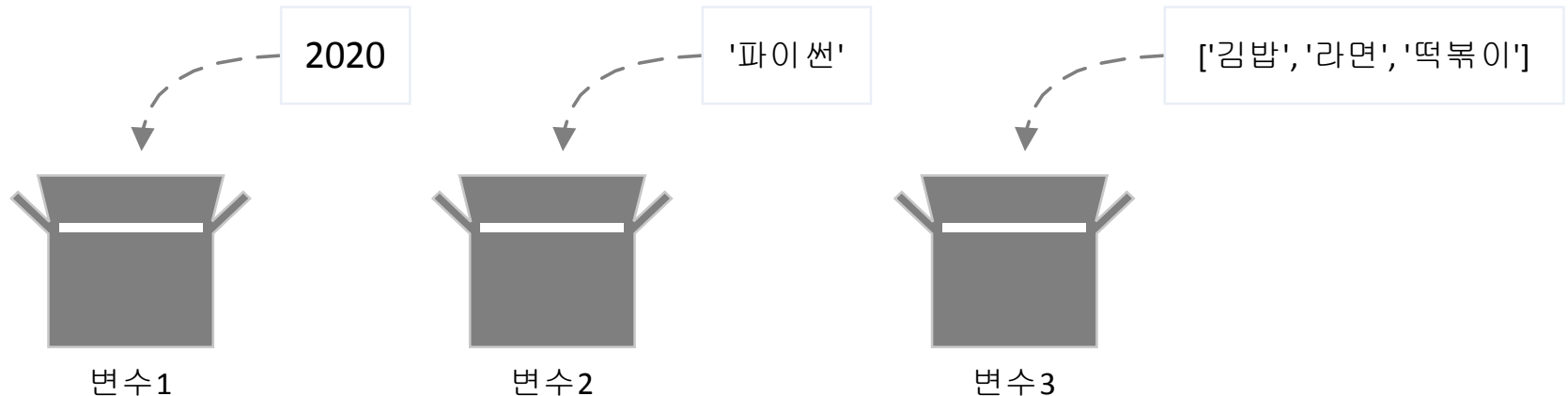


변수 및 데이터 형식

변수 (Variable)

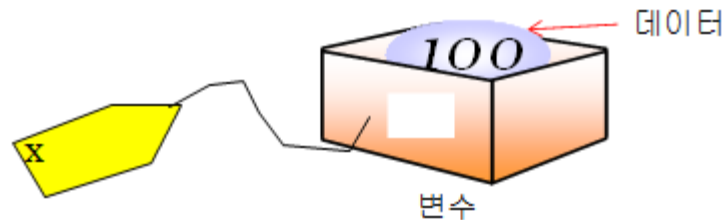
- ❖ 변수는 데이터를 저장하기 위한 메모리 공간
 - 데이터를 담는 상자로 생각할 수 있음
 - 수, 텍스트, 목록, 이미지 데이터 등을 담을 수 있음
 - 변수는 컴퓨터 메모리 공간에 생성
 - 해당 변수이름에 대한 고유한 저장공간이 확보됨



변수 생성 및 데이터 저장 (1)

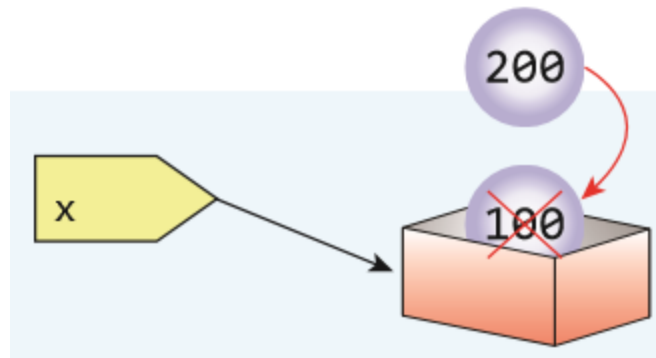
- ❖ 대입 연산자 ('=' 기호)는 변수에 데이터를 저장하라는 명령
 - 대입 연산자 오른쪽의 데이터값을 왼쪽의 변수에 할당하여 저장

```
>>> x = 100
```



- ❖ 생성된 변수에는 얼마든지 다른 값을 저장 가능

```
>>> x = 100
>>> x = 200
>>> print(x)
200
```



변수 생성 및 데이터 저장 (2)

❖ 한번에 여러 변수를 생성하여 각각의 값을 저장

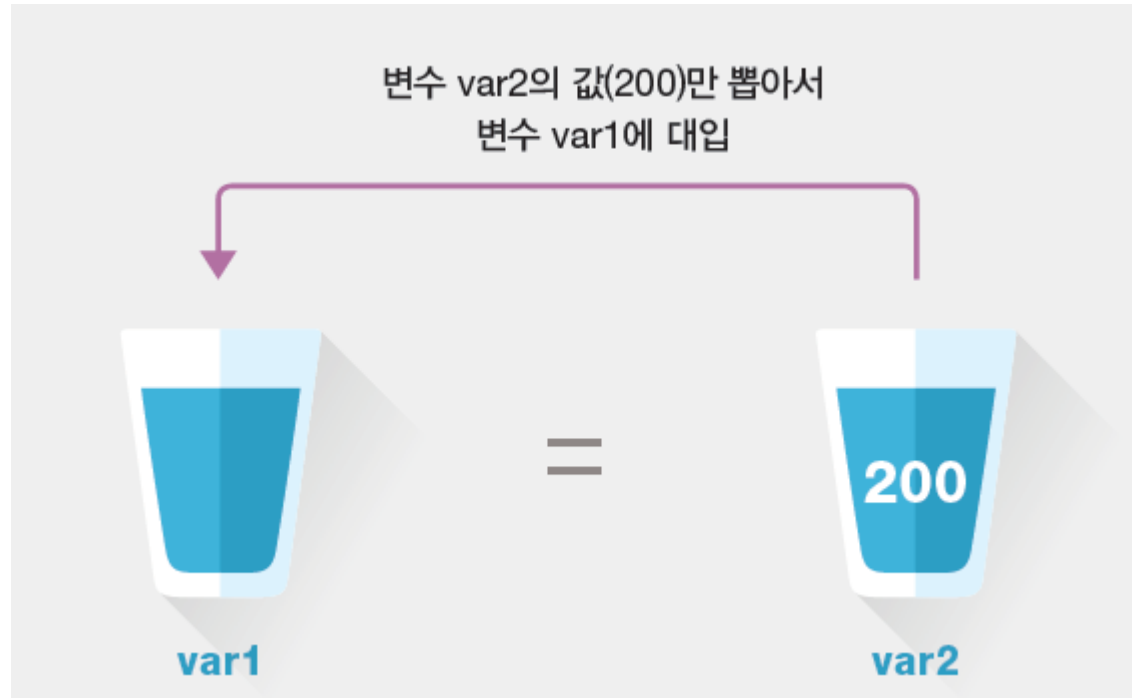
■ 변수의 갯수와 값의 갯수가 일치해야 함

```
>>> number_1, number_2 = 511
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    number_1, number_2 = 511
TypeError: 'int' object is not iterable
>>> number_1, number_2 = 2, 4, 5
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    number_1, number_2 = 2, 4, 5
ValueError: too many values to unpack (expected 2)
>>> number_1, number_2 = 6, 9
>>> number_1
6
>>> number_2
9
```

변수를 이용한 계산 (1)

❖ 변수의 값을 변수에 넣기

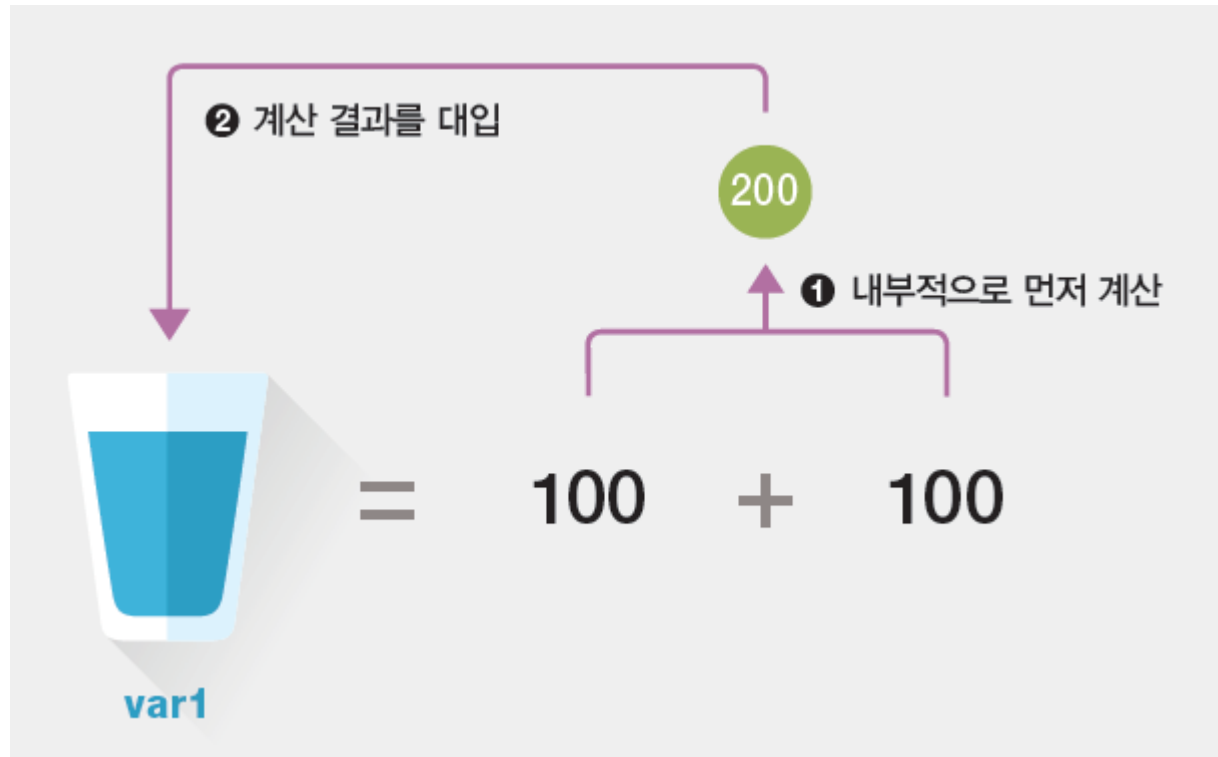
```
var2 = 200  
var1 = var2
```



변수를 이용한 계산 (2)

❖ 계산 결과를 변수에 넣기

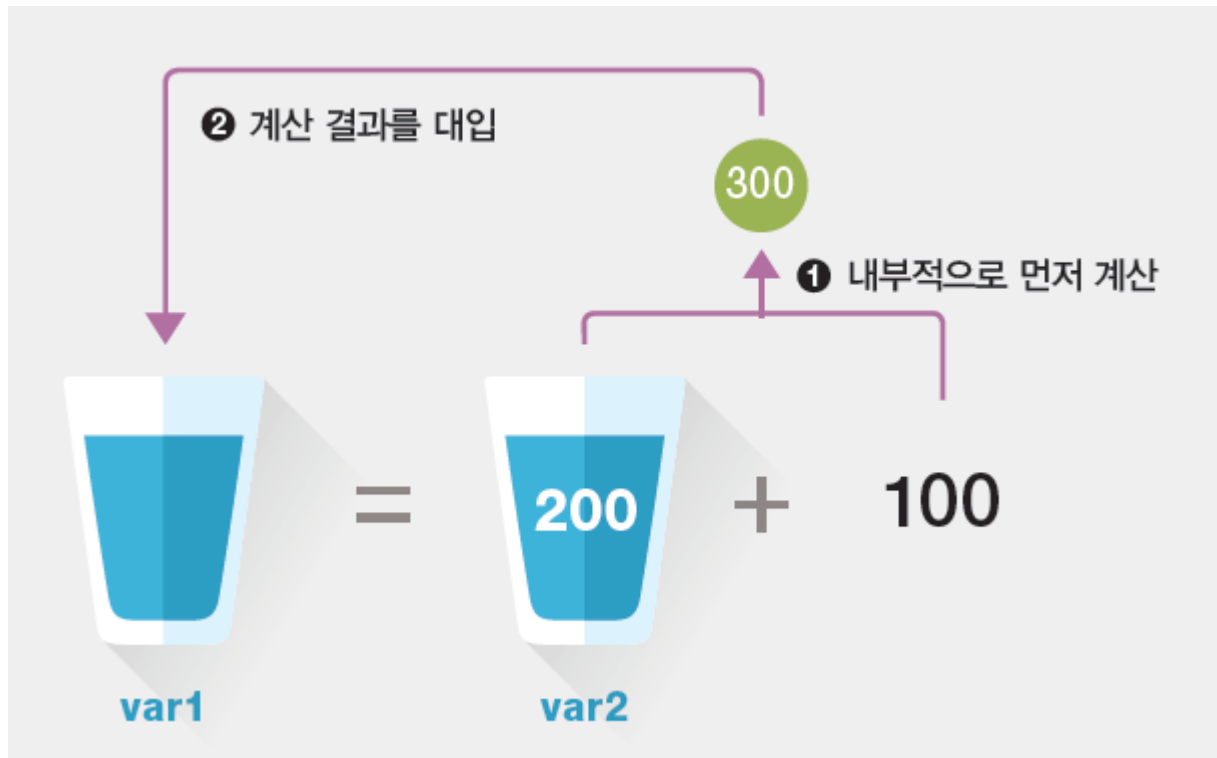
```
var1 = 100 + 100
```



변수를 이용한 계산 (3)

❖ 변수와 숫자의 연산을 변수에 넣기

```
var1 = var2 + 100
```



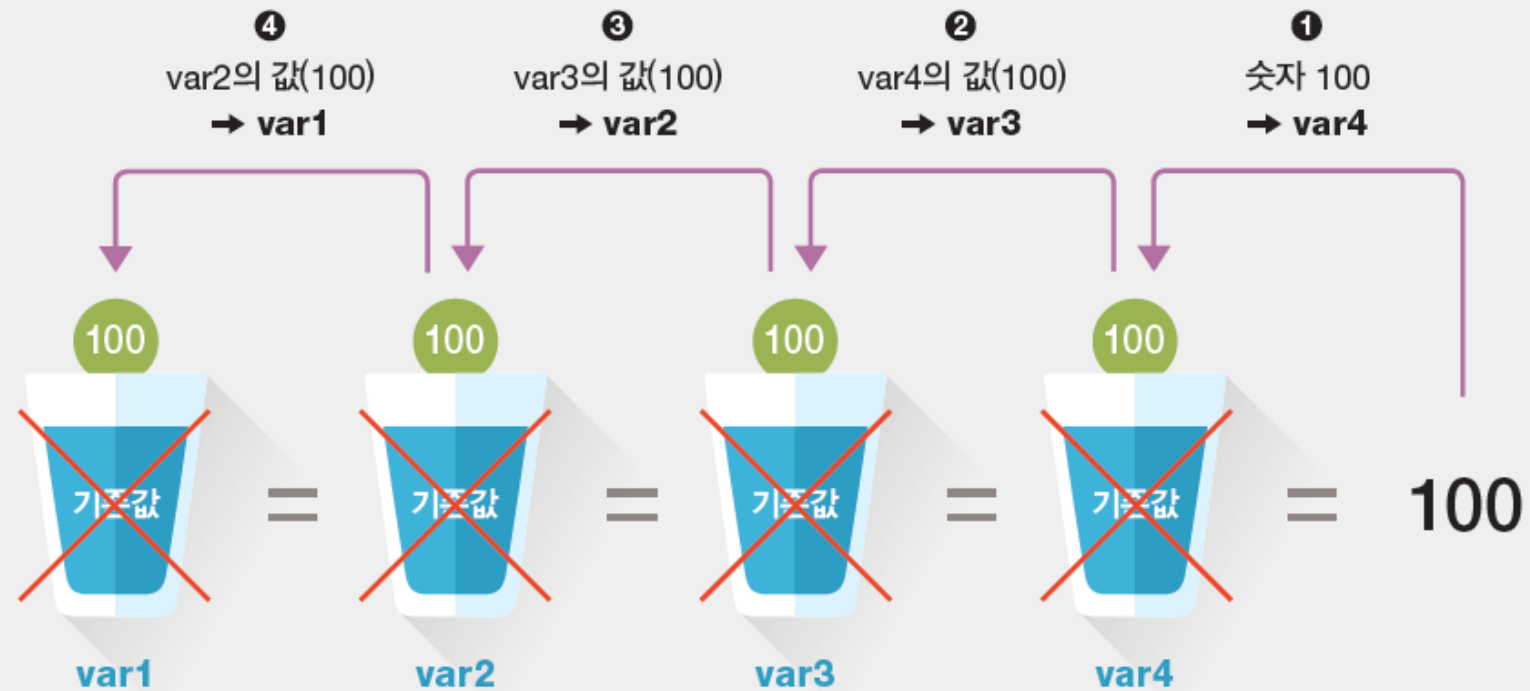
변수를 이용한 계산 (4)

❖ 연속된 값을 대입하는 방식

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

← 동일함 →

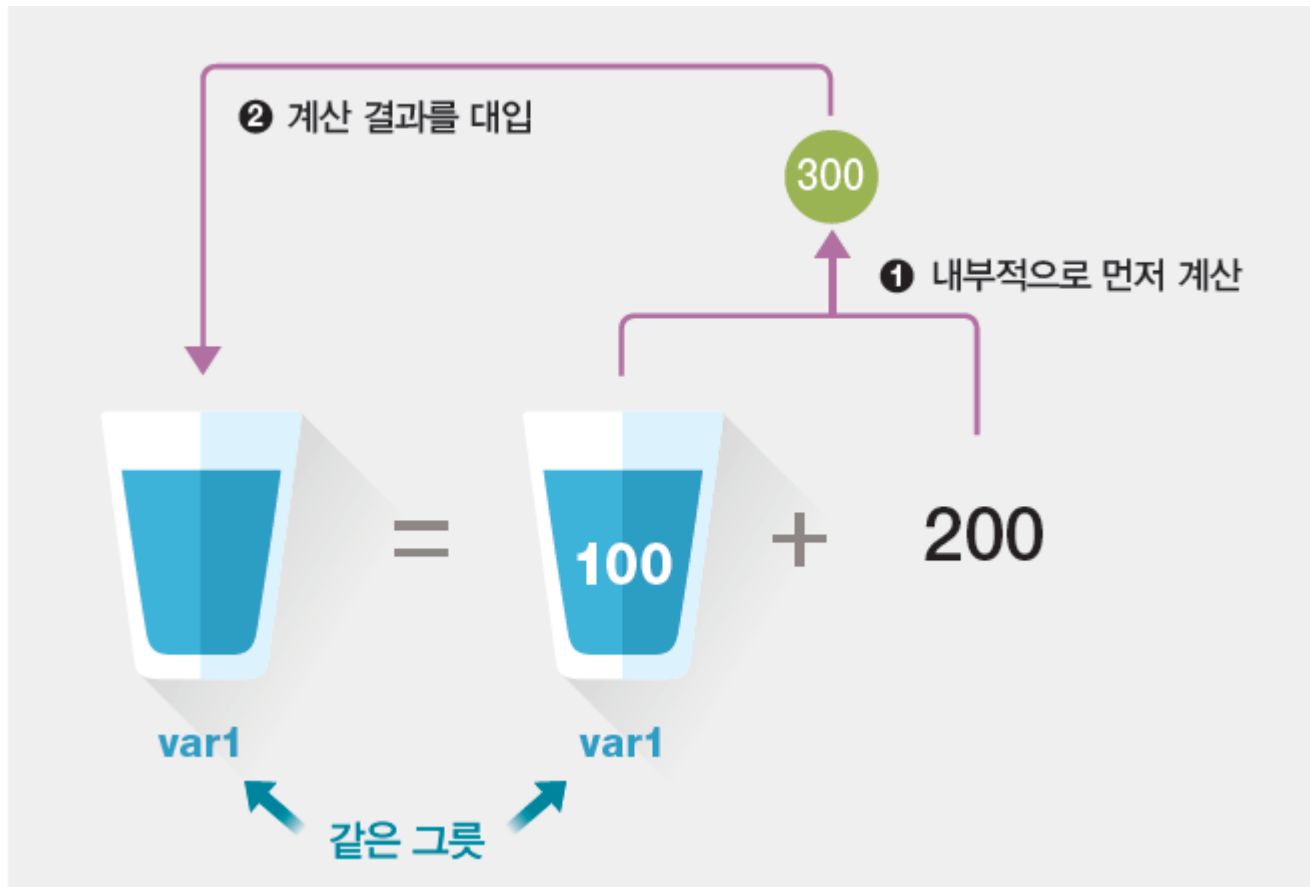
```
var4=100  
var3=var4  
var2=var3  
var1=var2
```



변수를 이용한 계산 (5)

- ❖ 자신의 값에 계산 결과를 대입하는 방식

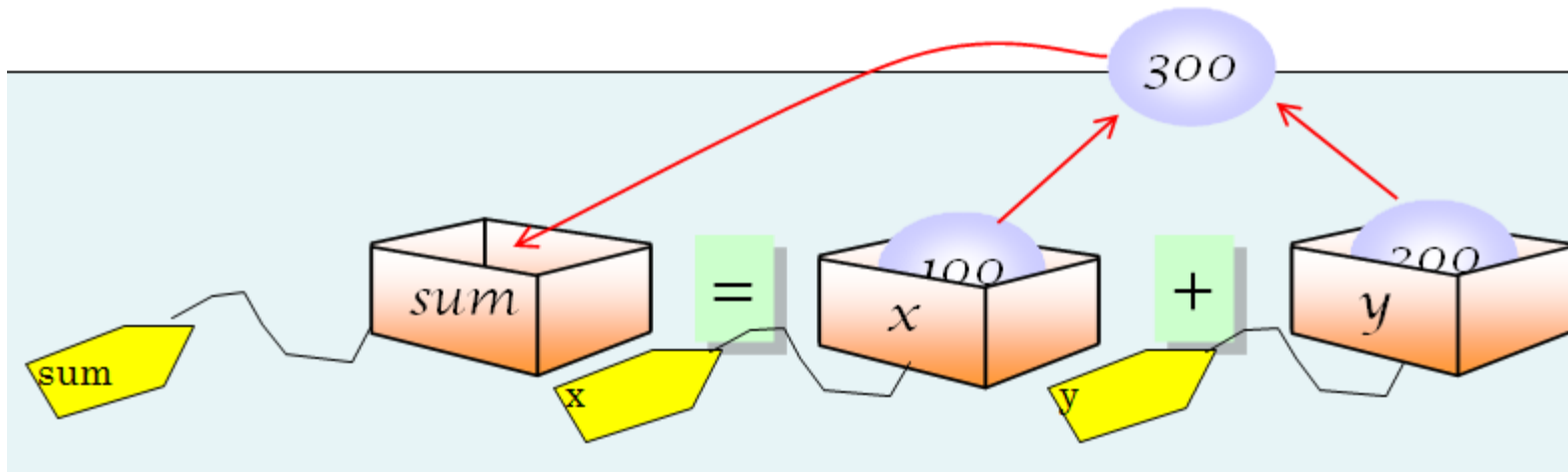
```
var1 = var1 + 200
```



변수를 이용한 계산 (6)

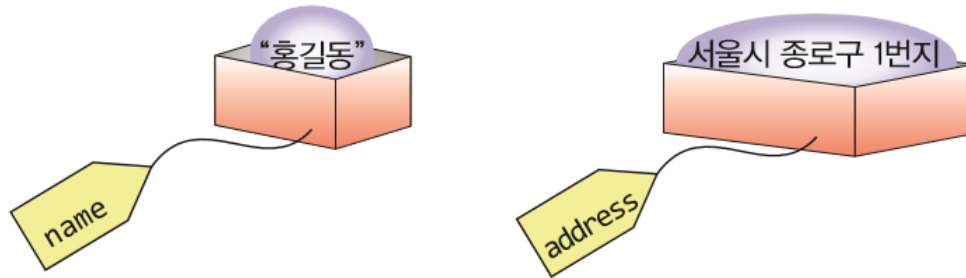
❖ 변수와 변수의 연산을 변수에 넣기

```
>>> x = 100  
>>> y = 200  
>>> sum = x + y  
>>> print(sum)  
300
```



변수에 문자열도 저장 가능

```
>>> name = "홍길동"  
>>> address = "서울시 종로구 1번지"
```



```
>>> print(name)  
홍길동  
>>> print(address)  
서울시 종로구 1번지
```

변수 이름 (식별자) 만드는 규칙

- ❖ 영문자, 숫자, 밑줄(_)로만 이루어짐
 - 영문자 및 밑줄(_)로만 시작해야 되며 숫자로 시작하면 안됨
 - 영문 대문자와 소문자를 구분
 - 이름 중간에 공백이 들어가면 안되므로 단어를 구분하기 위한 방법
 - 밑줄(_) 사용: 예) `social_security_number`
 - 낙타체 사용: 예) `socialSecurityNumber`
- ❖ 파이썬 지정단어(*Keyword or Reserved word*)들은 사용 불가
 - cf) 지정단어 목록
- ❖ 위의 규칙들을 위배할 경우 구문오류(*Syntax Error*)

파이썬 지정단어

- ❖ keyword를 import 하여 확인 가능
 - keyword.kwlist 명령어는 파이썬의 지정단어들을 나열해줌

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert',
'break', 'class', 'continue', 'def', 'del',
'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is',
'lambda', 'nonlocal', 'not', 'or', 'pass',
'raise', 'return', 'try', 'while', 'with',
'yield']
```

의미있는 변수명을 사용

- ❖ 해당 변수의 역할에 맞게 이름이 지어져야 프로그램을 검토할 때나 협업 시 타인들과 공유할 때 도움이 됨

예) 3개 시험성적의 총합과 평균을 구할 것

수학: 26점

영어: 54점

역사: 96점

```
a = 26
b = 54
c = 96
d = a + b + c
f = d / 3
```

```
math = 26
english = 54
history = 96
sum = math + english + history
average = sum / 3
```

변수의 데이터 형식(자료형)

- ❖ 파이썬에서는 변수의 데이터 형식 선언이 불필요
 - 변수에 값을 저장하는 순간 변수의 데이터 형식이 결정됨
 - 따라서 동일 변수에 어떤 종류의 자료도 저장 가능

```
>>> x = 10
>>> print("x =", x)
x = 10
>>> x = 3.14
>>> print("x =", x)
x = 3.14
>>> x = "Hello World!"
>>> print("x =", x)
x = Hello World!
```

사용가능한 데이터 형식

❖ 정수형 (*int*)

- 소수점이 없는 데이터

❖ 실수형 (*float*)

- 소수점이 있는 데이터

❖ Boolean형 (*bool*)

- 참(**True**)이나 거짓(**False**)만 저장

❖ 문자열형 (*str*)

- 값의 시작과 끝에 큰따옴표(" ")나 작은따옴표(' ')를 삽입
- 큰따옴표(")로 시작했다가 작은따옴표(')로 끝내면 *Syntax Error*

자료형을 알려주는 함수: `type()`

❖ 해당 변수 또는 값의 자료형을 확인하고자 할 때 사용

■ 사용형식: `type(argument)`

```
>>> n = 3
>>> type(n)
<class 'int'>
>>> n = 4.0
>>> type(n)
<class 'float'>
>>> type(10)
<class 'int'>
>>> type(10.0)
<class 'float'>
```

■ 자료형 확인이 필요한 상황

- 프로그램에서 변수에 직접 숫자를 대입하는 경우는 정수형 또는 실수형으로 인식
- 사용자들로부터 키보드를 통해 변수에 숫자를 입력 받는 경우는 문자열형으로 인식

자료형 강제 변환 (1)

❖ 데이터 형식을 인위적으로 변환 가능

- `variable_name = int(variable_name)`
 - `variable_name`에 해당하는 변수를 정수(*int*)형으로 강제 형 변환
- `variable_name = float(variable_name)`
 - `variable_name`에 해당하는 변수를 실수(*float*)형으로 강제 형 변환
- `variable_name = str(variable_name)`
 - `variable_name`에 해당하는 변수를 문자열(*str*)형으로 강제 형 변환

```
>>> n = 7
>>> n = float(n)
>>> type(n)
<class 'float'>
>>> n
7.0
```

```
>>> n = input('enter the number : ')
enter the number : 7
>>> n = int(n)
>>> type(n)
<class 'int'>
>>> n
7
```

자료형 강제 변환 (2)

```
>>> print(100+200)
```

```
300
```

```
>>> print("100"+"200")
```

```
100200
```

```
>>> t = input("정수를 입력하시오: ")
```

```
정수를 입력하시오: 100
```

```
>>> x = int(t)
```

```
>>> t = input("정수를 입력하시오: ")
```

```
정수를 입력하시오: 200
```

```
>>> y = int(t)
```

```
>>> print(x+y)
```

```
300
```

문자열형

- ❖ 컴퓨터에게는 숫자가 중요하지만 인간에게는 텍스트(**text**)가 중요
 - 따라서 컴퓨터를 이용한 텍스트의 처리도 무척 중요
- ❖ 문자열 (**string**)은 단어나 문장 등 문자들의 나열 또는 집합
 - 파이썬은 문자열 처리작업이 매우 용이하게 개발된 언어임
- ❖ 4가지 사용형식

' '	string 내에 " " 를 포함해야 하는 경우
" "	string 내에 ' ' 또는 ' 를 포함해야 하는 경우
''' '''	여러 문장을 사용하고, string 내에 " "를 포함해야 하는 경우
""" """	여러 문장을 사용하고, string 내에 ' ' 또는 ' 를 포함해야 하는 경우

문자열형 예제 (1)

```
>>> "Hello"
'Hello'
>>> msg = "Hello"
>>> msg
'Hello'
>>> print(msg)
Hello
```

*print()를 사용하여 출력하지 않는 경우에는
' '(작은따옴표)로 묶여서 출력됨*

```
>>> a = 'Hello, World.'
>>> a
'Hello, World.'
>>> b = "안녕하세요."
>>> b
'안녕하세요.'
>>> c = '''어서와 파이썬은 처음이지?'''
>>> c
'어서와 파이썬은 처음이지?'
>>> d = """Welcome to Python."""
>>> d
'Welcome to Python.'
>>> type(d)
<class 'str'>
```

문자열형 예제 (2)

→ "He is a smart boy." my teacher said

```
>>> print('He is a smart boy." my teacher said')
He is a smart boy." my teacher said
>>>
```

→ He's a smart and diligent boy.

```
>>> print("He's a smart and diligent boy.")
He's a smart and diligent boy.
>>>
```

→ print('He's a smart and diligent boy.')

```
>>> print('He's a smart and diligent boy.')
SyntaxError: invalid syntax
>>> # " " 대신 ' ' 를 사용하는 경우 Syntax 에러 발생
```

```
>>> print('' "He is a smart boy. He is a diligent boy." my teacher
said.'')
```

```
"He is a smart boy. He is a diligent boy." my teacher said.
```

```
>>> print("""He's a smart boy. He's a diligent boy. So I like him.""")
He's a smart boy. He's a diligent boy. So I like him.
```

특수문자열

- cf) 역슬래시 기호 대신 원화표시 기호 사용가능

특수 문자열	의미
\n	줄 바꿈 문자
\t	탭 문자
\\	역슬래시 자체
\"	큰따옴표 자체
\'	작은따옴표 자체

```
>>> print("말 한마디로\n천 냥 빚을 갚는다")
말 한마디로
천 냥 빚을 갚는다
```

```
>>> m='doesn't'
SyntaxError: invalid syntax
```

```
>>> m='doesn\'t'
>>> print(m)
doesn't
```

```
>>> m="doesn't"
>>> print(m)
doesn't
```

```
1 print("\n줄바꿈\n연습 ")
2 print("\t탭키\t연습")
3 print("글자가 \"강조\"되는 효과1")
4 print("글자가 \'강조\'되는 효과2")
5 print("\\\\\\\\ 역슬래쉬 세계 출력")
```

줄바꿈
연습

 탭키 연습
글자가 "강조"되는 효과1
글자가 '강조'되는 효과2
\\\\\\\\ 역슬래쉬 세계 출력

in 연산자 / str() 함수

❖ in 연산자

- 특정 문자열이 해당 문자열 안에 존재하는지를 확인

```
>>> a = 'Good Morning'
>>> 'Good' in a
True
>>> 'X' in a
False
>>> 'Evening' in a
False
```

❖ 숫자를 문자열로 변환하기 위해서는 str() 함수를 사용

```
a=100; b=100.123
str(a)+'1'; str(b)+'1'
```

출력 결과

```
'1001'
'100.1231'
```


문자열 병합 (*concatenation*)

❖ + 연산자 사용

```
>>> 'Hello ' + 'World!'  
'Hello World!'
```

```
>>> start = '=====  
>>> title = 'Python Program'  
>>> finish = '=====  
>>> print(start + title + finish)  
=====Python Program=====
```

❖ 문자열과 숫자는 합칠 수 없음

```
>>> print('나는 현재 ' + 21 + '살이다.')  
Traceback (most recent call last):  
File "<pyshell#1>", line 1, in <module>  
print('나는 현재 ' + 21 + '살이다.')  
TypeError: Can't convert 'int' object to str implicitly
```

```
>>> print('나는 현재 ' + str(21) + '살이다.')  
나는 현재 21살이다.
```

```
>>> print('원주율은 ' + str(3.14) + '입니다.')  
원주율은 3.14입니다.
```

문자열 반복(*iteration*)

❖ * 연산자 사용

```
>>> message = " Congratulations!"
>>> print(message*3)
Congratulations!Congratulations!Congratulations!
```

```
>>> print("="*50)
=====
```

```
>>> start = '=' * 10
>>> title = 'Python Program'
>>> finish = '=' * 10
>>> print(start + title + finish)
=====Python Program=====
```

문자열 *indexing*

❖ 문자열에서 특정 위치에 있는 문자를 추출하려면 대괄호 []에 첨자(*index*)번호를 입력해서 사용

- 사용형식: 문자열 변수명[*index*]
- *index*는 항상 0부터 시작
- 문자열내의 공백(*space*)에도 *index*가 부여됨

```
>>> a = 'Good Morning'
>>> a[0]
'G'
>>> a[8]
'n'
```

- 문자열에서 *index*를 이용한 변경을 허용하지 않음

```
>>> greeting = 'Hello, world!'
>>> greeting[0] = 'J'
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    greeting[0] = 'J'
TypeError: 'str' object does not support item assignment
```

- *index*로 음수를 입력하면 문자열의 끝에서부터 문자를 반환

문자열 *slicing*

❖ 문자열의 일부를 추출

- 사용형식: 문자열 변수명[*n:m*]
- *n*번째부터 *m*번째 사이의 부분, *n*은 포함하고, *m*은 뺀 부분을 지정
- 결과적으로 (*m-n*)길이의 문자(열) 생성

0	1	2	3	4	5	[6:10]				10	11
M	o	n	t	y		P	y	t	h	o	n
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
[-12:-7]											

```
>>> s = "Monty Python"
>>> print(s[6:10])
Pyth
```


[:n]	처음부터 n을 포함하지 않는 범위까지 지정해서 slice
[m:]	m번째부터 string의 끝까지 범위를 지정해서 slice
[m:문자열보다 큰 값]	m번째부터 string의 끝까지 범위를 지정해서 slice

```
>>> fruit = 'banana'
>>> fruit[:3]
'ban'
>>> fruit[3:]
'ana'
>>> fruit[3:999]
'ana'
```

문자열의 길이를 알려주는 함수: `len()`

- ❖ 사용형식: `len(문자열 변수명)`
 - 길이 측정시 공백(*space*)도 포함

I am a boy



10 글자

```
>>> a = 'Good Morning'
>>> len(a)
12
```

입력문함수 **input()**: 문자열 입력받기

❖ 2가지 형식

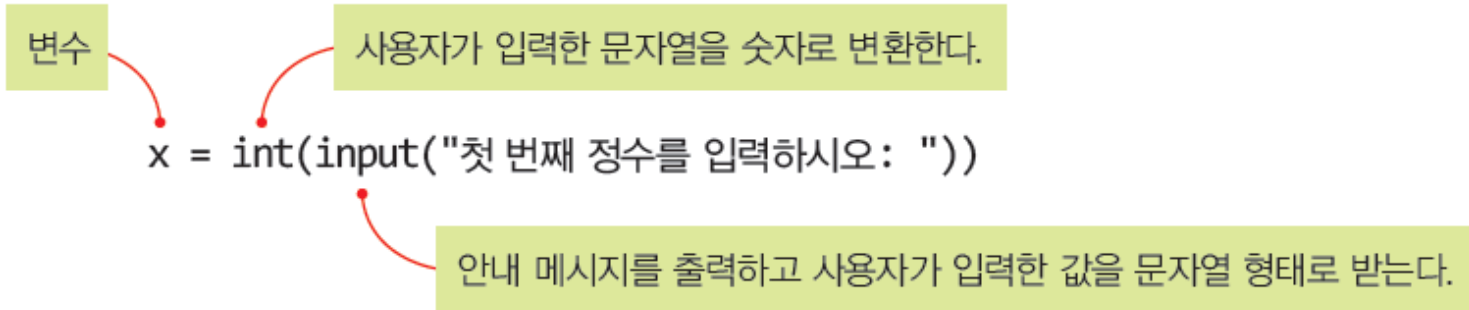
<code>variable_name=input()</code>	사용자로부터 입력을 받는다.
<code>variable_name=input('문자열')</code>	'문자열'에 해당하는 내용을 출력 후 사용자로부터 입력을 받는다.

```
>>> name = input()  
Gildong  
>>> name  
'Gildong'
```

```
>>> name = input('What is your first name? ')  
What is your first name? Gildong  
>>> name  
'Gildong'
```

입력문 함수 `input()`: 정수 입력받기

`input()` 사용법



```
>>> x = int(input("첫 번째 정수를 입력하시오: "))
첫 번째 정수를 입력하시오: 300
>>> y = int(input("두 번째 정수를 입력하시오: "))
두 번째 정수를 입력하시오: 400
>>> sum = x + y
>>> print(x, "과", y, "의 합은", sum, "입니다.")
100 과 200 의 합은 300 입니다.
```

출력문 함수 `print()`

❖ 3가지 형식

<code>print('문자열')</code>	'문자열'을 화면에 출력해준다.
<code>print(variable_name)</code>	변수 <code>variable_name</code> 에 해당하는 값을 화면에 출력해준다.
<code>print('문자열', variable_name)</code>	'문자열'과 변수 <code>variable_name</code> 에 해당하는 값을 연속해서 화면에 출력해준다.

❖ 서식을 지원하는 `print()` 함수 사용법

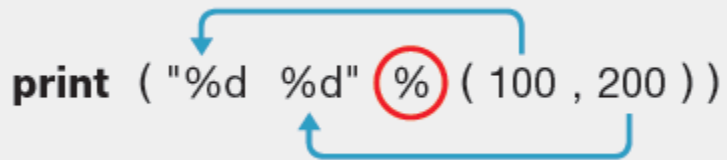
서식	값의 예	설명
<code>%d</code>	10, 100, 1234	정수
<code>%f</code>	0.5 , 1.0 , 3.14	실수(소수점이 붙은 수)
<code>%c</code>	"b", "한"	문자 한 글자
<code>%s</code>	"안녕", "abcdefg", "a"	한 글자 이상의 문자열

서식을 지원하는 `print()` 함수 (1)

`print("100")` → 100 (글자 “일영영”)
`print("%d" % 100)` → 100 (숫자)

`print("100+100")` → 100+100 (글자)
`print("%d" % (100+100))` → 200 (숫자)

❖ 서식의 개수와 `%` 뒤에 나오는 숫자(또는 문자)의 개수가 같아야 함

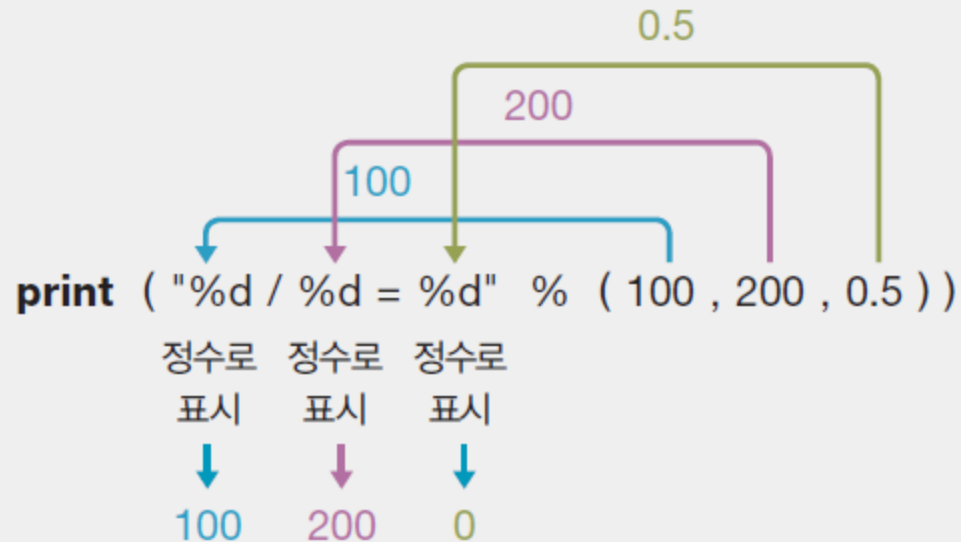


The diagram shows the code `print ("%d %d" % (100 , 200))`. The format string `"%d %d"` contains two `%d` specifiers. The argument tuple `(100 , 200)` contains two arguments. However, the third `%` symbol in the code is circled in red. Two blue arrows originate from this circled `%` and point to the two `%d` in the format string, indicating a mismatch or an error in the code structure.

`print ("%d %d" % (100 , 200))`

서식을 지원하는 `print()` 함수 (2)

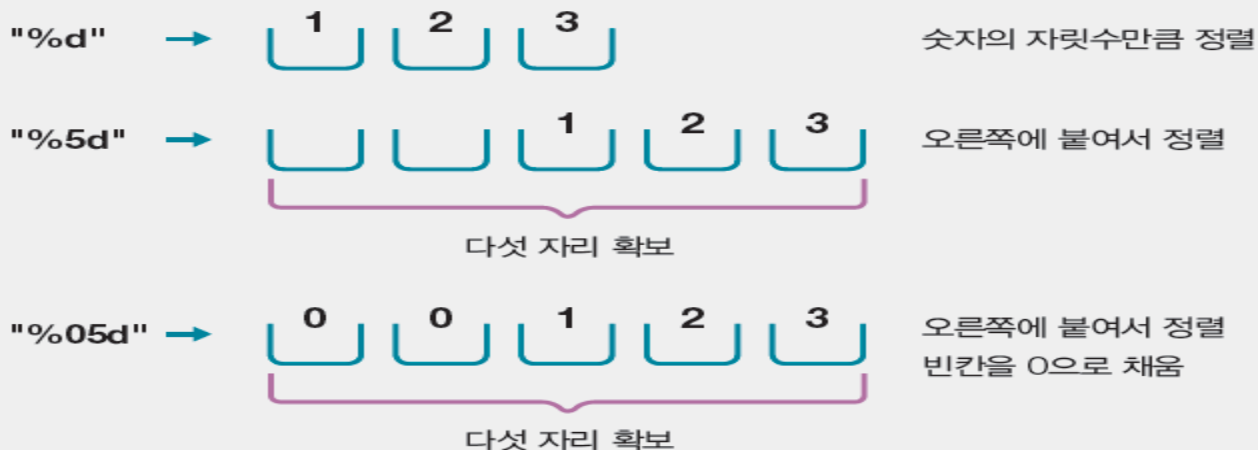
`print("%d / %d = %d" % (100, 200, 0.5))` → 100/200=0.5 가 아닌 100/200=0 이 나옴.
세 번째 숫자 0.5는 실수(소수점이 있는 수)이지만 보여주는 방식이 정수임



`print("%d / %d = %5.1f" % (100, 200, 0.5))` → 세 번째 %d 대신에 %f로 수정

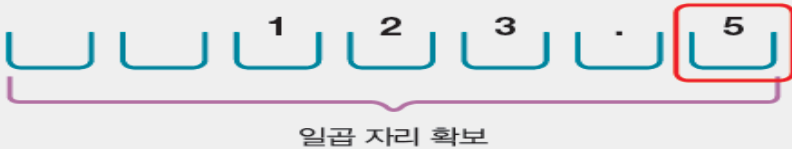
서식을 지원하는 print() 함수 (3)


1	print("%d" % 123)	123
2	print("%5d" % 123)	123
3	print("%05d" % 123)	00123
4		
5	print("%f" % 123.45)	123.450000
6	print("%7.1f" % 123.45)	123.5
7	print("%7.3f" % 123.45)	123.450
8		
9	print("%s" % "Python")	Python
10	print("%10s" % "Python")	Python




서식을 지원하는 print() 함수 (4)

"%f" →  소수점 아래 여섯 자리까지 무조건 출력

"%7.1f" →  일곱 자리 확보
소수점 아래 첫째 자리만 출력
소수점 아래 둘째 자리에서 반올림

"%7.3f" →  일곱 자리 확보
소수점 아래 셋째 자리까지 출력
오른쪽 빈칸은 0으로 채움

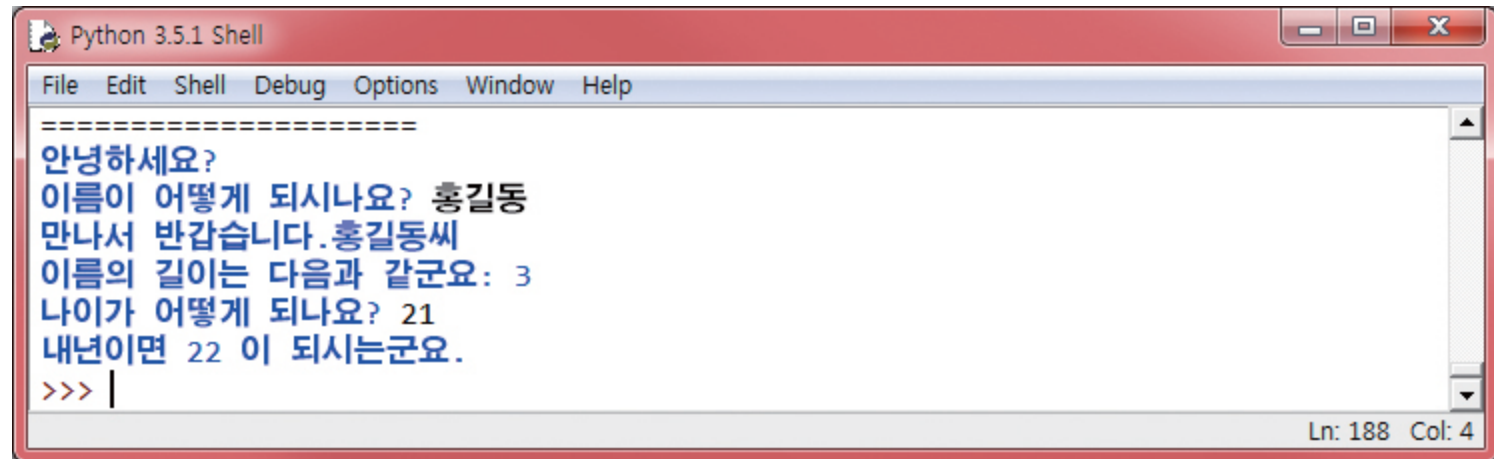
"%s" →  자릿수만큼 출력

"%10s" →  열 자리 확보
오른쪽 정렬

- 문자열에 변수값을 포함하여 출력

```
>>> price = 10000
>>> print("상품의 가격은 %s원입니다." % price)
상품의 가격은 10000원입니다.
```

문자열 예제 (1)



```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
=====
안녕하세요?
이름이 어떻게 되시나요? 홍길동
만나서 반갑습니다.홍길동씨
이름의 길이는 다음과 같군요: 3
나이가 어떻게 되나요? 21
내년이면 22 이 되시는군요.
>>> |
```


Ln: 188 Col: 4

```
print('안녕하세요?')
name = input('이름이 어떻게 되시나요? ')

print('만나서 반갑습니다.' + name + "씨")
print('이름의 길이는 다음과 같군요:', len(name))

age = int(input("나이가 어떻게 되나요? "))
print("내년이면", str(age+1), "이 되시는군요.")
```

문자열 예제 (2)



오늘의 연도를 입력하시오: 2016
오늘의 월을 입력하시오: 12
오늘의 일을 입력하시오: 25
오늘은 2016년 12월 25일입니다.

```
year = input("오늘의 연도를 입력하시오: ")  
month = input("오늘의 월을 입력하시오: ")  
date = input("오늘의 일을 입력하시오: ")  
print("오늘은", year+"년", month+"월", date+"일입니다.")
```

문자열 예제 (3)

경기장은 어디입니까?서울
이긴팀은 어디입니까삼성
진팀은 어디입니까?LG
우수선수는 누구입니까?홍길동
스코어는 몇대몇입니까?8:7

```
=====
오늘 서울 에서 야구 경기가 열렸습니다.
삼성 과 LG 은 치열한 공방전을 펼쳤습니다.
홍길동 이 맹활약을 하였습니다.
결국 삼성 가 LG 를 8:7 로 이겼습니다.
=====
```

```
# 사용자의 대답을 변수에 저장한다.
stadium = input("경기장은 어디입니까?")
winner = input("이긴팀은 어디입니까")
loser = input("진팀은 어디입니까?")
vip = input("우수선수는 누구입니까?")
score = input("스코어는 몇대몇입니까?")

# 변수와 문자열을 연결하여 기사를 작성한다.
print("")
print("=====")
print("오늘", stadium, "에서 야구 경기가 열렸습니다.")
print(winner, "과", loser, "은 치열한 공방전을 펼쳤습니다.")
print(vip, "이 맹활약을 하였습니다.")
print("결국", winner,"가", loser,"를 ", score,"로 이겼습니다.")
print("=====")
```