Clickseo.com

자료구조 및 알고리즘

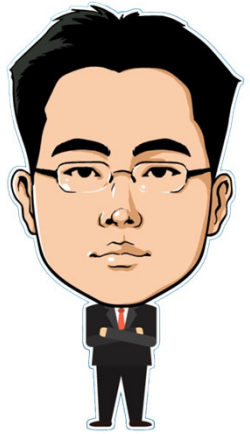
보고서 #03

리스트

Seo, Doo-Ok

Clickseo.com

clickseo@gmail.com



보고서 #03

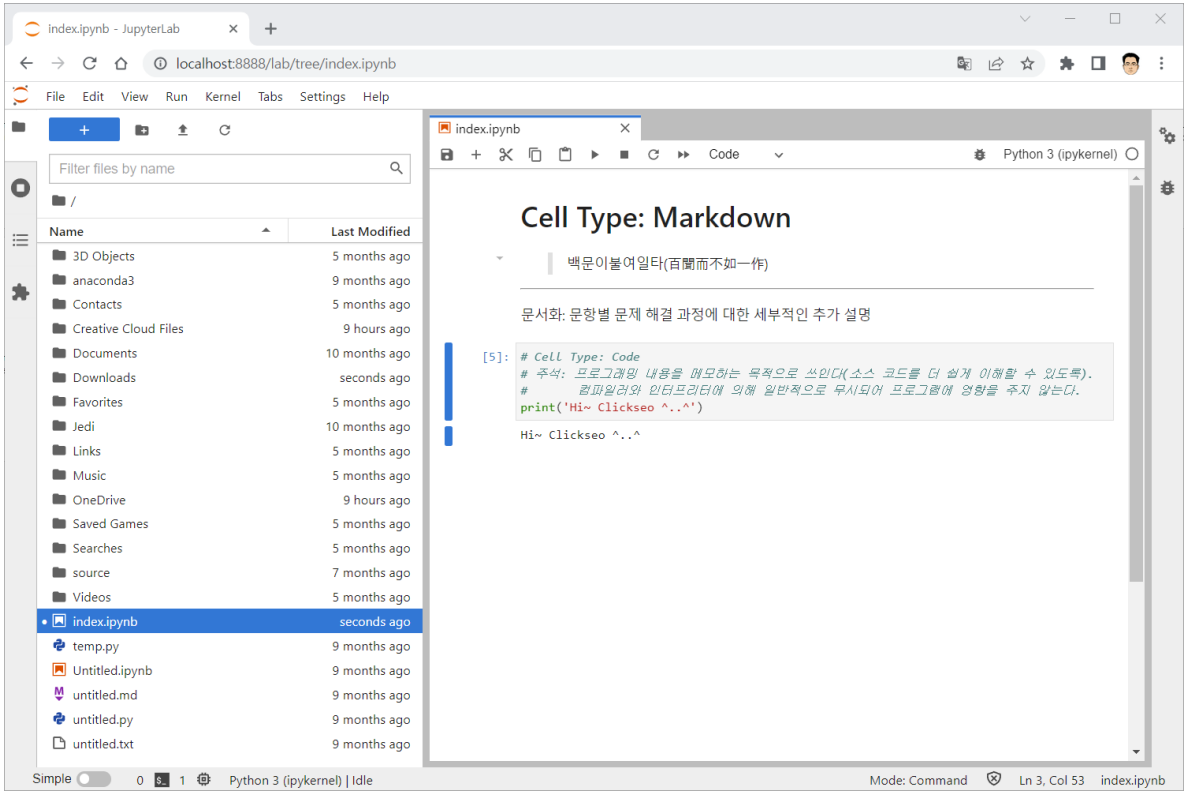
- **보고서 세부 내용**
  1. **데이터 처리:** 단순 연결 리스트
  2. **데이터 처리:** 이중 연결 리스트
  3. **스택 구현:** 단순 연결 리스트
  4. **큐 구현:** 단순 연결 리스트

- **제출방법: (eCampus) 과제 제출에 파일 업로드**
  - JupyterLab(Notebook) 또는 Google Colab: .ipynb 파일 제출
    - ✓ (Code Cell) 프로그램 소스 코드 및 주석
    - ✓ (Markdown 또는 Text Cell) 각 문항별 문제 해결 과정에 대한 세부적인 추가 설명(문서화)
  - 보고서 분량은 별도 제한 없음.

- **제출 마감: (8주차) 2025년 10월 26일(일) 23:59**

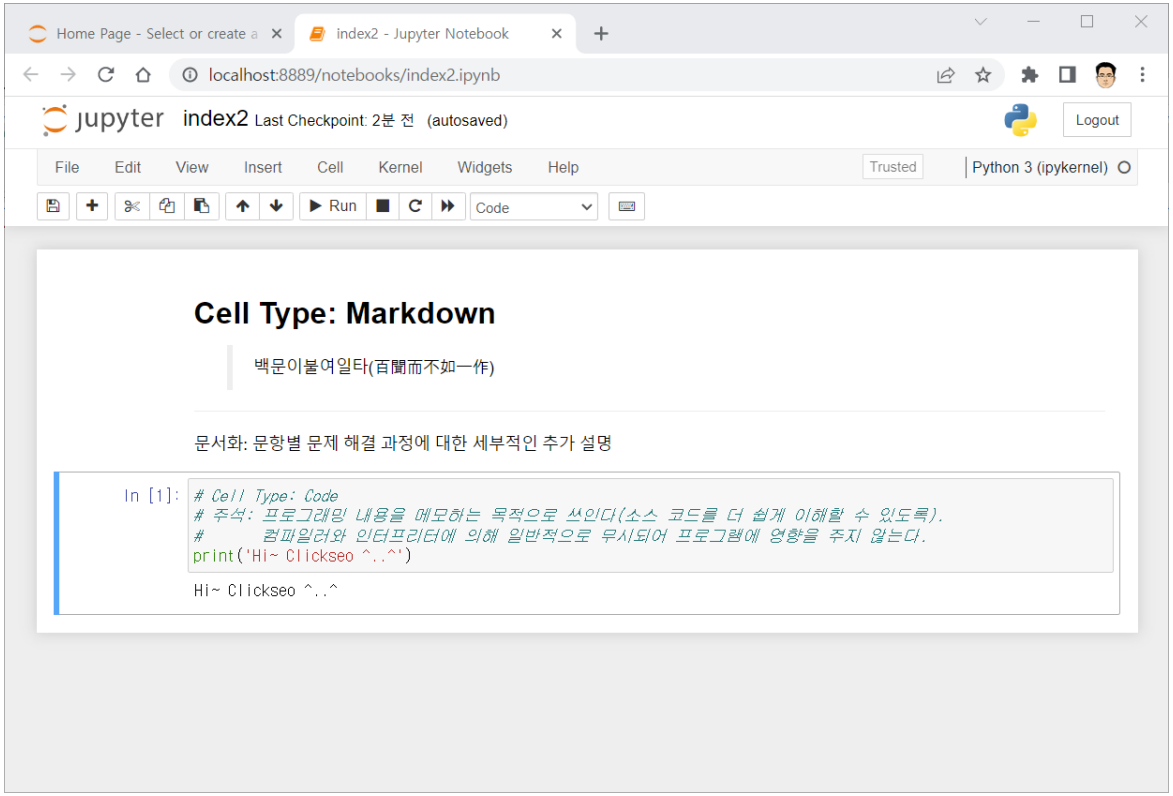
Project Jupyter (1/2)

- **JupyterLab**
  - **Cell Type:** Code(소스 코드 및 주석), Markdown(추가 설명)
    - **File > Download:** Notebook(.ipynb) 파일



Project Jupyter (2/2)

- **Jupyter Notebook**
  - **Cell Type:** Code(소스 코드 및 주석), Markdown(추가 설명)
    - **File > Download as > Notebook(.ipynb)**

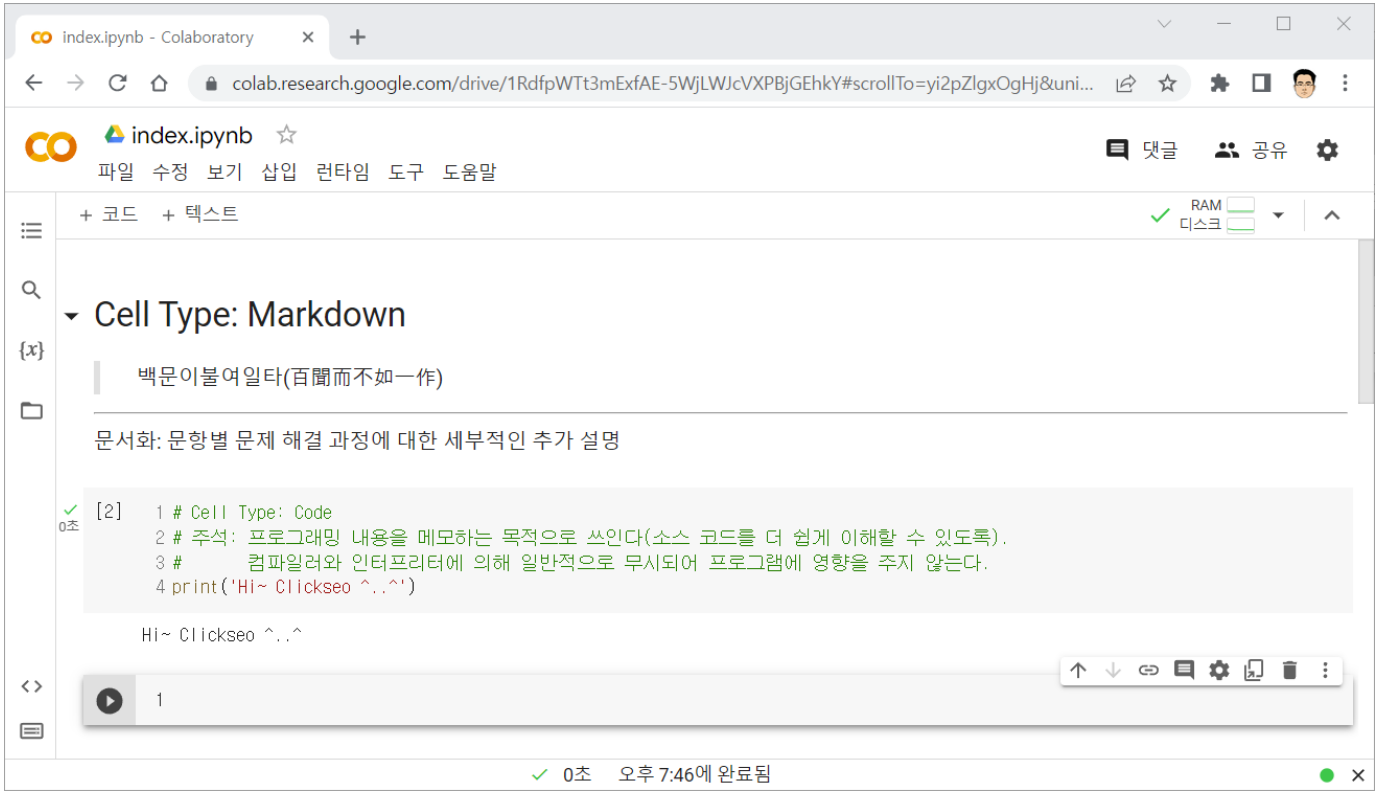


Google Colab

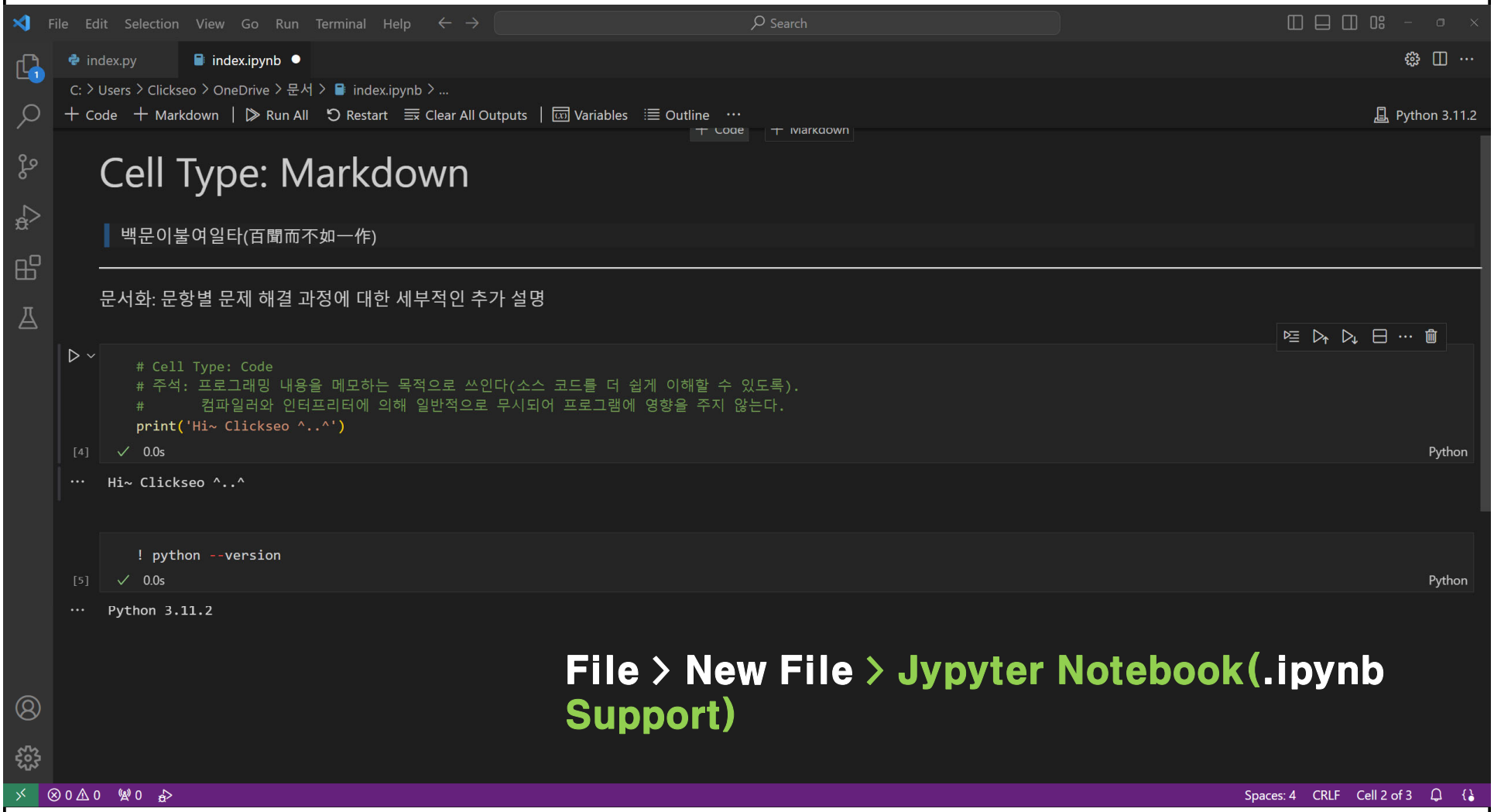
Google Colab

Cell Type: Code(소스 코드 및 주석 / 실행 결과), Text(추가 설명)

- **파일 > 다운로드 > .ipynb 다운로드**



Visual Studio Code



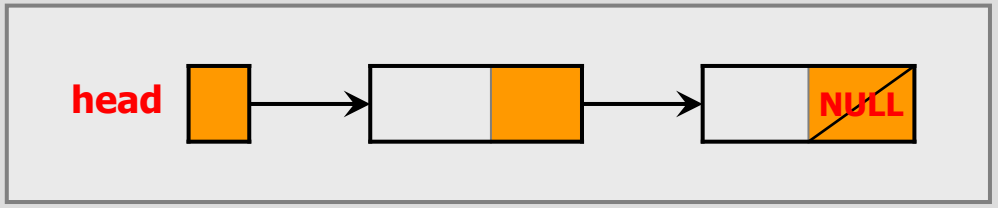
자료구조 및 알고리즘

연습문제: 리스트

연습문제 #01

- **데이터 처리:** 단순 연결 리스트
  - 임의의 정수 값을 입력 받아 출력하는 프로그램을 작성하세요.
    - 단순 연결 리스트를 클래스로 설계하여 작성하세요.
      - 단, 원소 삽입은 항상 마지막 노드로 삽입된다고 가정한다.
      - 또한 전체 원소에 대하여 삭제(메모리 반납)후 프로그램을 종료한다.
  - 프로그램 작성 시 클래스는 다음과 같다.

```
// 노드(SNode): data, link
class SNode :
    def __init__(self, data):
        self.__data = data
        self.__link = None

// 단순 연결 리스트: SLinkedList
class SLinkedList:
    def __init__(self):
        self.__head = None  # 첫 번째 노드
        self.__tail = None  # 맨 마지막 노드
        self.__count = 0    # 노드의 총 개수
```



  - 프로그램 실행 결과는 다음과 같다.

```
IDLE Shell 3.11.2
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 202
Type "help", "copyright", "credits" or "license
>>>
= RESTART: C:₩Users₩Clickseo₩OneDrive₩Clickseo₩
임의의 정수 입력(종료: 0): 1
임의의 정수 입력(종료: 0): 2
임의의 정수 입력(종료: 0): 3
임의의 정수 입력(종료: 0): 4
임의의 정수 입력(종료: 0): 5
임의의 정수 입력(종료: 0): 0

### 입력된 데이터 ###
1 ->> 2 ->> 3 ->> 4 ->> 5 ->>  NULL

노드의 총 개수: 5
첫번째 노드의 데이터: 1
마지막 노드의 데이터: 5
>>>
```

연습문제 #01: 클래스 설계

- **데이터 처리:** 단순 연결 리스트

```
// 노드: SNode(data, link)
class SNode:
    def __init__(self, data):
        self.__data = data
        self.__link = None
    def getData(self):          return self.__data
    def getLink(self):          return self.__link
    def setData(self, data):    self.__data = data
    def setLink(self, link):    self.__link = link

# 단순 연결 리스트: SLinkedList
class SLinkedList:
    def __init__(self):                     # 생성자
        self.__head = None                  # 첫 번째 노드
        # self.__tail = None                # 맨 마지막 노드
        # self.__count = 0                  # 노드의 총 개수
    def __del__(self):                      # 소멸자: 전체 노드 삭제
    def isEmpty(self) -> bool:              # 빈 리스트 여부 판단
    def countNode(self) -> int:             # 탐색: 노드의 총 개수(count)
    def frontNode(self) -> SNode:           # 탐색: 첫 번째 노드
    def rearNode(self) -> SNode:            # 탐색: 맨 마지막 노드
    def addRear(self, num) ) -> None:       # 삽입: 맨 마지막 노드
    def removeFront(self) -> None:          # 삭제: 첫 번째 노드
    def printLinkedList(self) ) -> None:    # 리스트의 전체 노드 출력
```

연습문제 #02

- **데이터 처리: 이중 연결 리스트**
  - 임의의 정수 값을 입력 받아 출력하는 프로그램을 작성하세요.
    - 단, 이중 연결 리스트를 이용하여 작성하세요.
      - 데이터는 항상 마지막 노드로 삽입된다고 가정한다.
      - 또한 전체 원소에 대하여 삭제(메모리 반납)후 프로그램을 종료한다.
  - 프로그램 작성 시 클래스 설계는 다음과 같다.

```
// 노드: DNode(data, Llink, Rlink)
class DNode:
    def __init__(self, data):
        self.__data = data
        self.__Llink = None
        self.__Rlink = None

// 이중 연결 리스트: DLinkedList
class DLinkedList:
    def __init__(self):                 # 생성자
        self.__head = None              # 첫 번째 노드
        self.__tail = None              # 맨 마지막 노드
        self.__count = 0                # 노드의 총 개수
```

```
IDLE Shell 3.11.2
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 202
Type "help", "copyright", "credits" or "license
>>>
= RESTART: C:₩Users₩Clickseo₩OneDrive₩Clickseo₩
임의의 정수 입력(종료: 0): 1
임의의 정수 입력(종료: 0): 2
임의의 정수 입력(종료: 0): 3
임의의 정수 입력(종료: 0): 4
임의의 정수 입력(종료: 0): 5
임의의 정수 입력(종료: 0): 0

### 입력된 데이터(순방향) ###
1 ->> 2 ->> 3 ->> 4 ->> 5 ->>  None


### 입력된 데이터(역방향) ###
5 ->> 4 ->> 3 ->> 2 ->> 1 ->>  None

노드의 총 개수: 5
첫번째 노드의 데이터: 1
마지막 노드의 데이터: 5

>>>
```

  - 프로그램 실행 결과는 다음과 같다.

연습문제 #02: 클래스 설계

- **데이터 처리:** 이중 연결 리스트

```
# 노드: Dnode(data, link, Rlink)
class DNode:
    def __init__(self, data):
        self.__data = data
        self.__Llink = None
        self.__Rlink = None
    def getData(self):          return self.__data
    def getLlink(self):         return self.__Llink
    def getRlink(self):         return self.__Rlink
    def setData(self, data):    self.__data = data
    def setLlink(self, Llink):  self.__Llink = Llink
    def setRlink(self, Rlink):  self.__Rlink = Rlink

# 이중 연결 리스트: DLinkedList
class DLinkedList:
    def __init__(self):                     # 생성자
        self.__head = None                  # 첫 번째 노드
        self.__tail = None                  # 맨 마지막 노드
        self.__count = 0                    # 노드의 총 개수
    def __del__(self):                      # 소멸자: 전체 노드 삭제
    def isEmpty(self) -> bool:              # 빈 리스트 여부 판단
    def countNode(self) -> int:             # 탐색: 노드의 총 개수
    def frontNode(self) -> DNode:           # 탐색: 첫 번째 노드(head)
    def rearNode(self) -> DNode:            # 탐색: 맨 마지막 노드
    def addRear(self, num) -> None:         # 삽입: 맨 마지막 노드
    def removeFront(self) -> None:          # 삭제: 첫 번째 노드(head)
    def printLinkedList(self) -> None:      # 출력(순방향): 리스트의 전체 원소(노드) 출력
    def printRevLinkedList(self) -> None:   # 출력(역방향): 리스트의 전체 원소(노드) 출력
```

연습문제 #03

- **스택 구현:** 단순 연결 리스트
  - 연결 자료구조를 이용하여 스택 구조를 구현하세요.
  - 프로그램 작성 시 클래스 설계는 다음과 같다.

```
# LinkedStack class: SNode, top, count
class LinkedStack:
    class SNode:
        def __init__(self, data, link=None):
            self.data = data
            self.link = link

    # 빈 스택 생성
    def __init__(self):
        self.__top = None
        self.__count = 0

    def __del__(self):                  # 스택 삭제: 모든 노드 삭제
    def empty(self) -> bool:            # 빈 스택 여부
    def size(self) -> int:              # 스택의 원소 개수
    def push(self, data) -> None:       # 데이터 삽입
    def pop(self) -> None:              # 데이터 삭제
    def top(self):              # 스택에서 맨 위의 데이터 반환
    def printStack(self) -> None:       # 스택의 전체 데이터 출력
```

  - 프로그램 실행 결과는 다음과 같다.

```
### 스택 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 3

      ##### 입력된 데이터 #####

STACK [ 5 4 3 2 1 ]
```

연습문제 #03: 실행 결과

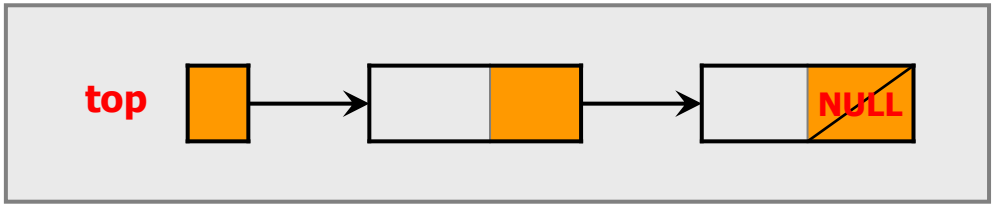
- **스택 구현:** 단순 연결 리스트
  - 프로그램 실행 결과는 다음과 같다.

```
 ### 스택 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 3

        ##### 입력된 데이터 #####

 STACK [ 5 4 3 2 1 ]
```

```
 ### 스택 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 2

삭제 된 데이터: 5
```

```
*IDLE Shell 3.11.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>>
================== RESTART: C:WUsersWclickW

 ### 스택 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 1
삽입 할 데이터 입력(종료: 0): 1
삽입 할 데이터 입력(종료: 0): 2
삽입 할 데이터 입력(종료: 0): 3
삽입 할 데이터 입력(종료: 0): 4
삽입 할 데이터 입력(종료: 0): 5
삽입 할 데이터 입력(종료: 0): 0
```

```
C:₩WINDOWS₩system3
계속하려면 아무 키나 누르십시오 . . .
```



연습문제 #04

- **큐 구현:** 단순 연결 리스트
  - 연결 자료구조를 이용하여 스택 구조를 구현하세요.
  - 프로그램 작성 시 클래스 설계는 다음과 같다.

```
# 클래스 설계: LinkedQueue
class LinkedQueue:
    class SNode:
        def __init__(self, data, link=None):
            self.data = data
            self.link = link

    # 빈 큐 생성
    def __init__(self):
        self._front = None
        self._rear = None
        self._count = 0
```

```
def __del__(self):                      # 큐 삭제
    def empty(self) -> bool:            # 빈 큐 여부
    def size(self) -> int:              # 큐의 원소 개수
    def push(self, data) -> None:       # 데이터 삽입
    def pop(self):                      # 데이터 삭제
    def front(self):          # 큐에서 첫 번째 데이터 반환
    def back(self):           # 큐에서 맨 마지막 데이터 반환
    def printQueue(self):     # 큐의 전체 데이터 출력
```

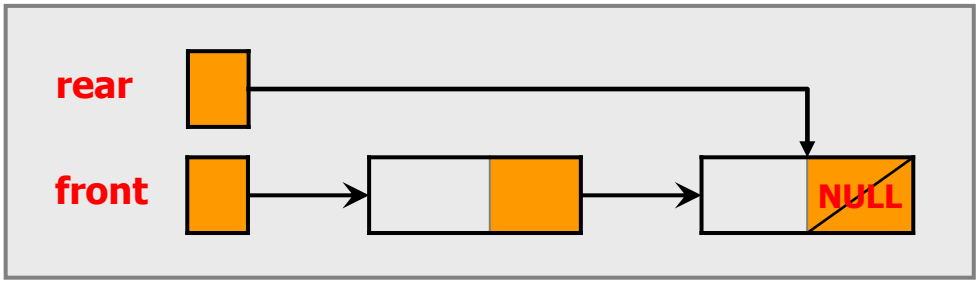
  - 프로그램 실행 결과는 다음과 같다.

```
 ### 큐 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 3

        ##### 입력된 데이터 #####

 QUEUE [ 1 2 3 4 5 ]
```

연습문제 #04: 실행 결과

- **큐 구현:** 단순 연결 리스트
  - 프로그램 실행 결과는 다음과 같다.

```
 ### 큐 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 3

        ##### 입력된 데이터 #####

 QUEUE [ 1 2 3 4 5 ]
```



IDLE Shell 3.11.2

File Edit Shell Debug Options Window Help

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "
>>>
================ RESTART: C:₩Users₩Click

 ### 큐 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 1
삽입 할 데이터 입력(종료: 0): 1
삽입 할 데이터 입력(종료: 0): 2
삽입 할 데이터 입력(종료: 0): 3
삽입 할 데이터 입력(종료: 0): 4
삽입 할 데이터 입력(종료: 0): 5
삽입 할 데이터 입력(종료: 0): 0
```

```
C:₩WINDOWS₩system3
계속하려면 아무 키나 누르십시오 . . .
```

```
 ### 큐 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 2

삭제 된 데이터: 1
```