Variable & List

TEAMLAB director
최성철

variable & memory

변수의 개요

- 가장 기초적인 프로그래밍 문법 개념

- 데이터(값)을 저장하기 위한 메모리 공간의 프로그래밍상 이름

```
>>> a = 5
>>> b = 3
>>> a + b
8
```

이 순간 컴퓨터에서
무슨 일이 일어난 일은?

[문제]

professor = “Sungchul Choi” 의 의미는”

① professor의 이름은 Sungchul Choi 이다.

② professor는 Sungchul Choi 이다.

③ professor와 Sungchul Choi는 같다.

④ professor에 Sungchul Choi를 넣어라

[문제]

professor = “Sungchul Choi” 의 의미는”

④ Professor에 Sungchul Choi를 넣어라

정확히는 Professor라는 변수에

“Sungchul Choi” 라는 값을 넣으라는 의미

[문제]


```
>>> a = 5
>>> b = 3
>>> a + b
8
```

그럼 변수는 어디에 저장될까?

[문제]

```
>>> a = 5
>>> b = 3
>>> a + b
8
```

그럼 변수는 어디에 저장될까?

변수(Variable)란?

수학식 2x + 7y = 14에서 변수는
x와 y를 의미함

프로그래밍에서 변수는

수학과 약간 다른 개념

변수(Variable)란?

프로그래밍에서는 변수는

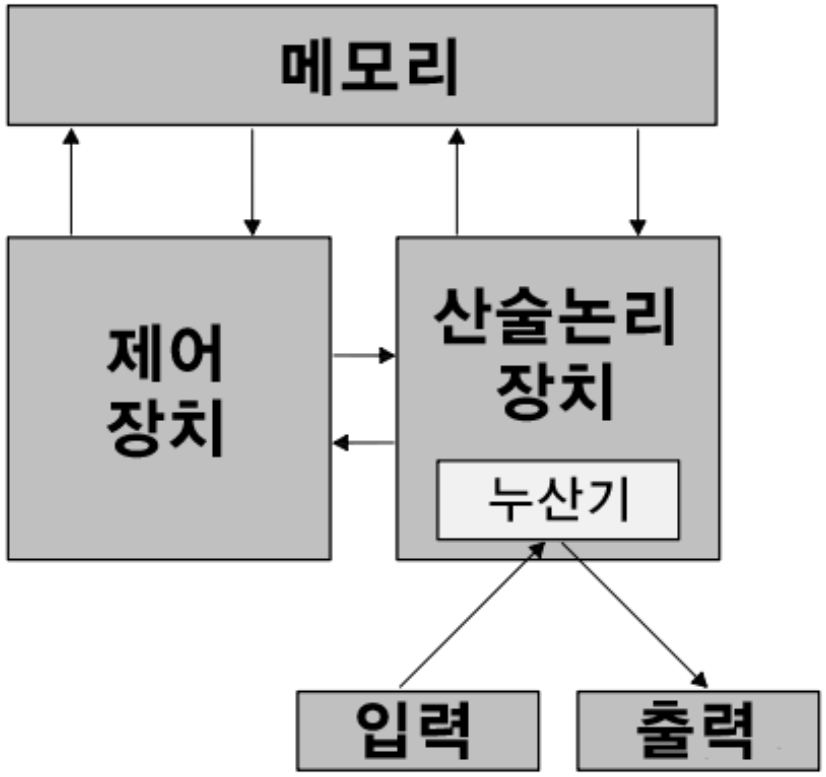
값을 저장하는 장소

변수는 메모리 주소를 가지고 있고

변수에 들어가는 값은 메모리 주소에 할당됨

[참고] 컴퓨터의 구조 – 폰 노이만(John von Neumann) 아키텍처

폰 노이만 아키텍처에서는
사용자가 컴퓨터에 값을
입력하거나 프로그램을
실행할 경우 그 **정보를 먼저 메모리에 저장**시키고
CPU가 순차적으로 그 정보를
해석하고 계산하여 사용자에게 결과값 전달



변수(Variable)란?

프로그래밍에서는 변수는

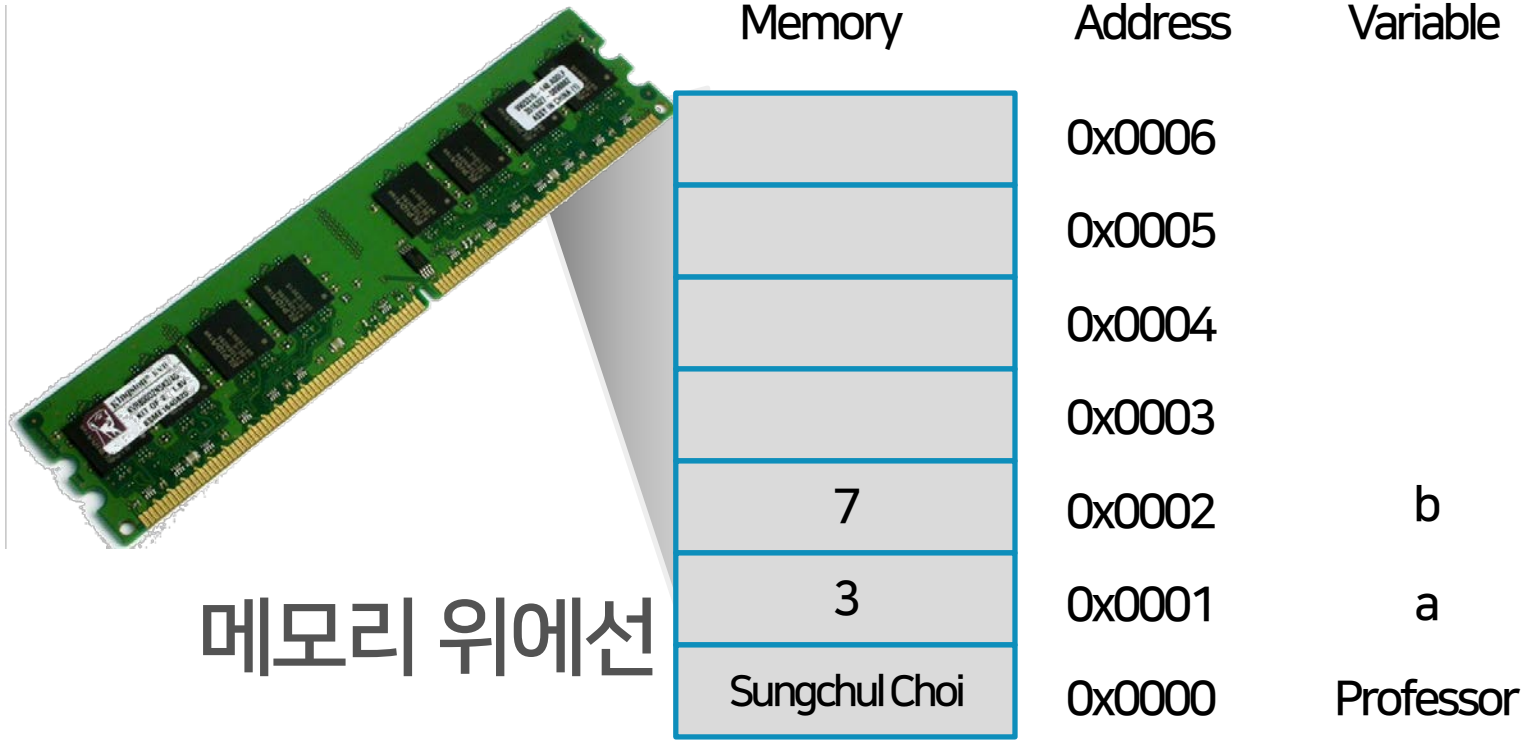
값을 저장하는 장소

변수는 메모리 주소를 가지고 있고

변수에 들어가는 값은 메모리 주소에 할당됨

변수와 메모리 주소

Professor = 'Sungchul Choi', a=3 , b=7



메모리와 변수

- 변수 – 프로그램에서 사용하기 위한 특정한 값을 저장하는 공간
- 선언 되는 순간 메모리 특정영역에 물리적인 공간이 할당됨
- 변수에는 값이 할당되고 해당 값은 메모리에 저장됨
- A = 8 의 의미는 “A는 8이다”가 아닌
  A라는 이름을 가진 메모리 주소에 8을 저장하라 임

변수 이름 작명법

- 알파벳, 숫자, 언더스코어(_) 로 선언 가능
  ex) data = 0, _a12 = 2, _gg = 'afdf'
- 변수명은 의미 있는 단어로 표기하는 것이 좋다
  ex) professor_name = 'Sungchul Choi'
- 변수명은 대소문자가 구분된다.
  ex) ABC와 Abc는 같지 않다
- 특별한 의미가 있는 예약어는 쓰지 않는다.
  ex) for, if, else 등

basic operations

Basic Operation (간단한 연산)

- 복잡한 프로그램을 작성하기 앞서 간단한 사칙연산과 문자열 처리 등의 기초적인 연산을 알아야 함
- 이를 위해 본 장에서는 아래 내용을 습득
  1) 기본 자료형 (primitive data type)
  2) 연산자와 피연산자
  3) 데이터 형변환
- 이를 통해 간단한 프로그램 작성의 기초를 익힘

기본 자료형 (primitive data types)

- data type : 파이썬이 처리할 수 있는 데이터 유형

유형			설명	예시	선언 형태
수치자료형	정수형	integer	양/음의 정수	1,2,3,100, -9	data = 1
	실수형	float	소수점이 포함된 실수	10.2, -9.3, 9.0	data = 9.0
문자형(문자형)		string	따옴표 (' / ")에 들어가 있는 문자형	abc, a20abc	data = 'abc' data ='a'
논리/불린 자료형		boolean	참 또는 거짓	True, False	data = True

자료형 사용 예시


```
>>> a = 1 # Integer
>>> b = 1 # Integer
>>> print (a, b)
1 1
>>> a = 1.5 # Float
>>> b = 3.5 # Float
>>> print (a, b)
1.5 3.5
>>> a = "ABC" # String
>>> b = "101010" # String
>>> print (a, b)
ABC 101010
>>> a = True # Boolean 대소문자 구분
>>> b = False # Boolean 대소문자 구분
>>> print (a, b)
True False
```

Dynamic Typing

코드 실행시점에 데이터의 Type을 결정하는 방법

JAVA

```java
package com.javatutorialhq.java.examples;

/*
 * Example program to test equality of Integer object
 */

public class IntegerTestEquality {

    public static void main(String[] args) {

        // instantiate Integer objects
        Integer firstInteger = 100;
        Integer secondInteger = new Integer(100);

        // test for equality
        System.out.println(firstInteger.equals(secondInteger));

    }

}
```

python

```python
first_integer = 100
second_integer = 100
print(first_integer == second_integer)
```

연산자(Operator)와 피연산자(operand)

- +, -, *, / 같은 기호들을 연산자라고 칭함
- 연산자에 의해 계산이 되는 숫자들은 피연산자라 칭함
- "3 + 2" 에서 3과 2는 피연산자, +는 연산자임
- 수식에서 연산자의 역할은 수학에서 연산자와 동일
- 연산의 순서는 수학에서 연산 순서와 같음
- 문자간에도 + 연산이 가능함 → concatenate

제곱승과 나머지 구하기

" ** " 는 제곱승 계산 연산자

```
>>> print (3 * 3 * 3 * 3 * 3)      # 3을 다섯 번 곱함
243
>>> print (3 ** 5)                  # 3의 5승
243
```

" % " 는 나머지를 구하는 연산자

```
>>> print (7 / 2)          # 7 나누기 2 (정수형 계산)
3.5
>>> print (7 % 2)          # 7 나누기 2의 나머지는
1
```

증가 또는 감소 연산

a += 1 는 a = a +1 과 같은 의미로 증가연산 (-=)

```
>>> a = 1                  # 변수 a 에 1을 할당
>>> a = a + 1              # a 에 1를 더한 후 그 값을 a에 할당
>>> print (a)              # a 출력
2
>>> a += 1                 # a 증가 연산
>>> print (a)              # a 출력
3
>>> a = a  -  1            # a 에 1를 뺀 후 그 값을 a에 할당
>>> a -= 1                    # a 감소 연산
>>> print (a)              # a 출력
1
```

만약 a = 1 일 때 a = 1 + 1 로 a에 다시 2가 할당(assign)됨

즉 좌변에 a는 할당 받는 변수 (variable)

우변에 a는 기존 a의 값(value)

데이터 형 변환: 정수형 ↔ 실수형

float()와 int() 함수를 사용하여 데이터의 형 변환 가능

```
>>> a = 10                  # a 변수에 정수 데이터 10을 할당
>>> print (a)               # a가 정수형으로 출력
10

>>> a = float(10)           # a를 실수형으로 변환 / 정수형은 int()
>>> print (a)               # a를 출력
10.0                        # a가 실수형으로 출력됨

>>> b = 3                   # b 에 정수 데이터 3 할당
>>> print (a / b)           # 실수형으로 a 나누기 b를 출력
3.33333333333               # 실수형 결과값 출력
```

10.3과 10.7 정수형으로 형변환 후 덧셈하면 결과값은?

데이터 형 변환: 정수형 ↔ 실수형

실수형에서 정수형으로 형 변환 시 소수점 이하 내림

```
>>> a = 10.7
>>> b = 10.3

>>> a= int(a)          # a를 정수형으로 형변환후 a에 할당
>>> b = int (b)        # b를 정수형으로 형변환후 b에 할당
>>> print (a+b)        # 정수형 a와 b의 합을 출력
20

>>> print (a)          # 정수형 a값 출력
10
>>> print (b)          # 정수형 b값 출력
10
```

데이터 형 변환: 숫자 ↔ 문자열

문자열로 선언된 값도 int(), float()함수로 형 변환 가능

```
>>> a = '76.3'                          # a에 문자열 76.3을 할당, ''은 문자열을 의미
>>> b = float(a)                        # a를 실수형으로 형 변환 후 b에 할당

>>> print (a)                           # a값 출력
76.3

>>> print (b)                           # b 값 출력
76.3
>>> print (a + b)                       # a와 b를 더함 그러나 문자열과 숫자열의
Traceback (most recent call last):      # 덧셈이 불가능하여 에러발생
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'float' objects
```

데이터 형 변환: 숫자 ↔ 문자열

a와 b를 실수형으로 덧셈하고, 문자열로 연결하려면? → 형을 맞춰라

```
>>> a = float(a)                # a를 실수형으로 형 변환 후 a에 할당
>>> b = a                       # 실수형 a 값을 b에 할당
>>> print (a + b)               # 두 실수형 더한 후 출력
152.6

>>> a = str(a)                  # 실수형 a 값 문자열로 변환 후 a 할당
>>> b = str(b)                  # 실수형 b 값을 문자열로 변환 후 b 할당
>>> print (a + b)               # 두 값을 더한 후 출력
76.376.3                        # 문자열간 덧셈은 문자열간 단순 연결
```

데이터 형 확인하기

type() 함수는 변수의 데이터 형을 확인하는 함수

```
>>> a=int(10.3)                # a는 정수형으로 10.3을 할당
>>> b=float(10.3)              # b는 실수형으로 10.3을 할당
>>> c=str(10.3)                # c는 문자열으로 10.3을 할당

>>> type(a)             # a의 타입을 출력
<class 'int'>
>>> type(b)             # b의 타입을 출력
<class 'float'>
>>> type(c)             # c의 타입을 출력
<class 'str'>
```

컴퓨터의 반올림 오차

아래와 같이 나오는 이유는 무엇일까?

Python 2.7 Only

```
>>> c = 38.8              # c에 실수형 38.8 할당
>>> print (c)             # c 출력
38.8
>>> c                     # c에 있는 값은?
38.799999999999997        # 응?
```

컴퓨터의 모든 값은 이진수로 변환되어 메모리에 저장

Python 2.7에서만 나오는 숫자 3.x 에선 정상으로 나옴

컴퓨터의 반올림 오차

0.1를 이진수 변환하여라

0.1 X 2 = 0.2 → 0

0.2 X 2 = 0.4 → 0

0.4 X 2 = 0.8 → 0

0.8 X 2 = 1.6 → 1

0.6 X 2 = 1.2 → 1

0.2 X 2 = 0.4 → 0 ……

0.0001100110011001100110011……(2)

https://ko.wikipedia.org/wiki/부동소수점

단순한 실수도 이진수로 변환하면 무한소수가 됨

반올림오차는 충분히 작아 반올림을 하여 일반적으로 문제가 되지 않음

[참고] 컴퓨터는 왜 이진수를 쓰나?

컴퓨터는 실리콘이라는 재료로 만든 반도체로 구성됨

반도체는 특정 자극을 줬을 때 전기를 통할 수 있게 하는 물질

Source : http://samsungsemiconstory.com/1

도체와 부도체에 반해 반도체는 전류의 흐름의 제어가 가능

전류가 흐를 때 1, 흐르지 않을 때 0으로만 숫자를 표현할 수 있음

이진수 한자리를 bit라 칭하고 8개의 bit는 1byte

list

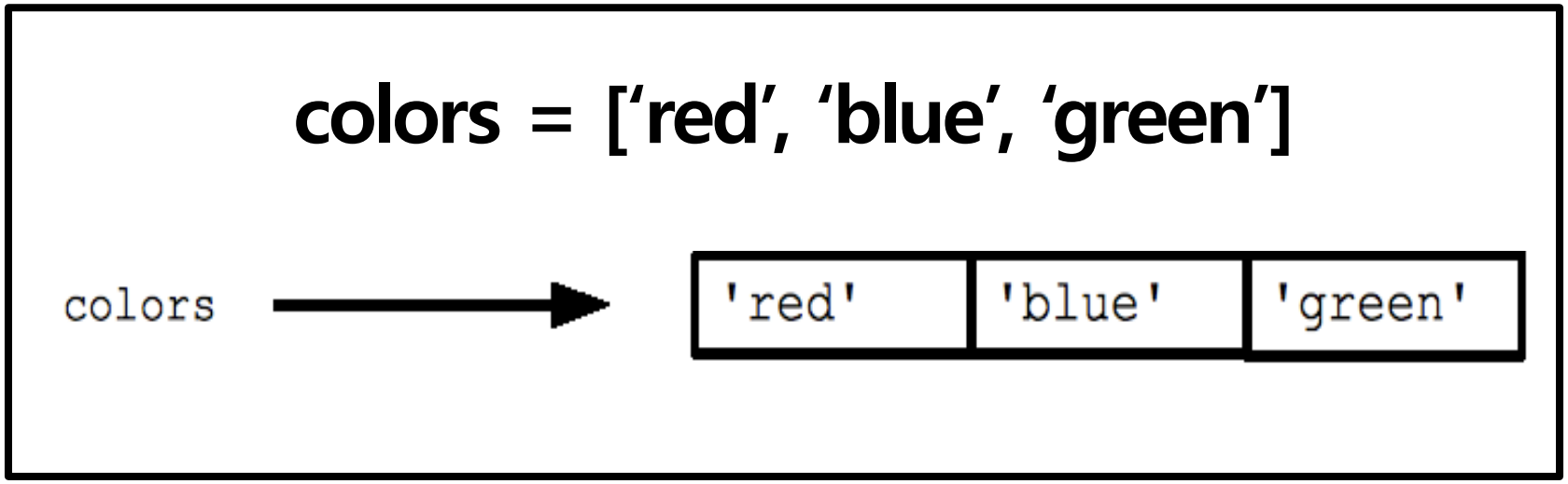
데이터가 100개 있다면 어떻게 관리할 것인가?

100명의 성적 관리를 위한 변수는 몇 개?

100개?　　1개?

List 또는 Array

- 시퀀스 자료형, 여러 데이터들의 집합
- int, float 같은 다양한 데이터 타입 포함



list의 특징

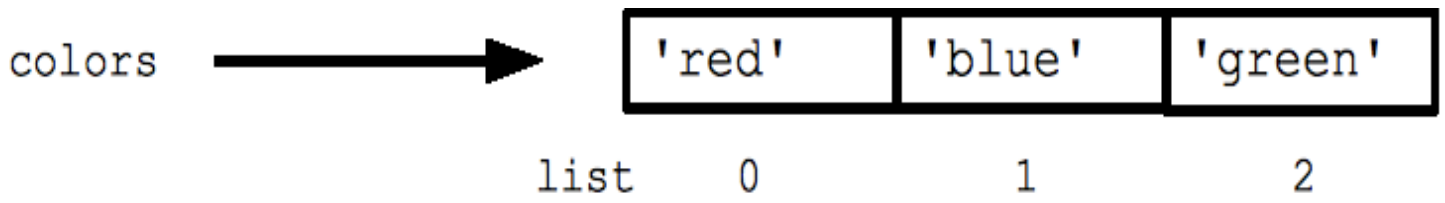
- 인덱싱 indexing
- 슬라이싱 slicing
- 리스트 연산
- 추가 삭제
- 메모리 저장 방식
- 패킹과 언패킹
- 이차원 리스트

인덱싱 (Indexing)

- list에 있는 값들은 주소(offset)를 가짐

  → 주소를 사용해 할당된 값을 호출

```
colors = ['red', 'blue', 'green']
print (colors[0])    # red
print (colors[2])    # green
print (len(colors))  # 3
                     # len은 list의 길이를 반환
```



슬라이싱 (Slicing)

- list의 값들을 잘라서 쓰는 것이 슬라이싱
- list의 주소 값을 기반으로 부분 값을 반환

```
cities = ['서울', '부산', '인천', '대구', '대전', '광주', '울산', '수원']
print (cities[0:6], " AND ", a[-9:]) # a 번수의 0부터 5까지, -9부터 끝까지
print (cities[:])          # a변수의 처음부터 끝까지
print (cities[-50:50])  # 범위를 넘어갈 경우 자동으로 최대 범위를 지정
print (cities[::2], " AND ", a[::-1]) # 2칸 단위로, 역으로 슬라이싱
```

리스트의 연산

- concatenation, is_in, 연산 함수들

```
>>> color = ['red', 'blue', 'green']
>>> color2 = ['orange', 'black', 'white']
>>> print (color + color2)   # 두 리스트 합치기
>>> len(color) # 리스트 길이
>>> color[0] = 'yellow' # 0번째 리스트의 값을 변경
>>> print (color * 2)  # color 리스트 2회 반복
>>> 'blue' in color2   # 문자열 ‘blue‘가 color2 존재 여부 반환
>>> total_color = color + color2
```

리스트의 연산 – 추가와 삭제

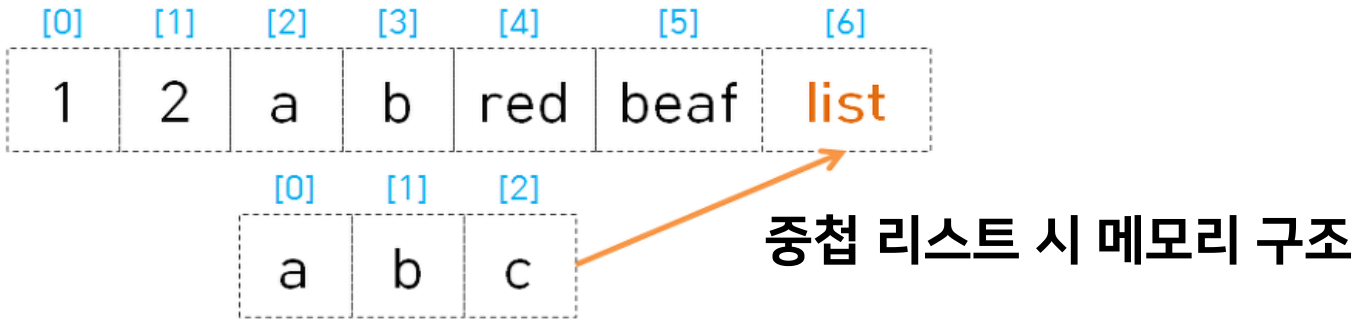
- append, extend, insert, remove, del 등 활용

```
 # 이전 장과 연결 돼서 실행
>>> color.append("white")        # 리스트에 “white” 추가
>>> color.extend(["black","purple"]) # 리스트에 새로운 리스트 추가
>>> color.insert(0,"orange")     # 0번째 주소에 “orange” 추가
>>> print (color)
['orange', 'yellow', 'blue', 'green', 'white', 'black', 'purple']
>>> color.remove("white")        # 리스트에 “white” 삭제
>>> del color[0]                 # 0번째 주소 리스트 객체 삭제
>>> print (color)
['yellow', 'blue', 'green', 'black', 'purple']
```

Python 리스트만의 특징

- 다양한 Data Type이 하나에 List에 들어감

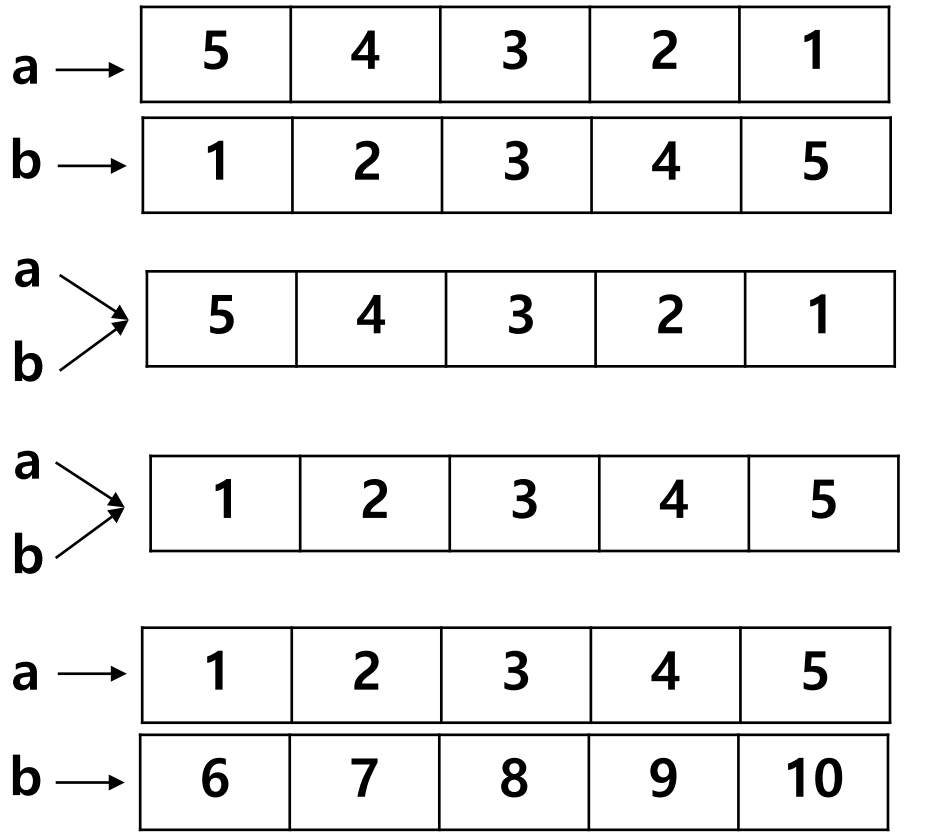
```
>>> a = ["color", 1, 0.2]
>>> color = ['yellow', 'blue', 'green', 'black', 'purple']
>>> a[0] = color # 리스트 안에 리스트도 입력 가능
>>> print (a)
[['yellow', 'blue', 'green', 'black', 'purple'], 1, 0.2]
```



리스트 메모리 저장 방식

- 파이썬은 해당 리스트 변수에는 리스트 주소값이 저장됨

```
>>> a = [5, 4, 3, 2, 1]
>>> b = [1, 2, 3, 4, 5]
>>> b = a
>>> print (b)
[5, 4, 3, 2, 1]
>>> a.sort()
>>> print (b)
[1, 2, 3, 4, 5]
>>> b = [6,7,8,9,10]
>>> print (a, b)
[1, 2, 3, 4, 5] [6, 7, 8, 9, 10]
```



"=" 의 의미는 같다가 아닌 메모리 주소에 해당 값을 할당(연결)한다는 의미

패킹과 언패킹

- 패킹 : 한 변수에 여러 개의 데이터를 넣는 것
- 언패킹 : 한 변수의 데이터를 각각의 변수로 반환

```
>>> t = [1, 2, 3]          #  1,2,3을 변수 t에 패킹
>>> a , b , c = t          #  t에 있는 값 1, 2, 3 을 변수 a, b, c에 언패킹
>>> print(t, a, b, c)   # [1, 2, 3] 1 2 3
```

이차원 리스트

- 리스트 안에 리스트를 만들어 행렬(Matrix) 생성

```
>>> kor_score = [49,79,20,100,80]
>>> math_score = [43,59,85,30, 90]
>>> eng_score = [49,79,48,60,100]
>>> midterm_score = [kor_score, math_score, eng_score]
>>> print (midterm_score[0][2])
20
```

	A	B	C	D	E
국어점수	49	79	20	100	80
수학점수	43	59	85	30	90
영어점수	49	79	48	60	100

- 이차원 리스트를 복사하는 방법?

End of Document
Thank You.