파이썬 프로그래밍 :: 데이터

IHS 서강대학교 SOGANG UNIVERSITY

자료형 (Data Types)

- 변수에 정수, 실수, 문자열 등의 다양한 형의 데이터를 대입하여 저장 가능
- python에서는 기본적으로 다음과 같은 데이터형을 제공
  - 숫자(numbers)
    - 정수(int) : 7, 123, -256, etc.
    - 실수(float) : 3.14, -1.2345, -3.0e+5, etc.
    - 복소수(complex) : 2.5+3.2j, 1+2j, etc.
  - Boolean : True, False
  - 문자열(string, str) : "Hello", 'Hello', etc
  - 리스트(list) : [5.5, -3, "Hello“]
  - 집합(set) : {1, 2.2, "Hello“}
  - 튜플(tuple) : (1, 2, 3, 4, 5)
  - 사전(dictionary, dict) : {'val1':1, 'val2':2}

자료형 (Data Types)

- 정수형 (int)
  - 정수형은 소수점이 없는 숫자 데이터(100, -123, 0,….)
  - int는 기본적인 정수 데이터 형식
  - 파이썬에서는 정수의 크기에 제한이 없음(이론적으로)

- 정수형 데이터 표현
  - 16진수는 0x나 0X (숫자 0)
  - 8진수는 0o나 0O(숫자+알파벳 오)
  - 2진수는 0b나 0B를 접두사로 붙여 표현

```
>>> print(0xFF, 0o77, 0b1111)
255 63 15
>>>
```

자료형 (Data Types)

- 실수형 (float)
  - 실수형은 소수점이 있는 숫자 데이터 (3.14, -2.7,….)
  - 실수의 두 가지 표현 방식
    - 고정 소수점 방식: 132.234, –0.023 등과 같이 소수점을 고정하여 표시한 방법
    - 부동 소수점 방식: 1.53e+5(=1.53×10^5), 3.5e–64(=3.5×10^{-64})

      등과 같이 지수형 (exponential form)으로 표시하는 방법

- 실수를 표현할 때 오차가 발생할 수 있음 (실수의 이진수 저장 방법 때문)

```
>>> 0.1 + 0.2              # 정확히 0.3이 아님
0.30000000000000004
>>> 4.3 – 2.7              # 정확히 1.6이 아님
1.5999999999999996
```

자료형 (Data Types)

- Boolean 형
  - 참(True)이나 거짓(False)만 저장
  - Boolean 형은 단독으로 사용하기 보다 if 조건문이나 while 반복문 등과 함께 주로 사용 (추후 설명)

- 문자열 형
  - 문자열은 양쪽을 큰따옴표(")나 작은따옴표(')로 감싼 문자들의 모임

```
>>> print(100+200)
300
>>> print("100"+"200")
100200
```

100+200 은 (정수+정수) 형태가 되어서 덧셈 연산.

"100"+"200"은 문자열과 문자열을 연결하라는 의미.

변수 (Variable)

- 변수는 객체를 가리키는 심볼
- 변수와 객체는 대입문/할당문(Assign Statement)에 의해 생성.

variable = value

할당연산자

- 할당 연산자의 왼쪽에는 무조건 변수만 올 수 있고,

  오른쪽에는 무엇이든 (value, 변수, 수식, 함수 등) 올 수 있음.

- x = 100 이라는 명령문은 100을 저장하는 정수형 object(객체)와

  이것을 가리키는 variable x를 생성하라는 명령.

- Python에서 symbol = 은 수학에서의 equality와 차이가 있음
- 할당 연산자의 오른쪽에 오는 변수는 반드시 값이 할당된 변수를 사용해야 함

```
>>> number_1 = number_2
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    number_1 = number_2
NameError: name 'number 2' is not defined
```

자료형 확인

- 내장 함수 type( )
  - Object의 데이터 형(data type)을 알려주는 함수.
  - 변수(variable)가 어떤 data type을 참조하는지 확인 가능.

```
>>> x = 10
>>> print(type(x))          # x가 참조하는 객체의 data type
<class 'int'>               # 정수
>>> x = 2.5
>>> print(type(x))
<class 'float'>             # 실수
>>> x = "Hello"
>>> print(type(x))
<class 'str'>               # 문자열
>>>
```

자료형 변환

- 정수, 실수, 문자열 등의 자료형들은 다른 자료형으로 변환될 수 있으며, python에서는 이를 위한 내장 함수를 제공
  - int() 함수
    - 소수점이 없는 숫자 형태의 문자열을 정수로 변환
    - 실수 형태의 문자열은 정수로 변환할 수 없음: error
    - 실수 값을 정수로 변환
  - float() 함수
    - 실수 형태의 문자열이나 정수를 실수로 형 변환
  - str() 함수
    - 실수나 정수를 문자열로 형 변환

```
>>> int ("34")
34
>>> int(34.5)
34
>>> int("34.5")
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    int("34.5")
ValueError: invalid literal for int() with base 10: '34.5'
```

기타 알아야 할 사항

- 하나의 statement를 여러 줄을 사용하여 작성할 때는 backslash(\ ) 를 사용한다.
  (한글 폰트에서는 ₩로 표시됨).

```
>>> a = 1 + 2 + 3 + ₩
        4 + 5 + 6 + ₩
        7 + 8 + 9
```

- 특정한 data type을 표시할 때 사용되는 (…), […], {…} 등의 괄호 내부에서는 backslash없이 줄을 바꾸어도 무방하다.

- 한 줄에 두 개 이상의 statement를 넣을 때는 semicolon(;)을 사용하여 구분한다.

```
>>> a = 1; b = 2; c = 3
>>> print(a, b, c)
1 2 3
>>>
```

기타 알아야 할 사항

- 필요한 경우 하나의 statement로 다수의 variable과 object를 생성할 수 있다.

```
>>> a, b, c = 5, 4.1, "Hello!"
>>> print(a, b, c)     # print 5 4.1 Hello!
5 4.1 Hello!
```

- 다수의 variable이 하나의 object을 참조하도록 하는 경우

```
>>> x = y = z = 5
>>> print(x, y, z)     # print 5 5 5
5 5 5
```

기타 알아야 할 사항

- 스와핑 코드, num1과 num2의 값을 서로 바꾸는 코드

script code

```
num1 = 10
num2 = 20
print(num1, num2)
tmp = num1
num1 = num2
num2 = tmp
print(num1, num2)
```

같은 의미의 명령어

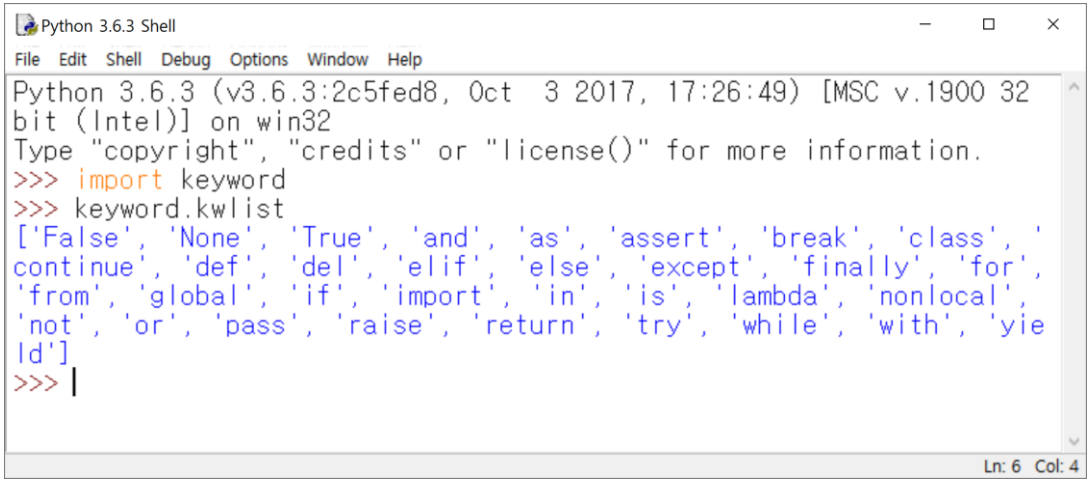
```
num1, num2 = num2, num1
```

변수명 규칙

- 변수명은 영어 대소문자, 숫자, 밑줄(_)로만 이루어짐
  - Money$: 문자 $는 사용할 수 없음
- 변수명은 영어 대소문자 또는 밑줄로만 시작
  - 7up, 5brothers : 숫자로 시작할 수 없음
- 파이썬 지정단어(Keyword, Reserved word)들은 사용 불가
- 파이썬에서는 대문자와 소문자를 구분
  - hour 와 Hour는 다른 변수

변수명 규칙

- 파이썬 지정단어 (Keyword/ Reserve Word)



  - 위의 지정 단어들은 변수명으로 사용 할 수 없음
  - import 명령문의 사용법은 추후에 설명

- 잘못된 식별자(변수명)

```
2nd_base      # 숫자로 시작할 수 없음
#money        # #과 같은 기호는 사용할 수 없음
varname$      # 특수 문자 $를 사용할 수 없음
id name       # 중간에 공백 문자를 사용할 수 없음
for           # 예약어 for를 변수명으로 사용할 수 없음
```

변수의 할당

- 다음과 같은 명령어를 실행하면

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

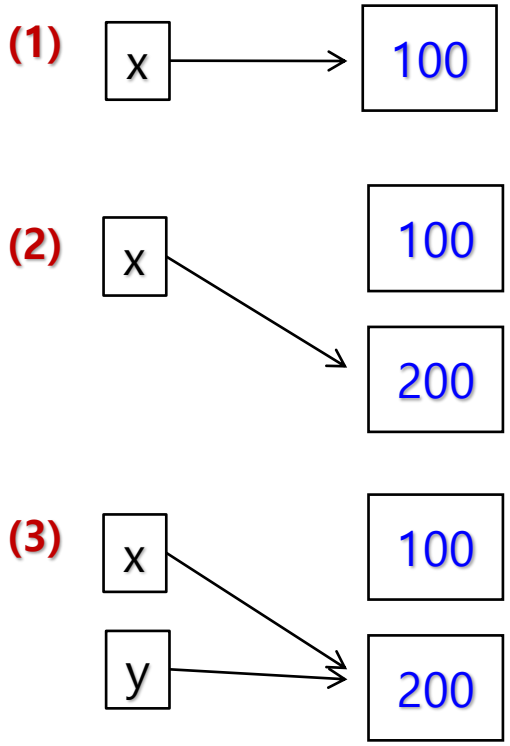
  - 정수 100 의 값을 가지는 객체가 메모리의 어딘가에 생성
  - 변수 x 역시 메모리의 어딘가에 생성되며, 100을 값으로 가지는 객체를 참조하게 됨

  - 우리는 간단히 'x 값은 100이다' 또는 'x는 100을 저장한다' 라고 한다.

변수의 할당

- 생성된 변수에 다른 값을 할당할 수 있음

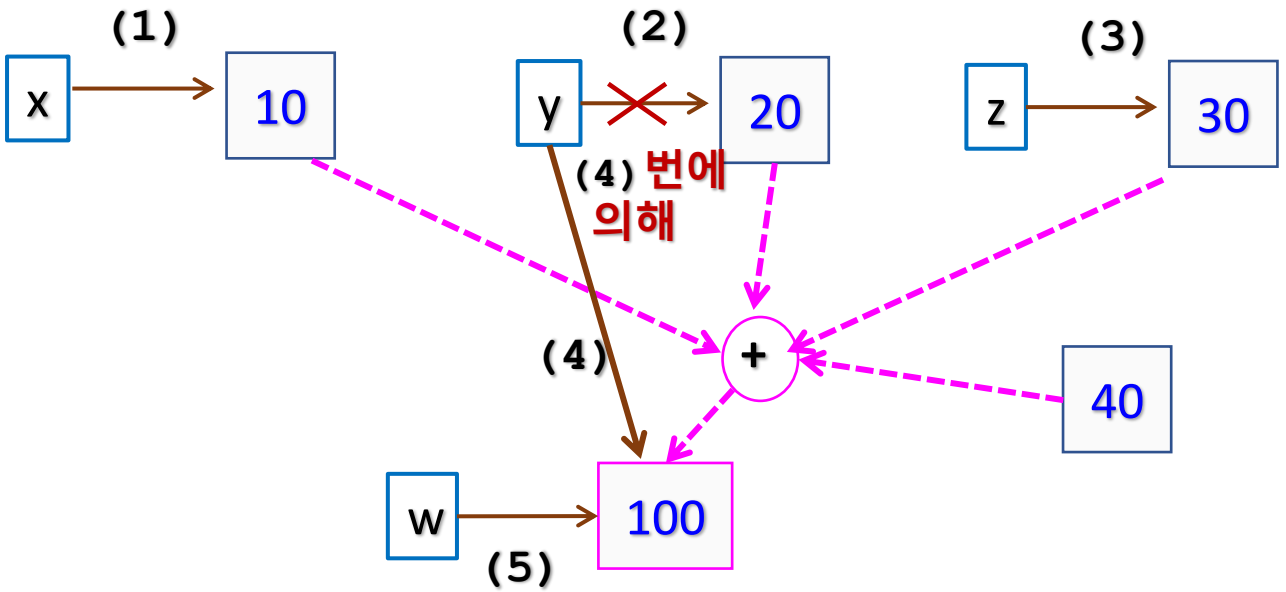
```
>>> x = 100     # (1)
>>> x = 200     # (2)
>>> print(x)
200
>>> y = x       # (3)
```



- y = x 명령어는 x가 가리키는 object를 같이 가리키는 variable y를 생성하는 것을 의미.
- y값 역시 200이다.

변수의 할당

```
>>> x = 10                    # (1)
>>> y = 20                    # (2)
>>> z = 30                    # (3)
>>> y = x + y + z +40         # (4)
>>> w = y                     # (5)
```



1. 3.14 를 변수에 할당한 후, 다음 순서대로 실행하는 script 작성할 것

(변수는 하나만 사용, ②번 작성시 강의 내용 다시 숙지 필요)

① 문자열로 변환 후, 값과 데이터 형을 출력

② 변환된 문자열을 정수로 변환 후, 값과 데이터 형을 출력

③ 변환된 정수를 실수로 변환 후, 값과 데이터 형을 출력

출력

```
3.14 <class 'str'>
3 <class 'int'>
3.0 <class 'float'>
```

2. 강의 자료 11쪽에서 보여준 두 변수의 값 교환 방법은 3개의 변수를 사용한다. 두개의 변수를 사용하여 변수의 값을 교환하는 script를 작성하라.

출력

```
10 20
20 10
```