변수(variable)

- 컴퓨터에서 처리되는 데이터들은 메모리의 어딘가에 실제로 존재함
- 파이썬에서 모든 데이터(값)는 객체이며 메모리에 존재
- 변수는 메모리에 저장된 값(Value)을 참조하는 이름
- 즉, 변수는 메모리에 생성된 객체(object)를 참조하기 위한 수단
- 파이썬에서 변수는 값이 처음 정해지는(할당되는) 순간에 생성
- 변수의 값은 초기화 된 후, 실행 중간에 변경 가능
- 또한, 객체 하나를 여러 개의 서로 다른 이름의 변수들로 참조 할 수 있음

```
x = 10    #변수 x가 정수 10을 값으로 하는 객체 참조
y = 10    #변수 y는 변수 x와 같은 객체 참조
x = 20    #변수 x는 정수 20을 값으로 하는 객체 참조
```

변수명

- 변수명은 영어 대소문자와 숫자, 밑줄(_)만 허용
- 소문자와 대문자는 서로 다른 변수명으로 취급
- 변수명에 공백을 허용하지 않으므로 단어를 구분하려면 밑줄 (_) 을 사용
- myNewCar와 같이 낙타체 표기법(대소문자로 단어 구별)을 사용하여 의미를 부여할 수 있음
- 변수명은 의미 있는 이름을 사용하는 것이 효과적

3개의 시험 성적의 총합과 평균을 구하여라.
- 수학: 26점
- 영어: 54점
- 역사: 96점

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

python script A

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

python script B

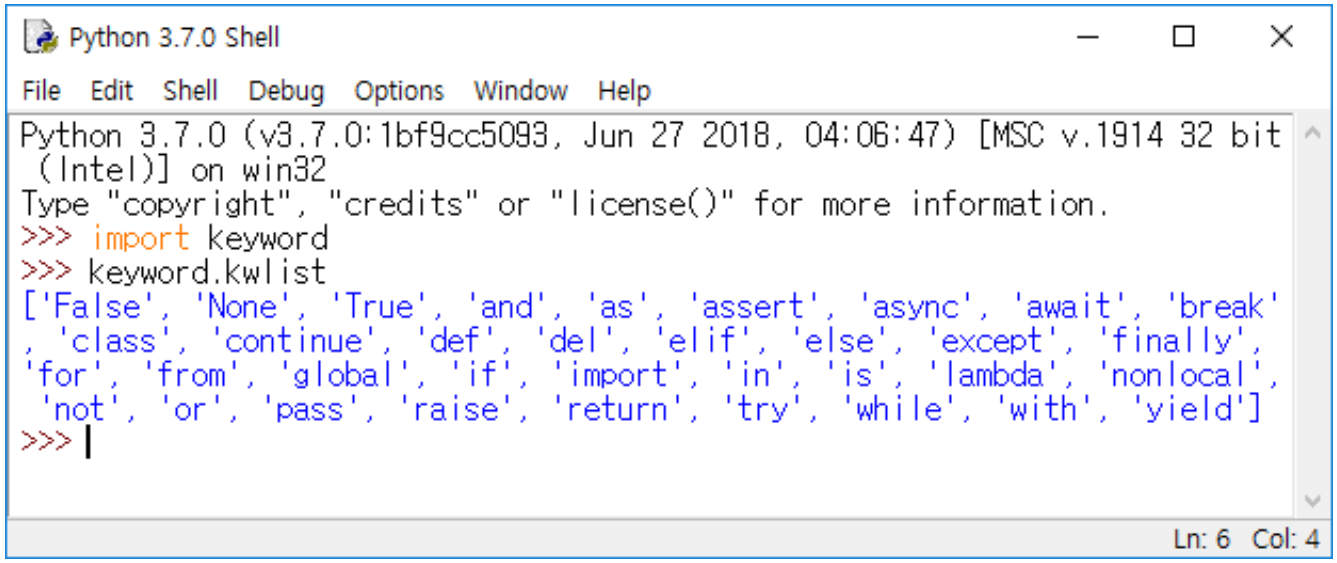
변수명

- 변수명(Naming) 규칙 정리

	변수명 규칙
1	변수명은 영어 대소문자, 숫자, 밑줄(_)로만 이루어짐 다른 기호를 사용하면 구문 에러(Syntax Error) (예) Money$, My Score : 문자 $와 공백은 사용하여 구문 에러
2	변수명은 영어 대소문자 또는 밑줄로만 시작해야 하며, 숫자로 시작 하면 안됨 (예) 7up, 5brothers : 숫자로 시작했기 때문에 구문 에러
3	파이썬 지정단어 (Keyword, Reserved word)들은 변수명으로 사용할 수 없음 (지정 단어 목록 참조)
4	파이썬에서는 대문자와 소문자를 구분 (hour 와 Hour는 다른 변수)

변수명

- 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
```

변수 할당문

- 변수는 대입문/할당문(assign statement)에 의해 생성

- 할당 연산자의 **왼쪽에는 변수만** 올 수 있고, 오른쪽에는 무엇이든 (값, 변수, 수식, 함수 등) 올 수 있음
- x = 100 이라는 명령어는 100을 저장하는 정수형 object(객체)와 이를 가리키는 변수 x를 생성하는 것
- python에서 symbol **=** 은 수학에서의 equality를 의미하지 않음
- 변수 초기화(initialization)란 변수에 처음 값을 할당하는 것을 의미함
- 할당 연산자의 오른쪽에 오는 변수는 반드시 값이 할당된 변수를 사용해야 함

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

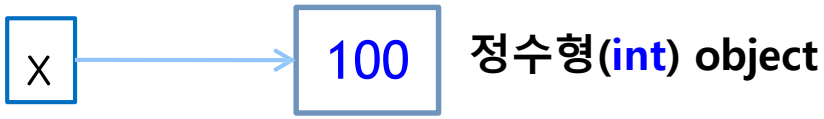
변수 할당문

- 다음과 같은 명령어를 실행하면

```
x = 100
```

① 정수 100 의 값을 가지는 객체가 메모리의 어딘가에 생성

② 변수 x 역시 메모리의 어딘가에 생성되며, 100을 값으로 가지는 객체를 참조하게 됨

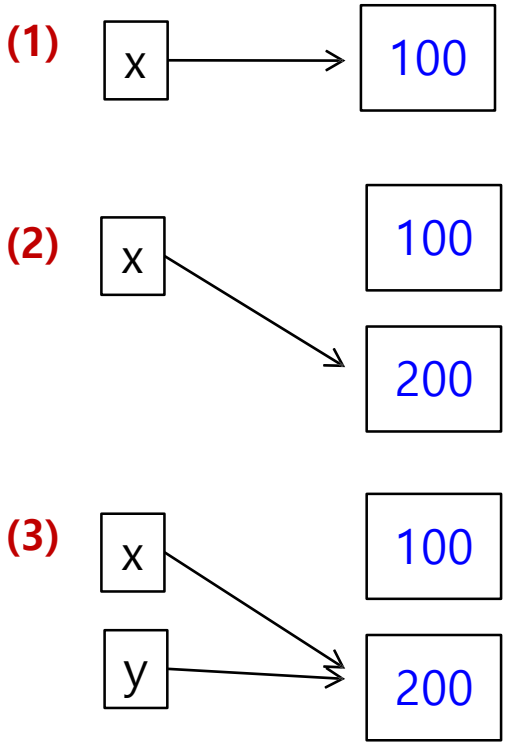


- 파이썬은 변수 선언문이 없으며, 할당문을 사용하여 변수에 값을 할당하는 순간 변수의 데이터형(자료형)이 결정됨
  - 변수에 다른 데이터형(자료형)의 값을 할당하는 순간 변수의 데이터형(자료형)도 같이 변경

변수 사용

- 생성된 변수에 다른 값을 할당할 수 있음

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



y = x 명령어는 x가 가리키는 객체(object)를 같이 가리키는 변수 y를 생성하는 것을 의미

데이터 형 (Data Types)

- 변수에는 정수, 실수, 문자열 등의 다양한 형의 데이터를 대입하여 저장 가능
- python에서는 기본적으로 다음과 같은 데이터형을 제공
  - 숫자(numbers)
    - 정수(integer, int) : 7, 123, -256,...
    - 실수(float) : 3.14, -1.2345, -3.0e5,...
    - 복소수(complex) : 2.5+3.2j, 1+2j, 3+1j,...
  - 논리값(Boolean) : True, False
  - 문자열(string, str) : "Hello", 'Hello',...
  - 리스트(list) : [1, 2.2, "Hello"]
  - 튜플(tuple) : (5.5, -3, "Hello")
  - 집합(set) : {1, 2, 3, 4, 5}
  - 사전(dictionary, dict) : {'val1':1, 'val2':2}

차후 설명

데이터 형 (Data Types)

- 정수형 (int)
  - 정수형은 소수점이 없는 숫자 데이터(100, -123, 0,....)
  - int는 기본적인 정수 데이터 형식
  - 파이썬 버전 3부터는 정수의 크기에 제한이 없음(이론적으로)

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

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

데이터 형 (Data Types)

- 여러 진법으로 정수형 표현

10진수	2진수	16진수
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

데이터 형 (Data Types)

- 여러 진법으로 정수형 표현

```
>>> hex(0)
'0x0'
>>> hex(255)
'0xff'
>>> a = 0xFF
>>> a
255
>>> b = 0x20
>>> b
32
>>> bin(0)
'0b0'
>>> bin(8)
'0b1000'
>>> a = 0b1001
>>> a
9
>>> b = 0b11111111
>>> b
255
```

```
>>> oct(8)
'0o10'
>>> oct(10)
'0o12'
>>> oct(64)
'0o100'
>>> a = 0o10
>>> a
8
>>> b = 0o12
>>> b
10
>>> c = 0o100
>>> c
64
```

hex(), bin(), oct()
함수들의 결과 값은
문자열 데이터형임

데이터 형 (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: 1.531e+35, 3.54e–64 등과 같이 지수형(exponential form)으로 표시하는 방법

```
>>> 3.1e+8
310000000.0
>>> 3.1e-3
0.0031
```

- 실수를 표현할 때 오차가 발생할 수 있음(실수 저장 방법 때문)

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

데이터 형 (Data Types)

- Boolean 형 (Boolean type)
  - True, False 값을 저장(참, 거짓을 의미)
  - Boolean 형은 단독으로 사용하기 보다 if 조건문 이나 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 결합하라는 의미

데이터 형 (Data Types)

- 내장 함수 type( )
  - 객체(object)의 데이터 형(자료형)을 알려주는 함수
  - 변수(variable)가 어떤 데이터 형을 참조하는지 확인할 수 있음

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

  - 같은 변수에 어떤 데이터 형의 자료도 저장할 수 있음

데이터형(자료형) 변환

- 정수, 실수, 문자열 등의 데이터형들은 다른 데이터형으로 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 값은 정수로 변환
  - float() 함수
    - 실수 형태 또는 정수 형태의 문자열을 실수로 변환
    - 정수 값은 실수로 변환
  - str() 함수
    - 실수나 정수 값을 문자열로 형 변환

```
>>> int("34.5")
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    int("34.5")
ValueError: invalid literal for int() with base 10: '34.5'
>>> int("34") + int(3.8)
37
>>> print("CT class " + str(10) + " 반")
CT class 10 반
```

여러 변수 할당문 (Multiple Assignments)

- 다수의 변수들이 동시에 하나의 객체를 참조할 수 있게 함

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

- 여러 변수에 서로 다른 값들을 동시에 할당 가능 (동시 할당문)

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

같은 의미의 명령어

```
>>> a = 5; b = 4.1; c = "Hello!"
```

한 줄에 두 개 이상의 명령어를 입력할 때는 semicolon(;)으로 구분해야 함

여러 변수 할당문 (Multiple Assignments)

- 변수 num1과 num2의 값을 서로 바꾸는 코드임

script code

```
num1 = 10
num2 = 20  print(
num1, num2)  tmp
= num1  num1 =
num2  num2 = tm
p  print(num1, nu
m2)
```

같은 의미의 명령어

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

출력

```
10 20
20 10
```