 추출 과정 실습

앞쪽의 소스의 isQueueEmpty() 함수를 없애고,
대신에 deQueue() 함수 안으로 그 기능을 모두 구현하자.
실행 결과는 다음과 같다.

실행 결과

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

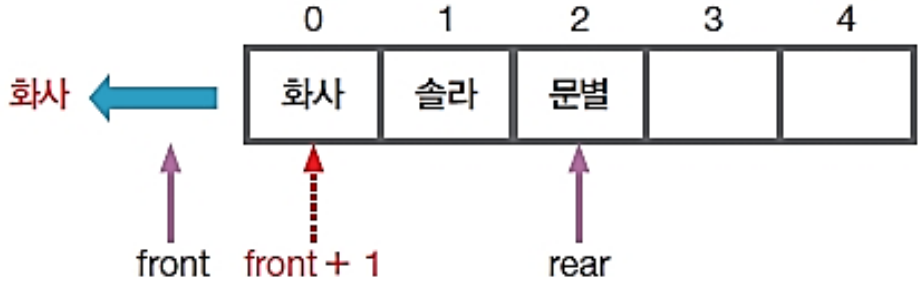
데이터 추출 과정 실습

4] 데이터 확인

peek(픽)

추출될 데이터를 큐에 그대로 두고 확인만 하는 것

- 데이터를 확인하는 peek 작동



4] 데이터 확인

큐에서 front+1 위치의 데이터를 확인하는 함수

```
1  def isQueueEmpty() :
…  # 생략(    앞쪽의    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
```

큐에서 front+1 위치의 데이터를 확인하는 함수

```
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]
```

기능 통합 버전

큐 작동을 위한 통합 코드

```
1  ## 함수 선언 부분 ##
2  def isQueueFull() :                    # 큐가 꽉 찼는지 확인하는 함수
3      global SIZE, queue, front, rear
4      if (rear == SIZE-1) :
5          return True
6      else :
7          return False
8
9  def isQueueEmpty() :                   # 큐가 비었는지 확인하는 함수
10     global SIZE, queue, front, rear
11     if (front == rear) :
12         return True
```

기능 통합 버전

큐 작동을 위한 통합 코드

```
13      else :
14          return False
15
16  def enQueue(data) :
17      global SIZE, queue, front, rear
18      if (isQueueFull()) :
19          print("큐가 꽉 찼습니다.")
20          return
21      rear += 1
22      queue[rear] = data
23
```

큐에 데이터를 삽입하는 함수

기능 통합 버전

큐 작동을 위한 통합 코드

```
24 def deQueue() :
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
```

큐에서 데이터를 추출하는 함수

기능 통합 버전

큐 작동을 위한 통합 코드

```
34 def peek() :
35     global SIZE, queue, front, rear
36     if (isQueueEmpty()) :
37         print("큐가 비었습니다.")
38         return None
39     return queue[front+1]
40
```

큐에서 front+1 위치의 데이터를 확인하는 함수

기능 통합 버전

큐 작동을 위한 통합 코드

```
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()
```

기능 통합 버전

큐 작동을 위한 통합 코드

```
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!")
```

기능 통합 버전

큐 작동을 위한 통합 코드

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
```

기능 통합 버전

큐 작동을 위한 통합 코드

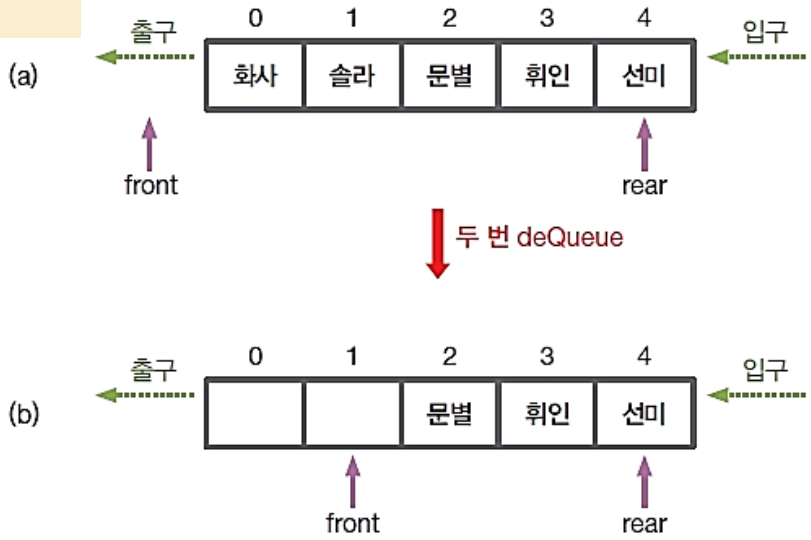
```
큐 상태 : [None, '솔라', '문별', None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 솔라
큐 상태 : [None, None, '문별', None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 문별
큐 상태 : [None, None, None, None, None]
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
큐가 비었습니다.
추출된 데이터 ==> None
큐 상태 : [None,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

5] 큐의 완성

기능 통합 버전 개선

큐가 꽉 찼는지 확인하는 함수의 문제점

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



5] 큐의 완성

기능 통합 버전 개선

큐가 꽉 찼는지 확인하는 함수를 다음과 같이 수정

❶ if (rear값 != 큐크기-1) :
    큐가 꽉 차지 않았음

❷ elif (rear값 == 큐크기-1) and (front == -1) :
    큐가 꽉 찼음

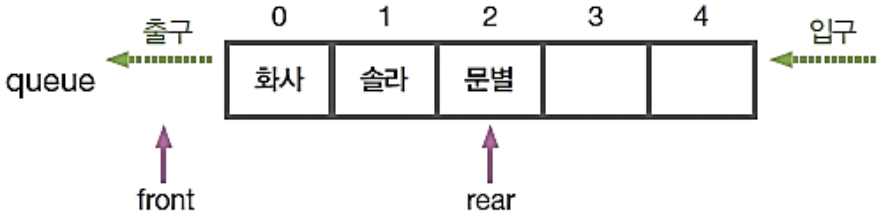
❸ else :
    앞쪽의 (b)와 같은 상태로, 데이터를 앞으로 당기면 큐가 꽉 차지 않음

5] 큐의 완성

기능 통합 버전 개선

1 rear ≠ 큐 크기-1인 경우

- rear 뒤에 여유가 있는 상태

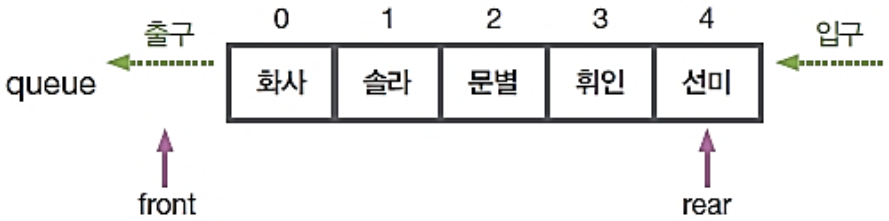


5] 큐의 완성

기능 통합 버전 개선

2 rear = 큐 크기-1, front = -1인 경우

- 큐가 꽉 찬 상태

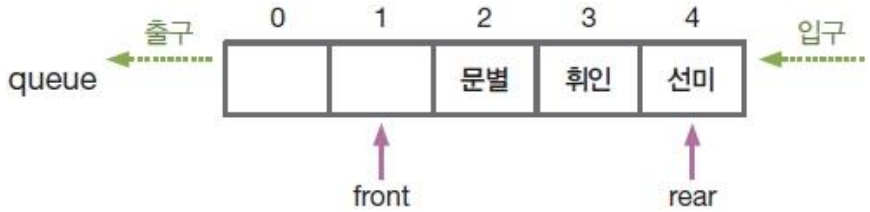


5] 큐의 완성

기능 통합 버전 개선

3 rear = 큐 크기-1, front ≠ -1인 경우

- rear는 끝이지만 앞쪽에 여유가 있는 경우



5] 큐의 완성

기능 통합 버전 개선

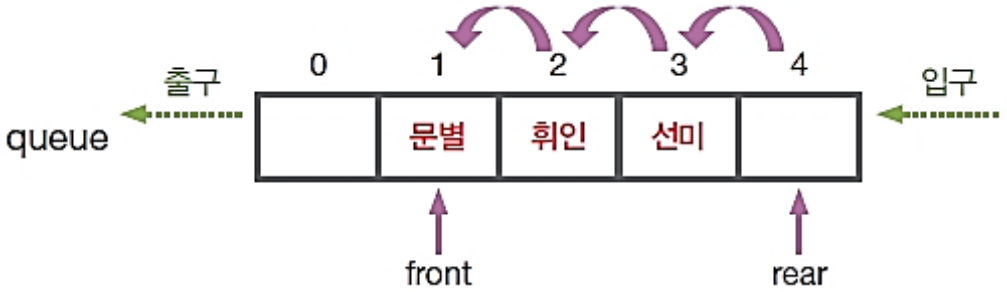
앞의 ❸인 경우는 다음을 수행해야 함

1. front+1 위치부터 마지막 칸까지 왼쪽으로 한 칸씩 이동
2. front 값에서 1을 빼기
3. rear 값에서 1을 빼기
4. 큐가 꽉 차지 않았다는 의미의 False를 반환

기능 통합 버전 개선

앞의 ❸인 경우는 다음을 수행해야 함

1 front+1 위치부터 마지막 칸까지 왼쪽으로 한 칸씩 이동

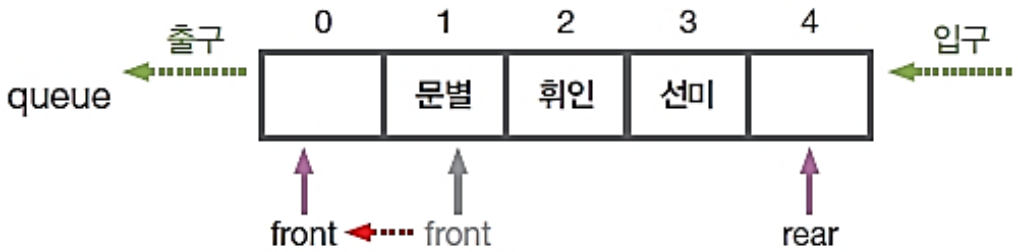


기능 통합 버전 개선

앞의 ❸인 경우는 다음을 수행해야 함

2 front 값에서 1을 빼기

- 즉, front를 왼쪽으로 한 칸 이동



기능 통합 버전 개선

앞의 ❸인 경우는 다음을 수행해야 함

3 rear 값에서 1을 빼기

- 즉, rear를 왼쪽으로 한 칸 이동

5] 큐의 완성

기능 통합 버전 개선

앞의 ❸인 경우는 다음을 수행해야 함

4 큐가 꽉 차지 않았다는 의미의 False를 반환

- 그러면 큐가 꽉 차지 않은 것이므로 마지막 칸에 데이터를 추가

기능 통합 버전 개선

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
```

기능 통합 버전 개선

큐가 꽉 찼는지 확인하는 함수 개선 버전

```
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]
```

개선된 큐 완성

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
```

개선된 큐 완성

큐 작동을 위한 통합 코드 수정

```
12            front -= 1
13            rear -= 1
14            return False
15
   # 생략(기능 통합버전의    9~67행과 동일)
75        print("프로그램 종료!")
```

개선된 큐 완성

큐 작동을 위한 통합 코드 수정

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
```

개선된 큐 완성

큐 작동을 위한 통합 코드 수정

```
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!
```

02 큐의 응용

1) 원형 큐의 개념
2) 원형 큐의 원리
3) 원형 큐의 구현
4) 큐의 응용 실습

1] 원형 큐의 개념

원형 큐

큐의 처음과 끝이 연결된 구조

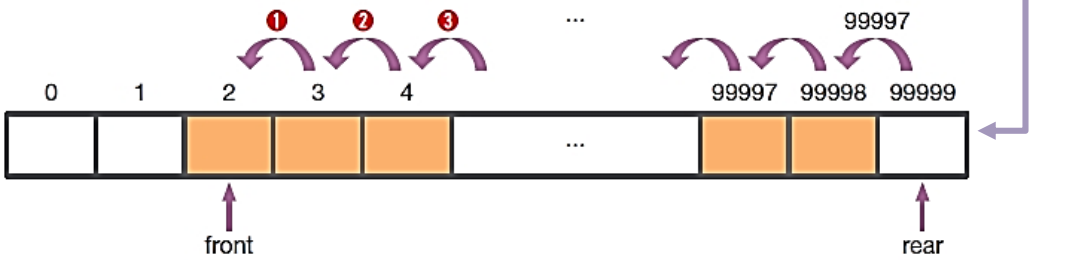
크기가 10만 개인 순차 큐의 앞쪽 일부를 제외하고
데이터가 꽉 찬 상태

앞쪽 일부를 제외하고 데이터가 꽉 찬 순차 큐에서 데이터 삽입

1단계

- 왼쪽으로 이동

순차 큐에서는
1단계에서 오버헤드 발생



앞쪽 일부를 제외하고 데이터가 꽉 찬 순차 큐에서
데이터 삽입

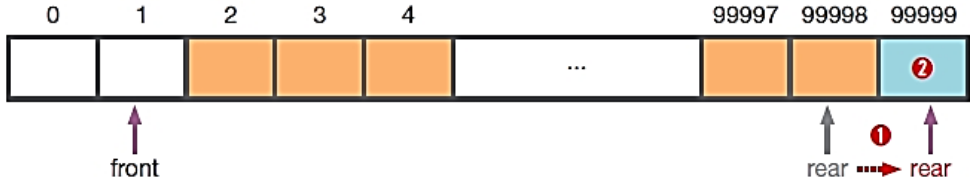
2단계

- front, rear 감소

앞쪽 일부를 제외하고 데이터가 꽉 찬 순차 큐에서
데이터 삽입

3단계

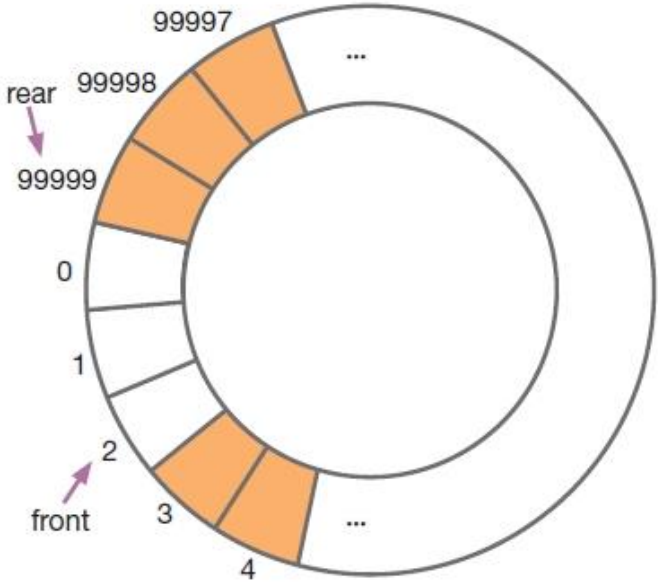
- rear 증가 후 데이터 삽입



1] 원형 큐의 개념

순차 큐를 구부려서 끝을 이은 원형 큐

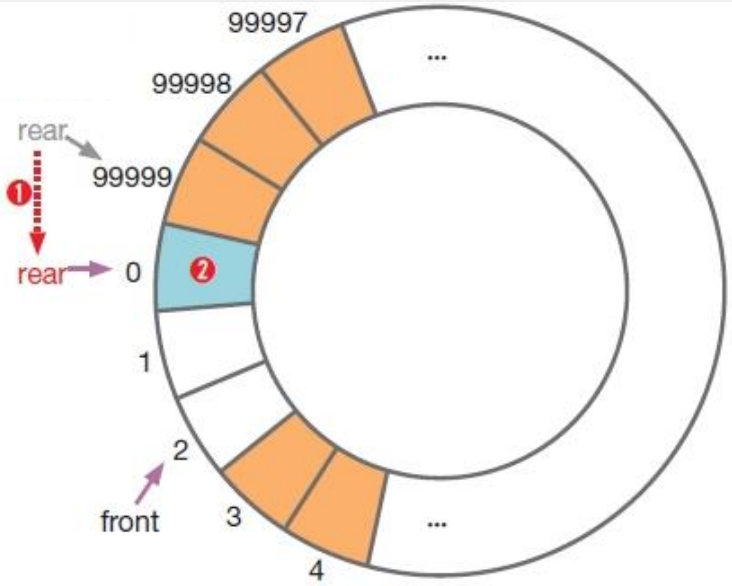
- 오버헤드가 발생하지 않음



앞쪽 일부를 제외하고 데이터가 꽉 찬 원형 큐에서
데이터 삽입

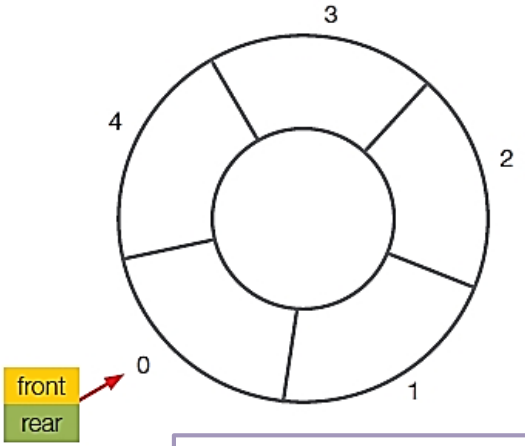
1단계

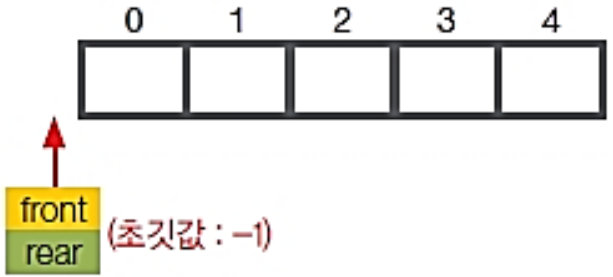
- rear 증가 후 데이터 삽입



원형 큐 초기화

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





순차 큐 초기화

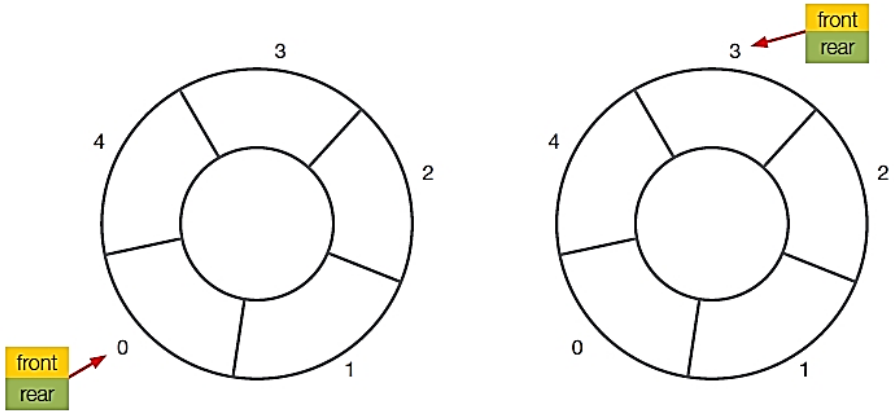
순차큐 = [None, None, None, None, None]
front = rear = -1

원형 큐가 빈 경우와 꽉 찬 경우

원형 큐가 비어 있는 상태

- front 와 rear가 동일하면 비어 있다는 의미

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

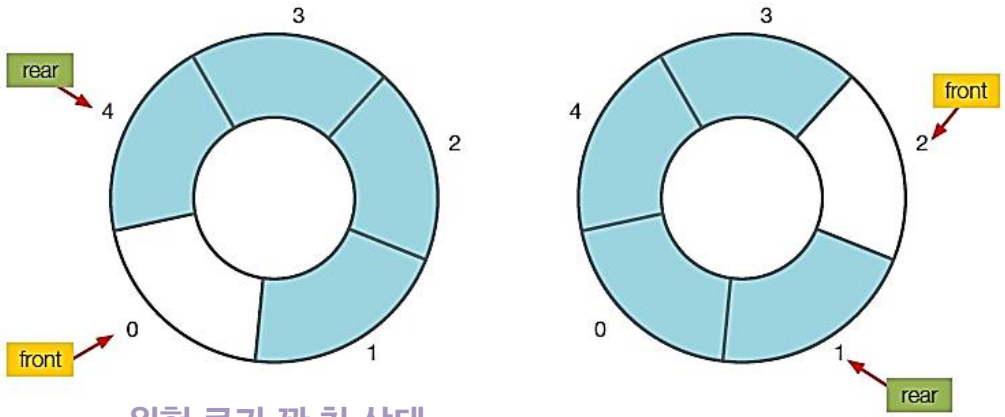


원형 큐가 빈 경우와 꽉 찬 경우

원형 큐가 꽉 찬 상태

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

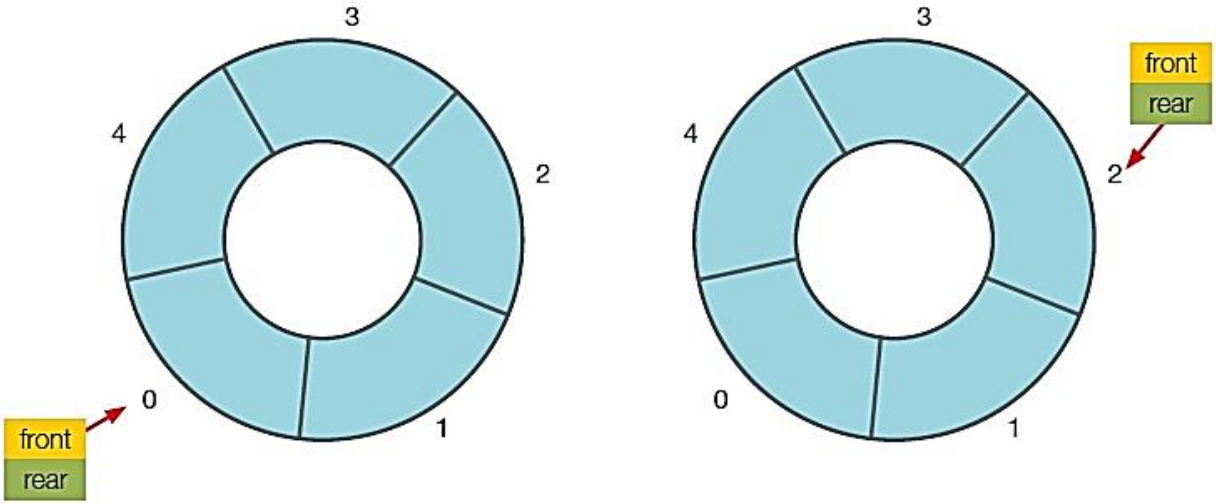
```
if ((rear값+1) % 5 == front값) :
    큐가 꽉 찼음
```



원형 큐가 꽉 찬 상태

원형 큐가 빈 경우와 꽉 찬 경우

원형 큐가 꽉 찼지만 큐가 비어 있다는 의미로 해석(잘못된 예)



원형 큐가 꽉 찼지만 빈 것으로 처리되는 잘못된 예

원형 큐의 데이터 삽입과 추출

원형 큐의 데이터 입력

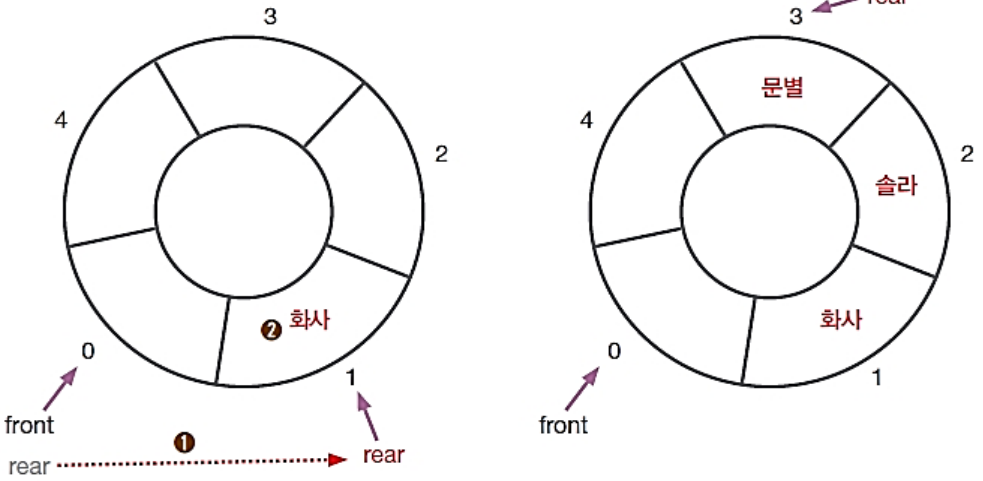
- 원형 큐에서 데이터를 삽입하는 예

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

원형 큐의 데이터 삽입과 추출

원형 큐의 데이터 입력

- 원형 큐에서 데이터를 삽입하는 예



원형 큐의 데이터 삽입과 추출

원형 큐의 데이터 추출

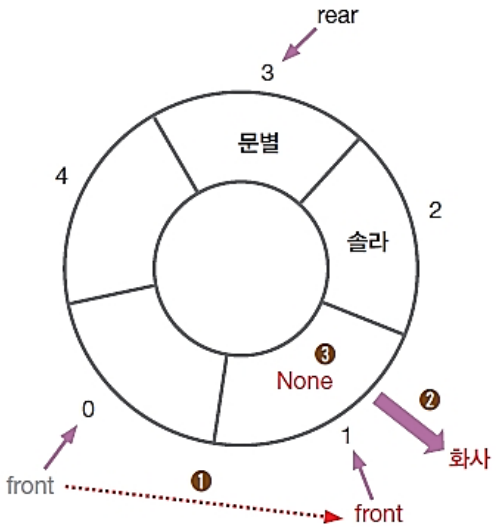
- 원형 큐에서 데이터를 추출하는 예

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

원형 큐의 데이터 삽입과 추출

원형 큐의 데이터 추출

- 원형 큐에서 데이터를 추출하는 예



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

```

원형 큐 작동을 위한 통합 코드

```
16 def enQueue(data) :
17     global SIZE, queue, front, rear
18     if (isQueueFull()) :
19         print("큐가 꽉 찼습니다.")
20         return
21     rear = (rear+1) % SIZE
22     queue[rear] = data
23
24 def deQueue() :
25     global SIZE, queue, front, rear
26     if (isQueueEmpty()) :
27         print("큐가 비었습니다.")
28         return None
29     front = (front+1) % SIZE
```

원형 큐 작동을 위한 통합 코드

```
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)]
```

원형 큐 작동을 위한 통합 코드

```
front = rear = 0

## 메인 코드 부분 ##
if __name__ == "__main__" :
    select = input("삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> ")

    while (select != 'X' and select != 'x') :
        if select == 'I' or select == 'i' :
            data = input("입력할 데이터 ==> ")
            enQueue(data)
            print("큐 상태 : ", queue)
            print("front : ", front, ", rear : ", rear)
```

원형 큐 작동을 위한 통합 코드

```
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
```

원형 큐 작동을 위한 통합 코드

```
69          select = input("삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> ")
70
71      print("프로그램 종료!")
```

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
```

원형 큐 작동을 위한 통합 코드

```
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.
```

원형 큐 작동을 위한 통합 코드

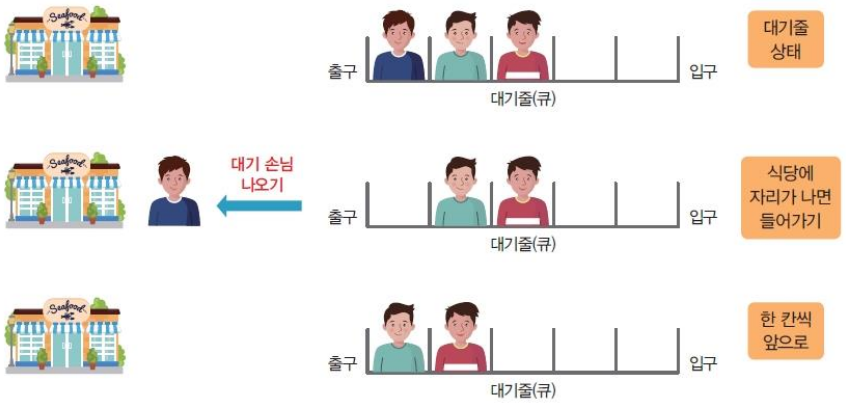
```
큐 상태 : [None, '화사', '솔라', '문별', '휘인']
front : 0 , rear : 4
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> E
추출된 데이터 ==> 화사
큐 상태 : [None, None, '솔라', '문별', '휘인']
front : 1 , rear : 4
삽입(I)/추출(E)/확인(V)/종료(X) 중 하나를 선택 ==> I
입력할 데이터 ==> 재남
큐 상태 : ['재남', None, '솔라', '문별', '휘인']
front : 1 , rear : 0
삽입(I)/추출(E)/확인(V)/종료(X)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

원형 큐 작동을 위한 통합 코드

```
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!
```

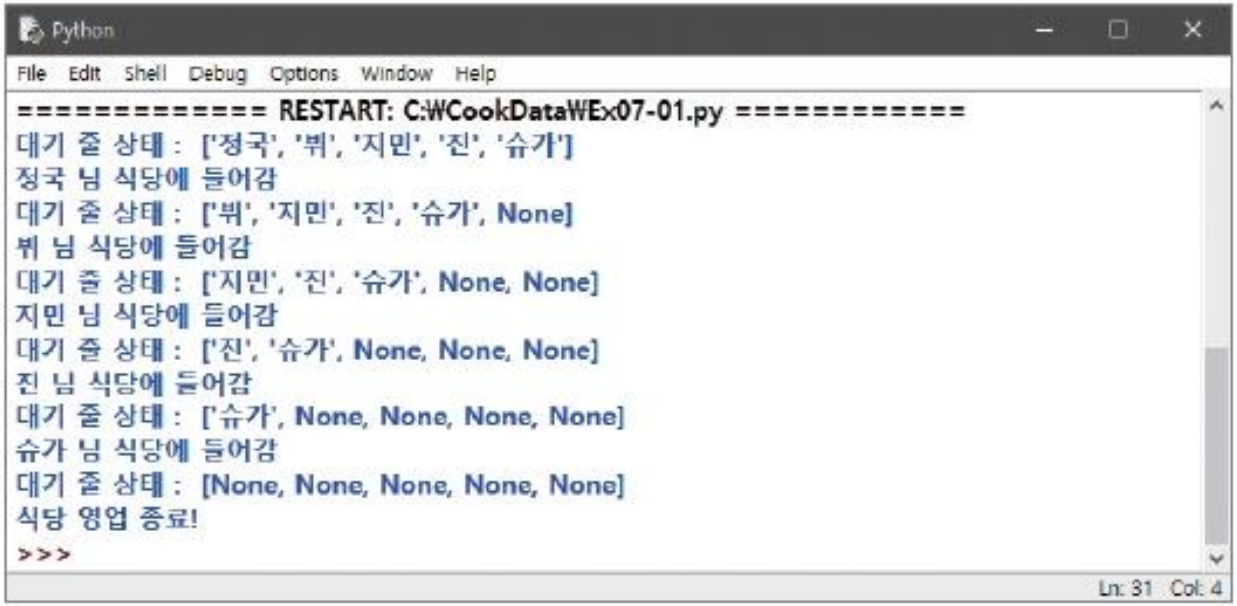
큐의 응용 실습

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.



큐의 응용 실습

실행 결과


Python
File Edit Shell Debug Options Window Help
========== RESTART: C:\CookData\Ex07-01.py ==========
대기 줄 상태 : ['정국', '뷔', '지민', '진', '슈가']
정국 님 식당에 들어감
대기 줄 상태 : ['뷔', '지민', '진', '슈가', None]
뷔 님 식당에 들어감
대기 줄 상태 : ['지민', '진', '슈가', None, None]
지민 님 식당에 들어감
대기 줄 상태 : ['진', '슈가', None, None, None]
진 님 식당에 들어감
대기 줄 상태 : ['슈가', None, None, None, None]
슈가 님 식당에 들어감
대기 줄 상태 : [None, None, None, None, None]
식당 영업 종료!
>>>
Ln: 31 Col: 4

큐의 응용 실습

학습평가

Q1 Q2

Q1

큐가 꽉 찼는지 확인하는 함수이다.
(1)에 적합한 코드는 무엇인가?

```
def isQueueFull():
  global SIZE, queue,
front, rear
  if (      1      ):
    return True
  else:
    return False
```

1. rear != SIZE - 1
2. rear == SIZE -1
3. rear != SIZE
4. rear == SIZE

학습평가

Q1 Q2

Q1

큐가 꽉 찼는지 확인하는 함수이다.
(1)에 적합한 코드는 무엇인가?

```
def isQueueFull():
  global SIZE, queue,
front, rear
  if (      1      ):
    return True
  else:
    return False
```

1. rear != SIZE - 1
2. rear == SIZE -1
3. rear != SIZE
4. rear == SIZE

정 답 2 rear == SIZE-1

해 설 rear 값이 '큐 크기-1'과 같다면 큐가 꽉 찬 상태입니다.

학습평가

Q2

큐가 비었는지 확인하는 함수이다.
(1)에 적합한 코드는 무엇인가?

```
def isQueueEmpty():
  global SIZE, queue,
front, rear
  if (     1     ):
    return True
  else:
    return False
```

1. front != rear - 1
2. front == rear -1
3. front != rear
4. front == rear

학습평가

Q2

큐가 비었는지 확인하는 함수이다.
(1)에 적합한 코드는 무엇인가?

```
def isQueueEmpty():
  global SIZE, queue,
front, rear
  if (     1     ):
    return True
  else:
    return False
```

1. front != rear - 1
2. front == rear -1
3. front != rear
4. ✓ front == rear

정 답 4 front == rear

해 설 front와 rear의 값이 같다면 큐가 비어 있는 상태입니다.

큐의 일반 구현

큐 초기화

- SIZE 값만 변경하면 원하는 크기의 빈 큐 생성(큐 초기화)

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

큐가 꽉 찾는지 확인하는 방법

- rear 값이 '큐 크기-1'과 같다면 큐가 꽉 찬 상태

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

큐의 일반 구현

큐가 비었는지 확인하는 방법

- front와 rear의 값이 같다면 큐가 비어 있는 상태

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

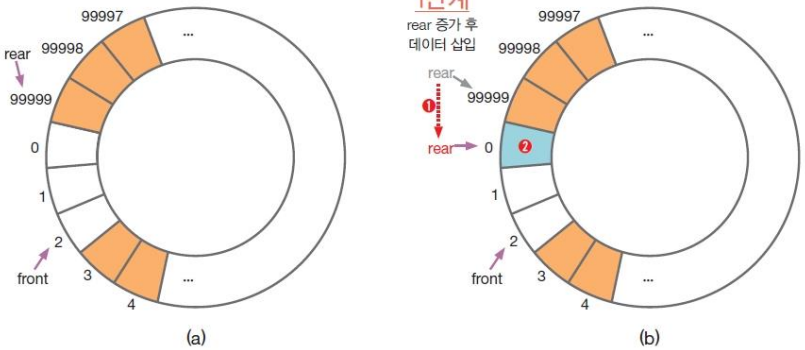
데이터 확인

- 추출될 데이터를 큐에 그대로 두고 확인만 하는 것 : peek(픽)

큐의 응용

원형 큐

- 큐의 처음과 끝이 연결된 구조
- 순차 큐를 구부려서 끝을 이은 원형 큐
- 앞쪽 일부를 제외하고 데이터가 꽉 찬 원형 큐에서 데이터 삽입



큐의 응용

원형 큐가 빈 경우와 꽉 찬 경우

- front 와 rear가 동일하면 비어 있다는 의미

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

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

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