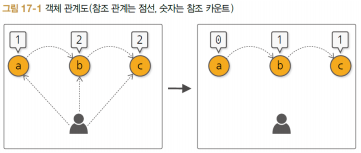
**딥러닝프레임워크 5주차**

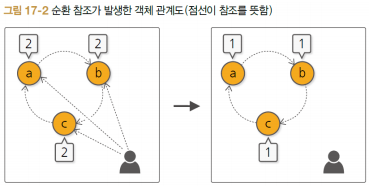
**201904192 김성중**

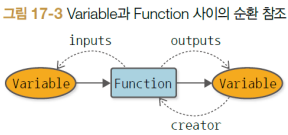
**메모리 관리**  
- 파이썬 메모리 관리  
필요 없어진 객체를 메모리에서 자동으로 삭제  
코드 작성에 따라 메모리 누수(memory leak) 또는 메모리 부족(out of memory) 문제 발생

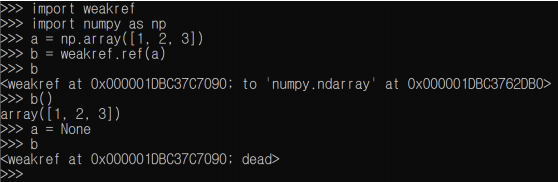
- 신경망의 메모리 관리   
큰 데이터를 다룸   
메모리 관리가 적절하지 못하면 실행 시간이 오래 걸리는 일이 자주 발생

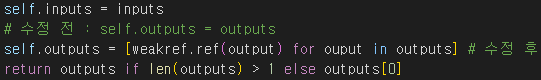
# 참조 수를 세는 방식  
# Garbage Collection(java에도 있는 방식) : 세대를 기준으로 쓸모없어진 객체를 회수하는 방식

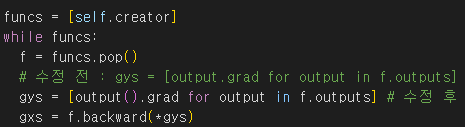
**참조 카운트**- 모든 객체는 참조 카운트가 0인 상태로 생성, 다른 객체가 참조할 때마다 1씩 증가  
- 객체에 대한 참조가 끊길 때마다 1씩 감소, 0이 되면 회수  


**순환 참조**  
- 참조 카운트로는 해결할 수 없는 문제  
- a = b =c =None 삭제 안됨  


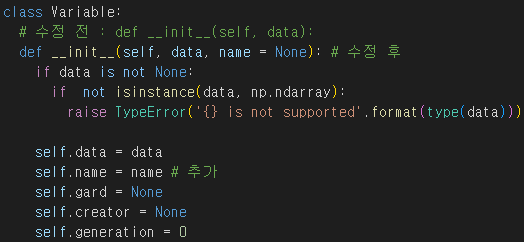
**GC(Garbage Collection)**  
- 메모리가 부족해지는 시점에서 자동 호출(gc.colect()로 명시적(강제적) 호출도 가능)  
- 메모리 해제를 GC에 미루다 보면 메모리 사용량이 커짐  
- DeZero에서는 순환참조를 만들지 않는 것이 좋음  


**약한 참조(weak reference)**  
- 다른 객체를 참조하되 참조 카운터는 증가시키지 않는 기능  
- 파이썬에서는 weakref.ref 함수를 사용하여 약한 참조를 만듬  
- b는 약한 참조이고, 약한 참조된 데이터에 접근하려면 b( )라고 쓰면 됨  
  
b는 약한참조, 약한 참조된 데이터에 접근하려면 b()라고 쓰면 됨  
a = None을 명시 후, b를 출력하면 dead라고 나옴 -> 인스턴스가 삭제됨

Weakref 구조를 Dezero에 도입  




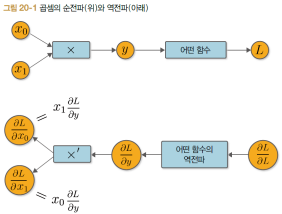
**변수 사용성 개선**

- Variable 클래스를 더욱 쉽게 사용하도록 개선  
# 변수 이름 지정  


**연산자 오버로드**

- 연산자를 지원하도록 Variable확장(덧셈, 곱셈 연산자)

- Variable 인스턴스를 ndarray 인스턴스처럼 사용하도록 구성(y = a \* b)  
#DeZero를 평범한 넘파이 코드를 작성하듯 사용 가능

곱셈의 순전파와 역전파  
- 역전파는 Loss의 각 변수에 대한 미분을 전파  


- Mul 클래스 구현  
