

Lecture 08

데이터의 저장구조_변수와 리스트



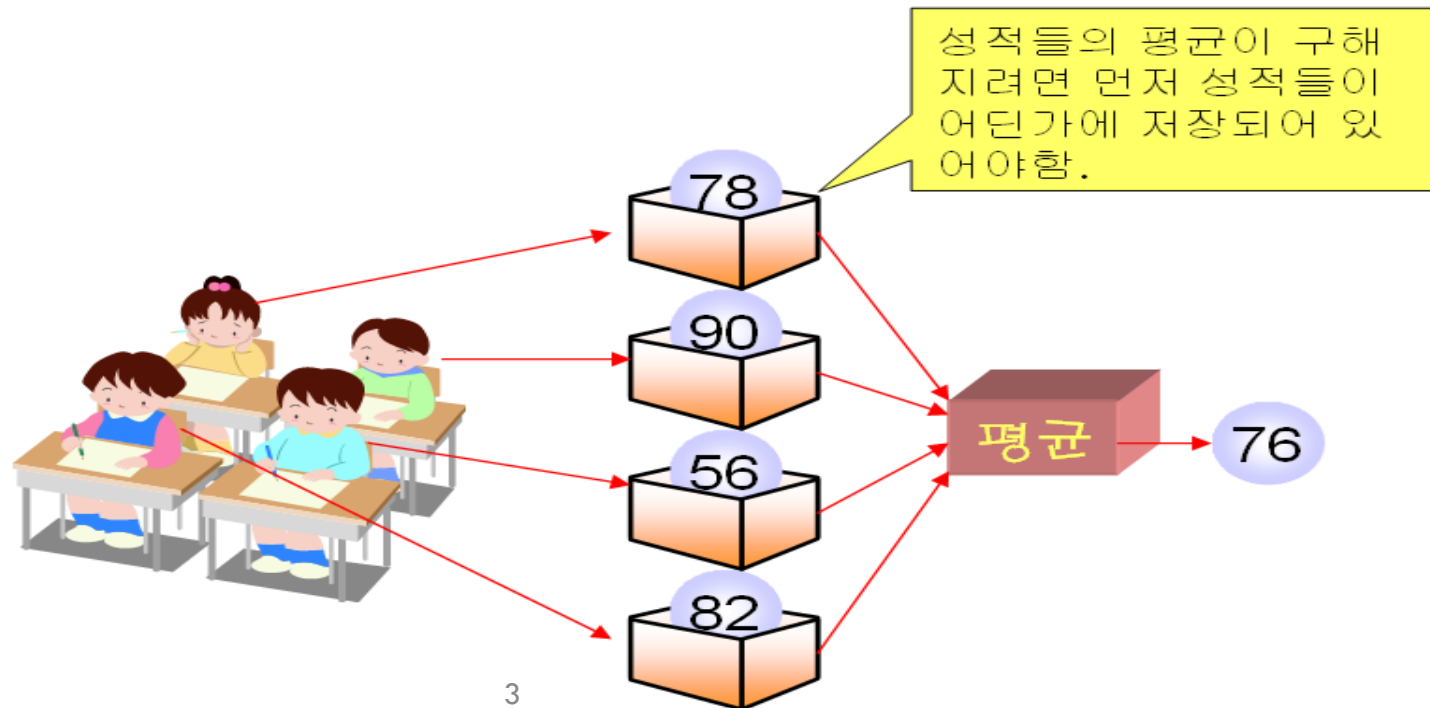
상명대학교
SANGMYUNG UNIVERSITY

학습목표

1. 변수의 의미와 특징을 말할 수 있다.
2. 파이선에서의 다양한 변수 활용법을 이해 할 수 있다.
3. 리스트의 의미와 특징을 말할 수 있다.
4. 튜플, 딕셔너리와 리스트의 차이점을 이해할 수 있다.

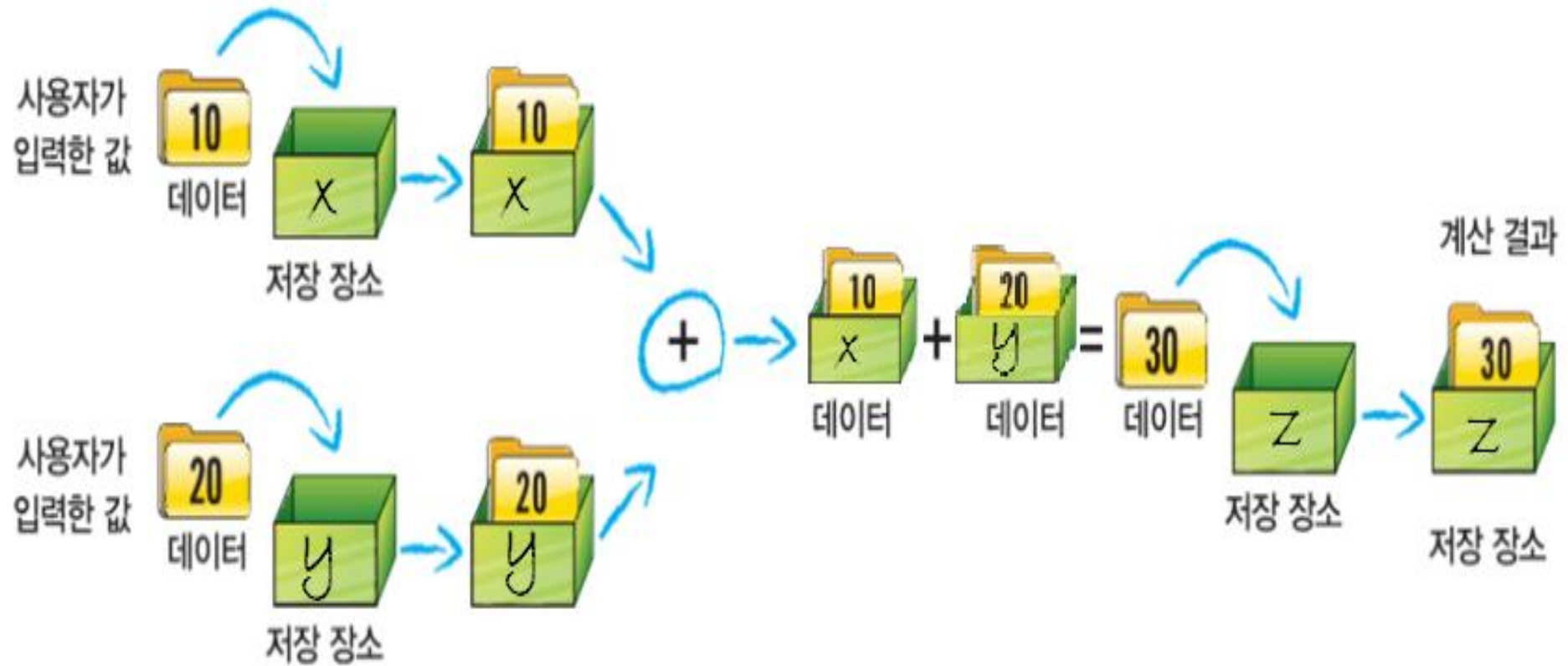
변수의 개요

- ▶ 변수를 만들기 위해서는 변수를 나타내는 고유의 이름인 변수 이름을 선언해야 함
- ▶ 선언되어 있지 않은 변수 이름을 사용하게 되면 컴퓨터는 해당 이름이 어떤 자료를 의미하는지 알 수 없기 때문에 오류(Error)를 일으키게 됨



변수의 개요




- ▶ 변수를 선언하고 나면 변수의 이름을 이용해서 메모리에 접근할 수 있음



변수의 개요

■ 변수의 표현

- ▶ 50과 20을 각각 a와 b의 변수에 담아 합계를 구한 값을 c에 넣는 과정

자료	설명	코딩
	변수 a에 50을 넣어라(대입하여라).	$a = 50$
	변수 b에 20을 넣어라(대입하여라).	$b = 20$
	변수 c에 $a+b$ 의 값을 넣어라(대입하여라).	$c = a + b$

변수의 개요

■ 자료 유형별 변수 선언과 자료 저장

- 정수형 변수

- ✓ `integer_variable = 123`

- 실수형 변수

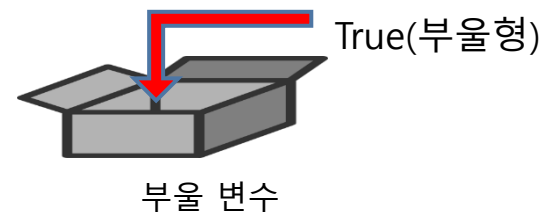
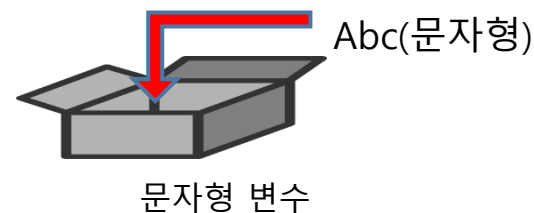
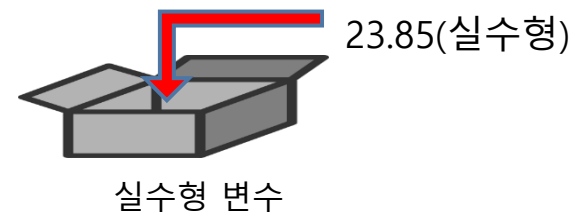
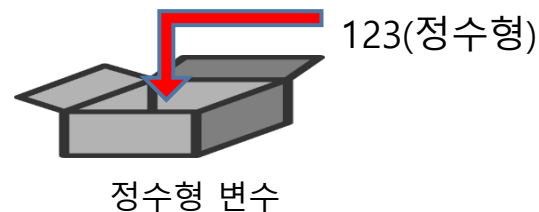
- ✓ `float_variable = 23.85`

- 문자형 변수

- ✓ `string_variable = "Abc"`

- 부울(bool)형 변수

- ✓ `bool_variable = True`



변수의 활용

- ▶ 파이썬(python)의 변수 선언

```
boolVar, intVar, floatVar, strVar=True, 0, 0.0, ""
```

- ▶ type() 함수를 사용하여 변수의 종류를 확인

```
type(boolVar), type(intVar), type(floatVar), type(strVar)
```

- ▶ 파이썬은 변수의 데이터 형식이 E

```
(<class 'bool'>, <class 'int'>, <class 'float'>, <class 'str'>)
```

```
myVar=100 → 정수형 변수를 생성함
```

```
type(myVar) → <class 'int'>가 출력됨
```

```
myVar=100.0 → 이 순간에 실수형 변수로 변경됨
```

```
type(myVar) → <class 'float'>가 출력됨
```

변수의 활용

■ 파이썬(python)의 정수 자료

- 정수형은 소수점이 없는 데이터 (기본적인 정수형 100, -123, 0 등)
- 파이썬은 변수의 선언이 없으며 변수에 값을 넣는 순간 변수의 데이터 형식이 결정됨
- 16진수는 0x나 0X로, 8진수는 0o나 0O(숫자+알파벳 오)로, 2진수는 0b나 0B로 표현

• 사용 예

`a = 0xFF`

`b = 0o77`

`c = 0b1111`

16진수 FF = $16^1 * 15 + 16^0 * 15 = 240 + 15 = 255$

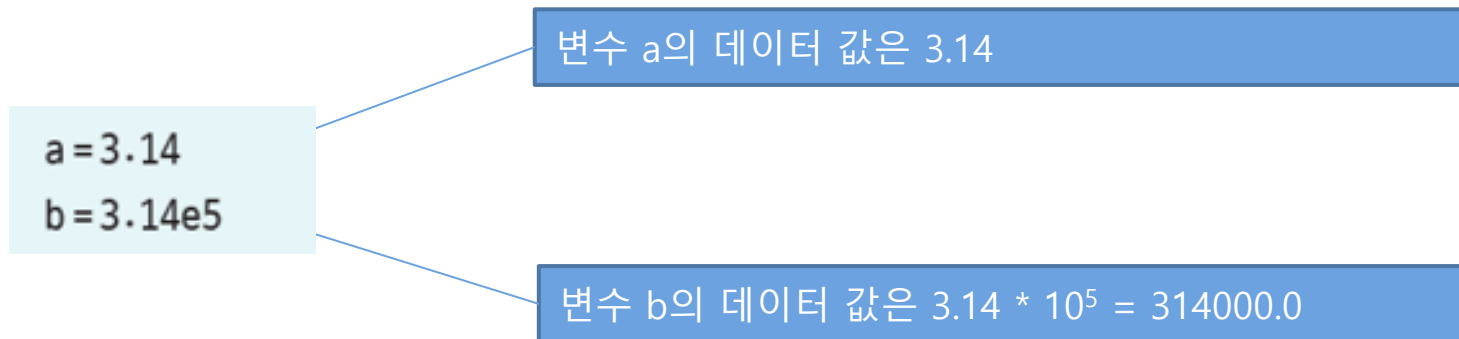
8진수 77 = $8^1 * 7 + 8^0 * 7 = 56 + 7 = 63$

2진수 1111 = $2^3 * 1 + 2^2 * 1 + 2^1 * 1 + 2^0 * 1 = 15$

변수의 활용

■ 파이썬(python) 의 실수 자료

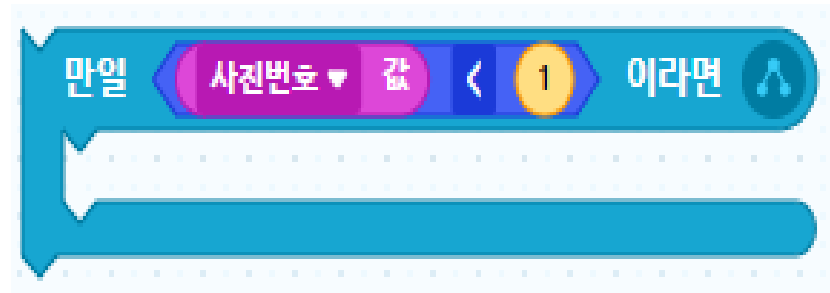
- 실수형은 3.14, -2.7처럼 소수점이 있는 데이터
- 사용 예



변수의 활용

■ 파이썬(python)의 부울(bool) 자료

- ▶ 참(True)이나 거짓(False)만 저장
- ▶ 사용 예



`a = True`

변수 a의 데이터 값에 True (참) 값을 저장

`a = (100 == 100)`
`b = (10 > 100)`

100이 100과 같은 값인가에 대한 비교, 결과는 True

10이 100보다 큰 값인가에 대한 검토, 결과는 False

변수의 활용

■ 파이썬(python)의 문자열 자료

- 문자열은 양쪽을 큰따옴표(")나 작은따옴표(')로 감싸서 사용

- 사용 예

```
ss = '상명' + '대학교'
```

변수 ss의 데이터 값은 '상명대학교'

```
s1 = '상명'  
s2 = '대학교'  
ss = s1 + s2
```

변수 ss의 데이터 값은 '상명대학교'

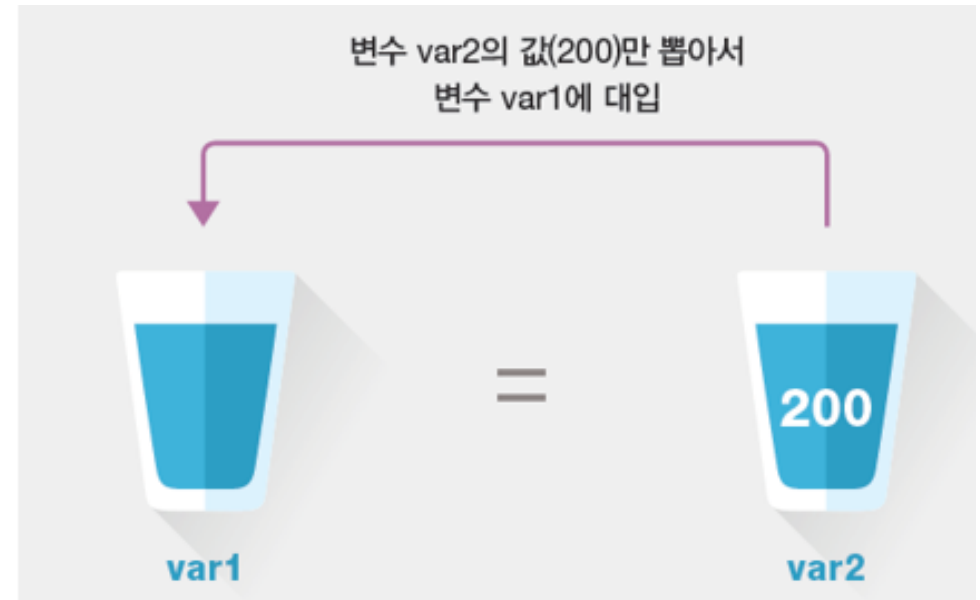
```
sss = '상명' * 3
```

변수 sss의 데이터 값은 '상명상명상명'

변수의 활용

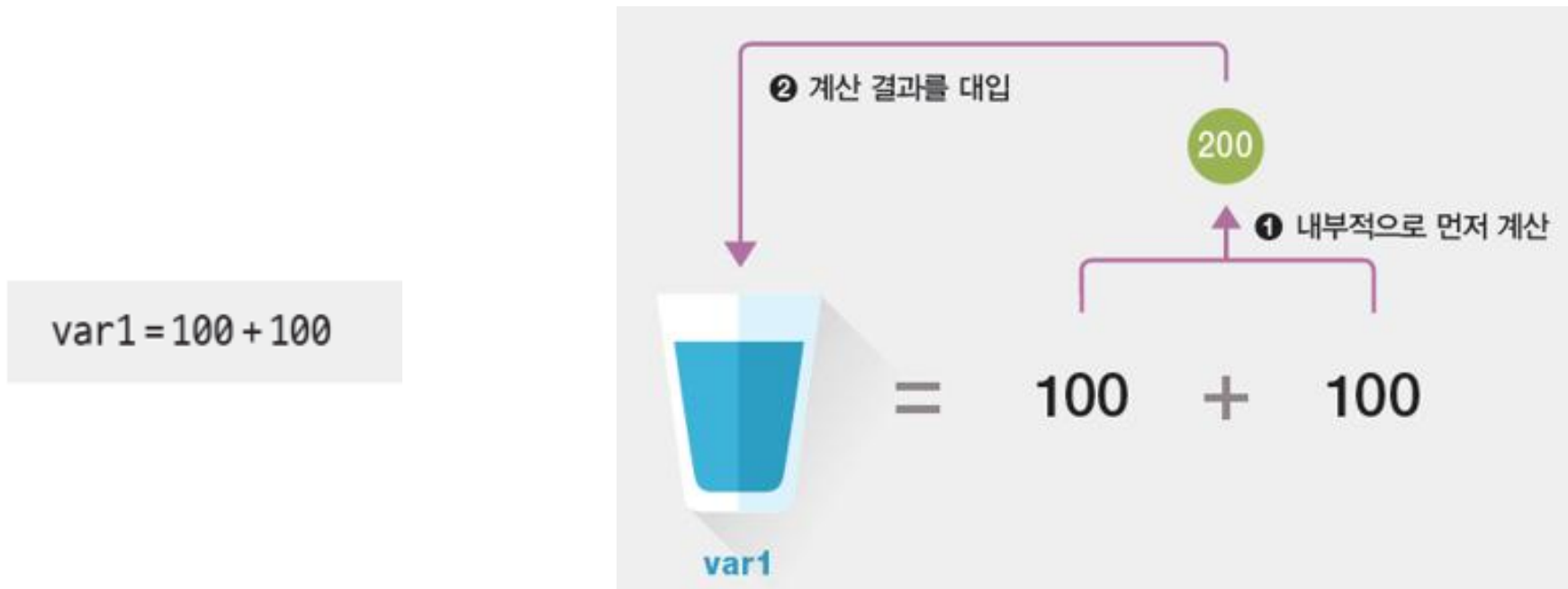
- 변수에 다른 변수의 데이터 값 저장하기
 - 변수 1인 var1과 변수 2인 var2가 있는 경우
 - 값을 저장할 때는 "=" (대입연산자) 사용
 - "var1 = var2"의 의미는
 - ✓ var2의 데이터 값을 var1에 저장
 - ✓ var2는 값을 가지고 있어야 함

```
var2 = 200  
var1 = var2
```



변수의 활용

- 계산을 수행한 후 그 결과 값을 변수의 데이터 값으로 저장하기
 - 숫자끼리의 계산 결과를 변수에 대입하는 경우



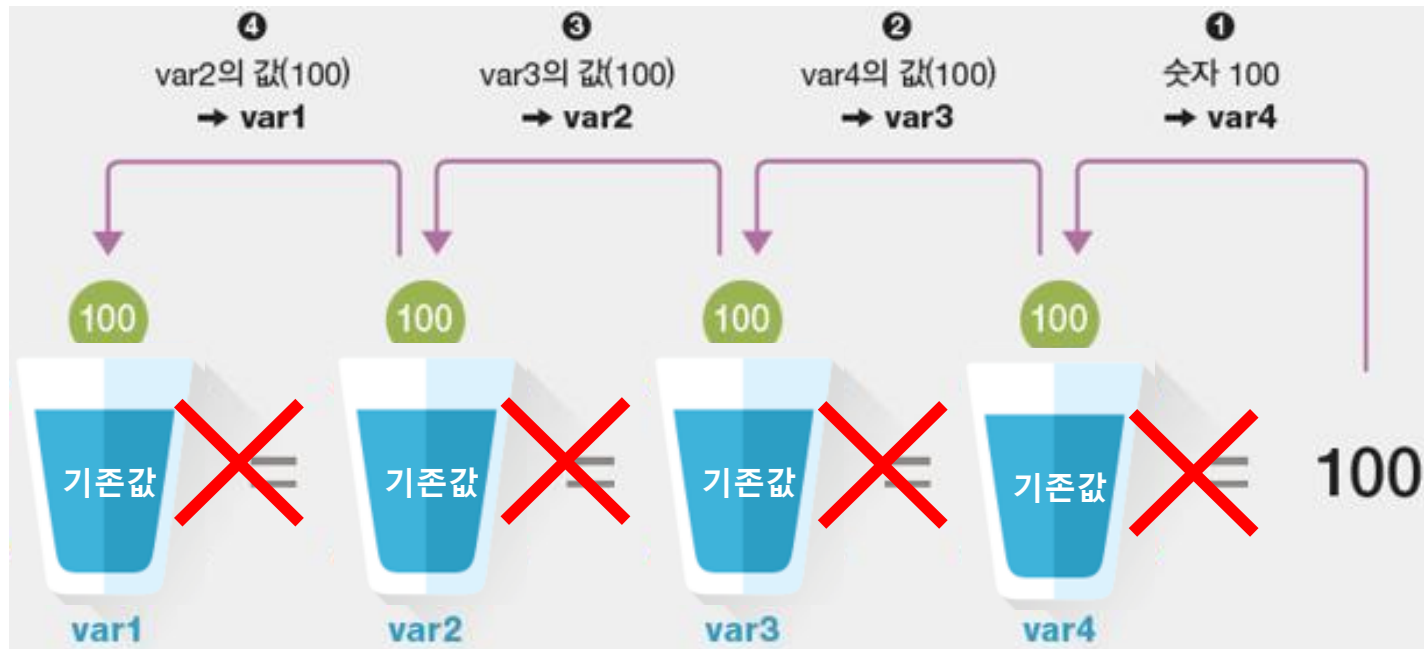
변수의 활용

- ▶ 연속된 값을 변수의 값으로 저장하기

```
var1=var2=var3=var4=100
```



```
var4=100  
var3=var4  
var2=var3  
var1=var2
```

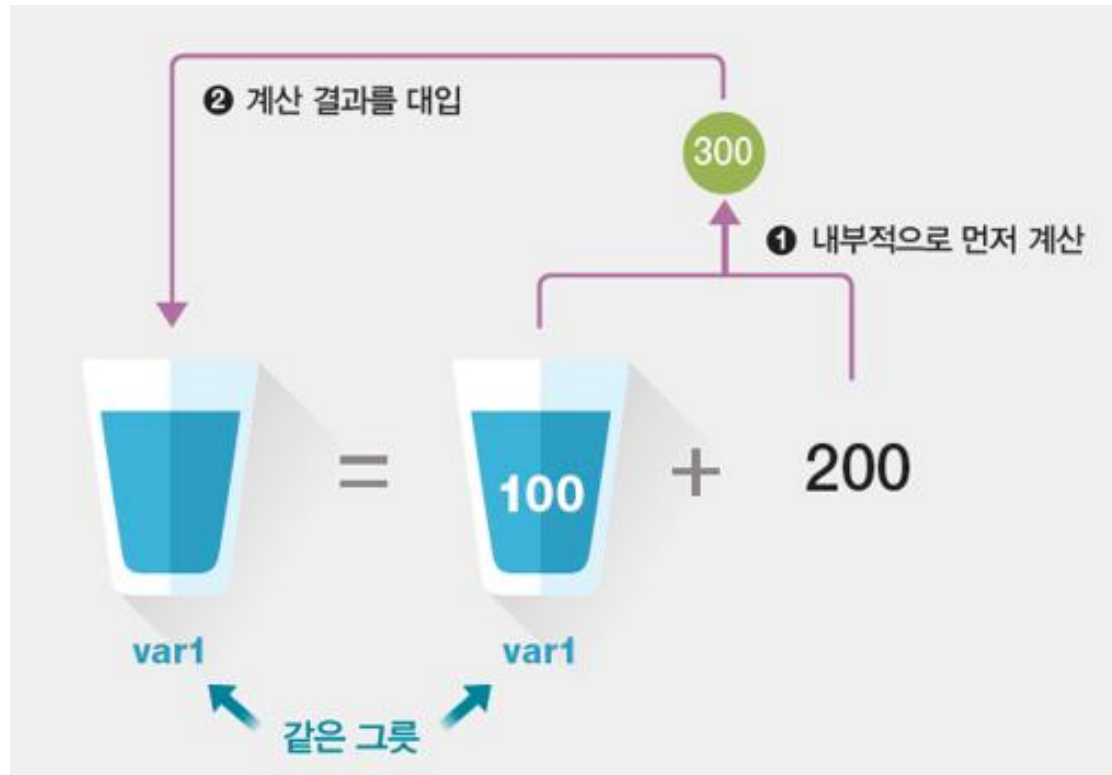


변수의 활용

- ▶ 변수 자신의 데이터 값에 계산을 수행한 수 결과 값을 새로 저장하기

`var 1=100`

`var1=var1+200`



변수의 활용

■ 파이썬(python)에서 변수 사용

- ▶ 변수 선언
- ▶ 변수를 사용하여 사칙 연산을 수행하고 결과 출력

▶ 출력물

```
a=10; b=20
```

```
print(a+b, a-b, a*b, a/b)
```

```
30 -10 200 0.5
```




리스트의 개요

■ 파이썬(python)에서 리스트 생성 방법

- 선언 방법

```
리스트이름 = [값1, 값2, 값3, ...]
```

```
aa = [ 10, 20, 30, 40 ]
```

- 사용 예

- aa[0], aa[1], aa[2], aa[3]으로 표현

- aa[0]에 저장된 데이터 값은 10

- 항목이 4개인 리스트인 경우 인덱스 번호는 1~4가 아닌 0~3을 사용

- 리스트 안에 저장되는 데이터 값은 동일한 자료형이 아니어도 되나 자료의 구조화를 위해서는 동일한 유형의 자료형을 권장함 Mylist=[1, 'abc', 2.0, '가나다', True]

파이선 리스트의 적용

■ 파이선(python)에서 리스트 사용 방법

- 리스트이름[시작:끝]로 리스트의 데이터 값 접근 가능
 - 끝으로 지정된 값은 (끝-1)에 해당하는 인덱스 번호를 의미
 - 시작 인덱스 번호 또는 끝 인덱스 번호 생략 가능
 - ❖ 시작이 생략된 경우는 0을 의미/ 끝이 생략된 경우는 마지막 데이터 값까지 포함

`mylist = [10, 20, 30, 40, 50]`

`mylist[0:5]`

`mylist[1:3]`

`mylist[2:]`

`mylist[:4]`

인덱스 번호 0~(5-1)인 4에 해당하는 [10, 20, 30, 40, 50]

인덱스 번호 1~(3-1)인 2에 해당하는 [20, 30]

인덱스 번호 2~마지막 데이터 값 [30, 40, 50]

인덱스 번호 0~(4-1)인 3에 해당하는 [10, 20, 30, 40]

파이선 리스트의 적용

■ 파이선(python)에서 리스트 활용 예

- 리스트끼리 더하기, 곱하기 연산 가능
 - 더하기는 서로 다른 리스트의 연결
 - 곱하기는 자신의 데이터 값 반복 확장

```
aa = [10, 20, 30]
```

```
bb = [40, 50, 60]
```

```
aa + bb
```

```
aa * 3
```

```
[10, 20, 30, 40, 50, 60]
```

```
[10, 20, 30, 10, 20, 30, 10, 20, 30]
```

파이선 리스트의 특징

■ 리스트

- ▶ 데이터 변경 가능(리스트 생성 후 추가/수정/삭제 가능)

```
>>> a = [1, 2, 3, 4]
```

```
>>> a
```

```
[1, 2, 3, 4]
```

```
>>> a[1]
```

```
2
```

[] 괄호가 리스트를 의미함을
이해해 주세요.

index 번호가 0부터 시작됨을
이해해 주세요.

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>>> a[0:5]
```

```
[1, 2, 3, 4, 5]
```

```
>>> a[5:]
```

```
[6, 7, 8, 9, 10]
```

```
>>> a[:3]
```

```
[1, 2, 3]
```

전체 중 일부를 출력하면 마지막은 n-
1까지 출력됨을 이해해 주세요.

: 붙었을 때 처음부터
혹은 끝까지 라는
의미를 이해해 주세요.

파이선 리스트의 특징

▶ 리스트의 결합

```
>>> a = [2, 4, 6, 8]
>>> b = [10, 12, 14]
>>> a + b
[2, 4, 6, 8, 10, 12, 14]
```

+ 가 연결의 의미가
있음을 이해해 주세요.

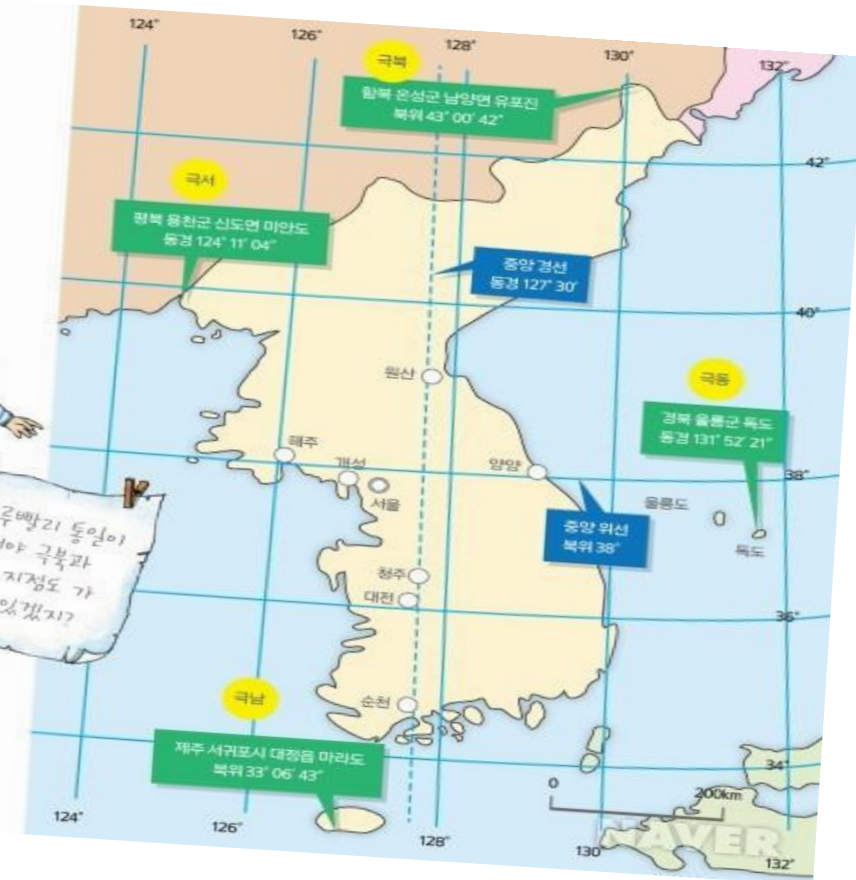
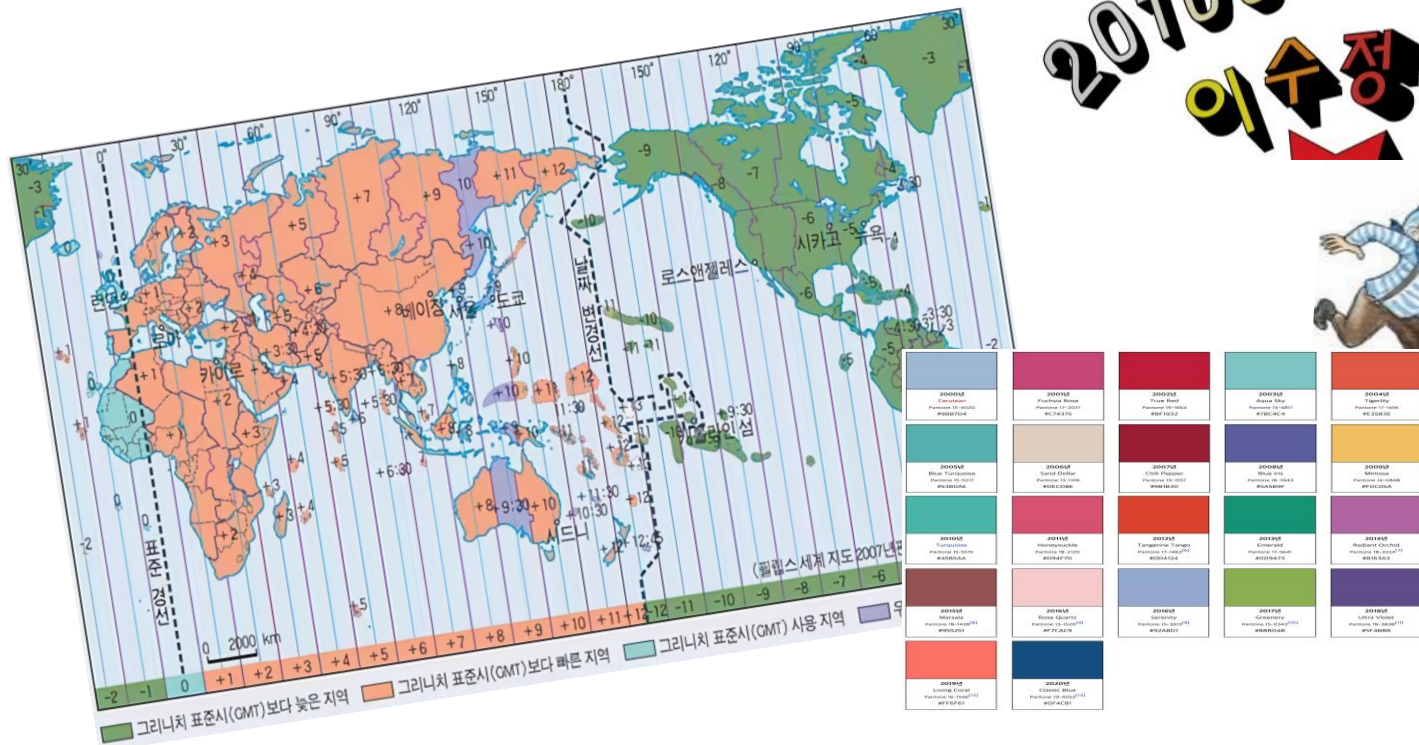
▶ 데이터 변경

```
>>> a = [2, 4, 5, 8]
>>> a[2] = 6
>>> a
[2, 4, 6, 8]
>>> a[3] = 10
>>> a
[2, 4, 6, 10]
```

리스트는 데이터가 변경될
수 있다는 걸 이해해 주세요.

리스트의 종류 중 튜플의 개요

- ▶ 자료값이 변경되면 혼란을 빚는 자료



리스트의 종류 중 튜플의 개요

■ 튜플의 개요

- 튜플은 리스트와 유사하나 내용 변경이 불가능함
- 원본 데이터를 그대로 유지해야 하는 경우 유용하게 사용됨



리스트의 종류 중 튜플의 특징

■ 튜플

- 데이터 변경 불가능(튜플 생성 후 추가/수정/삭제 불가능)
- 원본을 그대로 유지해야 하는 데이터의 경우 유용

```
>>> a = (1, 2, 3) # ()를 사용
>>> a
(1, 2, 3)
```

() 괄호가 튜플을 의미함을
이해해 주세요.

```
>>> a = (1, 2, 3, 4, 5, 6)
>>> a[:3]
(1, 2, 3)
>>> a[4:6]
(5, 6)
```

리스트의 종류 중 튜플의 특징

▶ 튜플의 결합

```
>>> a = (1, 2, 3)
>>> b = (4, 5, 6)
>>> c = a + b
>>> c
(1, 2, 3, 4, 5, 6)
```

▶ 데이터의 변경

```
>>> a = (1, 2, 3)
>>> a[0]
1
>>> a[0] = 0
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    a[0] = 0
TypeError: 'tuple' object does not support item assignment
```

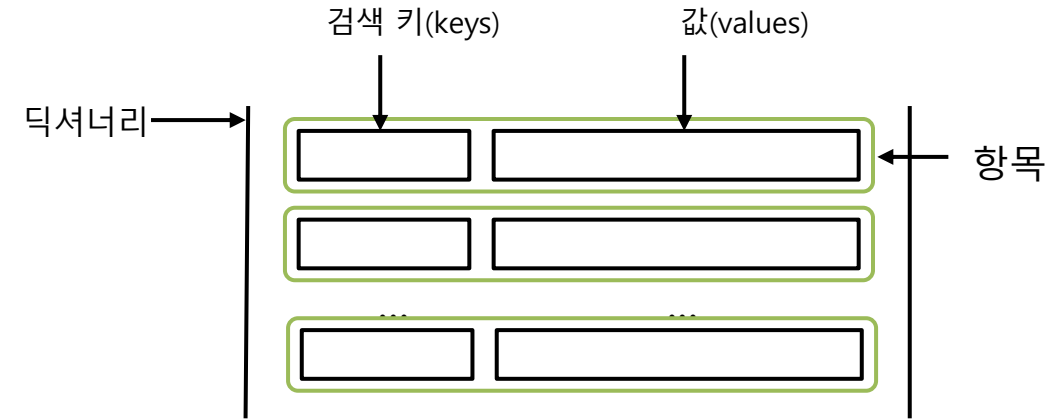
튜플은 데이터가 변경될 수 없으므로 오류가 생겼다는 걸 이해해 주세요.

리스트의 종류 중 딕셔너리의 개요

■ 딕셔너리

- 키(key)와 값(value)으로 구성되며 변경이 불가능

{ } 괄호가 딕셔너리를
의미함을 이해해 주세요.



```
>>> dic = {}  
>>> dic[ '학번' ] = '20250001'  
>>> dic[ '이름' ] = 'ANNE'  
>>> dic[ '성별' ] = '여자'  
>>> dic[ '학번' ]  
    '20250001'  
>>> dic  
{ '학번': '20250001', '이름': 'ANNE', '성별': '여자' }
```

키값과 values값을 삽입해서
딕셔너리를 만드는 과정임을
이해해 주세요.

리스트의 종류 중 딕셔너리의 특징

▶ 키와 값

```
>>> dic.keys()
```

키값을 출력하는 명령어임을
이해해 주세요.

```
dict_keys([ '학번', '이름', '성별' ])
```

```
>>> dic.values()
```

values값을 출력하는
명령어임을 이해해 주세요.

```
dict_values([ '20250001', 'ANNE', '여자' ])
```

▶ items()

```
>>> dic.items()
```

```
dict_items([( '학번', '20250001'), ( '이름', 'ANNE'), ( '성별', '여자') ])
```

리스트의 종류 중 딕셔너리의 특징

▶ in 연산자

```
>>> '학번' in dic.keys()
True
>>> 'Diana' in dic.values()
False
```

출력 결과가 True와 False로
나옴을 이해해 주세요.

▶ pop() 함수

```
>>> dic.pop('학번')
'20250001'
>>> dic
{'이름': 'ANNE', '성별': '여자'}
```

pop에 해당하는 키값을
찾아서 values 값을 삭제할 수
있음을 이해해 주세요.

변수와 리스트

- ▶ 변수의 종류
- ▶ 리스트의 특징
- ▶ 튜플의 특징
- ▶ 딕셔너리의 특징

