

# 1. 파이썬 기본

비주얼프로그래밍

2019년 2학기

# 파이썬

## ◆ 1990년 개발된 언어

- 귀도 반 로섬
- 휴가 중
- 고대 신화 파르나소스 산의 동굴에 살던 뱀
- 인터프리터형 언어, 플랫폼 독립적, 객체지향, 동적 타이핑
- 2001년부터 파이썬 소프트웨어 재단에서 관리
- 대학, 교육기관, 연구기관, 산업계
- 현재 미국에서 가장 많이 쓰는 언어 : 분야는?

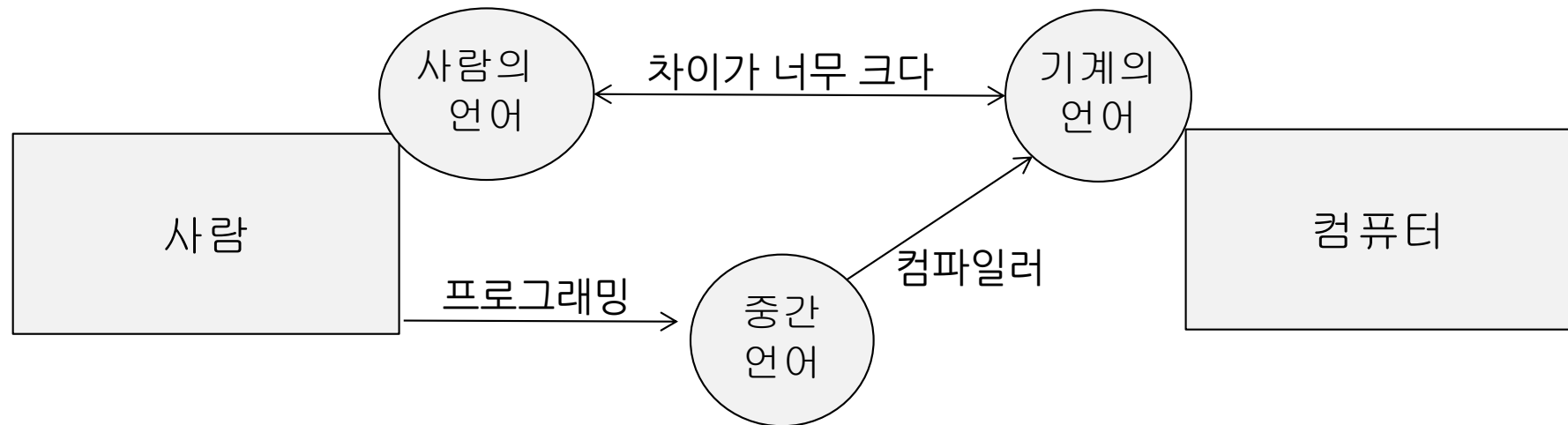


# 프로그래밍 언어

## ◆ 프로그램을 작성하기 위한 언어

- 사람이 컴퓨터에게 시키고 싶은 내용을 표현하기 위한 표기법
- 그 발전 단계에 따라 기계어, 어셈블리어, 고급언어로 분류

x 파이썬 코드를 실행하려면,  
파이썬 컴파일러가 있어야 함!

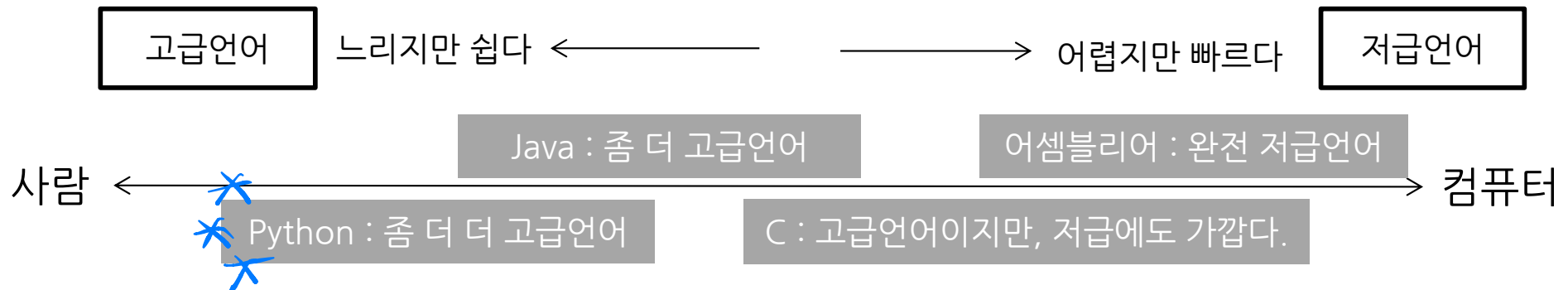


비교적 사람도 잘 이해할 수 있고 쉽게 기계의 언어로 변형할 수 있는 언어

컴파일러 : 중간 언어로 작성된 명령을 기계의 언어로 번역하는 도구

# 프로그래밍 언어

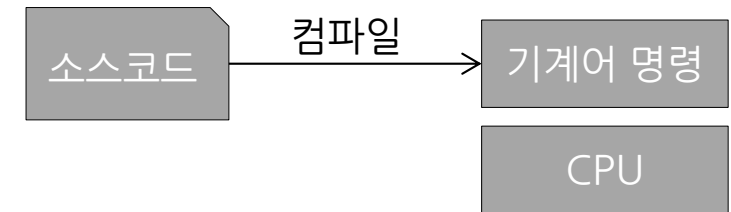
- ◆ 고급 언어(고수준 언어, high-level language)
  - 영어와 비슷한 구문으로 표현되며, 사람이 읽고 쓰기 쉽다.
  - 사람에게 가까울수록(사람이 쉽게 이해할 수 있음) 더 고급 언어이다.
- ◆ 저급 언어(저수준 언어, low-level language)
  - 기계에 가까운 언어. 2진수로 이루어진 기계어
  - 수행은 빠르지만, 사람이 이해하기 매우 어렵다.
- ◆ 대부분의 프로그래밍 언어는 고급언어이며, 최근 언어는 사람에게 더 가까워지는 추세이다.



# 컴파일러와 인터프리터

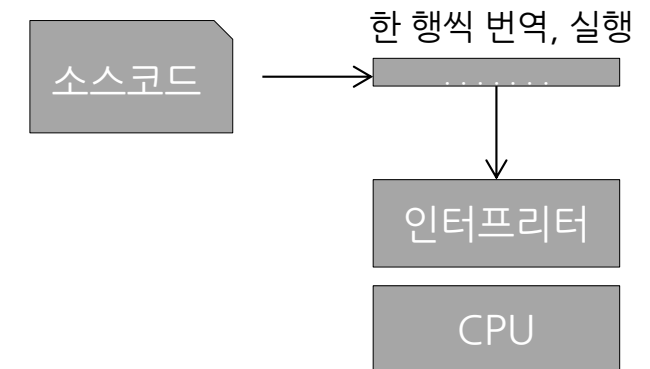
## ◆ 컴파일러

- 소스코드 전체를 기계어로 번역한 후, CPU에서 실행
- 성능이 좋다. 컴파일에 시간이 다소 소요된다.
- C, C++ 등



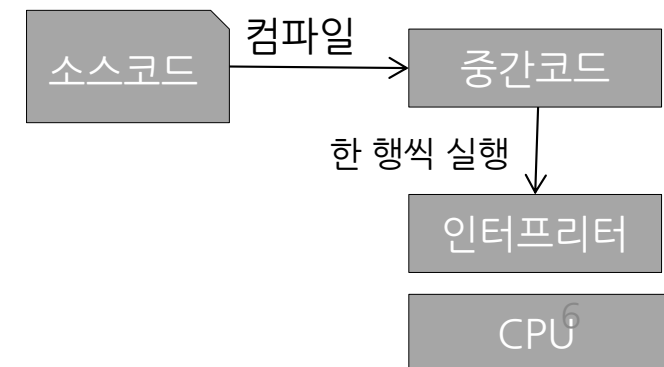
## ◆ 인터프리터

- 실시간으로 인터프리터가 소스코드를 한 행씩 번역 후 실행
- 다양한 기능을 제공하고, 융통성이 있다.
- 인터프리터가 항상 동작해야 한다. 성능에 한계가 있다.
- Python



## ◆ 중간 형태

- 소스코드 전체를 중간코드로 일괄 번역한 후, 한 행씩 실행
- 컴파일러와 인터프리터의 장점.
- Java



어차피 CPU는 기계어만 실행 가능

# 파이썬

## ◆ 인터프리터 언어이다.

- 중요하지는 않다. 왜냐하면 상황은 바뀌기 마련이니까.

## ◆ 프로그램을 실행하려면 인터프리터를 설치해야 한다.

- 무료

## ◆ 인터프리터형 언어는 왜 느릴까?

- 한 줄씩 번역한 후에 실행해야 하므로

\* 느린기준이 개발자기준!  
여러분들은 느리다고 느끼실수 없으실것!

## ◆ 느린데 왜 파이썬을 쓸까?

- 컴퓨터가 아주 빨라졌다. 그래서 실행도 비교적 빠르다.
- 다른 언어에 비해 실행은 느리지만, 프로그램 개발 시간은 짧다.
- 프로그래밍이 쉬워 누구나 프로그램을 만들 수 있다.

## ◆ 이해하기 쉽다.

```
if 4 in [1,2,3,4]:print("4가 있다")
```

“목록에 4가 있다면 있다고 출력한다.”

C 언어로 한다면?

```
for( i = 0 ; i < sizeof(Set) ; i++ )  
    if ( 4 == Set[i] ) printf("4가 있다\n");
```

## ◆ 문법이 쉽고 간단하다.

- 기본 문법의 종류의 수가 적다.
- 기본적인 자료형, 함수, 클래스, 라이브러리를 기본 제공한다.

=> 기존 언어의 기능은 포함하면서 단순화하였다.

믿을 수 없다면 C++을 배우자

# 특징

## ◆ 무료

- 언어
- 인터프리터
- 파이썬 관계된 것이 모두 무료는 아니다.

★ 없는것을 다 만들어내는데 아니라,  
남들이 만들어놓은것을 잘 가져다쓰면  
되는것임!! → 이것을 '모듈이라함'

## ◆ 간결하다.

- 최선의 길 1개만 제공하자.
- 결국 이해하기 쉽고 공동작업, 유지보수가 쉽다.

★ 외부모듈은 어떤언어로든 만듦.  
C++로 외부모듈을 만들면,  
파이썬 프로그래밍에다 집어넣어  
성능을 더 좋게 만들수 있음!

## ◆ 강력하다.

- 강력하게 만드는 요소는 외부 라이브러리(모듈)
- 좋은 **외부 모듈**이 많다. (왜일까?)

★ 파이썬 프로그래밍의 절반이  
파이썬에 어떤 좋은 부품이 있는지  
아는것이 절반!

\*\*\*



# 파이썬 프로그램

```
#simple.py
```

```
languages = ['python', 'perl', 'c', 'java']
```

```
for lang in languages:
```

```
    ← tab ↗ if lang in ['python', 'perl']:
        ← tab ↗ print("%s needs interpreter" % lang)
    elif lang in ['c', 'java']:
        print("%s needs compiler" % lang)
    else:
        print("should not reach here")
```

들여쓰기(tab or space 4번)는  
 굉장히 중요 → 들여쓰기하지 않으면  
 오류!!!

줄을 잘 맞추었다(필수적이면  
서 가독성을 향상)

소문자만 사용되었지만, 대소  
문자를 구별한다(case  
sensitive).

❌ 이해할 필요 없어요!  
대충 이런식이라 보면 됨!

# 파이썬으로 할 수 있는 일

## ◆ 할 수 있는 일

- 웹 프로그래밍
- 수치 연산 프로그래밍
- 데이터베이스 프로그래밍
- 데이터 분석(빅데이터), 시각화
- 머신 러닝, 딥 러닝

여러분들의 관심사는  
대부분 여기에 있습니다! 예상!

## ◆ 할 수 있으나 적절하지는 않은 일

- 사물 인터넷(IoT)
- 일반 유틸리티, GUI 프로그래밍
- 성능이 중요한 일

## ◆ 할 수 없는 일

- 시스템 프로그래밍
- 모바일

# 간단한 명령 하기

실습

```
>>> 1+2 (1과2를 더해라)
```

```
>>> a=1
```

```
>>> b=2
```

```
>>> a+b
```

```
>>> a = "Python"
```

```
>>> print(a)
```

```
>>> 분위기 있는 음악 틀어줘
```

```
File "<stdin>", line 1
```

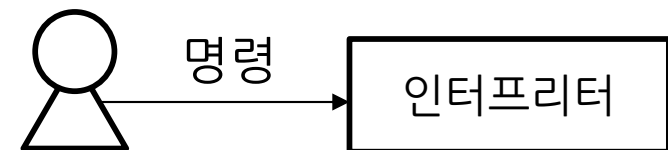
```
    분위기 있는 음악 틀어줘
```

```
    ^
```

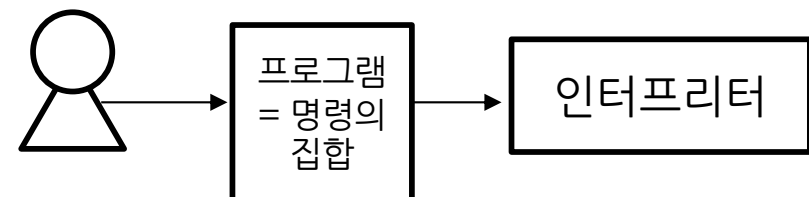
```
SyntaxError: invalid syntax
```

문법에러!

- 바로 명령을 입력하고 실행한다.
- 준비 작업 없이 바로 명령한다.
- 입력하는 명령이 모두 다 프로그램 코드이다.



명령을 모아놓으면 프로그램이다.



<= 에러  
Why 에러?  
(나중에 배울 것)

# 간단한 명령 하기



Terminal 혹은 명령 프롬프트에서 합니다!

구글 콜랩에서 하시면 됩니다!

```
>>> a = 3
```

```
>>> if a > 1 :
```

명령이 끝나지 않았다.

→ ...

```
print("a는 1보다 크다")
```

명령이 끝나지 않았다.

→ ...

공백 4개가 좋다.

줄바꿈 : 앞 명령을 끝내자

```
a는 1보다 크다.
```

1개여도 무관하나 공백 개수에 주의해야 한다.

... 은 인터프리터에게 직접  
입력할 때에만 표시된다.

↑  
실행 결과

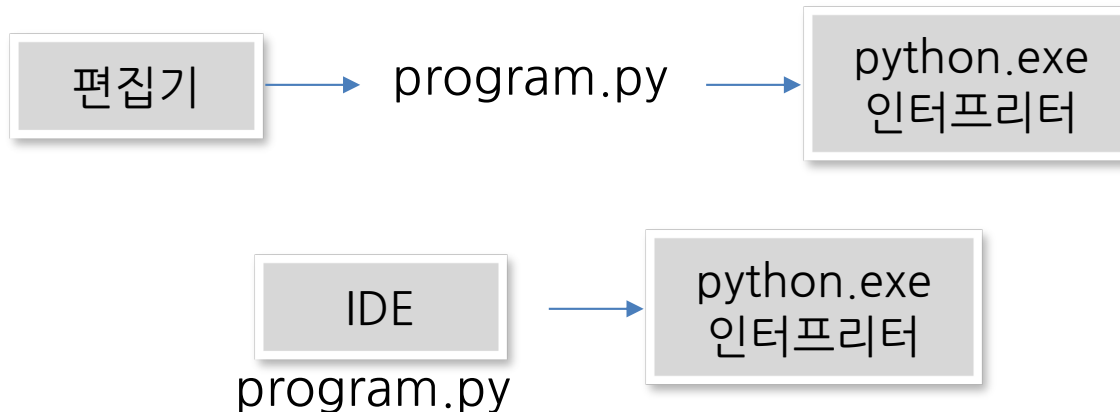
들여쓰기가 중요하다. Tab 키를 이용하자.

# 프로그램 편집기 / IDE

- ◆ 프로그램(명령)을 늘 한 행씩 입력할 수는 없다.
- ◆ 편집기로 프로그램을 입력한 후(프로그래밍) 한꺼번에 실행
- ◆ 윈도우 사용자라면 프로그램 입력할 때
  - Pycharm, iPython, IDLE(기본 내장)
  - 리눅스 사용자라면 vi, emacs

전반부에서는  
IDLE로도 충분!

더 능숙한 사용자가  
된다면



◆ 다음 파일을 작성하여 hello.py 로 저장하자.

python 프로그램 확장자

\* 한꺼번에 입력된 코드 실행하기

IDLE 실행 - New File - 저장 - run module 클릭!  
(단축키: F5 + fn(맥))

주석

```
#hello.py
```

```
"""
```

```
작성자 : 김승태
```

```
작성일 : 2017-12-18
```

```
기능 : 화면 출력
```

```
"""
```

```
print("Hello world")
```

```
print("python")
```

# : 한 줄 주석

""" """ 여러 줄 주석 :

↳ 많이 씁니다!!

↳ 주석은 코드 실행 안됨.  
코드 잘 알아보기 위해!

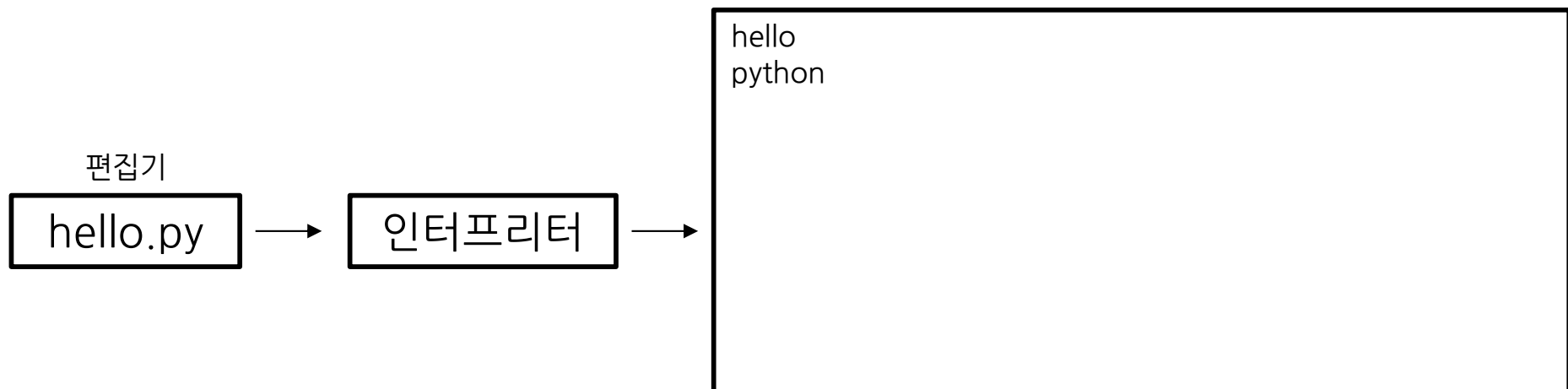
# IDLE를 쓸 때와 쓰지 않을 때

## 1. 직접 입력할 때

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18)
[MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>> print('hello')
hello
>>> print('python')
python
```

실행은 100% 동일  
프로그램 편집이 얼마나  
쉬운가의 차이일 뿐

## 2. \*.py를 실행할 때



# 함수 print 아주기본에 기본

◆ Python은 다양한 기능성 부품을 아주 아주 많이 제공한다.

– 이것을 함수라 한다. 사용법을 알아야 함수를 쓸 수 있다.

◆ print는 화면에 데이터를 출력하는 부품이다.

– print('hello')는 hello라는 문장을 화면에 표시하는 기능을 갖는 부품이다.

– 우리는 프로그램에서 이 기능(부품)을 가져와 사용했고, 이 부품에 'hello'라는 데이터를 공급했다.

