



- 프로그래밍 언어의 일종
  - 인터프리터 방식의 스크립트 언어 중 하나
  - 배우기 쉽고 결과도 바로 확인할 수 있어 초보자에게 적합
- 1989년, 귀도 반 로섬이라는 프로그래머가 크리스마스에 취미 삼아 만든 언어
- 사전적인 의미는 비단뱀으로 로고도 파란색과 노란색 비단뱀 두 마리가 서로 얹혀 있는 형태



### 재미있는 이야기

- 귀도 반 로섬은 1989년 크리스마스에 할 일이 딱히없어 파이썬을 개발하였다고 함
- “파이썬“이라는 이름은 코메디 프로그램 “Monty Python’s Flying Circus”에서 유래
- 파이썬의 원래 의미는 그리스 신화에 나오는 거대한 뱀
- 귀도는 파이썬 개발자를 찾는 헤드헌터로 부터 취업 제안 메일을 받은 적이 있음

객체 지향적(?)

동적(?) 타이핑 언어

## 객체 지향적 언어

: 실행 순서가 아닌 단위 모듈(객체) 중심으로 프로그램을 작성  
하나의 객체는 어떤 목적을 달성하기 위한 행동(method)와  
데이터(attribute)를 가지고 있음

## 동적 타이핑 언어

: 프로그램이 실행하는 시점 프로그램이 가지고 있는 데이터에  
대한 타입을 검사함

## ❖ 컴파일 방식과 인터프리터 방식 프로그래밍 언어

	언어	실행 속도	이식성
컴파일 방식	C, C++, Pascal, Ada	CPU가 실행할 수 있는 기계 코드로 컴파일되므로 <b>실행 속도 빠름</b>	원래 애플리케이션이 개발된 CPU/운영체제와 다른 CPU/운영체제로 옮기는 경우, 대체로 코드를 그대로 사용할 수 없음. <b>이식성 낮음</b>
인터프리터 방식	BASIC, python, Ruby, PHP, Perl	애플리케이션을 실행할 때 마다 인터프리터가 소스 코드를 기계 코드로 번역하는 절차를 거치므로 <b>실행 속도 느림</b>	인터프리터만 대상 CPU/운영체제를 지원한다면 애플리케이션 코드는 변경할 필요 없이 어떤 환경에서나 실행 가능. <b>이식성 높음</b>

## ❖ 파이썬의 특징 및 장점

- 배우기 쉽고 사용하기 쉬움
- 플랫폼에 독립적이므로 어느 운영체제에서나 사용 가능
- 오픈 소스이므로 무료로 사용 가능
- 고급 라이브러리를 대거 포함하며 서드 파티 라이브러리가 풍부
- 객체지향적이며 클래스를 지원
- C언어와의 접착성이 좋아 혼합 프로그래밍 가능
- 사물인터넷과 잘 연동
- 강력한 웹 프레임워크 사용 가능

## ❖ 단점

- 느린 속도
  - 스크립트 언어이므로 컴파일러 언어보다 느림 (C언어보다 최소 10배 이상 느린 성능)  
→ 이를 보완하기 위하여 많은 파이썬 패키지를 최적화시키고 있음
- 모바일 컴퓨팅 분야에 지원이 약하고 하드웨어 제어 등과 관련된 부분 사용이 어려움
  - 시스템 프로그래밍, 고성능 응용 프로그램, GUI, 모바일 분야에는 잘 사용하지 않음

# 인간 지향적인 간단한 문법



# 파이썬 - Why?

JAVA

```
int    myCounter = 0;
String myString = String.valueOf(myCounter);
if (myString.equals("0")) ...
```

PYTHON

```
myCounter = 0
myString = str(myCounter)
if myString == "0": ...
```

```
// print the integers from 1 to 9
for (int i = 1; i < 10; i++)
{
    System.out.println(i);
}
```

```
print the integers from 1 to 9
for i in range(1,10):
    print i
```

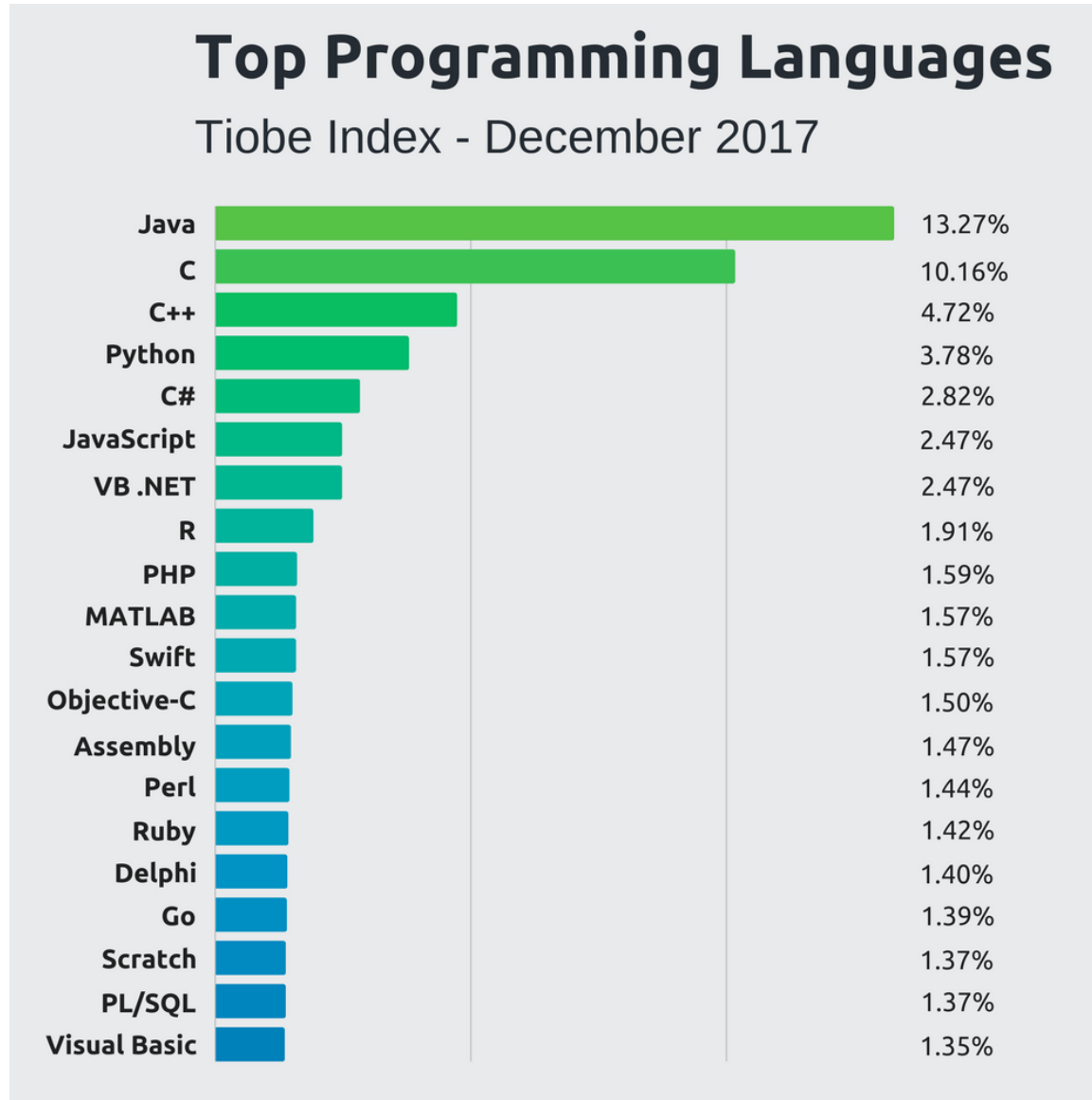
Brace 여부

프로그래밍을 몰라도 해석이 가능한 문법의 언어

# 다양한 라이브러리 넓은 활용범위











파이썬은 대부분의 라이브러리가 이미  
다른 사용자에 의해서 구현되어 있음 (특히 통계, 데이터 분석)

## 파이썬 - Why?



Indicator of the popularity of  
programming languages

# 파이썬 - Why?

Aug 2021	Aug 2020	Change	Programming Language		Ratings	Change
1	1			C	12.57%	-4.41%
2	3	▲		Python	11.86%	+2.17%
3	2	▼		Java	10.43%	-4.00%
4	4			C++	7.36%	+0.52%
5	5			C#	5.14%	+0.46%
6	6			Visual Basic	4.67%	+0.01%
7	7			JavaScript	2.95%	+0.07%
8	9	▲		PHP	2.19%	-0.05%
9	14	▲▲		Assembly language	2.03%	+0.99%
10	10			SQL	1.47%	+0.02%

## 파이썬 - Google loves Python.

*Python is big at Google. Since I don't want to bother with getting this blog reviewed by Google, I can't go into much detail, **but it's at a secure 3rd place after C++ and Java, and it's being used for everything from build tools to managing ads**. Name your third-party Python module and someone at Google is probably using it. So this is an exciting environment -- I get to see first-hand what truly large-scale Python development is like, and where the pain points are.*

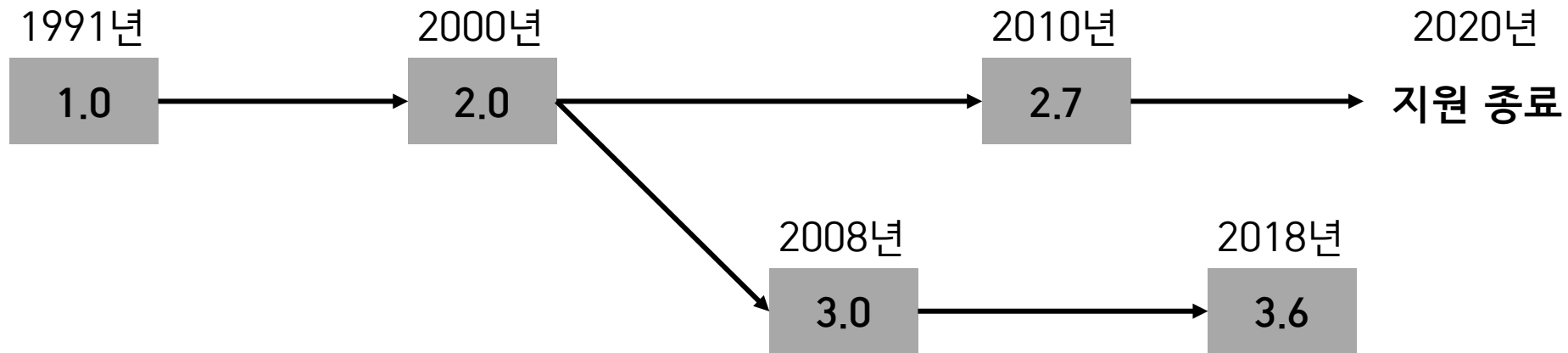
*- Guido van Rossum -*

## ❖ 개발환경 및 설치

- Linux Ubuntu ??

```
$ python3 --version
# 버전 확인
$ sudo apt-get install python3
# 설치
```

## ❖ 설치 시 주의할 점



- 설치 시 반드시 3.x 버전 이용
  - 2.X대와 3.x대는 호환되지 않음

## ❖ 대화식 모드

- REPL (Read - Eval - Print Loop)
- 파이썬 코드를 한 줄씩 입력해 가면서 테스트 해 볼 수 있는 입력창
- 짧은 코드를 테스트하기에 적합
- 상대적으로 간단

## ❖ 스크립트 모드

- 텍스트 파일에 일련의 명령을 작성하여 한꺼번에 순차적으로 실행하는 방식
- 길고 복잡한 코드를 작성하기에 적합
- 텍스트 파일 저장 시 .py 확장자 사용



## ❖ 대화식 모드

- ‘python3’ 을 입력하면 대화식 모드로 실행 가능
- 저장이나 실행 명령을 내릴 필요 없이 명령만 입력하면 결과를 즉시 볼 수 있음
- 매번 직접 입력해야 하기 때문에 길고 복잡한 프로그램을 짜기에는 불편
- 명령을 저장할 수 없고 약간씩 변형하거나 에러를 수정하기에 번거로움
- 대화식 모드에서 나오고 싶을 때는 ‘exit()’ 이용

```
>>> 2 * 3
6
>>> (1 + 2) * 3
9
>>> a = 3
>>> b = 4
>>> a + b
7
```

## ❖ 스크립트 모드

- ‘python3 *filename*’ 으로 실행 가능
- 텍스트 파일에 일련의 명령을 작성하여 순차적으로 실행하는 방식
- 명령어가 길고 복잡해도 상관없음
- 파일에 저장되어 있어 필요할 때 약간씩 바뀌가며 실행해 볼 수 있음
- 이미 작성해 놓은 코드를 복사하여 재사용하기도 편리
- 스크립트 파일 저장 시 .py 확장자로 저장

```
$ cat helloWorld.py
print('Hello, World!')
$ python3 helloWorld.py
Hello, World!
```

## ❖ 소스의 형식

```
>>> print('hello') # 정상적인 출력
hello
>>> print('hello') # error! 들여쓰기 규칙이 틀림.
  File "<stdin>", line 1
    print('hello')
    ^
IndentationError: unexpected indent
>>> 1 + 2 # python에서는 세미콜론을 붙이지 않음
3
>>> 1 + 2; # 그렇지만 붙여도 에러가 나지 않는다.
3
>>> a = 3; b = 4; a + b # 한 줄에 여러 명령을 쓰려면 붙여야 함
7
>>> # This is comment. # 주석은 다음과 같이 '#' 문자 사용
```

### ❖ 출력

```
print(출력 내용 [, sep = 구분자] [, end = 끝 문자])
```

```
>>> print(3 + 4) # 기본적인 출력문
7
>>> value = 1234
>>> print(value) # 변수도 출력 가능
1234
>>> print(value * 2)
2468
>>> a = 12; b = 34
>>> print(a, b) # 변수 두 개를 연이어 출력하면 두 값 사이에 공백 문자를 넣어서 띄움
12 34
>>> print(a, b, sep = ', ') # separator 지정도 가능
12,34
>>> print(a, b, sep = '') # 붙여쓰려면 sep = '' 이용
1234
>>> print(a, b, sep = ', ', end = '!\n') # 내용 마지막에 출력할 문자열 지정도 가능
12, 34!
```

### ❖ 입력

```
변수 = input ('질문 내용')
```

```
>>> name = input('이름이 뭐예요? ') # 기본적인 입력문. 값은 변수에 저장됨
이름이 뭐예요? 양갱이
>>> print(name)
양갱이
>>> age = input('몇 살이에요? ')
몇 살이에요? 4
>>> print(age)
4
>>> print(age + 1) # error! 사용자로부터 받은 값은 항상 문자열.
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

### ❖ 입력

```
변수 = int(input('질문 내용'))
```

```
>>> age = input('몇 살이에요? ')
몇 살이에요? 4
>>> print(age)
4
>>> print(age + 1) # 값을 그냥 받으면 문자열이므로 에러
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>> age = int(input('몇 살이에요? ')) # 받을때 int() 함수를 이용하면 정수 취급
몇 살이에요? 4
>>> print(age)
4
>>> print(age + 1) # 정상 출력
5
```

## ❖ 변수 사용

- 파이썬은 별도의 타입을 지정하지 않음

```
>>> job = 'student'
>>> age = 24
>>> print(job)
student
>>> print(age)
24
```

- 실행 중에 변수의 타입을 바꿀 수 있음 (Dynamic Typing)

```
>>> value = 1234
>>> type(value)
<class 'int'>
>>> value = 'computer'
>>> type(value)
<class 'str'>
```

## ❖ 정수형

- 별도의 길이 제한 없이 메모리가 허용하는 한 얼마든지 큰 수를 표현할 수 있음
- 98, 1200, -89, 123456789012345, 0x1a, 0o17, 0b1101 등

## ❖ 실수형

- 소수점 이하의 정밀한 값을 표현. 부호가 있어 음수도 표현 가능
- 67.5, 89.75, -3.1415, 9.46e12 등

## ❖ 복소수형

- 허수 표현도 가능
- $1+2j$ ,  $3+4j$  등



## ❖ 문자열

```
>>> 'Hello, World!' # 문자열은 따옴표로 감싼다.
'Hello, World!'
>>> "Hello, World!" # 큰 따옴표, 작은 따옴표 상관 없음
'Hello, World!'
>>> "Hello, World' # 섞어 쓰는 건 안됨
File "<stdin>", line 1
    "Hello, World'
        ^
SyntaxError: EOL while scanning string literal
```

## ❖ 긴 문자열

```
>>> print('''I think cats are so cute. # 긴 문자열은 따옴표 세 개로 감싼다.
... Because cats have triangular ears.
... So I love cats.''' )
I think cats are so cute.
Because cats have triangular ears.
So I love cats.
```

## ❖ 진위형

```
>>> a = 5
>>> b = a == 5 # Boolean type
>>> print(b)
True
>>> a = None # C언어의 null과 같은 의미
>>> print(a)
None
```

## ❖ 여러 컬렉션

- 여러 개의 값을 모아서 저장. 다른 언어의 배열이나 구조체와 유사

리스트(List) : 여러 개의 값을 하나의 변수에 순서대로 모아놓은 것

튜플(Tuple) : 리스트와 거의 비슷하지만 실행 중에 값을 변경할 수 없음

- 이 외에 딕셔너리, 집합 등의 컬렉션이 존재

```
>>> member = ['아이린', '슬기', '조이', '웬디', '예리'] # 리스트
>>> print(member)
['아이린', '슬기', '조이', '웬디', '예리']
>>> for m in member: print(m, " 짱") # for문으로 리스트의 요소를 하나씩 꺼내 순회 가능
아이린 짱
슬기 짱
조이 짱
웬디 짱
예리 짱
>>> member = ('아이린', '슬기', '조이', '웬디', '예리') # 튜플
>>> for m in member: print(m, " 짱") # 리스트와 똑같이 동작함
```

## ❖ 대입 연산자

- 파이썬은 변수를 선언하지 않고 바로 사용하며 대입되는 값에 따라 변수의 타입 결정
- 대입에 의한 초기화가 선언을 겸함

```
>>> a = 3
>>> s = 'computer'
>>> f = 3.1415
>>> a = (1 + 2) * 3
>>> b = a * f
```

## ❖ 산술 연산자

연산자	설명
+	더하기
-	빼기
*	곱하기
/	나누기
%	나머지
**	거듭제곱
//	정수 나누기

## ❖ 산술 연산자

```
>>> 3 + 4 # 기본적인 연산들
7
>>> 4 - 3
1
>>> 2 ** 10 # 거듭제곱 연산도 가능
1024
>>> candy = 400
>>> money = 1000
>>> money / candy # 일반적인 '/' 연산을 하면 항상 실수 return
2.5
>>> money // candy # 정수 나누기를 하려면 반드시 '/' 사용
2
```

### ❖ 복합 대입 연산자

- +=, -=, \*= 등
- But, 파이썬에는 ++, -- 연산자가 없음!

```
>>> print(money)
1000
>>> money += 1000 # ok
>>> print(money)
2000
>>> money *= 10 # ok
>>> print(money)
20000
>>> money++ # error!
      File "<stdin>", line 1
        money++
            ^
SyntaxError: invalid syntax
```

## ❖ 문자열 연산

문자열에는 +와 \* 연산자 사용 가능

- + : 두 문자열을 연결
- \* : 문자열을 정수 횟수만큼 반복

```
>>> s1 = 'I love '  
>>> s2 = 'cat '  
>>> print(s1 + s2) # 문자열끼리 + 연산 : 두 문자열을 연결  
I love cat  
>>> num = 22  
>>> print(s1 + num) # 문자열과 정수 + 연산 : error! 문자열과 정수는 연결할 수 없음  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: can only concatenate str (not "int") to str  
>>> print(s2 * 3) # 문자열의 * 연산 : 문자열을 정수 횟수만큼 반복  
catcatcat  
>>> print(s1 * s2) # 문자열끼리 * 연산 : error! 문자열과 문자열은 곱할 수 없음  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: can't multiply sequence by non-int of type 'str'
```



### ❖ 정수와 문자열

문자열과 정수를 연결하고 싶으면?

→ 타입 변환 필요! `str()` 함수를 이용하자

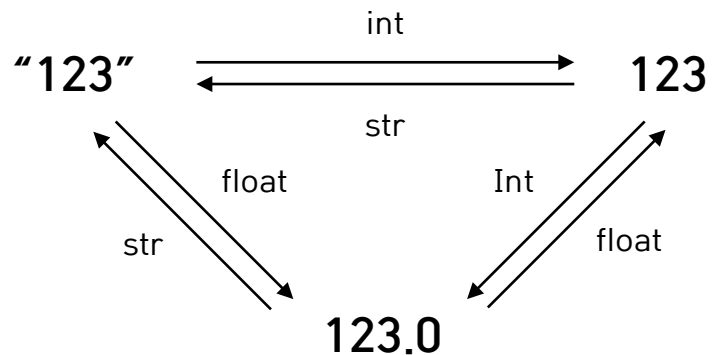
```
>>> print('I have ' + 2 + ' legs.') # error! 문자열과 숫자는 그냥 연결할 수 없음
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>> print('I have ' + str(2) + ' legs.') # str() 함수를 이용하면 가능
I have 2 legs.
```

반대로 문자열을 정수형으로 바꿀 수도 있다 → `int()` 함수 이용

```
>>> print(10 + '12') # error!
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> print(10 + int('12')) # int() 함수를 이용하면 문자열을 정수형으로 바꿀 수 있음
22
```

## ❖ 그 밖의 타입 변환 함수

- String - float - int



- 그 밖에 bool, list, tuple, dict 등의 변환 함수가 있음
- 컬렉션끼리도 서로 형태를 바꿔가며 변환 가능

## 조건문 if 조건문

### ❖ if문

조건문 뒤에 반드시 콜론

```
if 조건1:
    명령1
elif 조건2:
    명령2
else:
    명령3
```

생략 가능

명령문 앞쪽은 들여쓴다

파이썬은 형식이 엄격하므로 이 형태를 반드시 지켜야 함

## 조건문 if 조건문

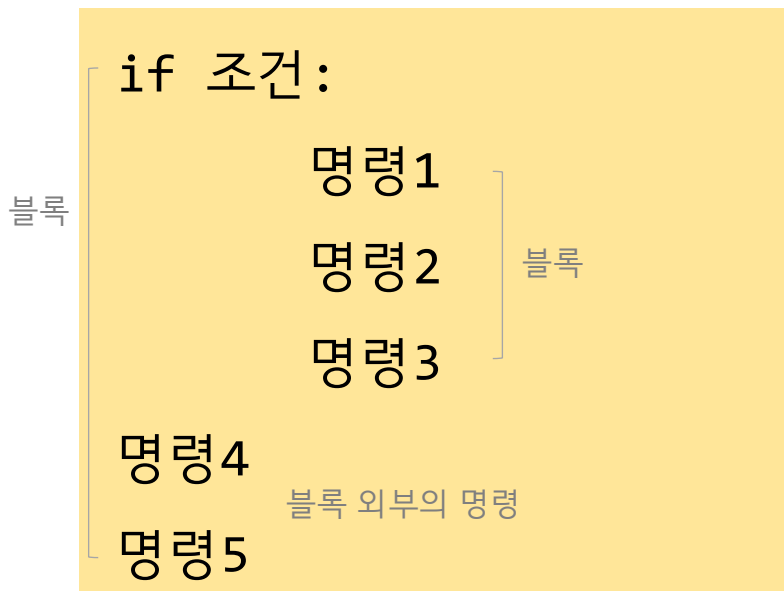
### ❖ if문

```
score = int(input('Input your score : '))

if score == 100:
    print('Your grade is A. Good job!')
elif score > 80:
    print('Your grade is B.')
elif score > 60:
    print('Your grade is C.')
else:
    print('Your grade is F. You have to take a class again...')
```

```
Input your score : 100
Your grade is A. Good job!
Input your score : 95
Your grade is B.
Input your score : 37
Your grade is F. You have to take a class again...
```

## ❖ 블록 구조



- 함께 실행 되는 하나의 코드 덩어리
- 들여쓰기로 블록을 구분
- 들여쓰기가 한 칸이라도 어긋나면 오류 발생
- 블록 안에 다른 블록이 들어갈 수 있음
- 내부의 블록은 외부의 블록에 종속적
- 파이썬 코드 전체를 하나의 블록으로 볼 수 있음