16차시	총10문제		연습: ☐	과제 : ☑	평가 : ☐

1. 스택 자료구조의 주요 특징은 무엇인가요?

a) FIFO (First In First Out)

b) LIFO (Last In First Out)

c) 정렬된 데이터 관리

d) 랜덤 접근 가능

2. 파이썬 리스트를 스택으로 사용할 때 요소를 제거하는 메서드는 무엇인가요?

a) pop()

b) remove()

c) del

d) discard()

3. 큐 자료구조의 주요 특징은 무엇인가요?

a) LIFO (Last In First Out)

b) FIFO (First In First Out)

c) 정렬된 데이터 관리

d) 랜덤 접근 가능

4. 덱(Deque)의 주요 특징은 무엇인가요?

a) 한쪽에서만 삽입과 삭제가 가능하다.

b) 양쪽 끝에서 삽입과 삭제가 가능하다.

c) 데이터를 자동으로 정렬한다.

d) LIFO만 지원한다.

5. 다음 자료 구조의 활용 예시 중 구현 방식이 다른 하나는 무엇인가요?

a) 함수 호출 스택

b) 프로세스 관리 (예: 작업 대기열)

c) 괄호 검사와 같은 문자열 처리

d) 웹 브라우저 방문기록 (뒤로가기)

6. 리스트를 이용하여 기본적인 스택(stack)을 구현하세요. (다음 내용을 고려할 것.)

push(x): 정수 x를 스택에 삽입

pop(): 스택에서 가장 위의 값을 제거하고 반환. 만약 스택이 비어 있다면 -1을 반환

top(): 스택의 가장 위에 있는 값을 반환. 만약 스택이 비어 있다면 -1을 반환

is_empty(): 스택이 비어 있으면 True, 아니면 False를 반환

```python
class Stack:
    def __init__(self):
        self.stack = []
    def push(self, x):
        self.stack.append(x)
    def pop(self):
        if self.is_empty():
            return -1
        return self.stack.pop()
    def top(self):
        if self.is_empty():
            return -1
        return self.stack[-1]
    def is_empty(self):
        return len(self.stack) == 0


s = Stack()
print(s.pop())
s.push(10)
s.push(20)
print(s.top())
print(s.pop())
print(s.pop())
print(s.pop())
print(s.is_empty())
```

```
PS C:\rokey> & C:/Users/hoone/AppData/L
grams/Python/Python313/python.exe c:/ro
.py
-1
20
20
10
-1
True
PS C:\rokey>
```

7. 후위 표기법(Postfix Notation, Reverse Polish Notation)으로 주어진 수식을 계산하는 프로그램을 작성하세요.

후위 표기법에서는 연산자가 피연산자 뒤에 위치합니다. 예를 들어, 3 4 +는 3 + 4를 의미하며, 결과는 7입니다.

연산자는 +, -, *, /만 고려합니다. (복수 연산자도 처리 가능해야 함.)

sol4.py > ...

```python
def evaluate_postfix(expression):
    stack = []
    tokens = expression.split()

    for token in tokens:
        if token.isdigit():
            stack.append(int(token))
        else:
            b = stack.pop()
            a = stack.pop()
            if token == '+':
                stack.append(a + b)
            elif token == '-':
                stack.append(a - b)
            elif token == '*':
                stack.append(a * b)
            elif token == '/':
                stack.append(int(a / b))

    return stack.pop()

print(evaluate_postfix("3 4 +"))
print(evaluate_postfix("5 1 2 + 4 * + 3 -"))
print(evaluate_postfix("10 2 8 * + 3 -"))
```

터미널

```
PS C:\rokey> & C:/Users/hoone/AppData/Lo
grams/Python/Python313/python.exe c:/roke
.py
7
14
23
PS C:\rokey>
```

8. 리스트를 사용하여 기본적인 큐(queue)를 구현하세요. (다음 내용을 고려할 것.)

enqueue(x): 정수 x를 큐의 끝에 추가

dequeue(): 큐의 앞에서 값을 제거하고 반환. 만약 큐가 비어 있다면 -1을 반환

front(): 큐의 앞에 있는 값을 반환. 만약 큐가 비어 있다면 -1을 반환

is_empty(): 큐가 비어 있으면 True, 아니면 False를 반환

```python
class Queue:
    def __init__(self):
        self.queue = []

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

    def dequeue(self):
        if self.is_empty():
            return -1
        return self.queue.pop(0)

    def front(self):
        if self.is_empty():
            return -1
        return self.queue[0]

    def is_empty(self):
        return len(self.queue) == 0

q = Queue()
print(q.dequeue())
q.enqueue(10)
q.enqueue(20)
print(q.front())
print(q.dequeue())
print(q.dequeue())
print(q.dequeue())
print(q.is_empty())
```

```
PS C:\rokey> & C:/Users/hoone/AppData/Loc
grams/Python/Python313/python.exe c:/roke
.py
-1
10
10
20
-1
True
PS C:\rokey>
```

9. 고정된 크기의 원형 큐를 구현하세요. (다음 내용을 고려할 것.)

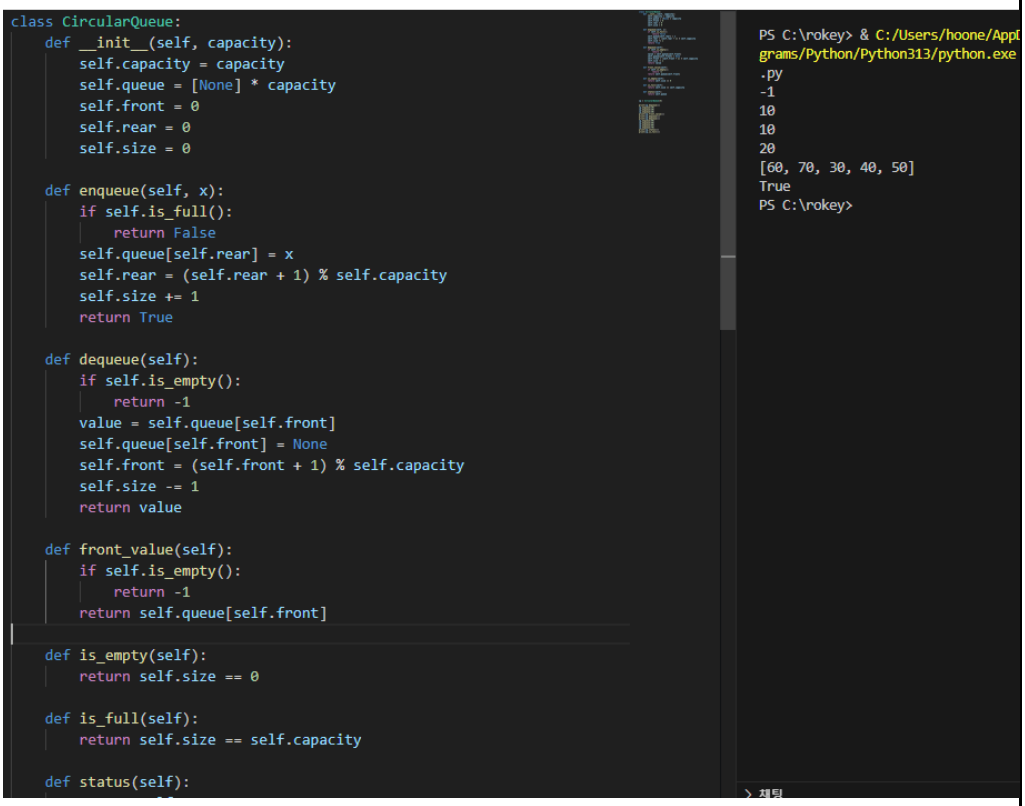
enqueue(x): 정수 x를 큐의 끝에 추가. (큐가 가득 차면 추가하지 않음)

dequeue(): 큐의 앞에서 값을 제거하고 반환. (큐가 비어 있다면 -1 반환)

front(): 큐의 앞에 있는 값을 반환. (비어 있다면 -1 반환)

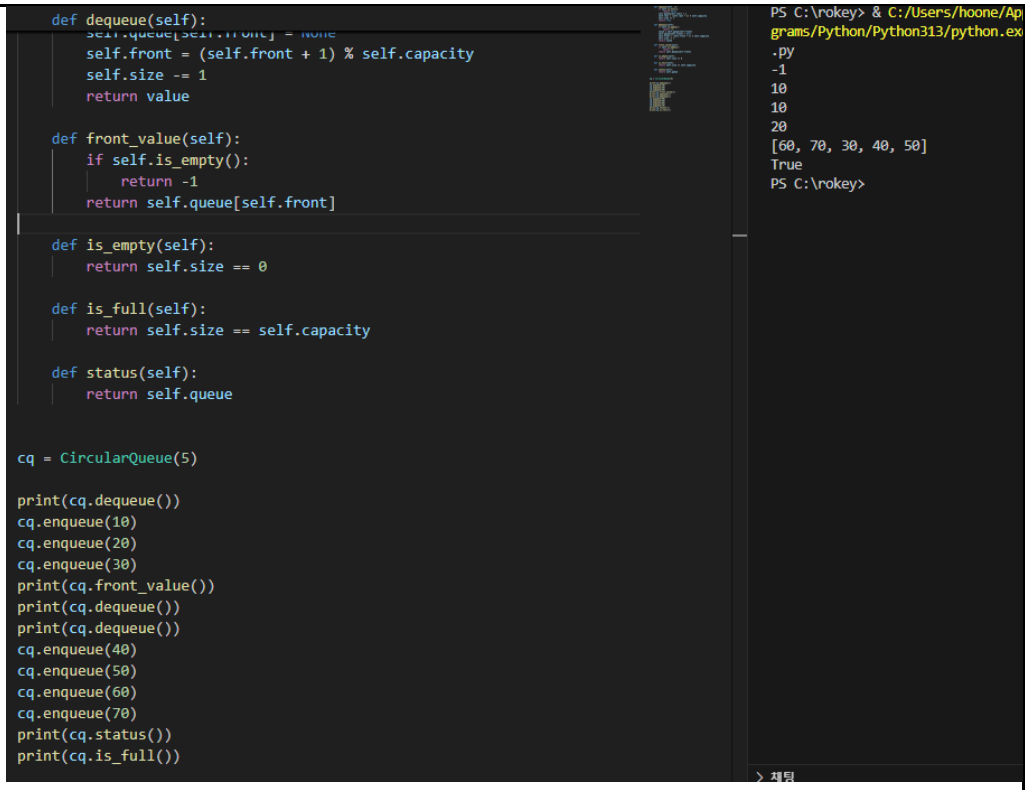
is_empty(): 큐가 비어 있으면 True, 아니면 False 반환

is_full(): 큐가 가득 차 있으면 True, 아니면 False 반환



```python
class CircularQueue:
    def __init__(self, capacity):
        self.capacity = capacity
        self.queue = [None] * capacity
        self.front = 0
        self.rear = 0
        self.size = 0

    def enqueue(self, x):
        if self.is_full():
            return False
        self.queue[self.rear] = x
        self.rear = (self.rear + 1) % self.capacity
        self.size += 1
        return True

    def dequeue(self):
        if self.is_empty():
            return -1
        value = self.queue[self.front]
        self.queue[self.front] = None
        self.front = (self.front + 1) % self.capacity
        self.size -= 1
        return value

    def front_value(self):
        if self.is_empty():
            return -1
        return self.queue[self.front]

    def is_empty(self):
        return self.size == 0

    def is_full(self):
        return self.size == self.capacity

    def status(self):
```

```
PS C:\rokey> & C:/Users/hoone/AppD
grams/Python/Python313/python.exe
.py
-1
10
10
20
[60, 70, 30, 40, 50]
True
PS C:\rokey>
```



```
    def dequeue(self):
        self.queue[self.front] = None
        self.front = (self.front + 1) % self.capacity
        self.size -= 1
        return value

    def front_value(self):
        if self.is_empty():
            return -1
        return self.queue[self.front]

    def is_empty(self):
        return self.size == 0

    def is_full(self):
        return self.size == self.capacity

    def status(self):
        return self.queue


cq = CircularQueue(5)

print(cq.dequeue())
cq.enqueue(10)
cq.enqueue(20)
cq.enqueue(30)
print(cq.front_value())
print(cq.dequeue())
print(cq.dequeue())
cq.enqueue(40)
cq.enqueue(50)
cq.enqueue(60)
cq.enqueue(70)
print(cq.status())
print(cq.is_full())
```

```
PS C:\rokey> & C:/Users/hoone/Ap
grams/Python/Python313/python.ex
.py
-1
10
10
20
[60, 70, 30, 40, 50]
True
PS C:\rokey>
```

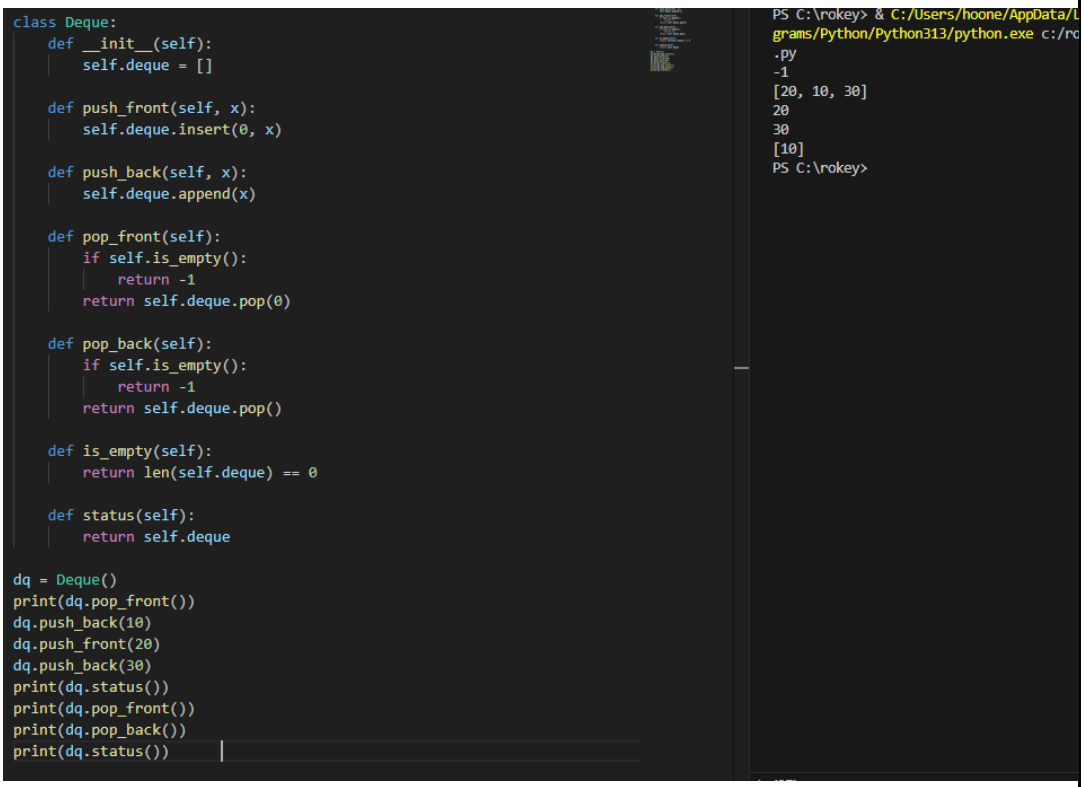
10. 리스트를 이용하여 덱(deque, 양방향 큐)을 구현하세요. (다음 내용을 고려할 것.)

push_front(x): 정수 x를 덱의 앞에 추가

push_back(x): 정수 x를 덱의 뒤에 추가

pop_front(): 덱의 앞에서 값을 제거하고 반환 (비어 있다면 -1)

pop_back(): 덱의 뒤에서 값을 제거하고 반환 (비어 있다면 -1)



```
class Deque:
    def __init__(self):
        self.deque = []

    def push_front(self, x):
        self.deque.insert(0, x)

    def push_back(self, x):
        self.deque.append(x)

    def pop_front(self):
        if self.is_empty():
            return -1
        return self.deque.pop(0)

    def pop_back(self):
        if self.is_empty():
            return -1
        return self.deque.pop()

    def is_empty(self):
        return len(self.deque) == 0

    def status(self):
        return self.deque

dq = Deque()
print(dq.pop_front())
dq.push_back(10)
dq.push_front(20)
dq.push_back(30)
print(dq.status())
print(dq.pop_front())
print(dq.pop_back())
print(dq.status())
```

```
PS C:\rokey> & C:/Users/hoone/AppData/L
grams/Python/Python313/python.exe c:/rc
.py
-1
[20, 10, 30]
20
30
[10]
PS C:\rokey>
```