파이썬 프로그래밍

파이썬 언어의 기본 문형

한국기술교육대학교
온라인평생교육원

▣ 파이썬 예약어 및 내장 함수

1. 예약어

• 예약어 (또는 키워드)
  - 파이썬에서 이미 문법적인 용도로 사용되고 있기 때문에 변수명 등의 식별자로 사용하면 안 되는 단어들

- 예약어는 Reserved(예약된) Words 또는 키워드라고 함
- 파이썬에서 이미 사용되고 있는(용도가 예약된) 단어들
- 이미 문법적인 용도로 사용 → 신택스(Syntax)
- 코딩할 때 활용할 예약어를 변수에 활용하면 안됨
- 예약어를 변수에 활용 → 에러는 없으나 고유 기능은 사라짐

▣ 파이썬 예약어 및 내장 함수

2. 예약어의 종류 알아보기

```
import keyword
print keyword.kwlist
print
print len(keyword.kwlist)
```

```
['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec',
'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print',
'raise', 'return', 'try', 'while', 'with', 'yield']
```

- 파이썬의 예약어 종류를 알아보는 방법은?
- keyword 모듈을 불러옴(import)
- keyword 모듈이 지원하는 kwlist를 출력(print)
- 이 때 print도 예약어이므로 다른 식별자로 사용하면 안됨
- print만 실행 → 한 줄을 띄워줌
- len → 특별한 모듈 추가 없이 사용할 수 있는 내장 함수
- keyword 모듈에 있는 kwlist에 몇 개의 단어가 있는지 알아봄
- 프로그램을 수행하면 콘솔에 수행결과가 나타남
- 수행결과는 문자열이 나열된 리스트 형태
- 리스트 안에 나열된 문자들 → 예약어
- 예약어의 개수는 31개(len 함수의 결과)
- 파이썬의 버전에 따라 예약어 종류 달라짐
- 파이썬 버전2.7에서는 31개 예약어 지원함
- 앞서 알아본 len이 내장 함수(Built-in Function)
- import keyword에서 keyword가 모듈
- 모듈을 불러올 때는 import(내장 함수) 사용
- 현재 프로그램에 사용된 예약어 → import, print
- keyword 모듈은 기본 인터프리터 환경에 추가 안됨
- 파이썬을 설치하면 수많은 모듈이 설치됨
- import keyword → 그 중 keyword 모듈을 사용하겠다
- 모듈 안에 있는 함수는 모듈을 import 해야 사용 가능

▣ 파이썬 예약어 및 내장 함수

3. 내장 함수

- 별도의 모듈(Module)의 추가 없이 기본적으로 제공되는 함수들
- 참고사이트
  - 내장(Built-in) 함수: http://docs.python.org/2/library/functions.html
- 대표적인 내장 함수
  - abs, max, min, pow, chr, str, range, type, …
- abs(x) : 수치형 자료 x에 대해 x의 절대값을 반환하는 함수

- 내장 함수는 모듈 추가 없이 활용 가능
- 파이썬 공식 홈페이지의 내장 함수 안내 페이지
- abs → 수치형 자료를 절대값으로 반환하는 내장 함수

```
print abs(3)
print abs(-3)
```

```
3
3
```

- max(s)
  - 시퀀스 자료형(문자열, 리스트, 튜플)을 입력받아 그 자료가 지닌 원소 중 최대값을 반환하는 함수

- max → 주어진 자료 중 최대값을 반환하는 내장 함수

▣ 파이썬 예약어 및 내장 함수

3. 내장 함수

```
print max(1,2)
print max([1, 2, 3])
print max("python")
```

```
2
3
y
```

- print max(1, 2) → 1과 2 중 더 큰 2를 반환
- print max([1, 2, 3]) → 리스트의 원소 중 제일 큰 3을 반환
- print max("python") → 문자열 중 아스키 코드값이 가장 큰 문자 반환
- max → 주어진 자료 중 최소값을 반환하는 내장 함수

• min(s)
  - 시퀀스 자료형(문자열, 리스트, 튜플)을 입력받아 그 자료가 지닌 원소 중 최소값을 반환하는 함수

```
print min(1,2)
print min([1, 2, 3])
print min("python")
```

```
1
1
h
```

- print min(1, 2) → 1과 2 중 더 작은 1을 반환
- print min("python") → 문자열 중 아스키 코드값이 가장 작은 문자 반환

▣ 파이썬 예약어 및 내장 함수

3. 내장 함수

• pow(x,y)
  - 수치형 자료형 x, y에 대해 x의 y승을 반환하는 함수

- pow(a, b) → a의 b승 값을 반환

```
print pow(2,4)
print pow(3, 3)
print pow(2, -1)
```

```
16
27
0.5
```

- print pow(2, 4) → 2의 4승, 즉, 16을 출력
- print pow(3, 3) → 3의 3승, 즉, 27을 출력
- print pow(2, -1) → 2의 -1승, 즉, 0.5를 출력

• chr(i)
  - 정수 형태의 ASCII코드 값을 입력으로 받아 그에 해당하는 문자를 반환하는 함수
  - 인수 i의 범위: 0부터 255까지

- chr → 아스키 코드값을 문자로 변환해 주는 함수

▣ 파이썬 예약어 및 내장 함수

3. 내장 함수

```
print chr(97)
print chr(65)
print chr(48)
```

```
a
A
0
```

- print chr(97) → 아스키 코드값이 97인 문자 → a
- print chr(65) → 아스키 코드값이 65인 문자 → A
- print chr(48) → 아스키 코드값이 48인 문자 → 0
- 이 때 0은 숫자 0이 아니라 문자로써의 0

```
• str(object)
  - 임의의 객체 object에 대해 해당 객체를 표현하는 문자열을 반환하는 함수
```

- str(a) → a를 출력(print)하면 어떻게 나타나는지를 반환

```
print str(3)
print str([1, 2])
```

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

- str(3) → 3이라는 객체를 출력했을 때 나타나는 결과 → 3
- str([1,2]) → [1, 2]를 출력했을 때 나타나는 결과 → [1, 2]
- str → 해당 객체를 표현하는 문자열로 반환해주는 함수

▣ 파이썬 예약어 및 내장 함수

3. 내장 함수

- range([start,]stop[,step])
  - 수치형 자료형으로 start, stop, step 등을 입력받아 해당 범위에 해당하는 정수를 리스트로 반환하는 함수
- 인수가 하나(stop)인 경우
  - 0부터 stop-1까지의 정수 리스트를 반환한다.
- 인수가 두 개(start, stop)인 경우
  - start부터 stop-1까지의 정수 리스트를 반환한다
- 인수가 세 개(start, stop, step)인 경우
  - start부터 stop-1까지의 정수를 반환하되 각 정수 사이의 거리가 step인 것들만 반환한다.

```
print range(10)
print range(3, 10)
print range(3, 10, 2)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[3, 4, 5, 6, 7, 8, 9]
[3, 5, 7, 9]
```

- range(10) → 0부터 10-1까지의 정수 리스트를 반환
- range(3, 10) → 3부터 10-1까지의 정수 리스트를 반환
- range(3, 10, 2) → 3부터 10-1까지 2개씩 건너 띈 정수 리스트를 반환
- range(3, 10, 2) → [3, 5, 7, 9]
- 5 – 3 = 2(step) → 앞, 뒤의 차이가 2씩 나게 함
- 7 – 5 = 2(step) → 앞, 뒤의 차이가 2씩 나게 함
- 9 – 7 = 2(step) → 앞, 뒤의 차이가 2씩 나게 함
- range 함수는 각각 원소의 차이가 step값만큼 나도록 할 수 있음
- range([start], stop, [step]) → start와 step은 생략 가능
- range(10) → stop만 10으로 기술한 것
- range(10) → 10-1까지의 정수 리스트를 반환
- start가 생략되었을 경우 0부터 시작
- range(a, b, c) → a부터 b-1까지 c씩 차이 나게 정수 리스트를 반환

▣ 파이썬 예약어 및 내장 함수

3. 내장 함수

```
print type(-1)
print type('abc')
print type([1, 2, 3])
```

```
<type 'int'>
<type 'str'>
<type 'list'>
```

- type(a) → a의 자료형을 반환
- print type(-1) → -1의 자료형을 반환 → int(정수형)
- print type('abc') → 'abc'의 자료형을 반환 → str(문자형)
- print type([1, 2, 3]) → [1, 2, 3]의 자료형을 반환 → list(리스트)
- type 함수를 사용하여 객체의 자료형을 알 수 있음

파이썬 프로그래밍

파이썬 언어의 기본 문형

한국기술교육대학교
온라인평생교육원

▣ 파이썬 식별자와 변수 사용

1. 식별자 만드는 법

- 파이썬 식별자는 변수, 함수, 모듈, 클래스 또는 객체를 식별하는데 사용되는 이름이다.
- 식별자의 조건
  - 대소문자 구별함
  - 식별자는 문자 A-Z 또는 a-z와 언더바(_)로 시작할 수 있다.
  - 식별자 첫 시작을 제외하고 식별자 내에 숫자(0~9)를 사용할 수 있다.
  - 특수문자 @, $, %등은 식별자에 올 수 없다.
  - 예를 들어 다음과 같은 것은 식별자가 될 수 없음 : 1abc, @file, %x

- 파이썬 식별자: 변수, 함수, 모듈, 클래스, 객체 식별에 사용되는 이름
- 가장 흔히 쓰는 식별자 → 변수 이름
- 프로그래밍 하면서 식별자를 정하게 됨
- 식별자 정할 때의 조건 1. 대소문자를 구별함
- 식별자를 정할 때 a(소문자)와 A(대문자)는 다름
- 변수 a와 A가 다르게 인식됨
- 조건 2. 식별자는 문자 또는 언더바(_)로 시작함
- aaa, AAAA, _aaa, _AAAA 모두 식별자로 사용 가능
- 조건 3. 식별자는 맨 처음을 제외하고 숫자 포함 가능
- _aaa2, _a2aa 모두 식별자로 사용 가능
- 그러나 2_a2aa는 식별자로 사용 불가
- 조건 4. 특수 문자는 식별자로 사용 불가
- 식별자로 사용 불가 → 1abc(숫자가 맨 앞), @file, %x(특수 문자 포함)

▣ 파이썬 식별자와 변수 사용

2. 변수명 만들 때 조심할 점

• 예약어, 내장함수, 모듈 이름을 변수명으로 만드는 일이 없도록 할 것

- 변수명 만들 때 예약어, 내장 함수, 모듈 이름으로 만들지 않도록 주의
- 예약어, 내장 함수, 모듈 이름은 이미 활용되고 있기 때문

```
print str(12345)
```

```
12345
```

- str은 주어진 객체를 출력해주는 함수

```
str = 'abc'


print str(12345)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback(most recent call last)
<ipython-input-2-ad06f2f248af> in <module>()
      1 str = 'abc'
      2
----→ 3 print str(12345)

TypeError: 'str' object is not callable
```

- 만약 str을 다른 것으로 정의하게 되면?
- 본래 str 함수가 가지고 있던 기능을 잃어버림

▣ 파이썬 식별자와 변수 사용

3. 변수의 생성 및 사용

• 파이썬에서 변수가 생성되는 시점은 해당 변수에 임의의 값이 할당될 때이다.

```
a = 1
print a
```

```
1
```

- 파이썬과 다른 언어의 큰 차이점 → 변수 생성 시 타입(type)을 적지 않음
- 변수에 값이 할당될 때 변수의 타입이 정해짐
- type 함수를 통해 a의 타입을 확인해보면 int(정수)로 나타남
- 이 때 a의 타입은 값이 할당이 되었을 때 int로 정해진 것
- 만약 a에 실수형(1.0)을 할당하면 타입은 float(실수)으로 나타남
- 동일한 a라도 할당된 값에 따라 타입이 바뀜
- 즉, 변수의 타입을 처음부터 정할 수 없음
- 이를 파이썬의 동적 특성, 동적 변수 할당이라 함

• 변수의 생성 없이 곧바로 사용할 수 없다.

```
print b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback(most recent call last)
<ipython-input-22-ab3a5d8kfk1075> in <module>( )
----→ 1 print b

NameError : name 'b' is not defined
```

- 변수 b 생성 않고 바로 print b → 에러 발생
- 변수 b가 생성되지 않았기 때문에 이해할 수 없음
- 반드시 변수에 값을 할당한 후 사용해야 함

▣ 파이썬 식별자와 변수 사용

4. 변수의 삭제

• del 이라는 예약어 사용

```
b = 2
print b

del b
print b
```

```
2

---------------------------------------------------------------------------
NameError                                 Traceback(most recent call last)
<ipython-input-24-026dc2933af4> in <module>( )
      3
      4 del b
----→ 5 print b

NameError : name 'b' is not defined
```

- del → 변수를 삭제 할 때 사용
- b에 2를 할당한 후 print b → 2가 정상적으로 출력됨
- del b → b라는 변수를 삭제
- 그 후 print b → 에러 발생

파이썬 프로그래밍

파이썬 언어의 기본 문형

한국기술교육대학교
온라인평생교육원

▣ 파이썬 기초 문형

1. 주석문

```
#이것은 주석입니다.
import sys #이것도 주석입니다.
```

- 주석문: #(샵) 뒤에 오는 문장
- 주석문은 첫 번째 줄에 올 수도 있고
- 일반적인 파이썬문 뒤에 주석문이 올 수도 있음
- 한 줄을 모두 차지하거나 명령문 뒤에 달렸거나 상관없이
- #(샵) 기호와 함께 사용되면 주석

▣ 파이썬 기초 문형

2. 연속라인

```
a = 1
b = 3
if (a == 1) and  \
(b == 3) :
        print 'connected lines'
```

```
connected lines
```

- a가 1이고 b가 3이면 'connected lines'라는 문구를 출력하는 예제
- 이 때 if부터 3까지는 한 줄이 되어야 함
- \ (백슬래시) → 나눠진 줄을 한 줄로 인식하도록 함
- \ (백슬래시)는 코딩이 길어져 한 화면에 나타나지 않을 때 사용

```
a = 1
b = 3
if (a == 1) and (b == 3) :
    print 'connected lines'
```

```
connected lines
```

▣ 파이썬 기초 문형

3. 할당문

```
a = 1
b = a
```

- 할당문 → 등호(=)를 사용한 연산
- a = 1 → a에 1을 할당
- b = a → b에 a를 할당 → a가 1이므로 b도 1
- 그러므로 a, b 모두 1을 할당

```
1 + 3 = a
```

```
File "<ipython-input-15-3aa35fdab4b7>", line 1
  1 + 3 = a
SyntaxError : can't assign to operator
```

- 주의! 등호 왼쪽에 표현식(1+3)이 오면 안됨(변수가 와야 함)
- 표현식은 등호 오른쪽에 올 수 있음
- 이 때 오른쪽의 표현식이 평가됨

```
a = 1
a = a + 1
print a
```

```
2
```

- a = a + 1 → 현재 a가 1이므로 1 + 1, 즉, 2 → a = 2
- 표현식은 등호 오른쪽에 올 수 있음

▣ 파이썬 기초 문형

3. 할당문

```
c, d = 3, 4
print c, d

x = y = z = 0
print x, y, z

e = 3.5; f = 5.6
print e, f
```

```
3 4
0 0 0
3.5 5.6
```

- c, d = 3, 4 → c와 d에 각각 3과 4를 할당
- x = y = z = 0 → 맨 오른쪽부터 이해해야 함
- z = 0 → z에 0을 할당
- y = z → z의 값을 y에 할당 → y = 0
- x = y → y의 값을 x에 할당 → x = 0
- print x, y, z → 0 0 0
- 세미콜론(;): 하나의 문장이 끝났음을 의미
- 두 문장을 한 줄에 이어 쓸 때 세미콜론(;)을 사용
- 세미콜론(;)은 프로그램 가독성 문제로 자주 활용되지 않음

▣ 파이썬 기초 문형

3. 할당문

• 두 변수의 값을 swap하는 방법

```
e = 3.5; f = 5.6
e, f = f, e
print e, f
```

```
5.6 3.5
```

- 스왑(swap): 값을 바꾸는 것을 의미
- e, f = f, e → e에 f를 할당, f에 e를 할당 → 값을 스왑하는 문장
- print로 결과 확인하면 두 값이 바뀐 것을 알 수 있음

• 아래에서 b = c + d는 식(Expression)이 아니라 문(Statement)이기 때문에 a에 할당될 수 없다.

```
a = (b = c + d)
```

```
File "<ipython-input-29-79d71c957091>", line 1
    a = (b = c + d)

SyntaxError: invalid syntax
```

- a = (b = c + d) → 잘못된 식
- (b = c + d) → 식이 아니라 문
- 문 자체가 a에 할당될 수 없음

▣ 파이썬 기초 문형

4. 확장 할당문

```
a = 1
a += 4
print a
```

```
5
```

- 확장 할당문: +=, -=, *=, /=
- a += 4는 a = a + 4와 동일
- a += 4 → a에 4를 더한 후 다시 a에 할당

```
a = 10
a -= 3
print a
```

```
7
```

- a -= 3 → a에 3을 뺀 후 다시 a에 할당

```
a = 10
a *= 2+3
print a
```

```
50
```

- a *= 2 + 3 → a에 2 + 3을 곱한 후 다시 a에 할당(2 + 3을 먼저 수행)
- 2 + 3 = 5, 그 다음 5 * 10 = 50, a에 50을 할당

▣ 파이썬 기초 문형

5. 객체와 할당

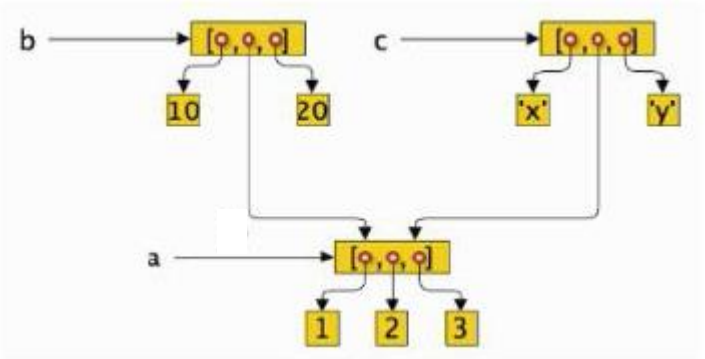
- 객체의 변수는 해당 객체의 레퍼런스를 지니고 있음
- a = 1이라는 Statement에서 a는 이름, 1은 객체이며 a 변수는 1이라는 객체를 가리킨다.
  - 즉, a 변수는 1 객체의 레퍼런스를 지니고 있음

- a = 1 → a는 변수 이름, 1은 객체
- a = 1 → a라는 변수가 어딘가에 있는 객체 1을 가리키는 것을 의미
- 가리킨다 → 레퍼런스를 지니고 있다

```
a = [1, 2, 3]
b = [10, a, 20]
c = ['x', a, 'y']

print a
print b
print c
```

```
[1, 2, 3]
[10, [1, 2, 3], 20]
['x', [1, 2, 3], 'y']
```



- a = [1, 2, 3] → 어딘가에 있는 1, 2, 3을 가리키는 형태
- 리스트 내부에는 1, 2, 3을 가리키는 레퍼런스 값이 존재
- 1, 2, 3을 가리키는 레퍼런스가 존재하는 리스트를 a가 가리킴

▣ 파이썬 기초 문형

5. 객체와 할당

- b도 어딘가에 있는 10, 20을 가리키는 레퍼런스 값이 존재
- 리스트 안에 있는 a는 [1, 2, 3]을 가리키는 형태
- c가 어떤 리스트를 가리키고 있음
- 리스트는 x, y라는 문자를 가리킴
- 리스트 안에 있는 a는 [1, 2, 3]을 가리키는 형태
- 파이썬에서는 모든 것이 다 객체
- 식별자 변수는 객체를 가리키는 형태
- a = [1, 2, 3] → print a → [1, 2, 3]
- b = [10, a, 20] → a가 문자가 아닌 리스트[1, 2, 3]으로 나타남
- c = ['x', a, 'y'] → a가 문자가 아닌 리스트[1, 2, 3]으로 나타남
- a = [1, 2, 3]에서 1은 인덱스 0, 2는 인덱스 1, 3은 인덱스 2

```
a[1] = 1000

print a
print b
print c
```

```
[1, 1000, 3]
[10, [1, 1000, 3], 20]
['x', [1, 1000, 3], 'y']
```

- a[1] = 1000 → 인덱스 1에 1000을 넣음
- 인덱스 1에 1000을 넣음 → 2 대신 1000을 넣음
- 리스트 a를 가리키고 있는 b, c 모두 변경됨
- b에 a가 연결되어 있기 때문에 a가 변하면 b도 변함
- 마찬가지로 c에 a가 연결되어 있기 때문에 a가 변하면 c도 변함

파이썬 프로그래밍

파이썬 언어의 기본 문형

한국기술교육대학교
온라인평생교육원

▣ 콘솔 입출력

1. 콘솔 입력

• 콘솔(Console)
  - 윈도우에서는 Command창, 리눅스/맥에서는 Terminal창
  - 각 IDE(예, 이클립스)에서는 별도의 콘솔 창이 제공됨
• raw_input(): 문자열 입력 내장함수

- 콘솔(Console): 윈도우의 커맨드 창, 리눅스/맥의 터미널 창을 의미
- 콘솔창: 내용을 입력 또는 출력하는 창
- 이클립스에서도 콘솔창을 제공하고 있음
- 프로그램 작성 후 실행하면 콘솔창에 내용이 출력됨
- 콘솔을 통해 내용을 입력 받는 함수 → raw_input(), input()

```
name = raw_input('name?')
```

```
name? 홍길동
```

- raw_input() → 콘솔창에서 문자열을 입력받는 내장 함수
- name = raw_input('name?') → 일단 콘솔창에 name? 나타남
- 콘솔창에 나타난 name? 뒤에 홍길동 입력 → name에 홍길동 할당됨

```
print name
```

```
홍길동
```

- 그 후 print name → name에 할당된 홍길동이 출력됨
- raw_input() → 콘솔창에서 문자열을 입력받는 내장 함수
- raw_input('텍스트') → 콘솔창에 텍스트 나타남
- raw_input() → 콘솔창에 1도 입력 가능
- 이 때 1은 숫자 1이 아닌 문자로써의 1
- 파이썬에서는 캐릭터와 문자열을 구분하지 않고 문자열로 취급
- 따라서 문자열을 입력받는 raw_input() 함수임에도 1이 입·출력된 것

▣ 콘솔 입출력

1. 콘솔 입력

• int(): 문자열을 정수로 변환하는 내장함수

```
k = int(raw_input('int : '))
print k
```

```
int : 12
12
```

- int(raw_input()) → 콘솔창에 입력된 값을 int 내장 함수에 넣음
- int() → 문자열을 숫자(정수)로 바꿈
- 이런 경우 raw_input() 값을 숫자 형태로 받아야 함
- raw_input('int:') → 콘솔창에 int: 나타남
- 콘솔창에 숫자 10 입력 후 엔터 → print k 실행 → 숫자 10 출력
- print k + 10 → k가 문자가 아닌 숫자이므로 연산이 가능함
- k는 int 내장 함수가 반환한 숫자 10이 들어있으므로 연산 가능
- raw_input() → 문자열을 입력받음 / int() → 문자열을 정수로 반환함

• input(): 정수, 실수, Expression 입력 내장함수

```
i = input('int : ')
print i
```

```
int : 45
45
```

- input() → 문자열 입력받는 raw_input()과 달리 연산식까지 입력 가능
- 이전 예제와 같이 input() 내장 함수를 활용

```
k = input('expr : ')
print k
```

```
expr : 30 + 50
80
```

▣ 콘솔 입출력

2. 콘솔 출력

• print 화면에 자료를 출력하는 보편적인 statement
• 여러 자료를 한꺼번에 출력할 때에는 콤마(,)를 사용

```
print 4 + 5, 4 - 2
```

```
9 2
```

- 콘솔창에 단순한 숫자가 아닌 식을 입력
- 콘솔창에서 식 입력 후 엔터 → 식이 계산된 결과 출력되어 변수에 할당
- 콘솔 출력은 print 예약어(Keyword)를 활용하여 가능
- print 4 + 5, 4 – 2 → 9 2
- print 수행 시 콤마(,) → 한 칸 띄어주는 역할
- 콤마(,) → 여러 자료를 한 번에 출력할 때 결과 사이를 한 칸 씩 띄어줌

• 세미콜론(;)은 순차적으로 입력된 각 statement를 분리함

```
print 1; print 2
```

```
1
2
```

- 세미콜론(;) → 각 문장의 줄을 분리
- 파이썬에서 한 문장은 기본적으로 한 줄을 차지
- 세미콜론(;) → 한 줄에 있는 두 문장을 두 줄에 나타나게 함
- 기본적으로 print는 한 줄을 바꾸어 줌, 즉, 개행이 됨

▣ 콘솔 입출력

2. 콘솔 출력

• 기본적으로 print는 마지막에 줄바꿈을 하지만 콤마(,)가 마지막에 있으면 줄바꿈을 하지 않는다.

```
print 1,
print 2
```

```
1 2
```

- 그렇다면 줄 바꿈 사이에 콤마(,)를 넣으면?
- 줄이 바뀌지 않고 한 줄에 두 결과가 나타남
- 콤마(,)는 한 칸만 띄어주기 때문에 개행이 없어지고 한 줄에 나타난 것

• + 연산자는 숫자와 문자열에 대한 연산을 지원하지 않는다.

```
print 12 + 'spam'
```

```
---------------------------------------------------------------------------
NameError                                 Traceback(most recent call last)
<ipython-input-71-84ee898a197b> in <module>( )
----→ 1 print 12 + 'spam'

TypeError : unspported operand type(s) for +: 'int' and 'str'
```

- 숫자와 문자를 더하면 어떻게 될까?
- 에러가 발생함
- 더하기(+) 연산자는 숫자와 문자를 합쳐주지 않음
- 파이썬은 수치형 자료와 문자형 자료를 합할 수 없음

```
print '12' + 'spam'
```

```
12spam
```

- print 12 + 'spam' → print '12' + 'spam' (숫자를 문자로 수정)
- print 12 + 'spam' → print str(12) + 'spam' (str 함수 사용)
- str → 숫자 12를 문자 12로 바꿔주는 내장 함수