Day3 [Basic Data Structure]

num = 10	num(variable), 10(object) - stores in stack
String str = “Test”	str(variable), “Test”(object) -stores in heap
test()	test(function) -stores in stack
num1 = 20.3	num1(variable), 20.3(object) -stores in stack

- 모든 변수나 함수는 선언하면 스택(Stack)으로 들어가게 되어있다.
- 클래스의 데이터값은 힙(Heap)으로 들어간다.

Test t

- “Test” is a class
- “t” is a variable, not an object
    - 초기화를 안해줬기 때문에 object가 아니다.
    - 변수와 함수가 필요하다.
    - 새로운 사용자 타입으로서 메모리를 할당 못한다.

Module

- 함수의 집합
- 사용하고자 하는 놈을 임포트
- from collection import deque

Two types of variables

1. **Class variable**
    a. 클래스가 가지고 있는 변수
2. **Reference variable (reference object variable)**
    a. 객체가 가지는 변수

Function Vs Methods

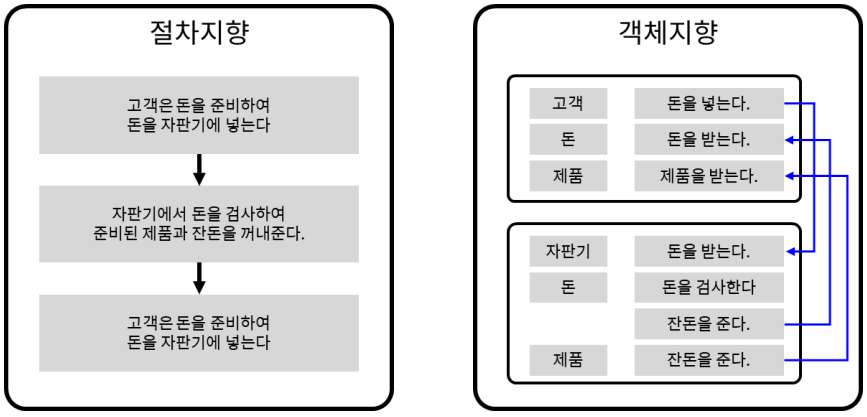
1. Same feature
    a. If the data goes in, modify it and bring out the output
2. PL(Precedual language)
    a. Use functions
3. OOP (JAVA, Python, C++)
    a. Use Methods

functions vs Methods

- **Methods(메서드)**
    - 클레스가 가지는 맴버 함수
    - 클라스의 객체(object) 없이는 사용(호출)할 수 없다.
- **Function(함수)**
    - 클레스랑 상관이 없다.
    - 클라스의 객체와 상관없이 사용할 수 있다.

1. Same feature
    a. If the data goes in, modify it and bring out the output
2. PL(Precedual language)
    a. Use functions
3. OOP (JAVA, Python, C++)
    a. Use Methods

절차적 언어 (Procedural Language) vs 객체지향 언어 (Object-Oriented Language, OOP)



1. **절차적 언어 (PL)**
    a. C, Pascal, BASIC
    b. 프로그램을 **"순차적인 명령의 집합"** 으로 봅니다.
    c. 함수(또는 프로시저)를 중심으로 코드를 작성합니다.
    d. 데이터와 함수가 분리되어 있습니다.
    e. **"어떻게(How)"** 에 집중합니다.
        i. 예: "버튼을 클릭하면 **A** → **B** → **C** 순서로 실행해라."
2. **객체 지향언어 (OOP)**
    a. Java, C++, Python, C#
    b. 프로그램을 "객체(Object)들의 상호작용" 으로 봅니다.
    c. 데이터(속성)와 함수(메서드)를 하나의 객체로 묶어 관리합니다.
    d. 상속, 다형성, 캡슐화, 추상화 같은 개념을 사용합니다.
    e. "무엇(What)" 에 집중합니다.
        i. 예: "버튼 객체는 클릭 시 이벤트를 처리한다."

OOP언어

- C++, JAVA, Python
- Class
    - Two components:
        1. 멤버 변수(member variable)
            a. member variable(field) that stores data
        2. 멤버 함수 (member methods)
            a. member function or method that uses a variable

    ex)
    Class Test;
    int test;
        - 클래스가 초기화가 되어있지 않아 사용할 수 없다.
        - 선언을 해도 메모리 할당이 되지 않는다.

- Object(객체)
    - 객체가 완성되기 전에는 단지 변수이다.

Class 는 사용자가 정의하는 새로운 Datatype의 일부분이다.
ex)

```
class Student {
    char name[12]
    int age
    char cell [15]
    float avg
}
```

name = 12 byte
age(int) = 4 byte
cell = 15 byte
avg(float) = 4 byte
⇒ 35 byte

- 파이썬에서는 제일 큰 데이터타입이 8byte이다.
- 여기서 파이썬 프로그램은 더 큰 데이터타입을 가지는게 불가능하다. 따라서, 사용자는 Class를 이용해 새로운 Datatype을 정의할 수 있다.
- 따라서 Student라는 Datatype을 하나 새로 만들어서 4가지 종류의 데이터타입을 가지게 한다 (35byte).

이 35byte의 타입을 담을 수 있는 변수를 만들어준다.

```
class Student {
    char name[12]
    int age
    char cell [15]
    float avg
}

Student std
```

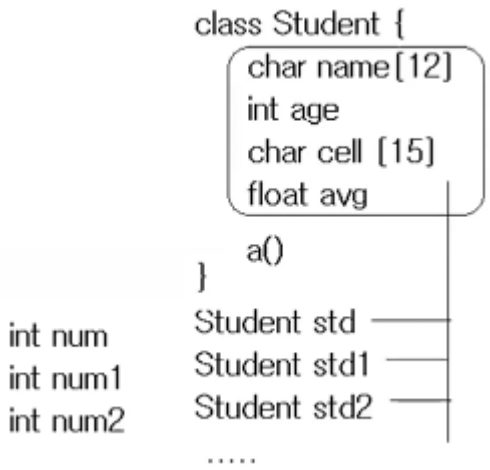
Student std는 Student형 변수가 된다.

- Student는 프로그램 자체적으로 정의되어있는 데이터타입이 아니다.
- 사용자가 정의한 Datatype이다.

reference = 클라스가 가지는변수(variable)

1. 참조한다라는 뜻
2. class variable

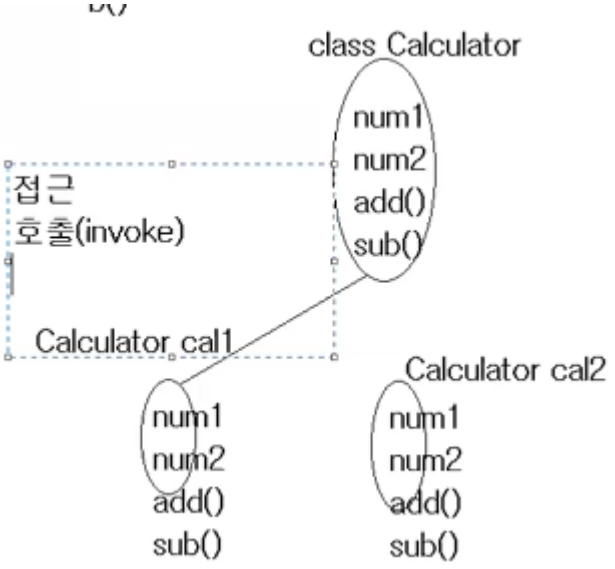
ex) Class의 Object 및 메서드 사용 예시:



std, std1, std2는 다 다른 객체(object) 이다.

- 선언되면 주소가 할당이 되는데 다 다른 주소다.
- a()라는 메서드를 사용할려면 클라스의 객체가 필요하다.

ex) Class의 접근, 호출 예시:



Calculator 라는 클라스의 객체, cal, cal2를 선언했다.

- Calculator 안에 있는 member variable를 그대로 접근(Access)할 수 있다.
- Calculator 안에 있는 member method를 그대로 호출(invoke)할 수 있다.
    - 끈을 연결해주는 기능을 생성자(Constructor) 라고 한다.
    - 객체를 초기화 해준다.

스택 / 큐 /

- 선형

트리 / 그래프

- 비선형

Day3 List(array)

a. 배열 리스트 (array list)
b. 링크드 리스트 (linked list)
    i. reference(주소)

파이썬 → 배열을 두가지로 나누어서 이야기 한다.

1) 리스트 (수정가능) ['a', 'b]
    a) 데이터에 한 번에 접근할 수 있으니 어디에 있는 지만 알면 빠르게 탐색할 수 있다. 이러한 접근 방식을 **<u>임의 접근</u>**이라고 이야기 한다.
2) 튜플 (읽기만) (a , b)

- 딕셔너리 → 자료구조 { key : value }

배열의 인덱스와 값을 일대일 대응해 관리하는 자료구조

- 어떤 위치에 있는 데이터든 한 번에 접근할 수 있다. (데이터를 찾기가 가장 빠르다)

파이썬 코테에서 주로 사용되는 배열의 선언방법:

ex)
```
arr = [0, 0 ,0, 0, 0, 0]                # 정수형 배열 리스트를 선언(길이값 6)
arr = [0] * 6
```

- 코테에서 일반적인 선언 방법

- 리스트 생성자를 사용하는법
```
arr = list(range(6))      # [0,1,2,3,4,5]
```

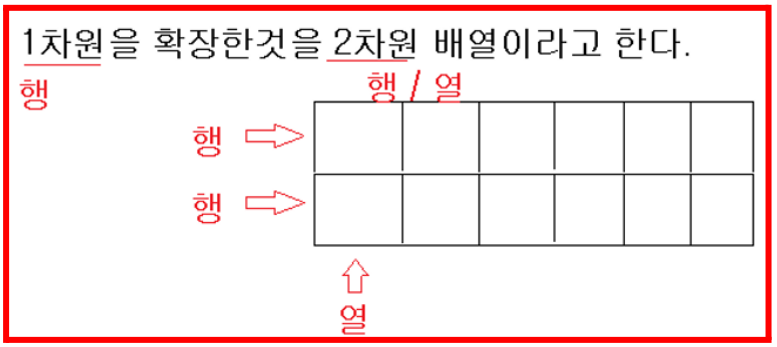
- 리스트 컴프리헨션 을 이용하는 방법 (0으로 초기화)
```
arr = [0 for i in range(6)]        # [0,0,0,0,0,0]

arr = [0 for _ in range(6)]        # for(초기값; 조건값; 증가값)
                                                i < 6;
```

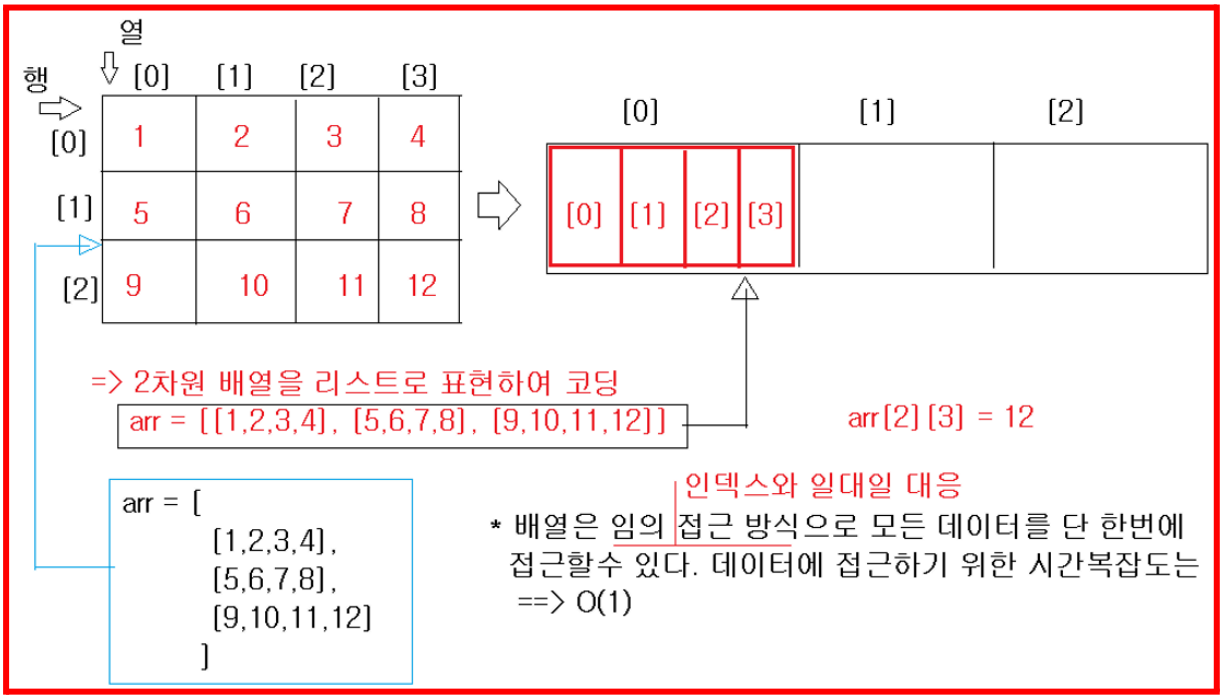
파이썬의 리스트는 동적으로 크기를 조절할 수 있다.

- 다른 언어에 비해 배열의 기능을 그대로 사용하면서 배열의 크기도 가변적으로 사용한다.
- 슬라이싱, 삽입, 삭제, 연결 등은 기본적으로 이해를 하여야 한다.

배열(리스트) → **1차원 배열, 2차원 배열**, 3차원 배열 ….. n차원 배열
배열은 차원과 상관없이 메모리에 연속적으로 할당



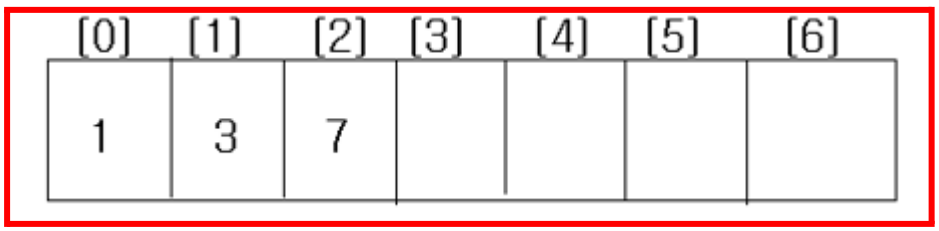
1차원 배열 arr[ ]
2차원 배열 arr[ ][ ]



파이썬의 배열은 유동적이다. (배열의 길이값)
일반적으로 배열의 길이값은 2가지로 나눈다.

1. 정적 배열
    a. 한번 길이값을 선언하면 조절 불가하다.
2. 동적(유동) 배열
    a. 길이값을 조절할 수 있다.

- **일반적으로 절차적언어는 정적 배열이고 객체지향 언어는 동적 배열이다.**



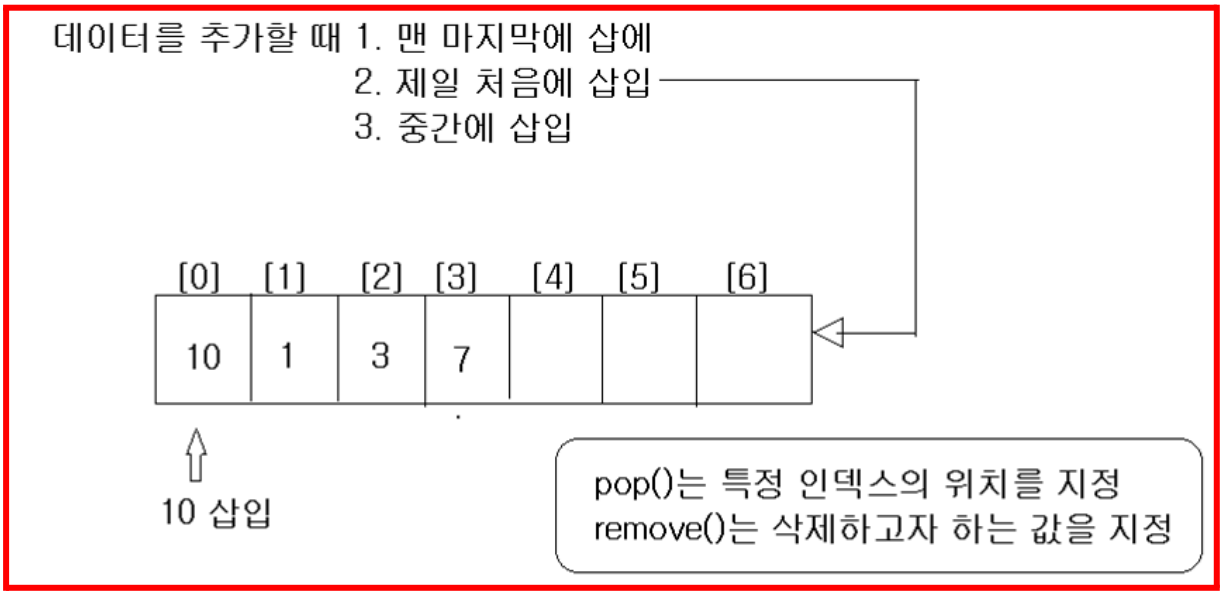
데이터를 리스트에 삽입할 때 (3가지 방법)

1. 맨 마지막에 삽입 - O(1)
    a. 값이 들어있는 마지막 인덱스의 다음 인덱스에 추가하면 되므로 연산이 한번만 필요하다.
2. 제일 처음에 삽입 - O(n)
    a. 데이터를 처음에 삽입하려면 이미 있는 데이터를 한번씩 뒤로 넘겨야 하므로 n번 넘겨야한다. 따라서 시간복잡도는 O(n)이다.
3. 중간에 삽입 - O(n)
    a. 리스트의 중간에 데이터를 삽입하면, 해당 위치부터 모든 데이터를 한 칸씩 뒤로 이동해야 함.
    b. 최악의 경우 맨 앞에 삽입하는 것과 동일한 연산량이 발생 → O(n).
4. ex)
   arr[3]에 10이라는 데이터를 삽입할때 시간 복잡도는 O(1)
    - 인덱스값을 지정함으로서 한번만 연산을 하면된다.
        - 시간복잡도 O(1) - 10,000번 안에 데이터에 접근
        - 시간복잡도 O(2) - 20,000번 안에 데이터에 접근

ex)
list = [10, 20, 30] # 리스트 선언
list.append(40) # 40이라는 데이터를 리스트에 삽입

element = list.pop(2) # 2번째 인덱스에 있는 값(30)을 제거한후 element에 할당

pop()는 <u>특정 인덱스의 위치를 지정</u>해야함
remove()는 <u>삭제하고자 하는 값을 지정</u>해야함



문제) 정수 배열을 정렬해서 반환하는 함수를 작성하시오

arr = [1, -5, 2, 4, 3] 출력 => [-5, 1, 2, 3, 4]

```
def solution(arr):
    arr.sort() # 정렬하는 메서드이다. 리스트의 원본 자체의 값을 바꾼다.
    return  arr

def solution(Arr):
    sorted_list = list(sort(arr)) # 리스트 원본은 그대로 보존하고 결과값만 구현
    return sorted_list
```

sort() 메서드를 사용하지 않고 정렬 알고리즘으로 작성할 수 있다.

-그러나 너무 비효율적으로 나온다.

문제) 정수 배열을 하나 받는다. 그 중 배열의 중복값을 제거하고 배열 데이터를 내림차순으로 정렬해서 반환하는 함수를 작성하시오

	입력	출력
-	[4,2,2,1,3,4]	- [4,3,2,1]

문제] 정수 배열을 하나 받는다. 그 중 배열의 중복값을 제거하고 배열 데이터를 내림차순으로 정렬해서 반환하는 함수 작성하시요

입력] [4, 2, 2, 1, 3, 4] 결과값으로는 [4, 3, 2, 1]

1) 중복 제거 함수 set()
2) 내림 차순 명령어 reverse = True

```
def solution(arr):
    list1 = list(set(arr))
    list1.sort(reverse = True)
    return list1
```

문제) 정수 배열 num이 주어질때 리스트 num에서 서로 다른 인덱스에 있는 2개의 수를 뽑아 더해 만들 수 있는 모든 수를 배열에 오름차순으로 정렬하는 solution을 작성하시오.

예를들어 [2, 1, 3, 4, 1] ⇒ [2, 3, 4, 5, 6, 7]

합의 수가 같으면 중복 제거한다.

```
def solution(num):
    ret = []
    for i in range(len(num)):
        for j in range(i + 1, len(num)):
            ret.append(num[i] + num[j])

    ret = sorted(sort(ret))
    return ret
```

코드 연습:
https://colab.research.google.com/drive/1AoAQ6Kh_PcLVzPvWp8HQuFwgvKb90auf#scrollTo=oOQWRNYa7WZk