

# 파이선 프로그래밍 (CLTR268003)

---

Seolyoung Jeong, Ph.D.

경북대학교 IT대학 컴퓨터학부

# Lecture Overview

## ◆ 교과목 /교과목 번호

- 파이선 프로그래밍 (CLTR268 003)

## ◆ 강의 진행

- 이론 설명 및 실습
- 경북대 학습관리시스템 (<http://lms.knu.ac.kr/>)
  - 강의노트 및 공지사항
  - 실습 및 과제 제출

## ◆ 실습 진행

- PC실습실 (IT융복합관 309호)
- 개인PC(노트북 등)에도 반드시 설치할 것

## ◆ 튜터 (실습 및 과제 채점)

- 김재성 ([horsequake@knu.ac.kr](mailto:horsequake@knu.ac.kr))

# Syllabus

## ◆ Assessment :

- 중간고사 30%
- 기말 고사 30%
- 실습 및 과제 30%
- 출석 및 수업태도 10%

## ◆ 주의사항

- 수업일수 1/4이상 결석인 경우 F 학점 (학사 규정 참조)
- 결석 2회 혹은 지각 4~6회 시 -1점 감점
- 결석 3회 혹은 지각 7회 이상 시 -2점 감점
- 과제, 시험, 출석 등 부정행위 시 0점 처리
  
- C, C++, Java 등 기초 프로그래밍 경험 필수
- IT 전공자(컴퓨터,전자) 및 연계/융합전공 등 프로그래밍 관련 전공자

# Lecture Schedule (변경 가능)



차시	내용
01	Python 기본문법, 입출력 서식
02	기본연산자, 조건문, 반복문, 리스트, 튜플
03	문자열, 리스트 함수
04	함수, 모듈 및 기본 라이브러리
05	클래스
06	쓰레드 & 프로세스
07	중간고사
08	디버깅 및 예외처리, Turtle
09	Tkinter
10	웹크롤링 및 지도데이터 표현
11	파이선 기반 머신러닝 프로젝트
12	데이터 분류
13	모델 훈련
14	파이선 기반 딥러닝
15	기말고사



# 참고자료

- ◆ 본 강의는 공식 Python Tutorial Release 문서 (버전3.7)를 기반으로 진행됨 (<https://docs.python.org/3/index.html>)
- ◆ 참고사이트
  - <https://www.python.org/about/gettingstarted/>
  - <https://wiki.python.org/moin/BeginnersGuide/Programmers>
- ◆ 참고서적
  - <https://wiki.python.org/moin/IntroductoryBooks>
  - Python Fundamentals, Ryan Marvin, 2018
  - Python Crash Course: A Hands-On, Project-Based Introduction to Programming, Eric Matthes, 2015
  - Think Python: How to Think Like a Computer Scientist, Allen B. Downey, 2015
  - Learning Python, 5th Edition, Mark Lutz, 2013

---

**파이썬이란?**

# 파이썬(Python)이란?

- ◆ 1991년 귀도 반 로섬(Guido Van Rossum)이 발표
- ◆ 플랫폼 독립적인 인터프리터 언어이며,
- ◆ 객체 지향적, 동적 타이핑 대화형 언어
- ◆ 초기에는 C언어로 구현되었었음



Guido Van Rossum



Python Logo

파이썬은 ‘피톤’이라는 이름으로 알려진, 고대 그리스 신화에 나오는 거대한 뱀의 이름. 피톤은 Python을 고대 그리스어로 읽은 것이며, 영어를 그대로 읽으면 ‘파이선’이 됨.

사실, 파이썬이라는 이름은 파이썬을 만든 귀도 반 로섬(Guido van Rossum)이 자신이 좋아하는 영국 코미디 프로인 ‘몬티 파이선의 날아다니는 서커스(Monty Python’s Flying Circus)’에서 따왔다고 함. 물론 여기서의 파이선도 피톤을 의미.

# 왜 파이썬을 배우는가?

## ◆ 인간 지향적인 간단한 문법

### JAVA

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

```
for (int i = 1; i < 10; i++)
    System.out.println(i);
```

### Python

```
myCounter = 0;
myString = str(myCounter);
If myString == "0": ...
```

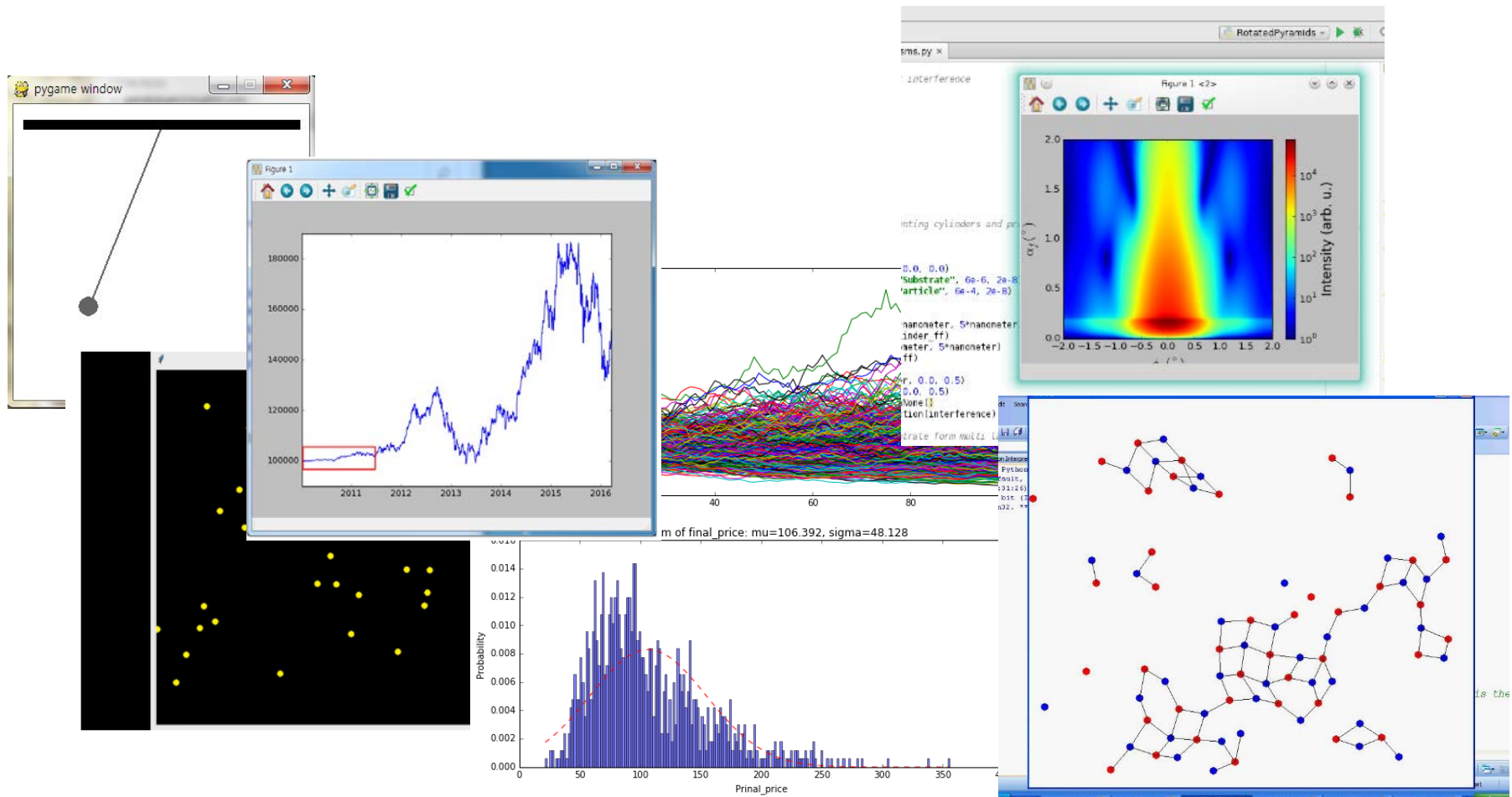
```
for i in range(1,10):
    print i
```

## ◆ 프로그래밍을 몰라도 해석이 가능한 문법의 언어



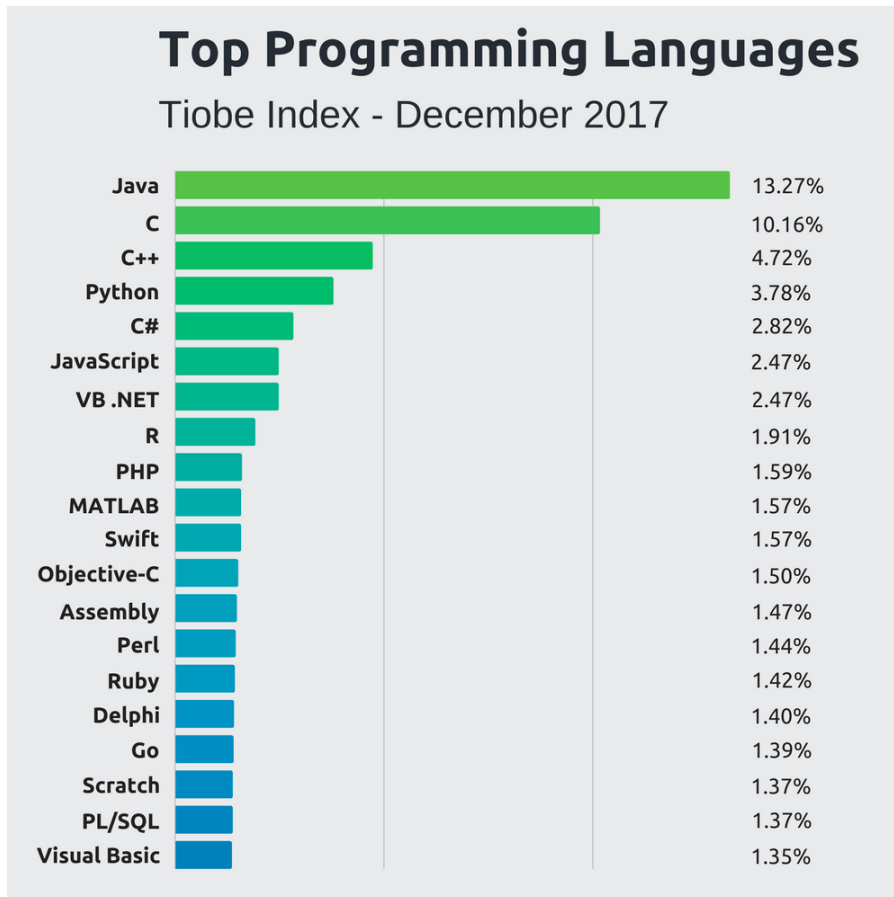
# 왜 파이썬을 배우는가?

- ◆ 다양한 라이브러리 넓은 활용 범위
- ◆ 파이썬은 대부분의 라이브러리가 이미 다른 사용자에게 의해서 구현되어 있음 (특히 인공지능, 통계, 데이터 분석, 그래프 등)



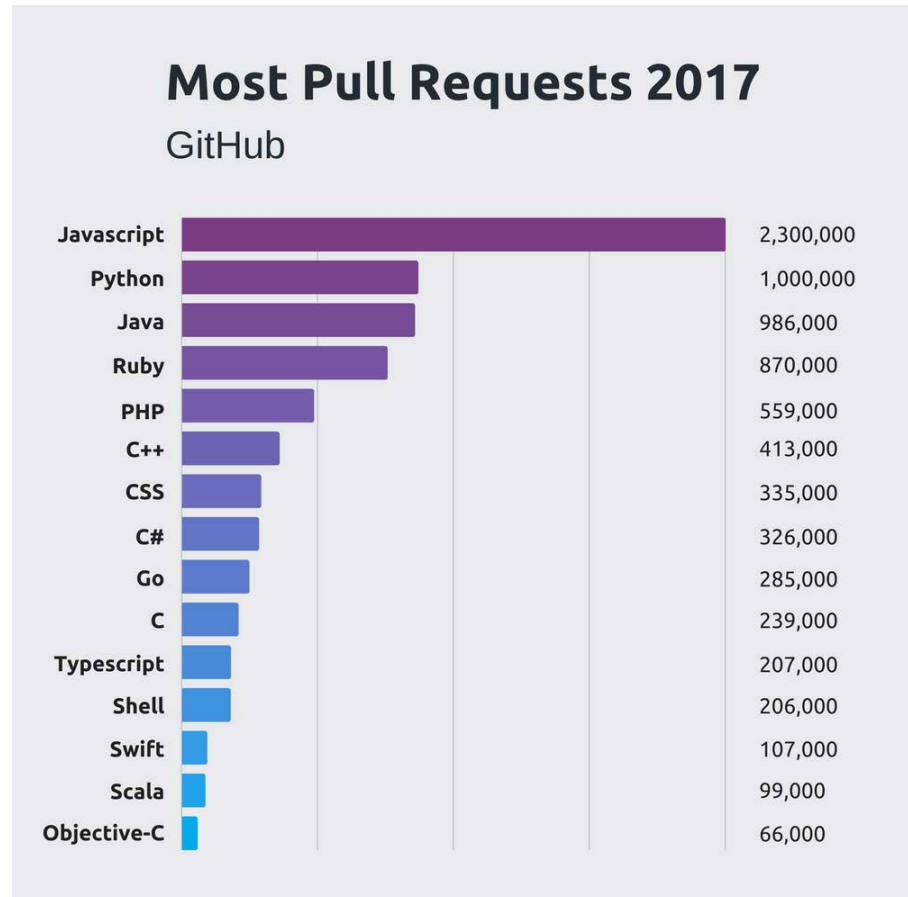
# 왜 파이썬을 배우는가?

## ◆ Top Programming Languages (2017)



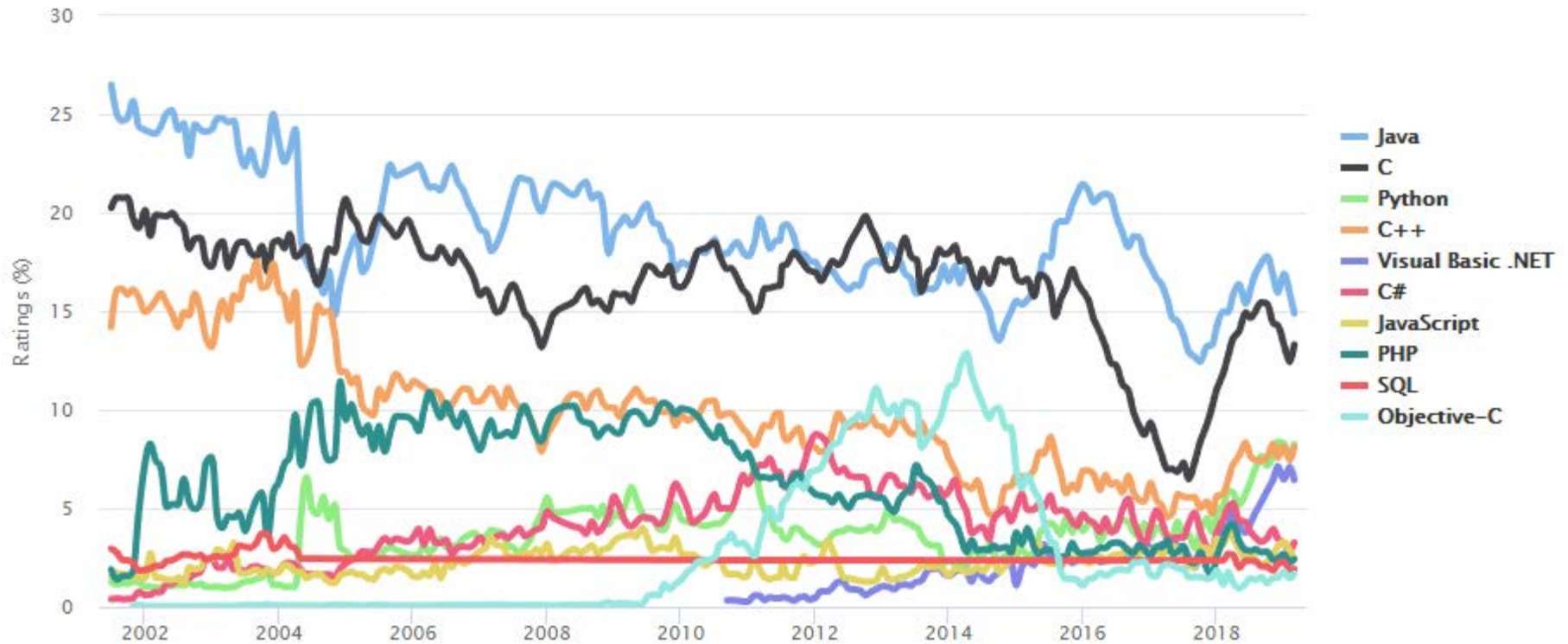
## ◆ Most Pull Requests in GitHub (2017)

- GitHub : 오픈소스 프로젝트 사이트



# 왜 파이썬을 배우는가?

## ◆ History of Programming Language Ratings



# 파이썬 설치 및 툴

# 파이썬 설치 (3.7.4)

◆ <http://www.python.org/>

The screenshot shows the Python.org homepage. At the top, there's a navigation bar with links: Python, PSF, Docs, PyPI, Jobs, and Community. Below this is the Python logo and a search bar. A red circle ① highlights the 'Downloads' link in the main navigation menu. A dropdown menu is open, showing options: All releases, Source code, Windows (highlighted with a red box and red circle ②), Mac OS X, Other Platforms, License, and Alternative Implementations. To the right of the dropdown, a section titled 'Download for Windows' is visible. It contains a link for 'Python 3.7.4' (highlighted with a red box and red circle ③), a note that Python 3.5+ cannot be used on Windows XP or earlier, and a link to view the full list of downloads. At the bottom of the page, there are four columns: 'Get Started' (with a power icon), 'Download' (with a download icon), 'Docs' (with a book icon), and 'Jobs' (with a briefcase icon). Each column has a brief description of its content.

Python

PSF

Docs

PyPI

Jobs

Community

python™

Donate

Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

# Python 3: Sim

>>> print("Hell

Hello, I'm Pyth

# Input, assign

>>> name = inpu

>>> print('Hi,

What is your na

Python

Hi, Python.

All releases

Source code

Windows

Mac OS X

Other Platforms

License

Alternative Implementations

Download for Windows

Python 3.7.4

Note that Python 3.5+ cannot be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)

Get Started

Whether you're new to

Download

Python source code and installers

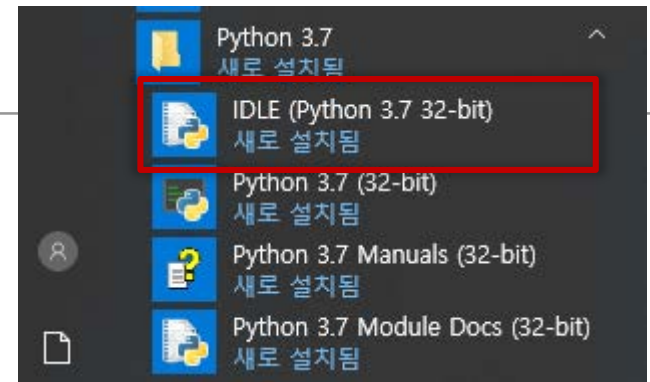
Docs

Documentation for Python's

Jobs

Looking for work or have a Python

# 기본 Python 툴



## ◆ IDLE (Interactive Development Environment)

- 파이썬 개발을 위한 셸 (대화형 인터프리터) – 기본 툴

## ◆ Python 3.7 (command line)

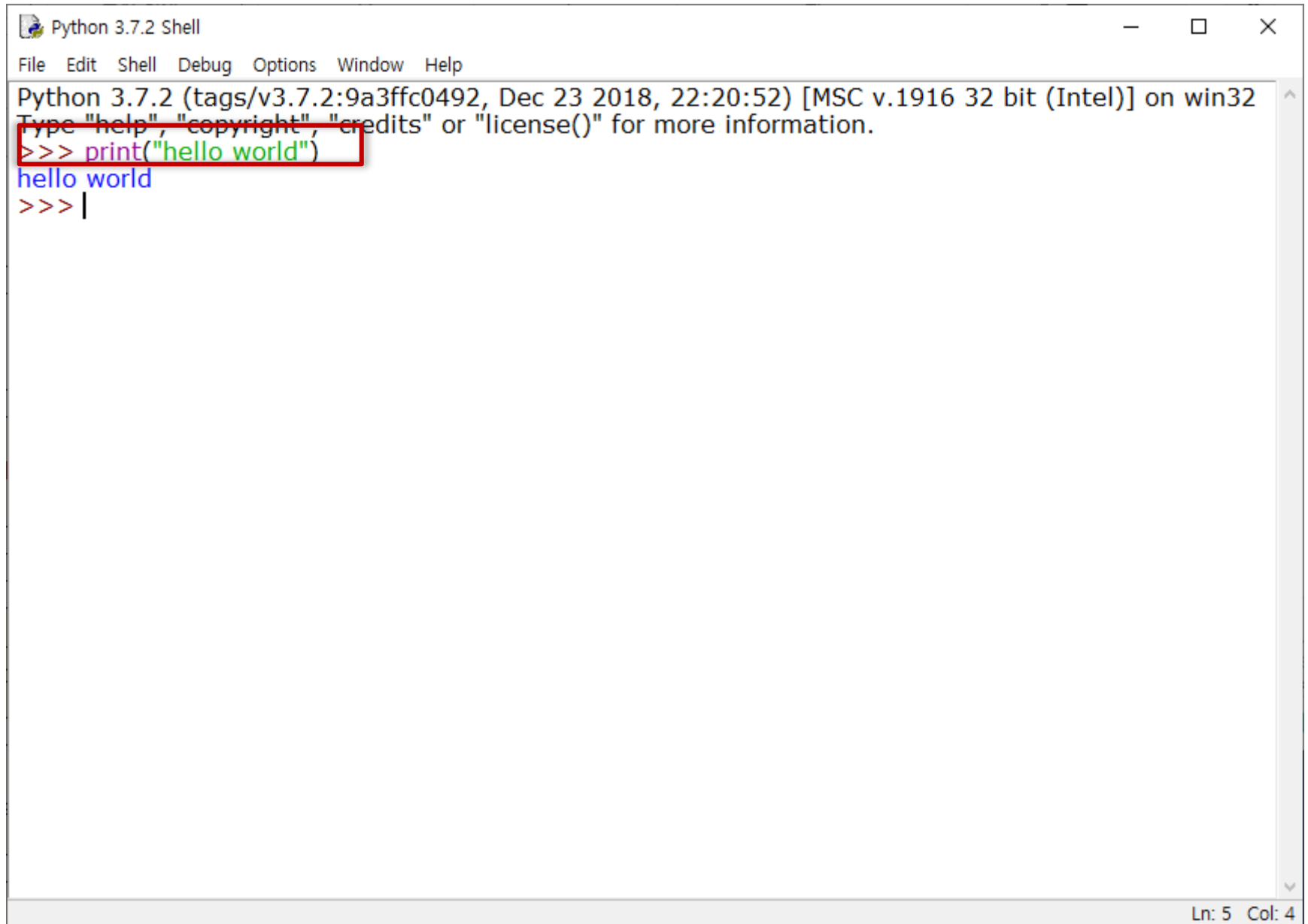
- IDLE과 유사한 기능. 윈도우 창이 아닌 명령 프롬프트에서 파이썬 인터프리터 동작

A screenshot of a Windows command prompt window. The title bar shows 'C:\Python34\python.exe'. The window content displays the Python 3.4.3 startup text: 'Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32'. Below this, it says 'Type "help", "copyright", "credits" or "license" for more information.' and shows the prompt '>>>'.

## ◆ Python 3.7 Manuals

## ◆ Python 3.7 Module Docs

# 파이썬 쉘에서 코딩 및 (즉시)실행



The image shows a screenshot of a Python 3.7.2 Shell window. The window has a title bar that says "Python 3.7.2 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following content:

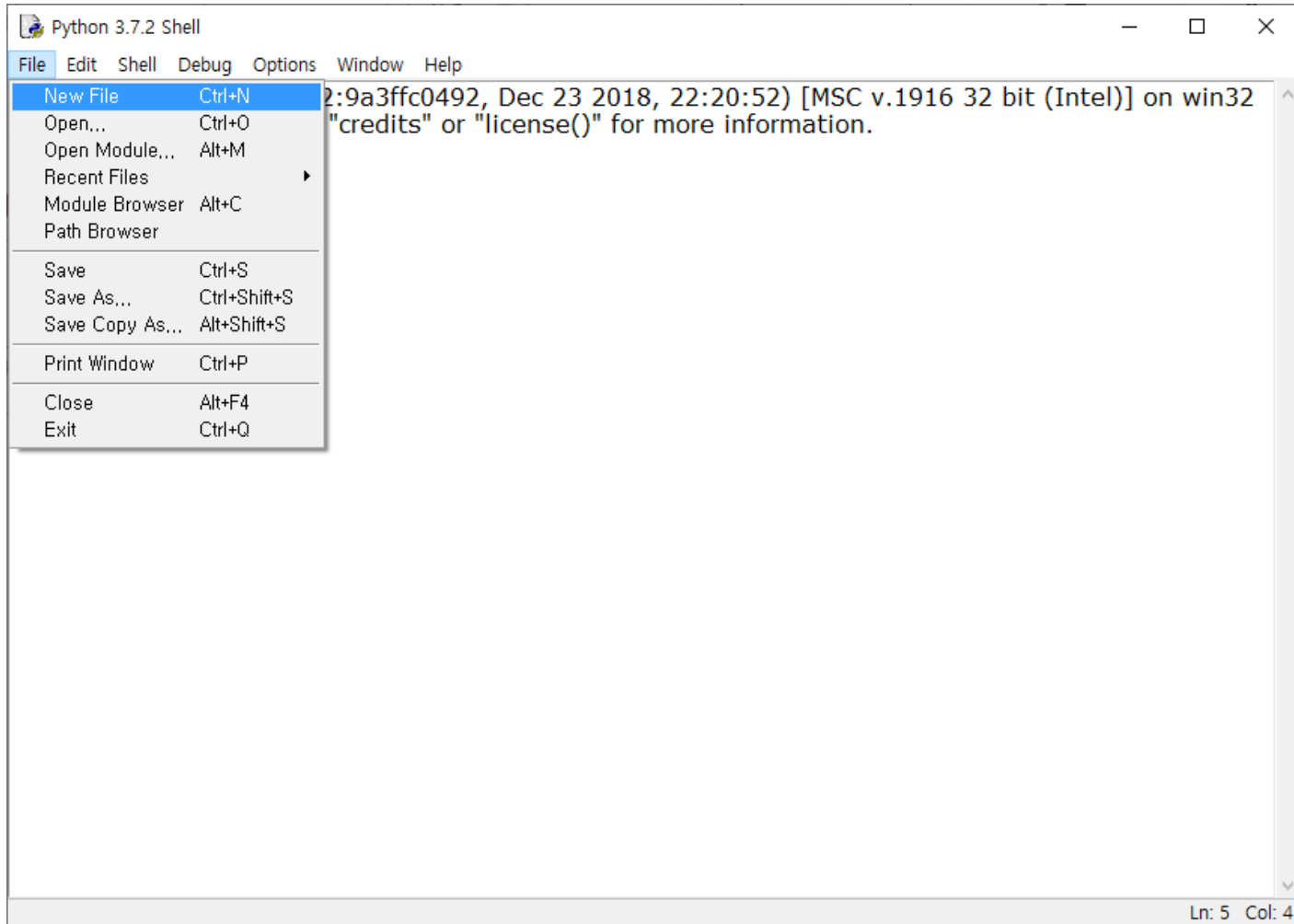
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("hello world")
hello world
>>> |
```

The line `>>> print("hello world")` is highlighted with a red rectangular box. The output `hello world` is displayed on the line immediately following the command. The prompt `>>>` is followed by a vertical bar `|` on the next line, indicating the shell is ready for the next command.

At the bottom right of the window, the status bar shows "Ln: 5 Col: 4".

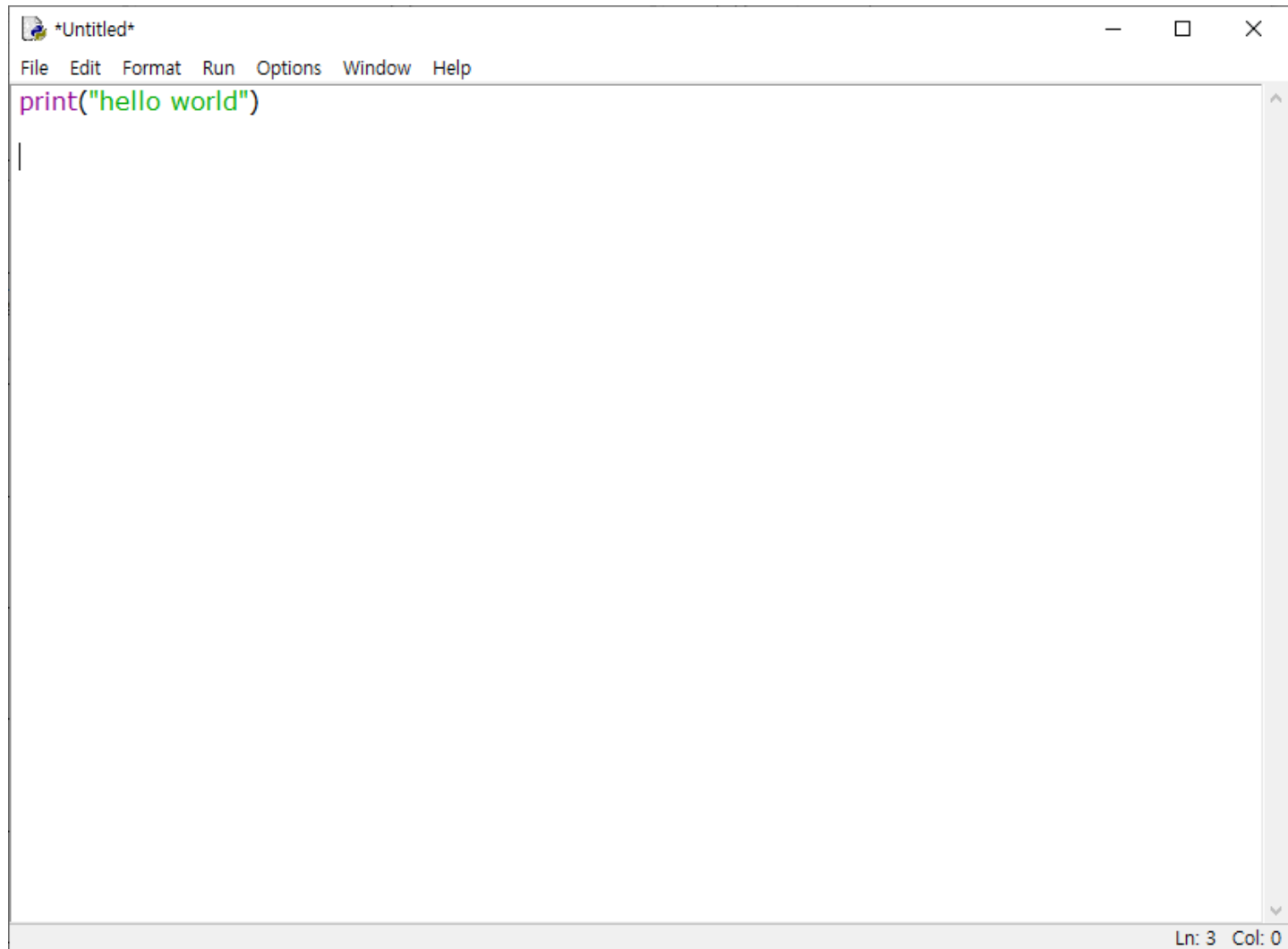
# ※ Python 소스(.py) 작성

## ◆ File → New File





# 소스 코드 작성

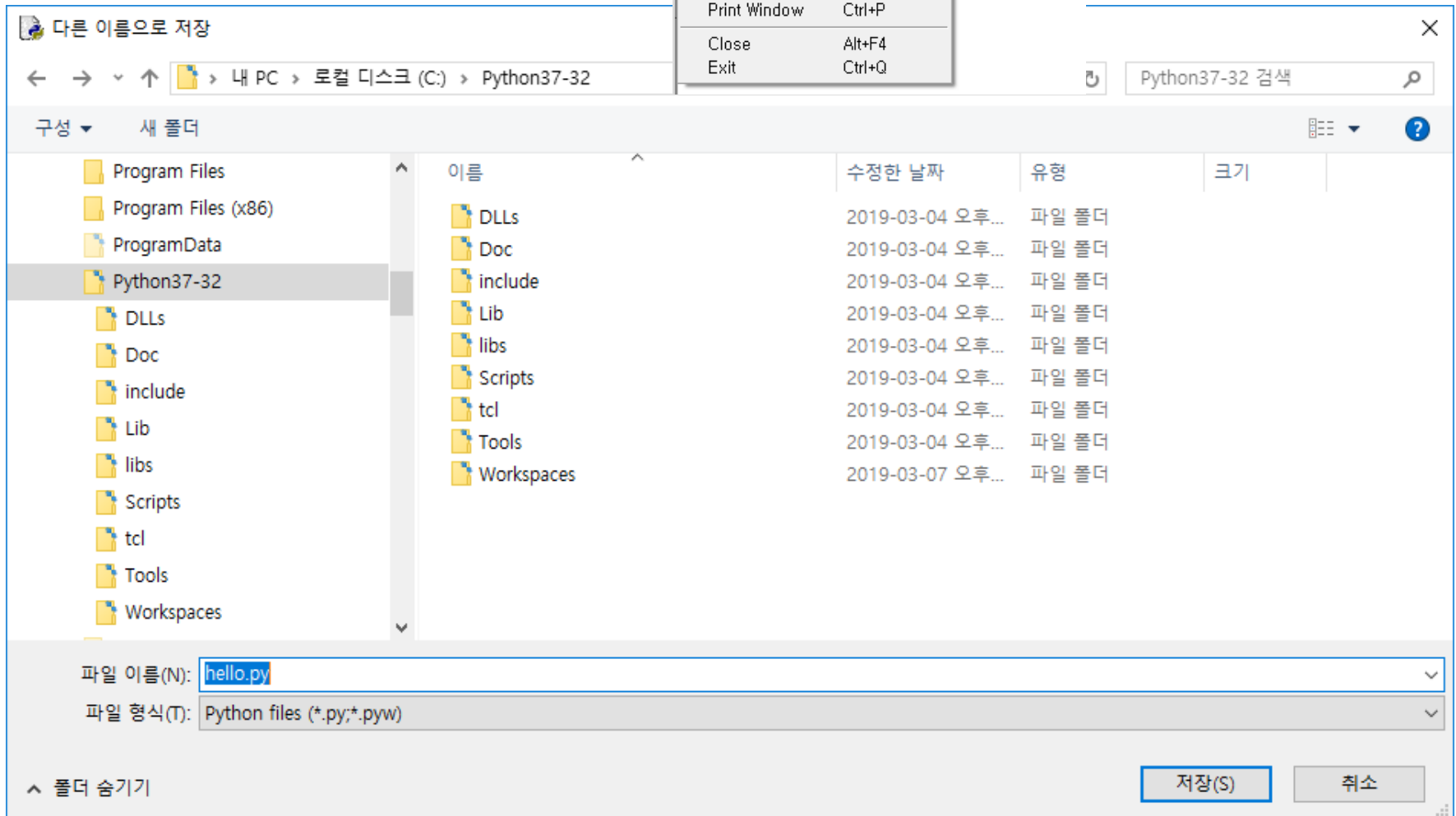


A screenshot of a code editor window titled '\*Untitled\*'. The window has a menu bar with the following options: File, Edit, Format, Run, Options, Window, and Help. The main editing area contains a single line of Python code: `print("hello world")`. The code is color-coded: `print` is in purple, `(` is in green, `"hello world"` is in green, and `)` is in green. A vertical cursor is positioned at the end of the line. The status bar at the bottom right indicates 'Ln: 3 Col: 0'.

```
print("hello world")
```

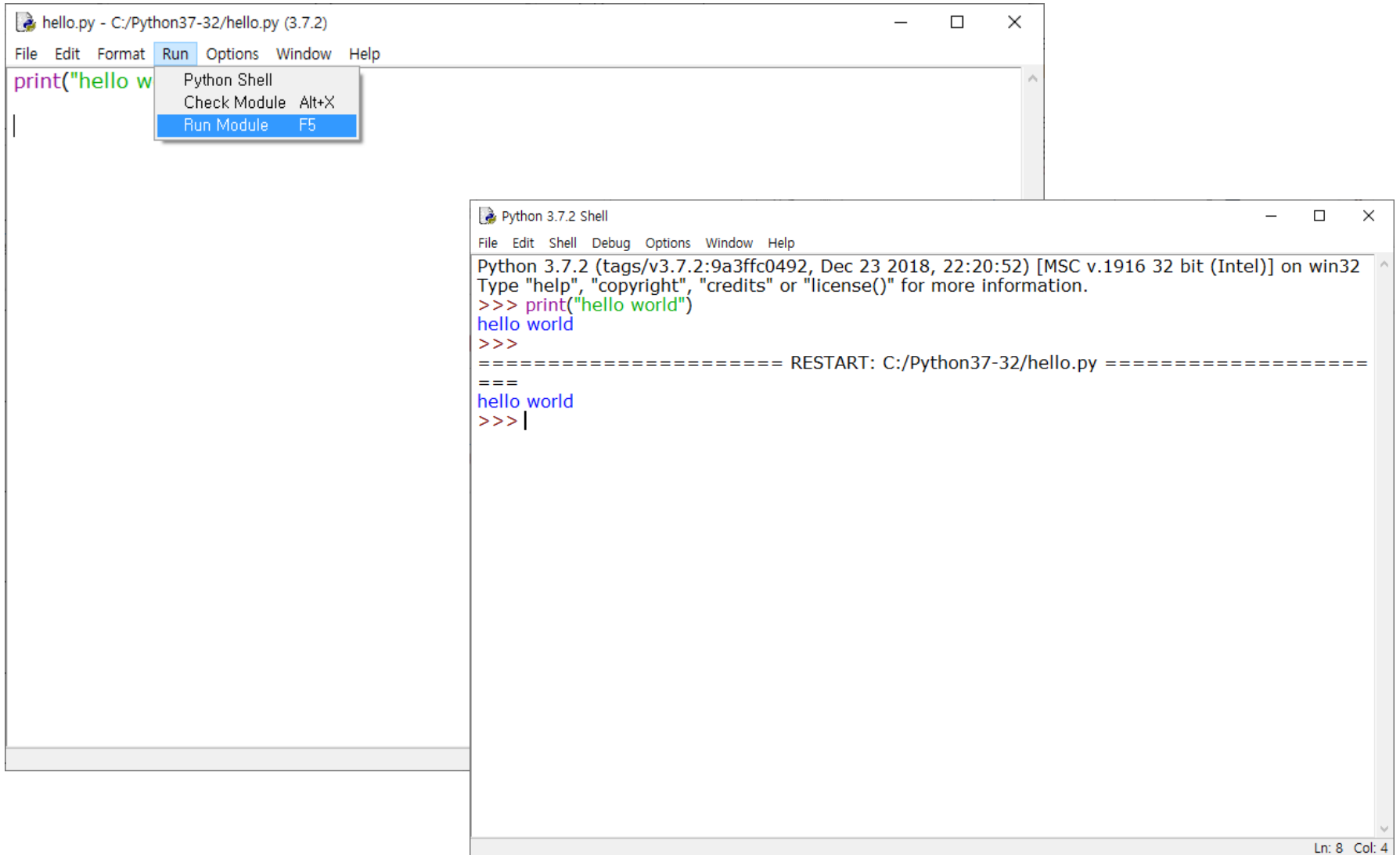
# 소스 저장

- ◆ File → Save As...
- ◆ 파일이름 : hello.py



# Python 코드 실행

## ◆ Run → Run Module : Shell에서 코드 실행 확인



# IDE for Python

- ◆ 파이썬은 스크립트 언어 특성상 IDE 활용 필요성이 타 언어에 비해 상대적으로 낮음
- ◆ 그러나, 최근 파이썬을 활용한 웹 개발 등 대규모 개발이 지속적으로 진행되면서 IDE의 활용이 높아지고 있음
- ◆ Python 프로그래밍을 위한 통합 개발 환경(IDE)
  - Pycharm (<http://jetbrains.com/pycharm/>)
  - Wing IDE
  - PyDev (Eclipse 기반)
  - Sublime Text
  - ...

# Jupyter Notebook

- ◆ 코드, 수식, 시각화 및 설명 텍스트가 포함된 문서를 작성하고 공유 가능
- ◆ 오픈소스 웹 애플리케이션 (웹 브라우저 기반)
- ◆ 데이터 정리, 변환, 수치 시뮬레이션, 통계 모델링, 데이터 시각화, 머신 러닝 등에 사용
- ◆ 장점
  - 데이터 시각화 용이
  - 코드 공유 용이
  - 실시간 실행 (대화형) 가능
  - 코드 샘플 기록
- ◆ 단점
  - IDE 기능 없음
  - 디버깅, 모듈관리 등을 지원하는 본격적인 파이썬 개발 환경은 아님
  - 세션 상태 저장 안됨 : 노트북을 불러올 때마다 코드를 다시 실행해야만 상태 복원 가능

# 수업 진행...

---

- ◆ 전반부 : PyCharm 또는 Python IDLE
- ◆ 후반부 : Jupyter Notebook (in Anaconda)

# 파이썬 기본 문법

# 파이썬 기본 문법

## ◆ 세미콜론

- 많은 프로그래밍 언어들은 구문이 끝날 때 ;(세미콜론)을 붙여야 함
- 파이썬은 세미콜론을 붙이지 않아도 문법에러 발생 안함  
(붙여도 상관 없음)

```
>>>  
>>> print("hello world")  
hello world  
>>>  
>>> print("hello world");  
hello world  
>>>  
>>> print("hello"); print("1234")  
hello  
1234  
>>>
```

- 한 줄에 여러 문장을 사용할 때 세미콜론으로 구분 가능



# 파이썬 기본 문법

## ◆ 주석

- 프로그램 실행에는 영향을 주지 않음 (인터프리터가 해석하지 않음)
- 코드에 대한 설명 작성
- 특정 코드를 임시로 사용하지 않도록 만들 때 사용

```
1
2      # hello world 출력
3      print("hello world")
4
5      """
6      print("test1")
7      print("test2")
8      print("test3")
9      """
10
```

---

**print() 출력**

# print() 함수

## ◆ print() 함수 사용법

- 값 여러 개를 ,(coma)로 구분하여 출력할 수 있음
- 값 사이에 space(한칸 띄워쓰기) 자동 추가

```
>>>  
>>> print("hello", "world")  
hello world  
>>>
```

- 문자열 뿐만 아니라, 숫자 출력도 가능

```
>>> print(100)  
100  
>>> print(100, 200, 300)  
100 200 300  
>>>
```

- 따옴표는 숫자가 아니라 문자임

```
>>> print(100)  
100  
>>> print("100")  
100  
>>>
```

- 숫자 100
- 문자 일영영

# 서식을 지원하는 print() 함수 사용법

## ◆ 서식은 앞에 '%'가 붙음 (치환연산자, 포매팅)

```
print("안녕하세요")
```

→ 안녕하세요

```
print("100")  
print(100)  
print("%d" % 100)
```

→ 글자 100 (일영영)

→ 숫자 100

→ 숫자 100

```
print("100 + 100")  
print(100 + 100)  
print("%d" % (100+100))
```

→ 글자 100 + 100

→ 숫자 200

→ 숫자 200

# print() 문자열 변환

변환	뜻
'd'	부호 있는 정수 십진 표기.
'i'	부호 있는 정수 십진 표기.
'o'	부호 있는 8진수 값.
'u'	쓸데없는 유형 -- 'd' 와 같습니다.
'x'	부호 있는 16진수 (소문자).
'X'	부호 있는 16진수 (대문자).
'e'	부동 소수점 지수 형식 (소문자).
'E'	부동 소수점 지수 형식 (대문자).
'f'	부동 소수점 십진수 형식.
'F'	부동 소수점 십진수 형식.
'g'	부동 소수점 형식. 지수가 -4보다 작거나 정밀도 보다 작지 않으면 소문자 지수형식을 사용하고, 그렇지 않으면 십진수 형식을 사용합니다.
'G'	부동 소수점 형식. 지수가 -4보다 작거나 정밀도 보다 작지 않으면 대문자 지수형식을 사용하고, 그렇지 않으면 십진수 형식을 사용합니다.
'c'	단일 문자 (정수 또는 길이 1인 문자열을 허용합니다).
'r'	문자열 ( <code>repr()</code> 을 사용하여 파이썬 객체를 변환합니다).
's'	문자열 ( <code>str()</code> 을 사용하여 파이썬 객체를 변환합니다).
'a'	문자열 ( <code>ascii()</code> 를 사용하여 파이썬 객체를 변환합니다).
'%'	인자는 변환되지 않고, 결과에 '%' 문자가 표시됩니다.

# 서식을 지원하는 print() 함수 사용법

## ◆ 서식의 갯수와 %(구분자) 뒤에 나오는 숫자(또는 문자)의 갯수가 같아야 함

- 구분자 '%'는 서식 '%'와 다름

```
print("%d" % (100, 200))  
print("%d %d" % (100))
```

```
C:\Python34\python.exe D:/Workspace/Python/Test01/sample_01.py  
Traceback (most recent call last):  
  File "D:/Workspace/Python/Test01/sample_01.py", line 1, in <module>  
    print("%d" % (100, 200))  
TypeError: not all arguments converted during string formatting
```

### ➔ 오류 발생

첫 번째 행에는 %d가 하나밖에 없는데 숫자는 두 개 (100, 200)  
두 번째 행에는 %d가 두 개인데, 숫자는 하나 (100) 밖에 없음

```
print ( "%d %d" % ( 100 , 200 ) )
```



➔ 서식과 숫자는 반드시 대응

# 서식을 지원하는 print() 함수 사용법

## ◆ 정수(%d) 외에 자주 사용되는 서식

```
print("%d %d %d" % (100, 200, 0.5))
```

→ “100 200 0.5” ??

→ “100 200 0”

세 번째 숫자 0.5는 실수(소수점이 있는 수)이지만, 보여주는 방식이 정수(%d)임



→ 서식과 숫자의 불일치 상황

# print()의 서식

## ◆ print()에서 사용하는 서식

서식	값의 예	설명
%d, %x, %o	10, 100, 1234	정수(10진수, 16진수, 8진수)
%f	0.5 , 1.0 , 3.14	실수(소수점이 붙은 수)
%c	"b", "한"	문자 한 글자
%s	"안녕", "abcdefg", "a"	한 글자 이상의 문자열

## ◆ 세 번 째 '%d' 대신 '%f'로 수정

→ "100 200 0.5" ??

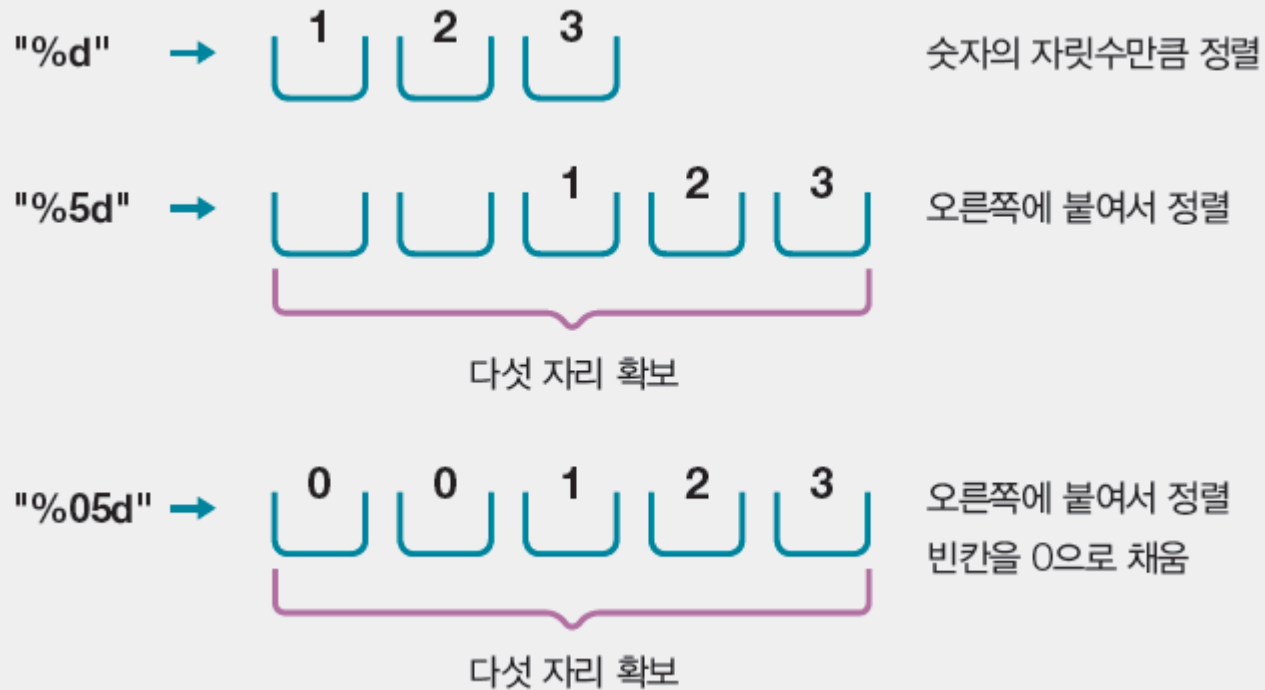
```
print("%d %d %f" % (100, 200, 0.5))
```

→ %.1f : 소수점 이하 한 자리 실수

```
print("%d %d %.1f" % (100, 200, 0.5))
```

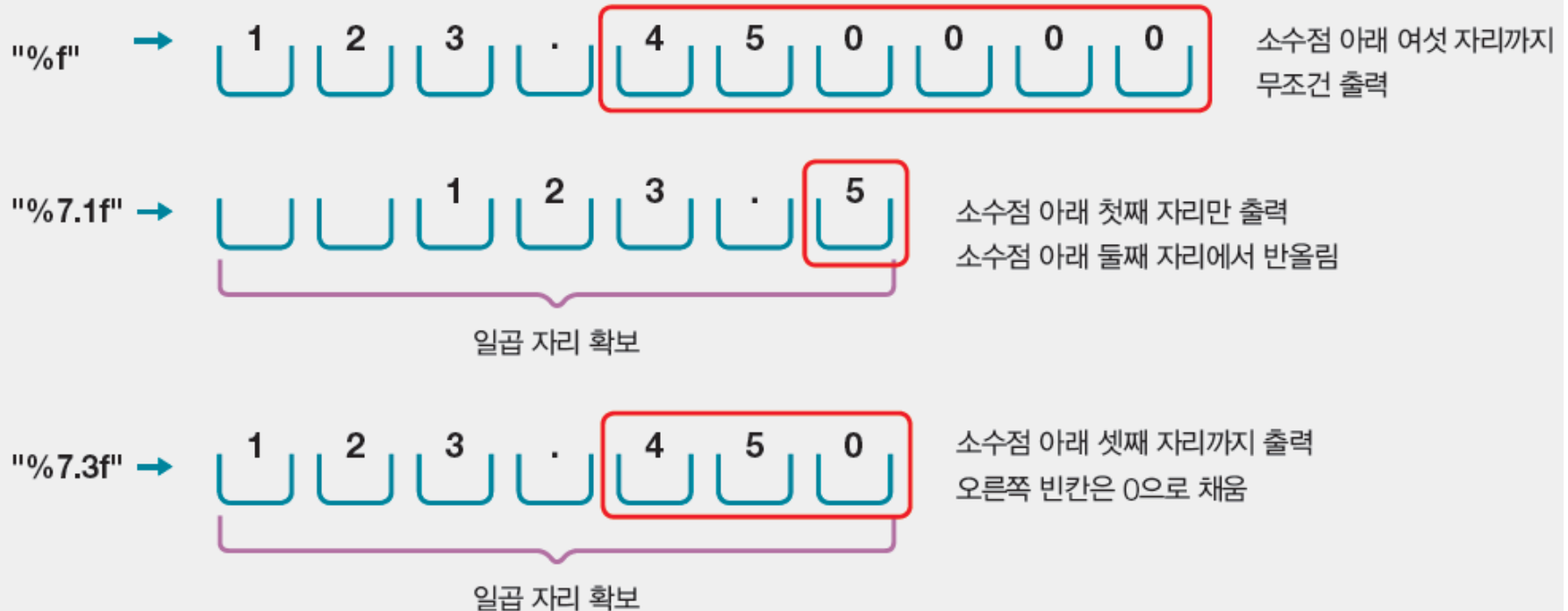


# 정수형 데이터 서식



# 실수형 데이터 서식

- ◆ **'%7.1f'**는 소수점을 포함한 전체 일곱 자리를 확보하고, 소수점 아래는 한 자리만 차지한다는 의미



# 문자열형 데이터 서식

"%s"



P y t h o n

자릿수만큼 출력

"%10s"



          P y t h o n

오른쪽 정렬

열 자리 확보

# print() 서식 출력 연습

```
print("%d" % 123)
print("%5d" % 123)
print("%05d" % 123)

print("%f" % 123.45)
print("%7.1f" % 123.45)
print("%7.3f" % 123.45)

print("%s" % "Python")
print("%10s" % "Python")
```

## → 실행 결과

```
123
  123
00123
123.450000
  123.5
123.450
Python
  Python
```

# print() 서식 출력 연습

```
>>>
>>> print("%g" % 1.234)
1.234
>>>
>>> print("%f" % 1.234)
1.234000
>>>
>>>
>>>
>>> print('%(language)s has %(number)03d quote types.' % {'language': "Python", "number": 2})
Python has 002 quote types.
>>>
```

# 문자열 출력

- ◆ 작은따옴표 ‘...’ / 큰따옴표 “...” 둘 다 사용 가능
- ◆ 복수라인 출력 : 기호를 세 번 (“...” 혹은 ““...””) 사용

```
print('python programming')
print("What's your name?")
print('''Characters of name?
    1. Platform independent
    2. Interpreter
    3. Object-Oriented Programming
    4. Dynamic typing''')
print("""What's your name?
    My name is Steve""")
```

## → 실행 결과

```
python programming
What's your name?
Characters of name?
    1. Platform independent
    2. Interpreter
    3. Object-Oriented Programming
    4. Dynamic typing
What's your name?
    My name is Steve
```

# 다양한 이스케이프 문자

이스케이프 시퀀스	의미
<code>\newline</code>	역 슬래시와 개행 문자가 무시됩니다
<code>\\</code>	역 슬래시 (\)
<code>\'</code>	작은따옴표 (')
<code>\"</code>	큰따옴표 (")
<code>\a</code>	ASCII 벨 (BEL)
<code>\b</code>	ASCII 백스페이스 (BS)
<code>\f</code>	ASCII 폼 피드 (FF)
<code>\n</code>	ASCII 라인 피드 (LF)
<code>\r</code>	ASCII 캐리지 리턴 (CR)
<code>\t</code>	ASCII 가로 탭 (TAB)
<code>\v</code>	ASCII 세로 탭 (VT)
<code>\ooo</code>	8진수 <i>ooo</i> 로 지정된 문자
<code>\xhh</code>	16진수 <i>hh</i> 로 지정된 문자

# 이스케이프 문자 활용

```
print("\n줄바꿈\n연습")
print("\t탭키\t연습")
print("글자가 \"강조\"되는 효과1")
print('글자가 "강조"되는 효과1')
print("글자가 \'강조\'되는 효과2")
print("글자가 '강조'되는 효과2")
print("\\\\\\\\ 역슬러쉬 세개 출력")
print(r"\n \t \" \\ 을 그대로 출력")
```

## → 실행 결과

```
줄바꿈
연습
    탭키    연습
글자가 "강조"되는 효과1
글자가 "강조"되는 효과1
글자가 '강조'되는 효과2
글자가 '강조'되는 효과2
\\\\\\\\ 역슬러쉬 세개 출력
\\n \\t \\" \\ 을 그대로 출력
```



# 변수 계산 및 출력

## ◆ 변수 선언

- $a = 100$  (a 변수에 100 대입)
- $b = 50$  (b 변수에 50 대입)

## ◆ 변수 계산

- $result = a + b$  (result 변수에  $a+b$  계산결과 대입)

```
>>>  
>>> a=100  
>>> b=50  
>>>  
>>> result=a+b  
>>> print(result)  
150  
>>>
```

## ◆ 모든 변수 출력

```
>>>  
>>> print(a)  
100  
>>> print(b)  
50  
>>> print(result)  
150  
>>> print(a, b, result)  
100 50 150  
>>>
```

# 여러 개의 문자 출력 (변수, 문자열)

→ 파일 생성 및 저장: PY01.py”

```
a = 100
b = 50

result = a + b
print(a, "+", b, "=", result)

result = a - b
print(a, "-", b, "=", result)

result = a * b
print(a, "*", b, "=", result)

result = a / b
print(a, "/", b, "=", result)
```

→ 실행 결과

```
100 + 50 = 150
100 - 50 = 50
100 * 50 = 5000
100 / 50 = 2.0
```

---

**input() 함수**

# input() 함수

- ◆ 사용자로부터 키보드 입력을 받을 수 있음

```
print("input your name : ")  
  
str_name = input()  
  
print("my name is %s !!!" % str_name)
```

→ 실행 결과

```
input your name :  
python  
my name is python !!!
```

# input() 함수로 값 입력 받기

→ 파일 불러오기: PY01.py”

수정

```
a = input()
b = input()

result = a + b
print(a, "+", b, "=", result)

result = a - b
print(a, "-", b, "=", result)

result = a * b
print(a, "*", b, "=", result)

result = a / b
print(a, "/", b, "=", result)
```

→ 오류 발생

```
100
50
100 + 50 = 10050
Traceback (most recent call last):
  File "D:/Workspace/Python/Test01/sample_01.py", line 7, in <module>
    result = a - b
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

# input() 함수

- ◆ input() 함수로 입력 받은 값은 **정수가 아니라 문자열**
- ◆ int() 함수 : **정수로 변환** (타입캐스팅)

수정

```
a = int(input())  
b = int(input())  
  
result = a + b  
print(a, "+", b, "=", result)  
  
result = a - b  
print(a, "-", b, "=", result)  
  
result = a * b  
print(a, "*", b, "=", result)  
  
result = a / b  
print(a, "/", b, "=", result)
```

# 설명이 추가된 입력 함수

수정

```
a = int(input("첫 번째 숫자를 입력하세요: "))  
b = int(input("두 번째 숫자를 입력하세요: "))
```

```
result = a + b  
print(a, "+", b, "=", result)
```

```
result = a - b  
print(a, "-", b, "=", result)
```

```
result = a * b  
print(a, "*", b, "=", result)
```

```
result = a / b  
print(a, "/", b, "=", result)
```

## → 실행 결과

첫 번째 숫자를 입력하세요: 100

두 번째 숫자를 입력하세요: 50

100 + 50 = 150

100 - 50 = 50

100 \* 50 = 5000

100 / 50 = 2.0

# 서식을 이용한 문자열 print() 연습

## ◆ 문자 출력

```
print("안녕하세요")
```

## ◆ 문자열 변수 출력

```
str_who = "홍길동"  
print("%s님, 안녕하세요" % str_who)
```

## ◆ 키보드 입력값(문자열) 출력

```
str_who = input()  
print("%s님, 안녕하세요" % str_who)
```

## ◆ 키보드 입력 안내문구 추가

```
str_who = input("이름을 입력하세요:")  
print("%s님, 안녕하세요" % str_who)
```



# 서식을 이용한 숫자 print() 연습

## ◆ 숫자출력

```
print(100)
```

## ◆ 정수 변수 출력

```
num = 100  
print("숫자는 %d" % num)
```

## ◆ 두 개의 정수 변수 출력

```
num1 = 100  
num2 = 200  
print("첫번째 숫자는 %d이고, 두번째 숫자는 %d이다" % (num1, num2))
```

## ◆ 키보드 입력값(문자열) 출력 – 에러

```
num = input()  
print("숫자는 %d" % num)
```

## ◆ 키보드 입력값(문자열) → 정수로 변환후 출력

```
num = int(input())  
print("숫자는 %d" % num)
```

# 서식을 이용한 숫자 print() 연습 2)

## ◆ 키보드 입력 안내문구 추가

```
num = int(input("숫자를 입력하세요: "))  
print("숫자는 %d" % num)
```

## ◆ 두 개의 정수 입력

```
num1 = int(input("첫번째 숫자를 입력하세요: "))  
num2 = int(input("두번째 숫자를 입력하세요: "))  
print("첫번째 숫자는 %d이고, 두번째 숫자는 %d이다" % (num1, num2))
```

## ◆ 입력한 두 개의 정수 합 출력

```
num1 = int(input("첫번째 숫자를 입력하세요: "))  
num2 = int(input("두번째 숫자를 입력하세요: "))  
sum_num = num1 + num2  
print("합은 %d입니다." % sum_num)
```

또는...

```
num1 = int(input("첫번째 숫자를 입력하세요: "))  
num2 = int(input("두번째 숫자를 입력하세요: "))  
print("합은 %d입니다." % (num1 + num2))
```

# format() 함수

- ◆ 좀 더 발전된 스타일로 문자열 포맷 지정 가능

```
print("I eat {0} apples".format(3))  
print("I eat {0} apples".format("five"))  
  
number = 7  
print("I eat {0} apples".format(number))  
  
print("I eat {num} apples".format(num=9))
```

## → 실행 결과

```
I eat 3 apples  
I eat five apples  
I eat 7 apples  
I eat 9 apples
```

# format() 함수

```
print("%d %5d %05d" % (123, 123, 123))  
print("{0:d} {1:5d} {2:05d}".format(123, 123, 123))
```

- ◆ 두 번째 행에서 { } 안의 0, 1, 2는 format() 안의 0, 1, 2번째 값에 대응  
(주의: %d에서 %를 떼고 d로 표시)

`print( "{0:d} {1:5d} {2:05d}".format( 123 , 123 , 123 ) )`

↑   ↑   ↑  
0번째   1번째   2번째

123   □□123   00123

# format() 함수

## ◆ format 함수 정렬 :

- >(오른쪽정렬) 또는 <(왼쪽정렬) 숫자(전체자리수)

```
>>>
>>> '{:>5}'.format('123')
' 123'
>>> '{:>10}'.format('123')
'      123'
>>> '{:<10}'.format('123')
'123 '
>>> '{:<10}'.format('12.3')
'12.3 '
>>> '{:.3f}'.format(12.38898)
'12.389'
>>>
```

```
>>>
>>> '{:>05}'.format('123')
'00123'
>>> '{:>010}'.format('123')
'0000000123'
>>>
```

# Any Questions... Just Ask!



*“Knowledge is only part of understanding.  
Genuine understanding comes from hands-on experiences.”*  
- Prof. Seymour Papert, MIT