컴퓨터에 데이터를 저장하는 방법

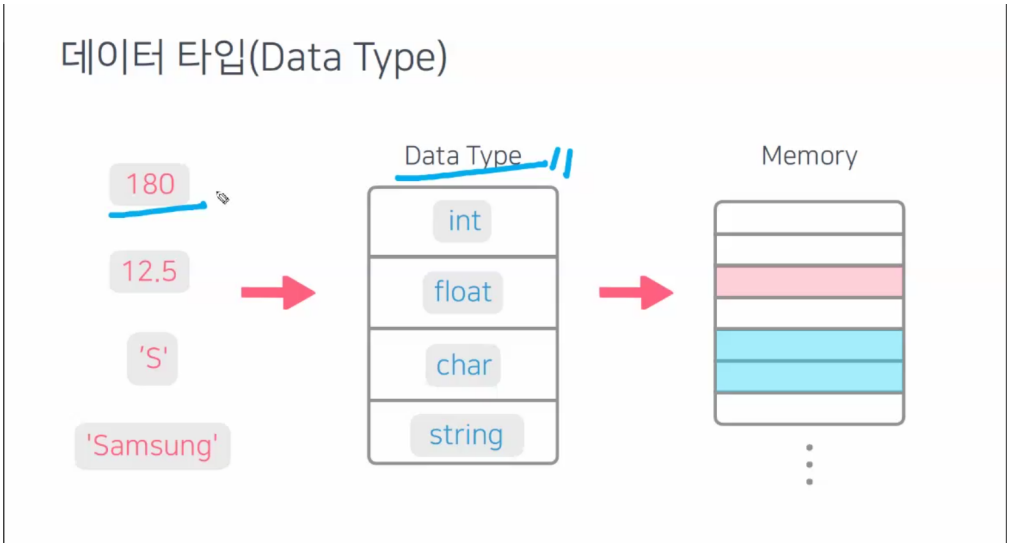
스토리지

- 데이터가 영구히 저장되는 곳
- HDD, SSD, USB, CD
- 용량이 크지만 속도가 느림
- 당장 필요하지 않은 데이터

메모리

- 데이터가 임시적으로 저장되는 곳
- 용량이 작지만 속도가 매우 빠름
- 당장 필요한 데이터
- 메모리를 효율적으로 사용하자.
- 데이터가 저장 되는 위치 Random
- RAM (Random Access Memory)
- 각 칸마다 주소값이 지정되어 있음.

데이터 타입



데이터를 메모리에 저장하기 전에

데이터 타입을 지정하고 그 후에 메모리에 저장한다.

데이터 타입이란 :

더 다양한 데이터 구조를 표현하고 싶어

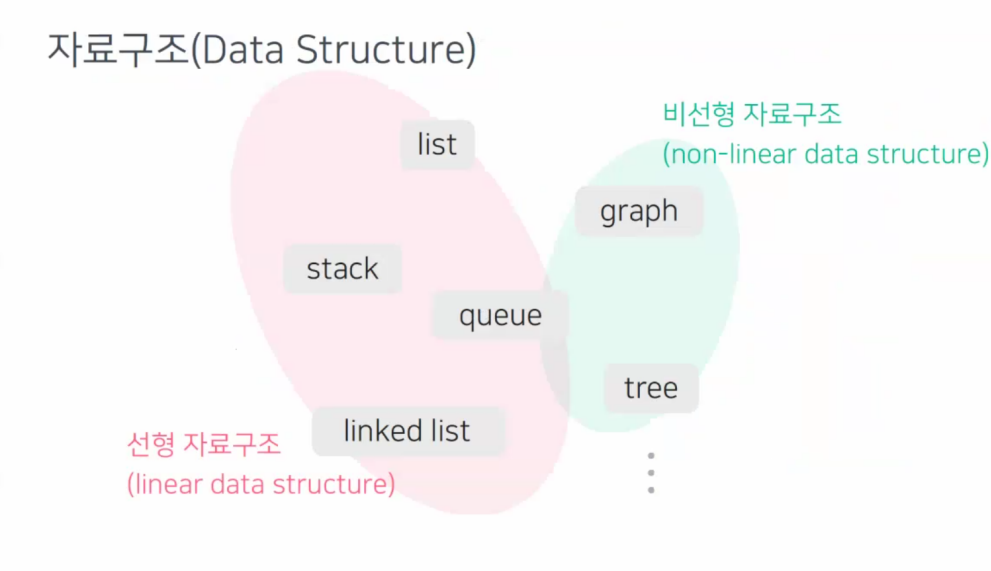
내가 데이터 타입을 정의하고 싶은데?

-> User-defined data type

클래스를 이용해서 나만의 데이터 타입을 선언

자료구조 (Data Structure)

컴퓨터의 메모리를 효율적으로 사용할 수 있도록 데이터를 저장하고 구성하는 방법



자료구조의 장점

데이터의 구조를 전체적으로 파악해서 효율적으로 가용가능

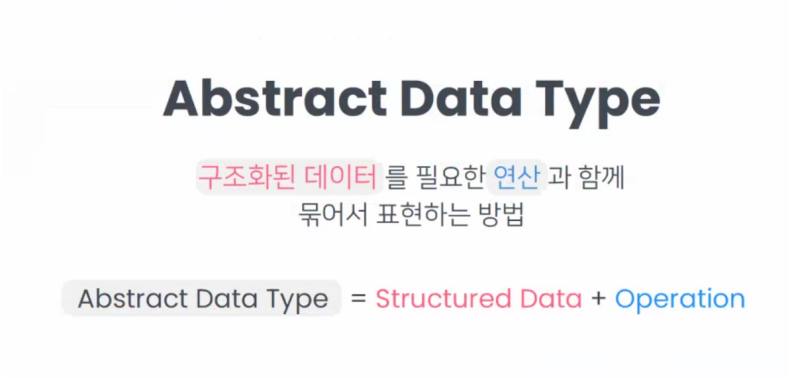
데이터를 표현하기 쉽고, 이해하기 쉽고, 활용하기도 쉽다.

추상 데이터 타입

데이터가 가지고 있는 공통점을 모으고

각각의 데이터가 필요한 구조, 연산을 따로 할당

Abstract Data Type



Stack

물건을 쌓아 올린듯, 자료를 쌓아 올린 형태의 자료구조

스택 (stack)

물건을 쌓아 올린 듯 자료를 쌓아 올린 형태의 자료구조

1) 가장 마지막에 들어간 것이, 가장 처음에 나온다.

후입선출
LIFO (Last-In-Fist-Out)

2) 가장 위에서만 데이터의 삽입 & 삭제가 일어난다

top : 삽입과 삭제가 일어나는 지점

pop: 데이터를 뽑아 내는 것

push : 하나의 데이터를 삽입하는 것

함수 콜 스택

```
def f(n):
    if n == 1:
        return 1
    else:
        return n * f(n-1)
```

스택의 연산 (Operations of stack)

①	CreateStack	스택을 생성하는 연산, size 필요
②	IsEmpty	스택이 현재 비어있는지를 확인 하는 연산, True False 리턴
③	IsFull	스택이 현재 꽉 차있는지를 확인 하는 연산, True, False 리턴
④	Push	스택에 새로운 데이터 요소를 삽입 하는 연산
⑤	Pop	스택에서 가장 위에있는 요소를 제거 하는 연산, 데이터 반환
⑥	Peek	스택에서 가장 위에있는 요소를 반환 하는 연산

스택의 연산 (Operations of stack)

①	CreateStack	스택을 생성하는 연산, size 필요
②	IsEmpty	스택이 현재 비어있는지를 확인 하는 연산, True False 리턴
③	IsFull	스택이 현재 꽉 차있는지를 확인 하는 연산, True, False 리턴
④	Push	스택에 새로운 데이터 요소를 삽입 하는 연산
⑤	Pop	스택에서 가장 위에있는 요소를 제거 하는 연산, 데이터 반환
⑥	Peek	스택에서 가장 위에있는 요소를 반환 하는 연산

스택의 데이터 구조 (Data structure of stack)

①	top	스택의 가장 위에 있는 위치를 저장하고 있는 데이터
②	size	스택의 크기를 저장하고 있는 데이터
③	items	스택에 담길 데이터를 저장 할 데이터 구조

```
class Stack:
    ...
    size = n
    top = -1
    items = [None] * size
```

DFS = Stack 사용

BFS = Queue 사용