단국대학교 SW중심대학

창의적 사고와 코딩

변수 소개

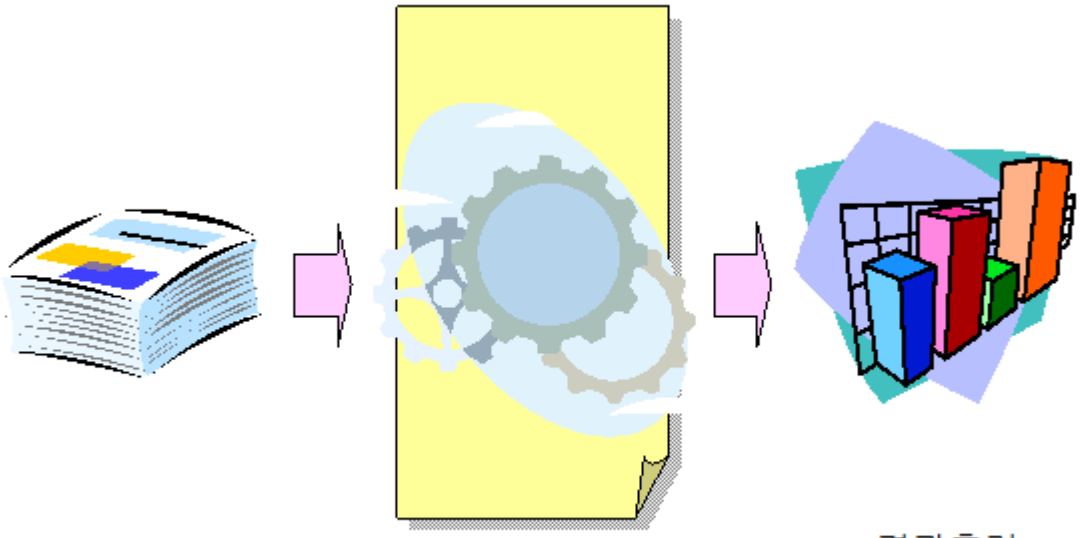
박 소 현

sohyunpark@dankook.ac.kr

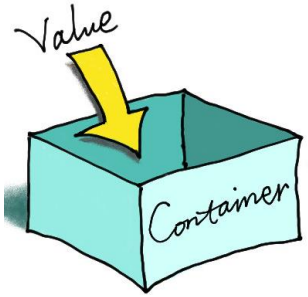
변수를 소개합니다

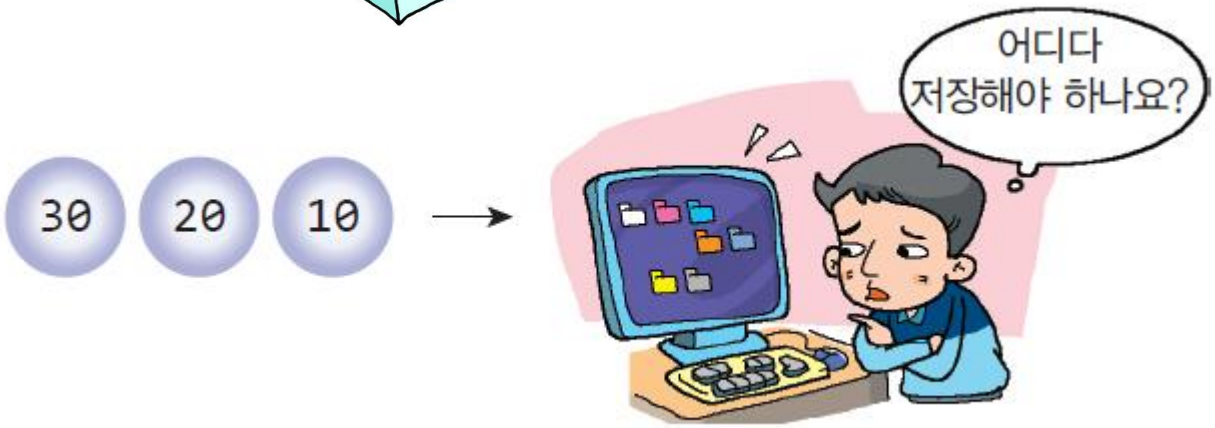
일반적인 프로그램의 구조

- 일반적인 프로그램은 외부로부터 데이터를 받아서(입력단계), 데이터를 처리한 후에(처리단계), 결과를 화면에 출력(출력단계)한다.



- 변수(variable)는 값을 저장하는 상자로 생각할 수 있다.
- 변수는 컴퓨터 메모리 공간에 만들어 진다.

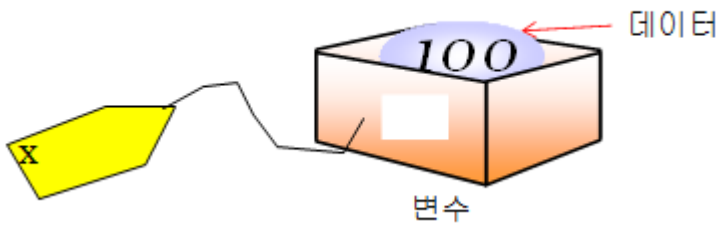




변수 생성

- 파이썬에서 변수를 생성하려면 다음과 같이 한다.

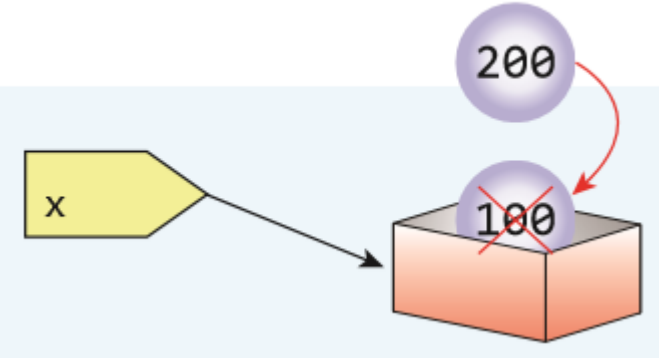
```
>>> x = 100
>>>
```



변수의 사용

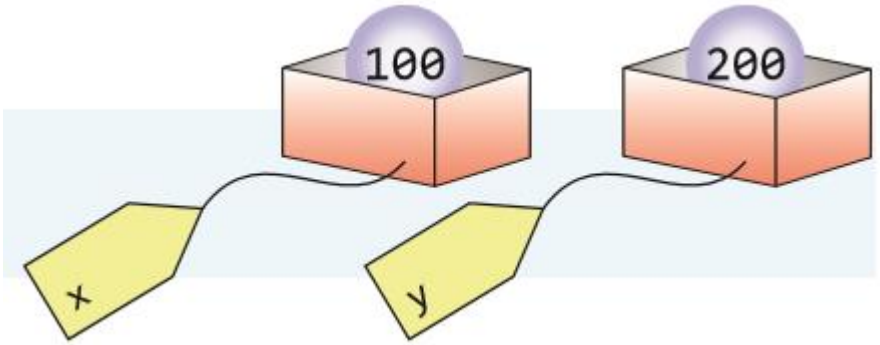
- **생성된 변수에는 얼마든지 다른 값을 저장할 수 있다.**

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



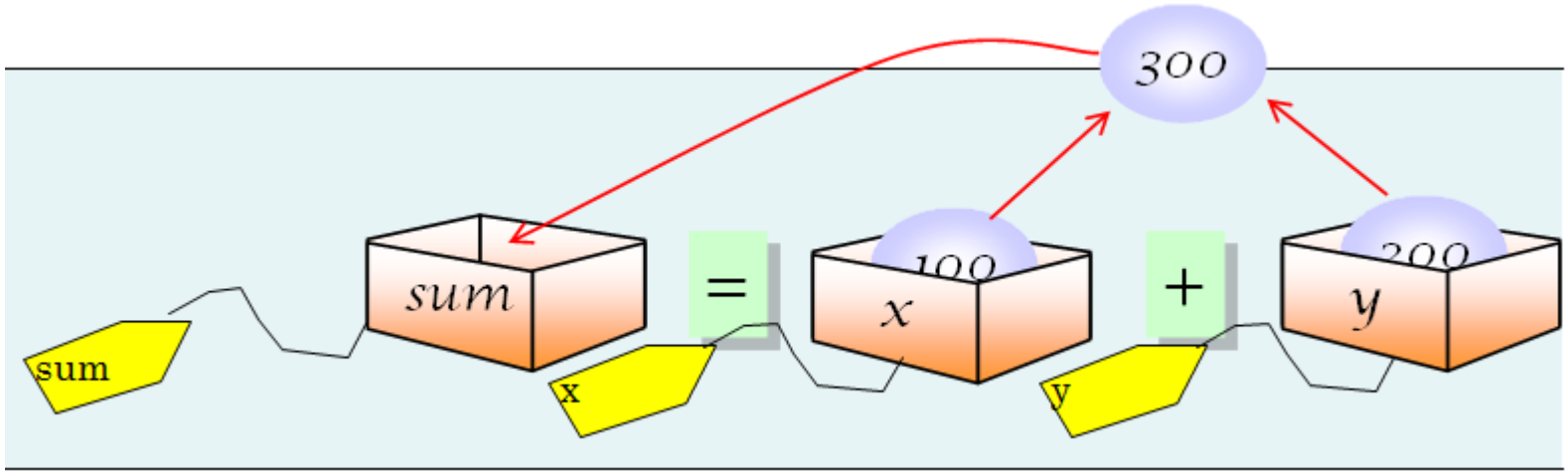
변수 2개 생성

```
>>> x = 100
>>> y = 200
```



변수를 이용한 계산

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



주의!!

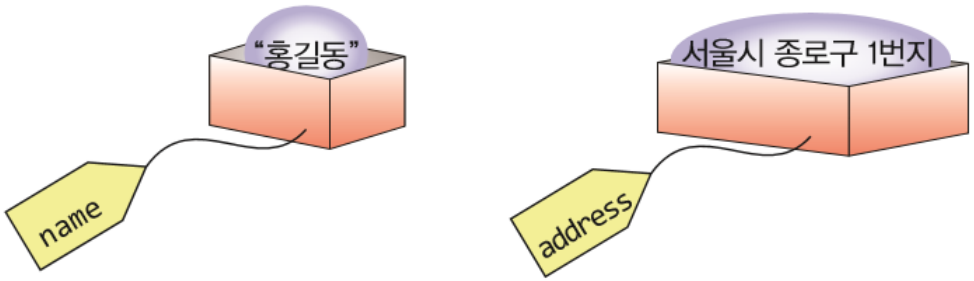
경고

입문자들이 가장 많이 틀리는 문제 중의 하나가 =을 '양변이 같다'고 해석하는 것이다. 파이썬에서 = 기호는 "변수에 값을 저장하라"라는 의미이다. 혼동하지 않도록 하자. 등호는 ==와 같이 표시한다.

문자열도 변수에 저장할 수 있다!

- **파이썬의 변수에는 정수뿐만 아니라 문자열도 저장할 수 있다.**

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




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

도전문제

무엇이 출력될까?

```
>>> x = 7
>>> y = 6
>>> print(x + y)
```

```
>>> x = '7'
>>> y = '6'
>>> print(x + y)
```

할당되지 않은 변수는 에러

- **변수는 사용되기 전에 반드시 할당 되어야 함**

다음 명령문을 직접 실행시켜본 후 결과를 비교해보자.

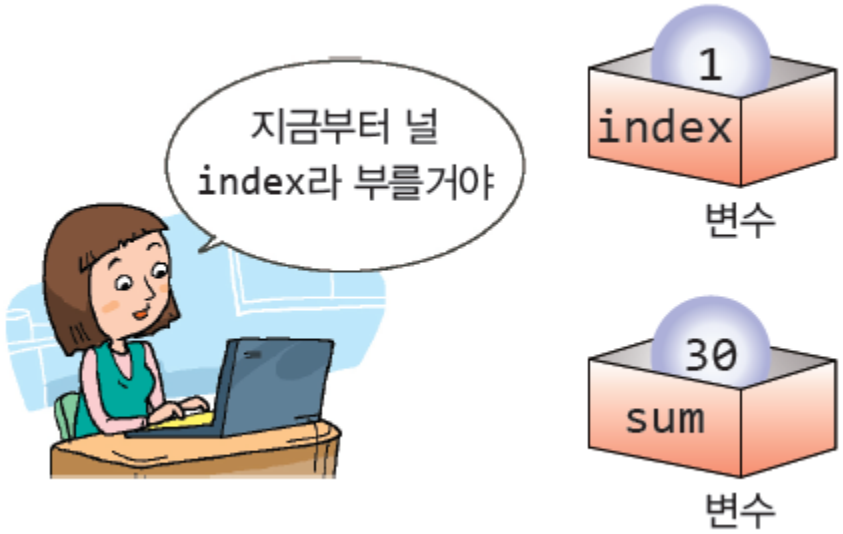
1) number

2) number = 5

```
>>> number
Traceback(most recent call last):
  File "<pyshell#0>", line 1, in <module>
    number
NameError: name 'number' is not defined
>>> number = 5
>>> number
5
>>>
```

변수의 이름

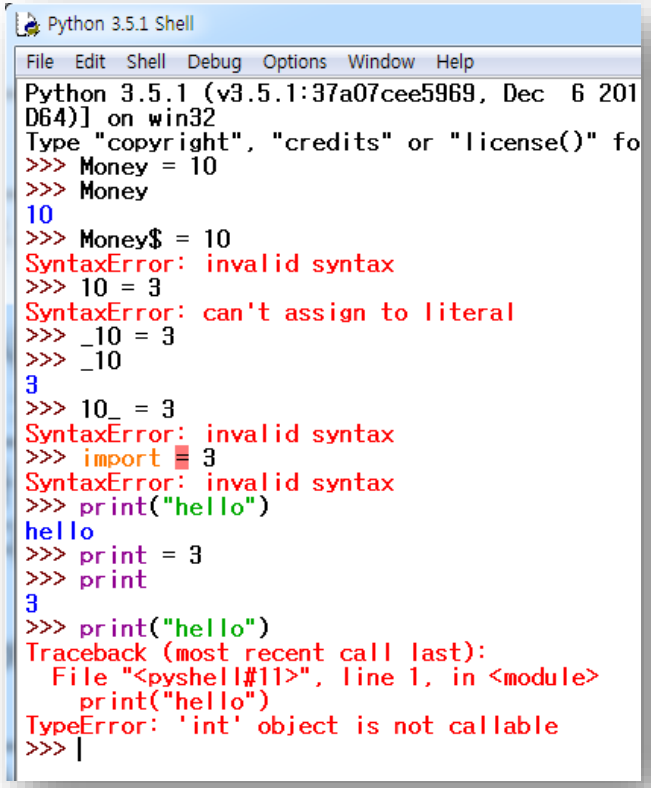
- 의미 있는 이름을 사용
- 소문자와 대문자는 서로 다르게 취급된다.
- 변수의 이름은 영문자와 숫자, 밑줄(_)로 이루어진다.
- 변수의 이름 중간에 공백이 들어가면 안 된다. 단어를 구분하려면 밑줄(_)을 사용 한다.



변수명 (Naming) 규칙

- **변수 이름을 결정할 때에 고려해야 할 규칙**

	변수명 규칙
1	변수의 이름은 문자, 숫자 그리고 Underscore(_)로만 이루어진다. 다른 기호를 사용하면 구문 에러(Syntax Error)이다. (예) Money$: 구문 에러, $는 사용할 수 없다.
2	변수명은 문자 또는 Underscore로만 시작해야 한다. 즉 숫자로 시작하면 안된다. (예) 7up, 5brothers : 숫자로 시작했기 때문에, 역시, 구문 에러이다.
3	파이썬 지정단어 (Keyword, Reserved word)들은 변수명으로 사용할 수 없다. (지정 단어 목록 참조)
4	파이썬에서는 대문자와 소문자를 구분한다. (예) hour 와 Hour는 다른 변수이다.



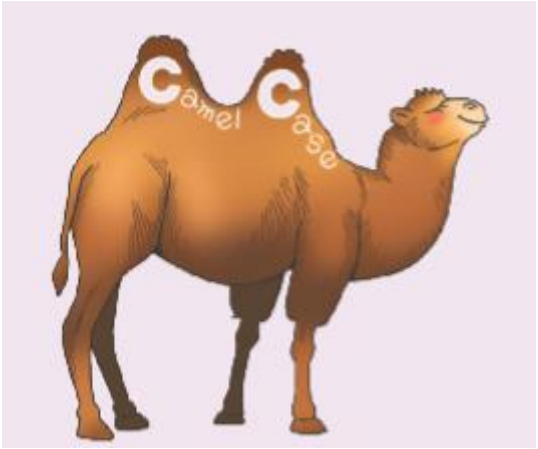
식별자

```
sum                          # 영문 알파벳 문자로 시작
_count                       # 밑줄 문자로 시작할 수 있다.
number_of_pictures    # 중간에 밑줄 문자를 넣을 수 있다.
King3                        # 맨 처음이 아니라면 숫자도 넣을 수 있다.
```

```
2nd_base (X)                 # 숫자로 시작할 수 없다.
money# (X)                   # #과 같은 기호는 사용할 수 없다.
```

낙타체

- 낙타체는 변수의 첫 글자는 소문자로, 나머지 단어 의 첫 글자는 대문자로 적는 방법이다. 예를 들면, myNewCar처럼 첫 'm'은 소문자로, 나머지 단어들의 첫 글자는 대문자로 표기한다



[참고] 파이썬 지정 단어 (Keyword, Reserved Word)

- **keyword를 import 하면 파이썬에서 지정한 단어들을 확인 할 수 있음**

  – 파이썬 지정 단어는 변수명으로 사용 할 수 없음에 유의하자.

keyword를 import하여 파이썬 지정 단어를 확인해 보자.

1) keyword.kwlist 명령어는 파이썬 지정 단어를 배열 형태로 나열해 준다.

2) len 함수를 사용해서 파이썬 지정 단어의 개수도 확인해 보자.

```
>>> 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']
>>> len(keyword.kwlist)
33
>>>
```

[실습] 올바른 변수명이 아닌 이유

다음의 변수명으로 임의의 값을 선언해 본 뒤 결과를 확인해보자.

1) money$
2) 7up
3) False

```
>>> money$ = 2
SyntaxError: invalid syntax
>>> 7up = 26
SyntaxError: invalid syntax
>>> False = 'True Love'
SyntaxError: can't assign to keyword
```

- 아래의 '변수명 ' 이 올바른지 O/X로 표현해보자.

>>> True ( )

>>> 3apples ( )

>>> elif ( )

>>> new_score ( )

>>> Brother ( )

[참고] 의미 있는 변수명

- **변수명은 의미 있게 만들어져야 함**
  - 그 변수의 역할에 맞게 이름이 지어져야, 프로그램을 검토할 때나 협업 시 공유할 때 도움이 됨

예제

3개의 시험 성적이 주어졌을 때, 시험 성적의 총합과 평균을 구하여라.

- 수학: 26점
- 영어: 54점
- 역사: 96점

예제 실습 A

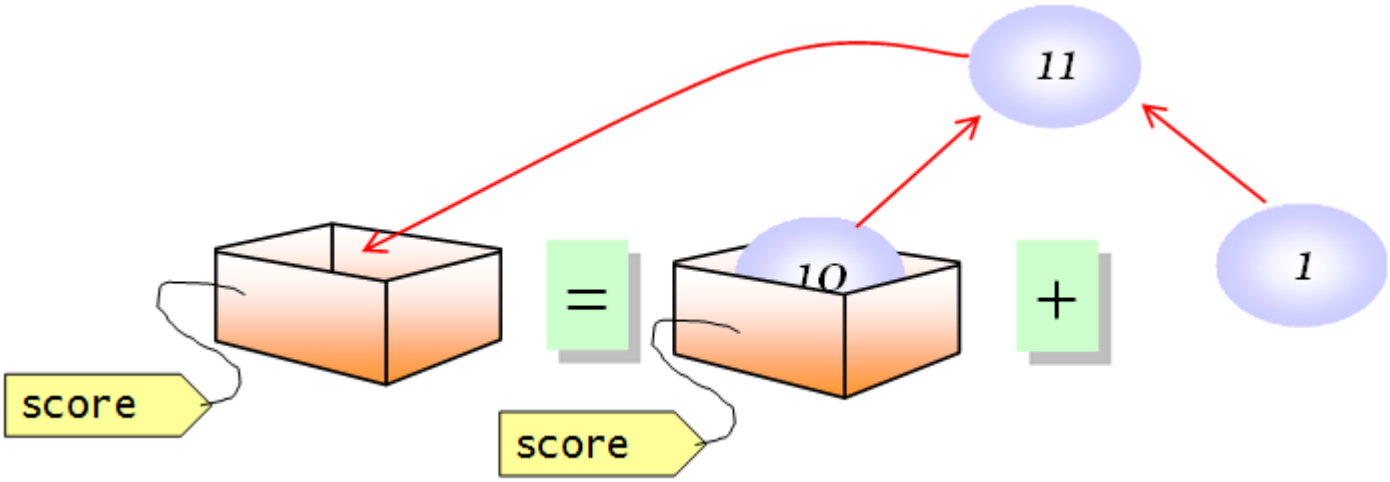
```
a = 26
b = 54
c = 96
d = a + b + c
f = d / 3
```

예제 실습 B

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

이런 것도 가능하다!

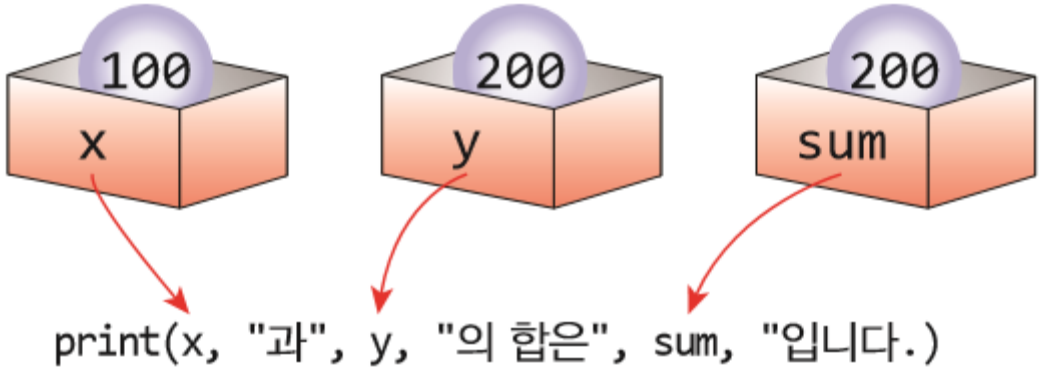
```
score = 10
score = score + 1
```



여러 값을 함께 출력하기

```
x = 100
y = 200
sum = x + y
print(x, "과", y, "의 합은", sum, "입니다.")
```

```
100 과 200 의 합은 300 입니다.
```



한번에 여러 변수 할당(Multiple Assignments)

- 한번에 여러 변수에 할당 시 변수와 값의 개수가 일치해야 함

[예제 3-15] 한 번에 여러 변수 할당 연습

다음 명령문을 직접 실행시켜본 후 결과를 비교해보자.

1) number_1, number_2 = 511

2) number_1, number_2 = 2, 4, 5

3) number_1, number_2 = 6, 9

```
>>> 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
>>>
```

변수에 변수 할당

- 할당문의 오른쪽에 문자가 올 때는 반드시 먼저 값을 할당 받은 후,

  할당문의 오른쪽에 문자를 사용해야 한다.

[예제] 변수에 변수를 할당 1

다음 명령문을 직접 실행시켜본 후 결과를 비교해보자.

1) number_1 = number_2

2) number_2 = 511
   number_1 = number_2

```
>>> number_1 = number_2
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    number_1 = number_2
NameError: name 'number_2' is not defined
>>> number_2 = 511
>>> number_1 = number_2
>>> number_1
511
>>> number_2
511
>>>
```

변수에 변수 할당

- 같은 경우로 문자를 사용할 때는 초기값을 할당해야 함

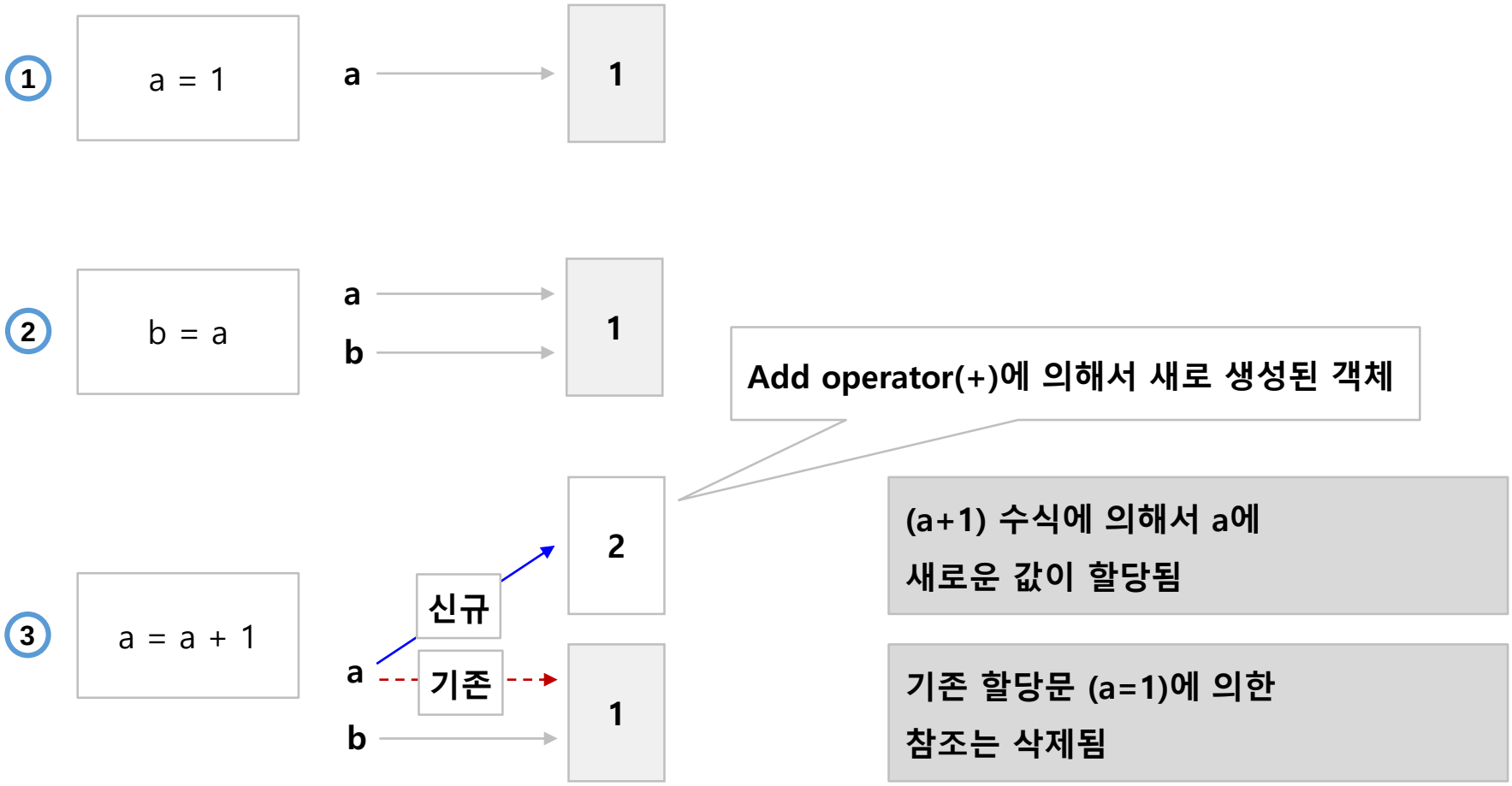
[예제] 변수에 변수 할당 2

다음 명령문을 직접 실행시켜본 후 결과를 비교해보자.

1) number_1 = number_2 + 2

```
>>> number_1 = number_2 + 2
Traceback (most recent call last):
          File "<pyshell#0>", line 1, in <module>
                    number_1 = number_2 + 2
NameError: name 'number_1' is not defined
>>>
```

[참고] 할당문의 실행과정


1
a = 1
a
1
2
b = a
a
b
1
3
a = a + 1
신규
a
기존
b
2
1
Add operator(+)에 의해서 새로 생성된 객체
(a+1) 수식에 의해서 a에
새로운 값이 할당됨
기존 할당문 (a=1)에 의한
참조는 삭제됨

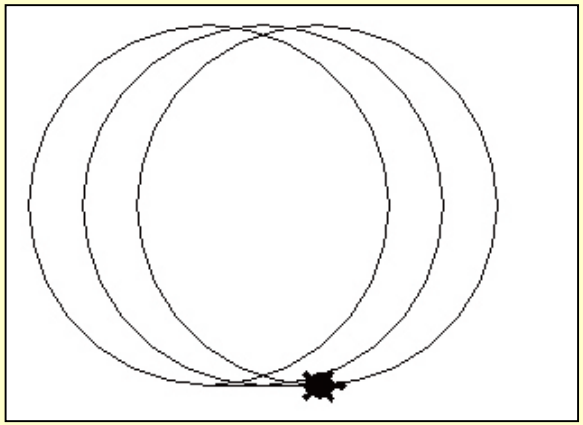
- 다음 코드의 괄호 안 결과를 예측하여 보자

```
>>> number1 = 5
>>> number2 = number1
>>> number1 = number1 + 2
>>> number1
(                    )
>>> number2
(                    )
```

Lab: 변수는 어디에 유용할까?

- 다음과 같이 터틀 그래픽을 사용하여 반지름이 100픽셀인 3개의 원을 그리는 프로그램이 있다고 하자.

```
import turtle
t = turtle.Turtle()
t.shape("turtle")

radius = 100
t.circle(radius) # 반지름이 100인 원이 그려 진다.
t.fd(30)
t.circle(radius) # 반지름이 100인 원이 그려 진다.
t.fd(30)
t.circle(radius) # 반지름이 100인 원이 그려 진다.
```



코드 변경

- 하지만 갑자기 원의 반지름을 50으로 변경하여서 다시 그려야 한다면 어떨까?

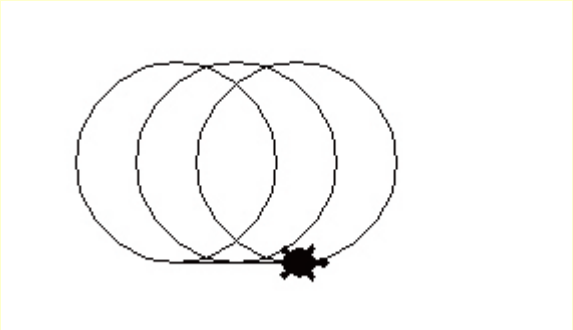
-> 원의 반지름이 변수로 표현되었기 때문에 쉬운 방법이 있다. 변수만 변경하면 된다.

```
t.circle(100)
t.fd(30)
t.circle(100)
t.fd(30)
t.circle(100)
```

```
radius = 100
t.circle(radius)
t.fd(30)
t.circle(radius)
t.fd(30)
t.circle(radius)
```

Solution

```
import turtle
t = turtle.Turtle()
t.shape("turtle")

radius = 50
t.circle(radius) # 반지름이 50인 원이 그려 진다.
t.fd(30)
t.circle(radius) # 반지름이 50인 원이 그려 진다.
t.fd(30)
t.circle(radius) # 반지름이 50인 원이 그려 진다.
```



입력문을 위한 함수

입력문을 위한 함수

- **Python에서 많이 사용하는 입력함수는 2가지 형식으로 나눌 수 있다.**

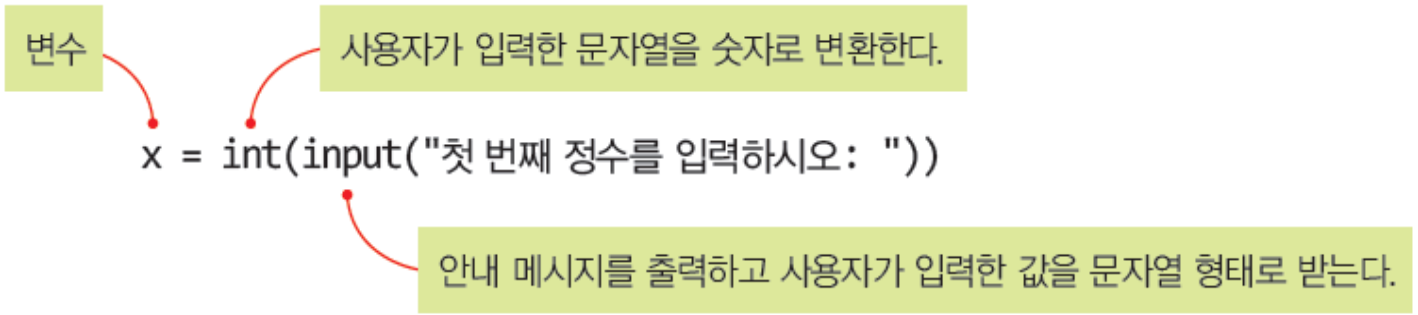
variable_name=input()	사용자로부터 입력을 받는다.
variable_name=input('문자열')	'문자열'에 해당하는 내용을 출력 후 사용자로부터 입력을 받는다.

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

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

사용자로부터 정수 입력받기

input() 사용법

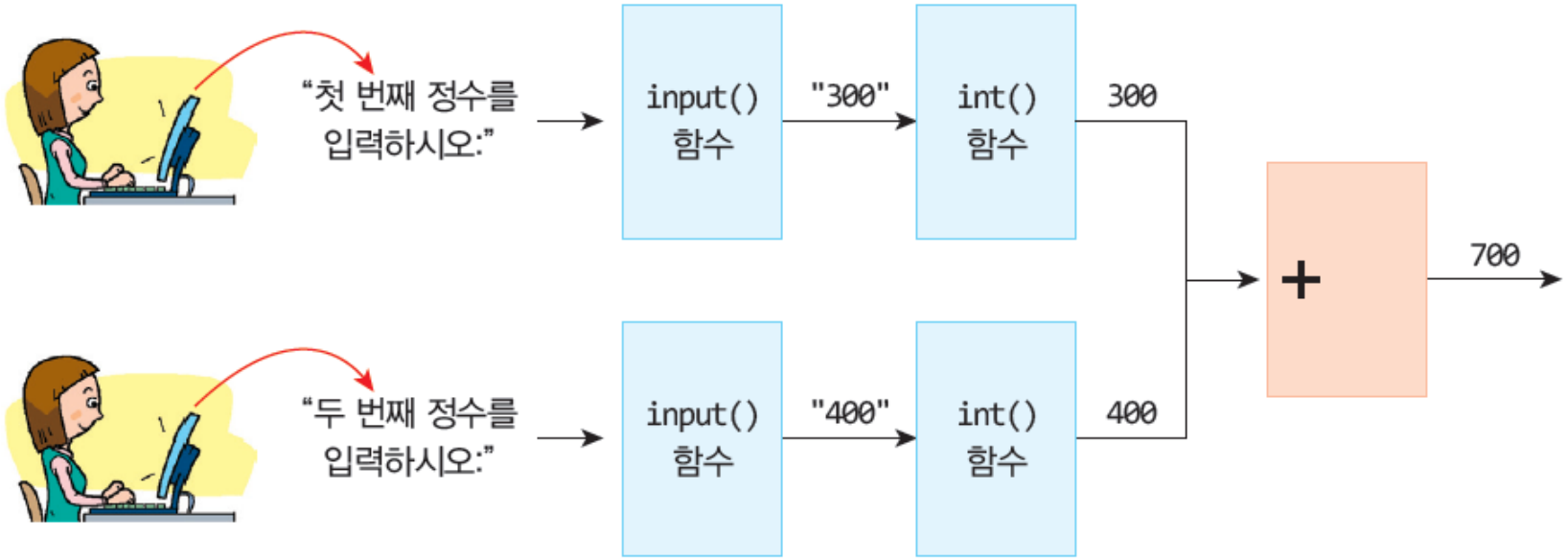


완전한 코드

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

```
첫 번째 정수를 입력하시오: 300
두 번째 정수를 입력하시오: 400
100 과 200 의 합은 300 입니다.
```

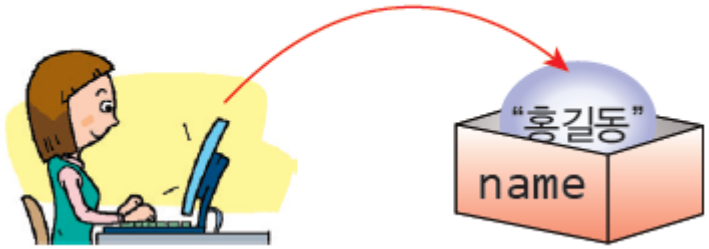
완전한 코드


"첫 번째 정수를 입력하시오:"
input() 함수
"300"
int() 함수
300
"두 번째 정수를 입력하시오:"
input() 함수
"400"
int() 함수
400
+
700

사용자로부터 문자열 입력받기

```
name = input("이름을 입력하시오: ")
print(name, "씨, 안녕하세요?")
print("파이썬에 오신 것을 환영합니다.")
```

```
이름을 입력하시오: 홍길동
홍길동 씨, 안녕하세요?
파이썬에 오신 것을 환영합니다.
```



도전문제

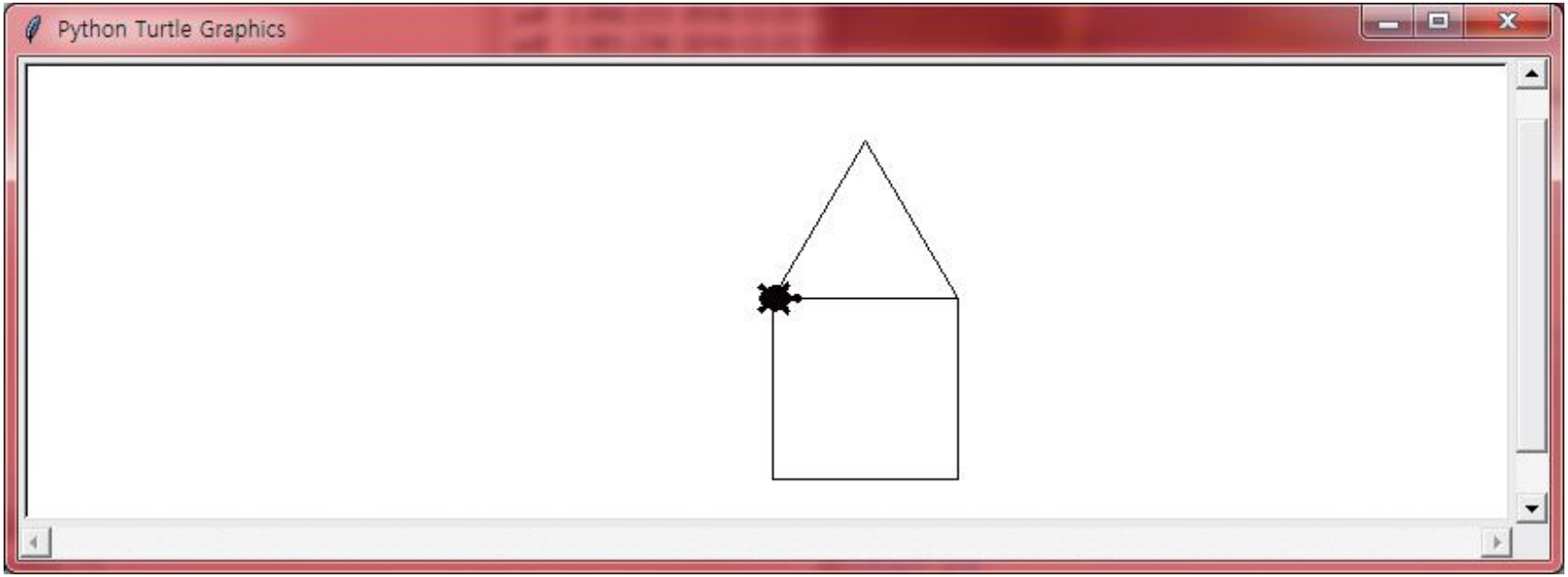
사용자의 이름을 물어보고 이어서 2개의 정수를 받아서 덧셈을 한 후에 결과를 출력하는 다음과 같은 프로그램을 작성해보자.

```
이름을 입력하시오: 홍길동
홍길동 씨, 안녕하세요?
파이썬에 오신 것을 환영합니다.
첫 번째 정수를 입력하시오: 300
두 번째 정수를 입력하시오: 400
300 과 400 의 합은 700 입니다.
```

Lab: 집그리기

- **우리는 사용자로부터 집의 크기를 입력받아서 크기에 맞는 집을 그려보자.**

집의 크기는 얼마로 할까요? 100



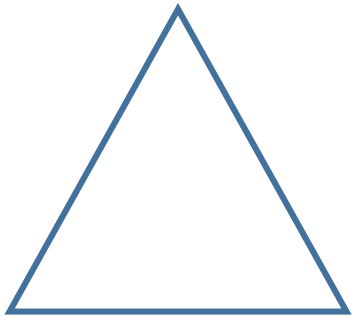
Solution

2.py - C:/Users/SSO/Desktop/2.py (3.5.1)

File Edit Format Run Options Window Help

```
import turtle
t = turtle.Turtle()
t.shape("turtle")

size = int(input("집의 크기는 얼마로 할까요? "))

t.forward(size) # size 만큼 거북이를 전진시킨다.
t.right(90) # 거북이를 오른쪽으로 90도 회전시킨다.
t.forward(size)
t.right(90)
t.forward(size)
t.right(90)
t.forward(size)

t.right(90)

t.forward(size)
t.left(120)
t.forward(size)
t.left(120)
t.forward(size)
t.left(120)
```

Ln: 23 Col: 0

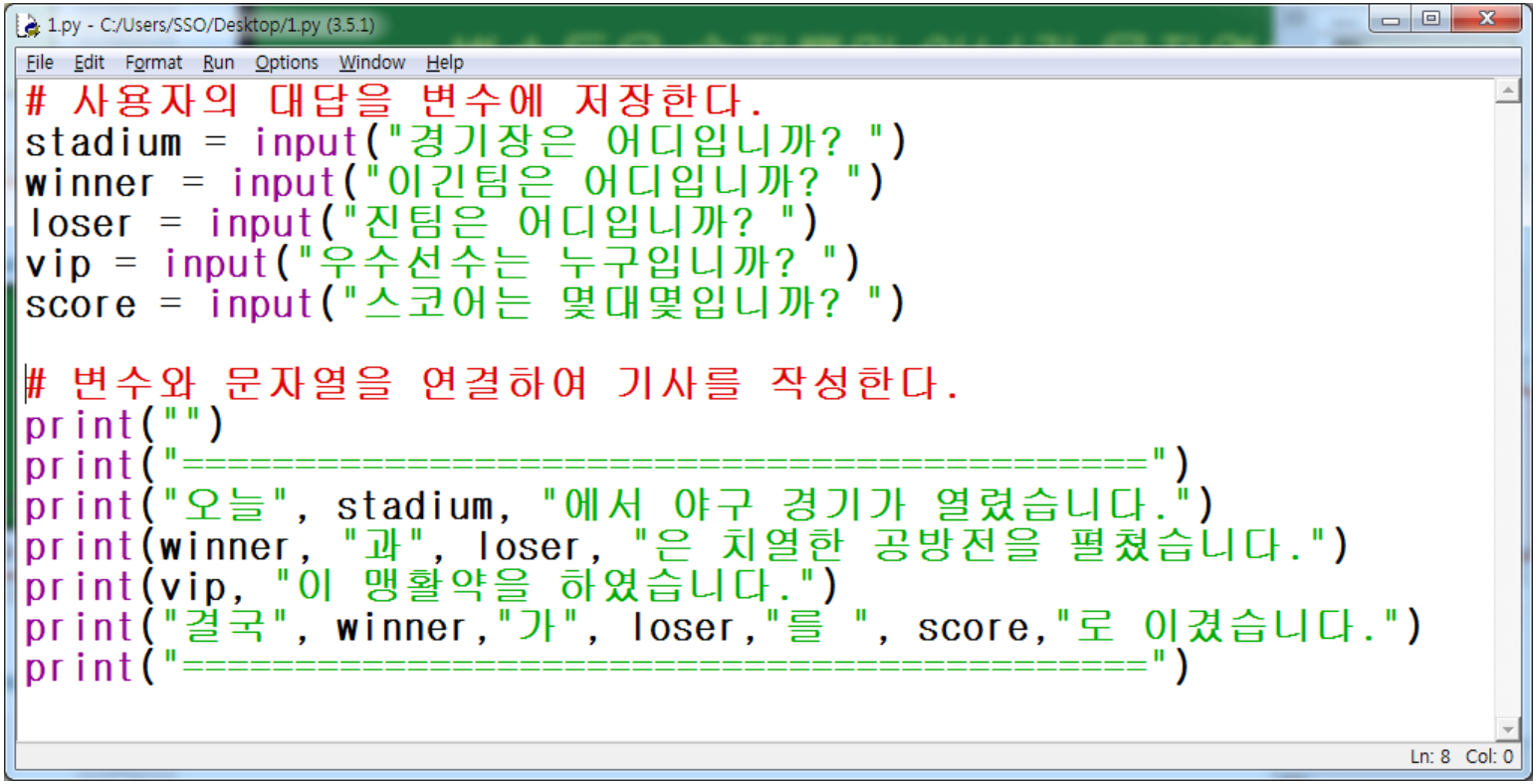


Lab: 로봇 기자 만들기

- 사용자에게 경기장, 점수, 이긴 팀, 진 팀, 우수 선수를 질문하고 변수에 저장한다. 이들 문자열에 문장을 붙여서 기사를 작성한다.

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  6 2015, 01:54:25) [MSC v.1900 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/SSO/Desktop/1.py ==========
경기장은 어디입니까? 서울
이긴팀은 어디입니까? 삼성
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.
==========================================
>>>
                                                                    Ln: 17  Col: 4
```

Solution


1.py - C:/Users/SSO/Desktop/1.py (3.5.1)
File Edit Format Run Options Window Help
# 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("==")
Ln: 8 Col: 0

이번 장에서 배운 것

- 컴퓨터에서는 변수를 사용하여 어떤 것을 컴퓨터 메모리 안에 저장할 수 있다.
- 변수들은 이름을 가지고 있다.
- 변수들은 숫자뿐만 아니라 문자열도 저장할 수 있다. 사실은 어떤 것이든지 저장이 가능하다.

Q & A