

파이썬 프로그래밍

2차시

파이썬 설치와
파이썬 셸 실행



⚠️ 학습개요

- ... 파이썬 개발환경 설치
- ... 파이썬 쉘(IDLE) 사용 방법

⚠️ 학습목표

- ... 파이썬 개발환경을 설치할 수 있다.
- ... 파이썬 쉘(IDLE)을 실행할 수 있다.
- ... 쉘에서 파이썬 코딩을 할 수 있다.

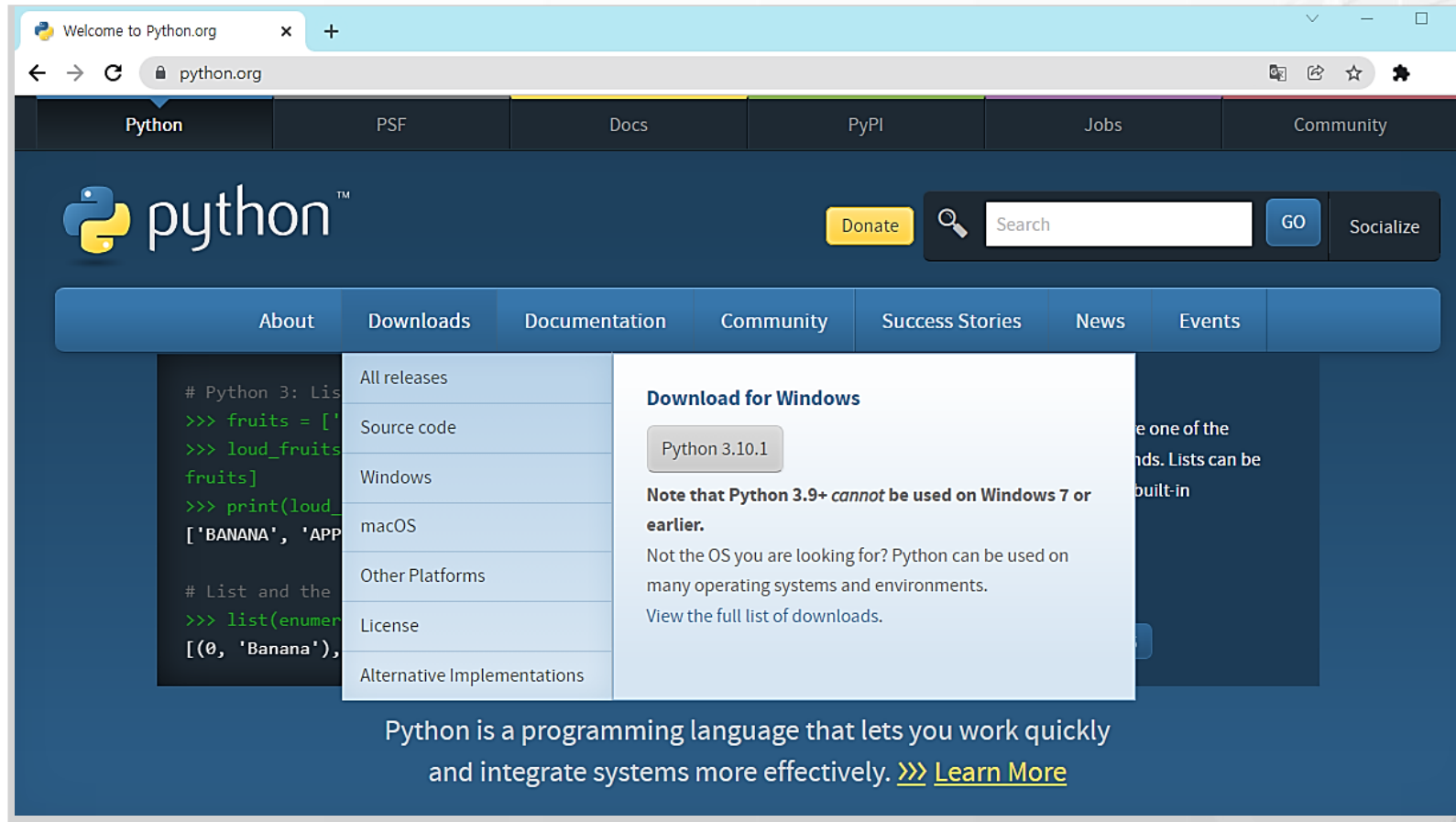
Chapter 1.

파이썬 개발환경 설치

P Y T H O N P R O G R A M M I N G

! 파이썬 개발 도구 설치

+ 파이썬 홈페이지(www.python.org)



⚠ 파이썬 개발 도구 설치

+ 파이썬 개발 도구의 다운로드와 실행 1

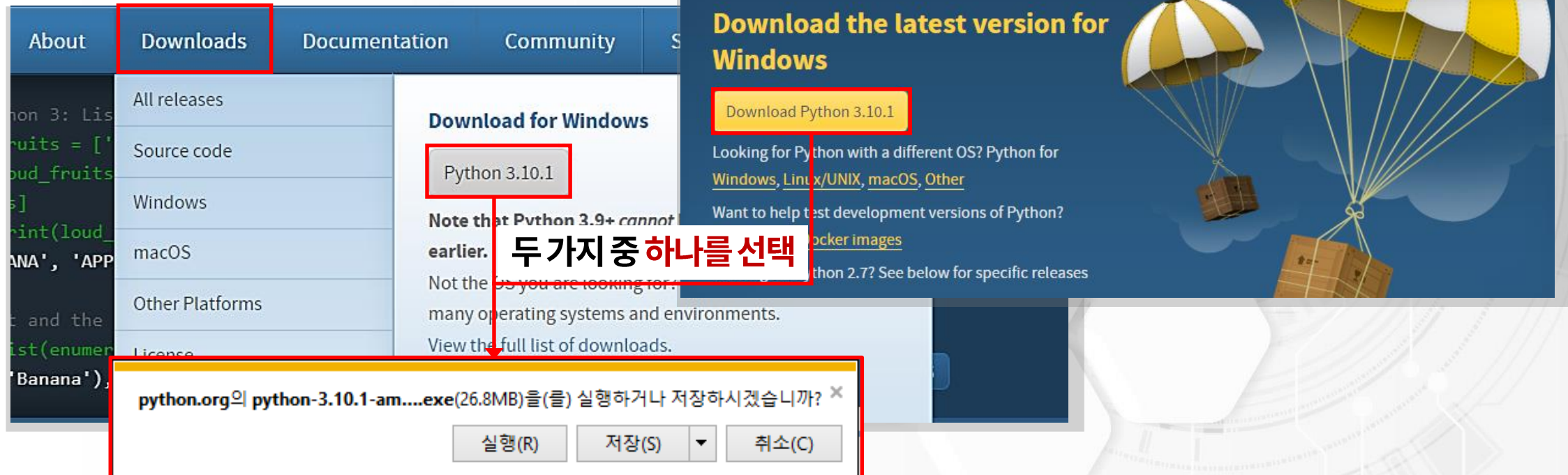
- 최신 버전 내려 받아 설치
- 현재 3.10, 내려 받은 파일 이름
 - `python-3.10.1-amd64.exe`



⚠ 파이썬 개발 도구 설치

+ 파이썬 개발 도구의 다운로드와 실행 1

- 최신 버전 내려 받아 설치
- 현재 3.10, 내려 받은 파일 이름
 - `python-3.10.1-amd64.exe`

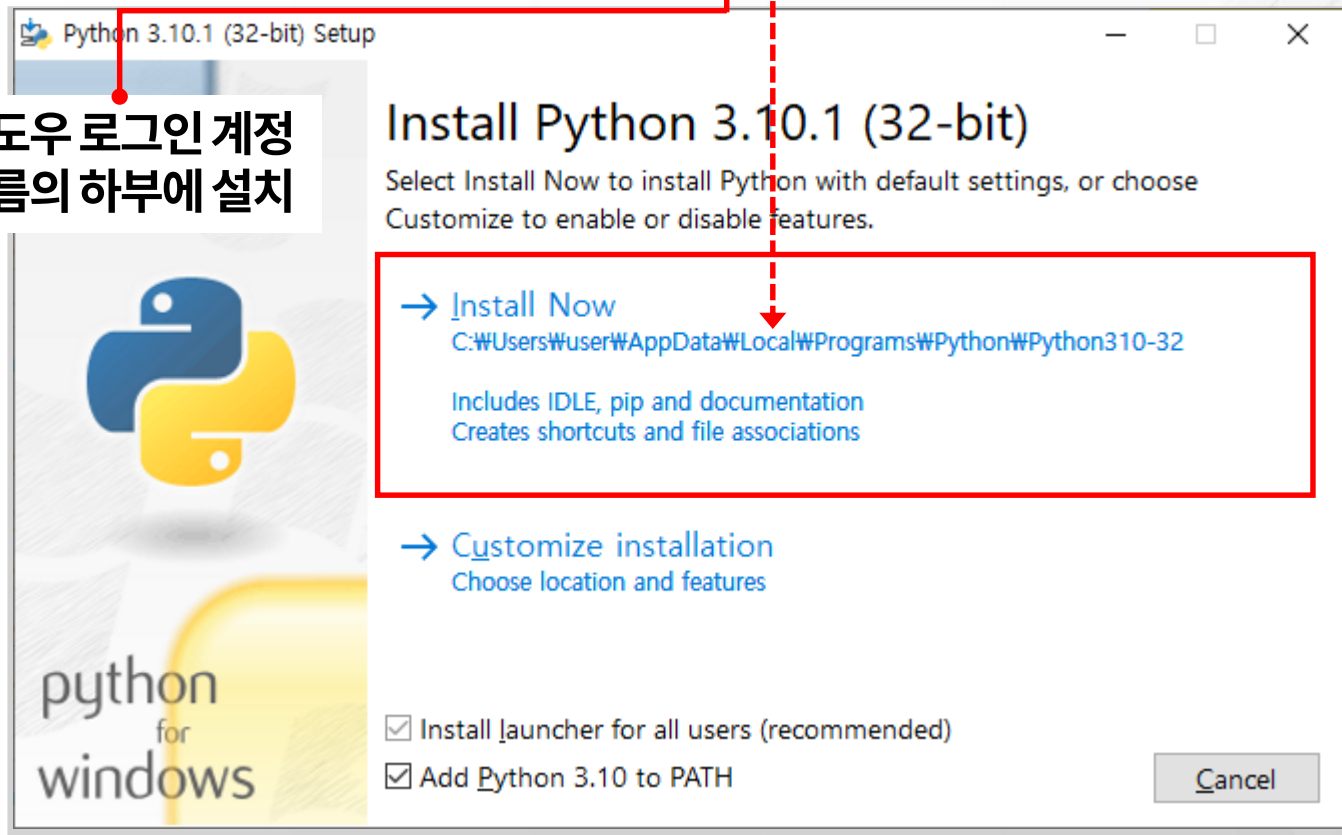


⚠ 파이썬 개발 도구 설치

+ 파이썬 설치 시작 화면 2

- 기본 파이썬 설치 폴더: C:\Users\user\AppData\Local\Programs\Python\Python310-32

윈도우 로그인 계정
이름의 하부에 설치



⚠ 파이썬 개발 도구 설치

+ 파이썬 설치 시작 화면 2

- 기본 파이썬 설치 폴더: C:\Users\user\AppData\Local\Programs\Python\Python310-32





좀 더 알아봅시다!

+ 파이썬 설치 폴더 살펴 보기

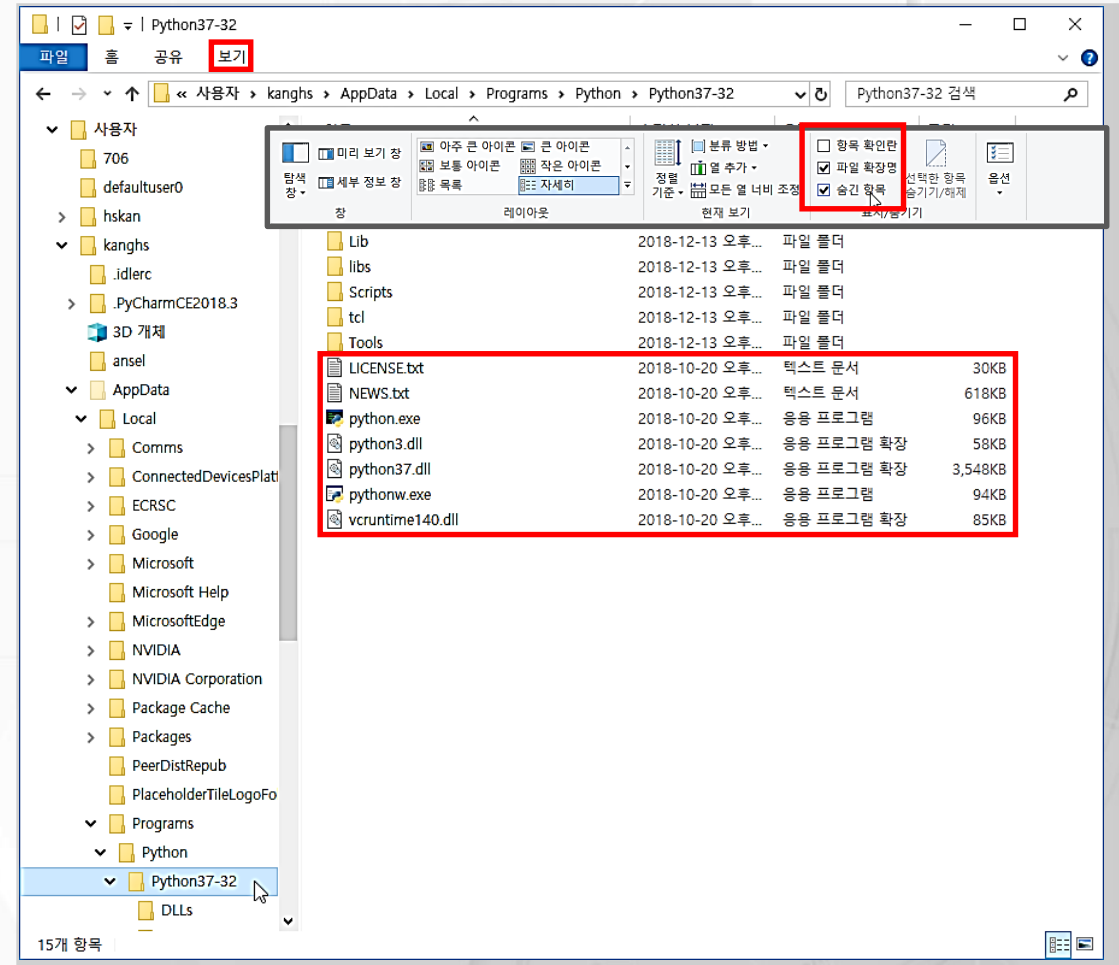
- 파이썬이 설치된 폴더를 살펴보려면 탐색기의 메뉴[보기]에서 체크박스 ☒ 숨긴 항목]을 선택한다.
- 파이썬 소스등과 같은 다양한 파일의 확장자를 보기 위해 체크박스 ☒ 파일 확장명]도 선택해 보자.
- 자신의 파이썬 설치 폴더를 찾아보자.



좀 더 알아봅시다!

+ 파이썬 설치 폴더 살펴 보기

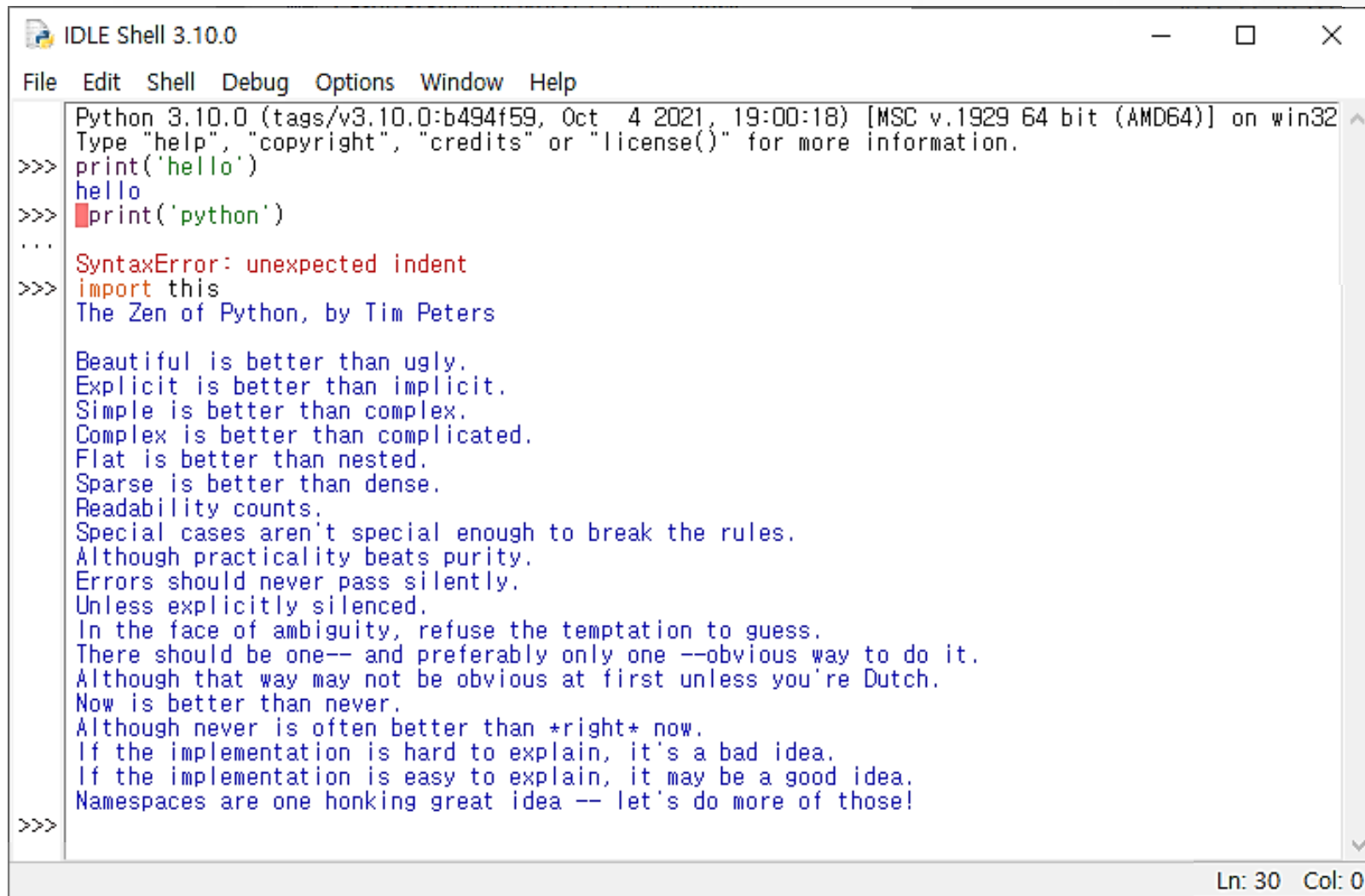
- 파이썬이 설치된 폴더를 살펴보려면 탐색기의 메뉴[보기]에서 체크박스 ☒ 숨긴 항목]을 선택한다.
- 파이썬 소스등과 같은 다양한 파일의 확장자를 보기 위해 체크박스 ☒ 파일 확장명]도 선택해 보자.
- 자신의 파이썬 설치 폴더를 찾아보자.



[그림2-1] 탐색기에서 파이썬 설치 폴더 이동



[예습] 파이썬 쉘(IDLE)의 실행



```
IDLE Shell 3.10.0
File Edit Shell Debug Options Window Help
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('hello')
hello
>>> print('python')
...
SyntaxError: unexpected indent
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

>>>
```

Ln: 30 Col: 0

Chapter 2.

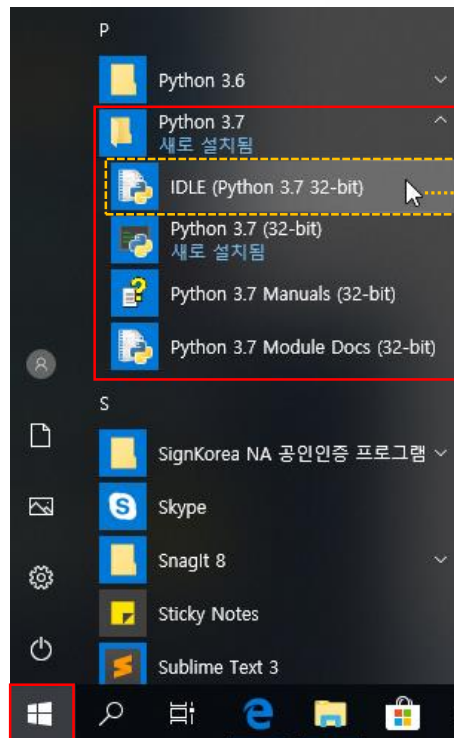
파이썬 쉘(IDLE) 사용 방법

P Y T H O N P R O G R A M M I N G

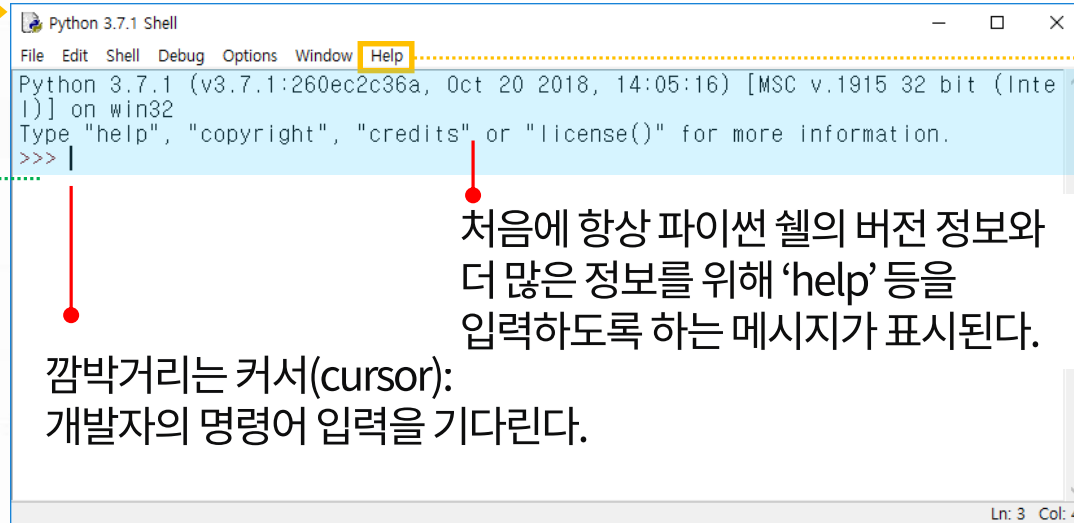
! 파이썬 쉘의 실행

+ '파이썬 쉘 IDLE'을 실행

- 설치된 [Python 3.7] - [IDLE (Python 3.7 32-bit)] 선택

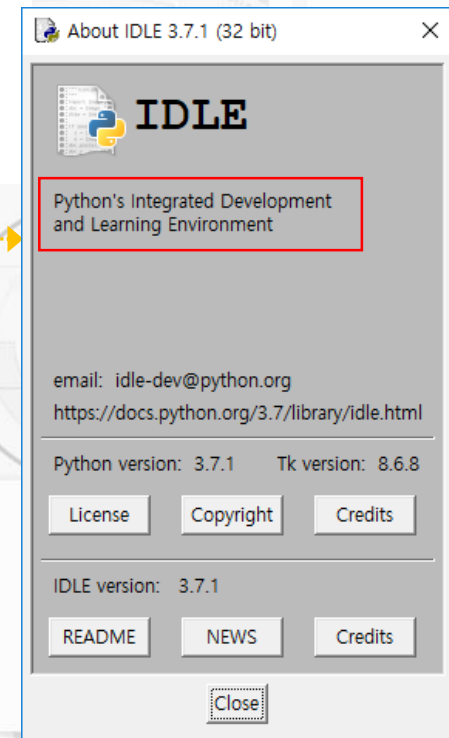


>>> (프롬프트, Prompt): 사용자의 명령이나 문장의 입력을 기다린다는 의미이며, 사용자는 한 줄 정도의 파이썬 문장을 입력한 후 [Enter]를 누른다.



깜박거리는 커서(cursor):
개발자의 명령어 입력을 기다린다.

처음에 항상 파이썬 쉘의 버전 정보와 더 많은 정보를 위해 'help' 등을 입력하도록 하는 메시지가 표시된다.

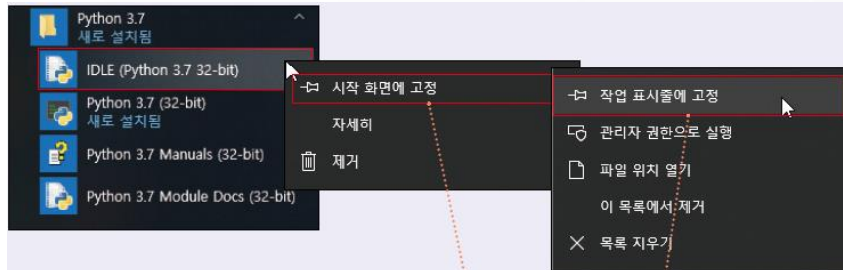


[그림2-2] 파이썬의 실행 메뉴와 [파이썬 쉘 IDLE]

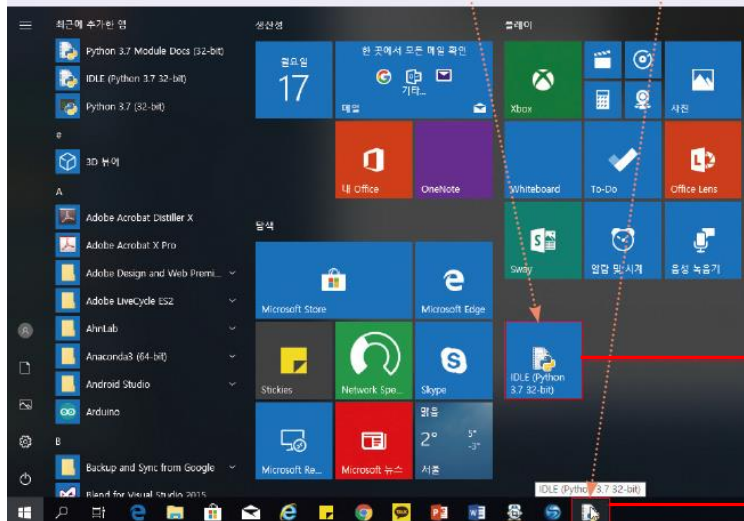


좀 더 알아봅시다!

+ 파이썬 쉘 IDLE 바로 실행 만들기과 검색에서 실행 명령어 'idle'입력



- 파이썬 실행 메뉴를 마우스 오른쪽 버튼으로 누른 후 [시작 화면에 고정]을 누르면 오른쪽 [시작화면]에 아이콘 메뉴가 표시되고, 다시 [자세히] - [작업 표시줄에 고정]을 누르면 파이썬 실행 메뉴를 [작업표시줄]에 고정할 수 있다.



IDLE을 바로 실행할 수 있는 메뉴가 만들어진다.

작업 표시줄에 IDLE을 바로 실행할 수 있는 메뉴가 만들어진다.

[그림2-3] 파이썬 바로 실행 만들기

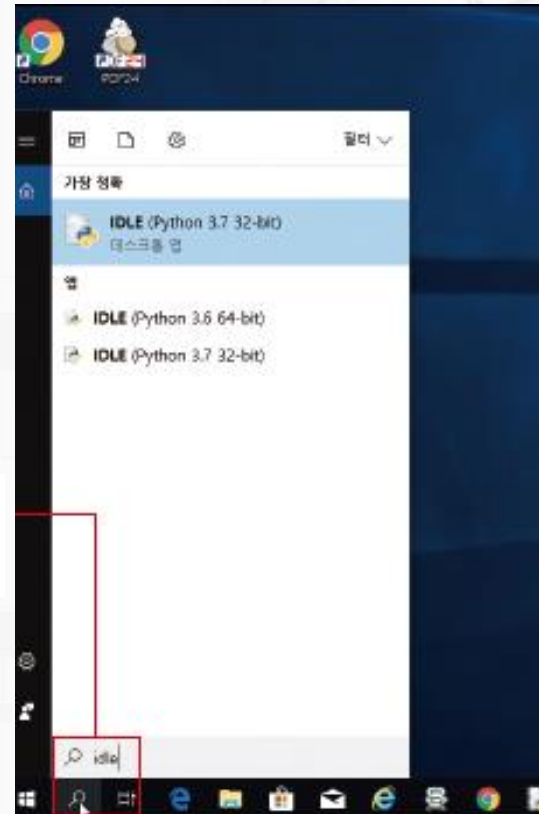


좀 더 알아봅시다!

+ 파이썬 쉘 IDLE 바로 실행 만들기과 검색에서 실행 명령어 'idle'입력

- [Windows로고 키] +[S]를 눌러 검색 창에 'idle'을 입력하면 바로 '파이썬 쉘 IDLE'이 실행된다.

실행 명령어인 'idle'을
직접 입력해 실행한다.



[그림2-4] 윈도우 검색창에 'idle'입력

⚠ 글자 Hello World!를 출력하는 첫 대화형 코딩

+ print 다음에 '('Hello World!')'를 입력

- 'Hello World!'와 같이 출력하고자 하는 글자의 앞뒤에 작은따옴표(또는 홑따옴표)를 붙인다.
- 작은따옴표를 붙인 글자를 문자열 또는 스트링(string)이라 한다.
- print('Hello World!')를 입력한 후 [Enter]를 눌러 결과를 알아보자.

[코딩실습] 글자Hello World! 출력

난이도 기본

1. `>>> print(' Hello World! ')`
2. Hello world!

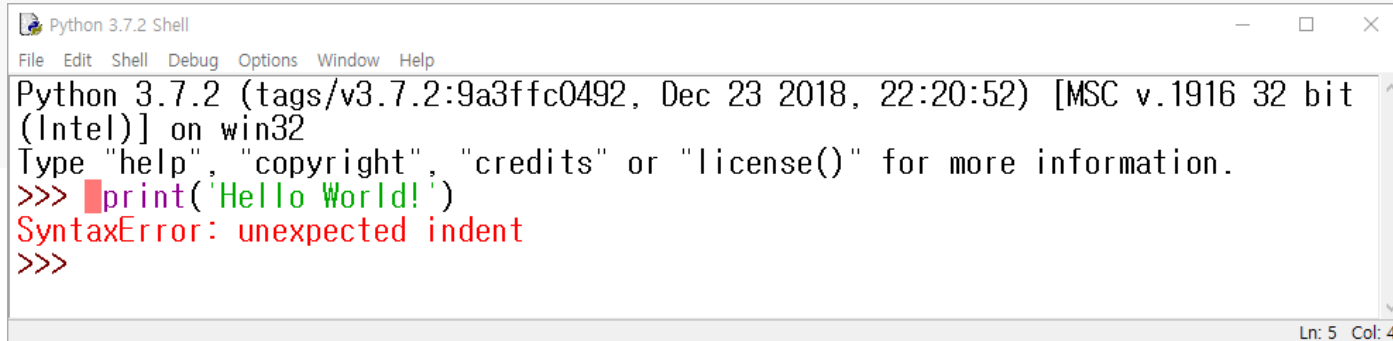
⚠ 글자 Hello World!를 출력하는 첫 대화형 코딩

+ print 다음에 '('Hello World!')'를 입력

- 'Hello World!'와 같이 출력하고자 하는 글자의 앞뒤에 작은따옴표(또는 홑따옴표)를 붙인다.
- 작은따옴표를 붙인 글자를 문자열 또는 스트링(string)이라 한다.
- print('Hello World!')를 입력한 후 [Enter]를 눌러 결과를 알아보자.

[코딩실습] 글자Hello World! 출력

난이도 기본

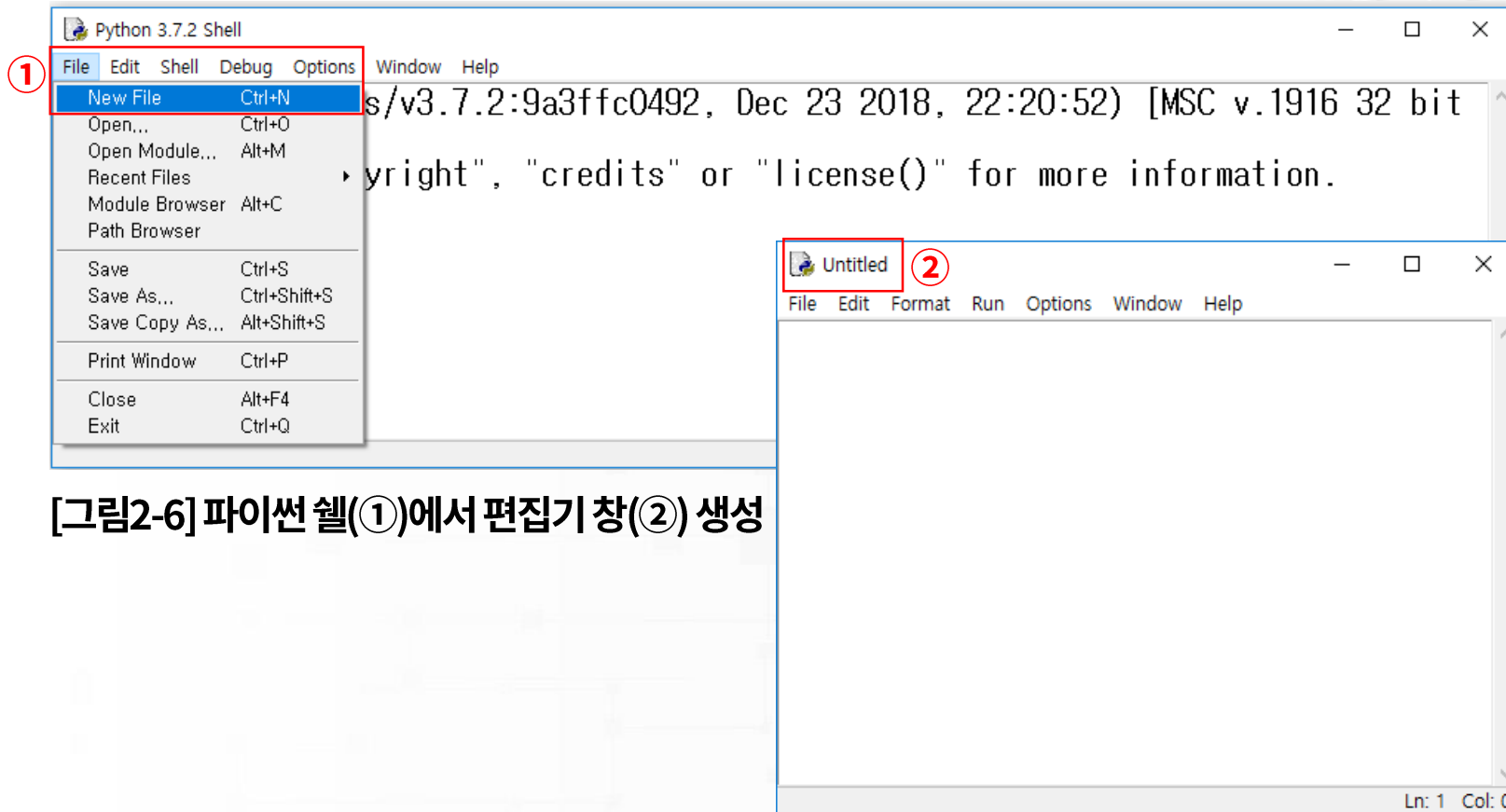


```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('Hello World!')
SyntaxError: unexpected indent
>>>
```

[그림2-5] 맨 앞에 공백이 있으면 무조건 오류, 첫 칸부터 명령어 입력

⚠ 편집기에서 첫 파일 프로그래밍

+ 코딩한 파일을 저장할 편집기 생성



[그림2-6] 파이썬 쉘(①)에서 편집기 창(②) 생성

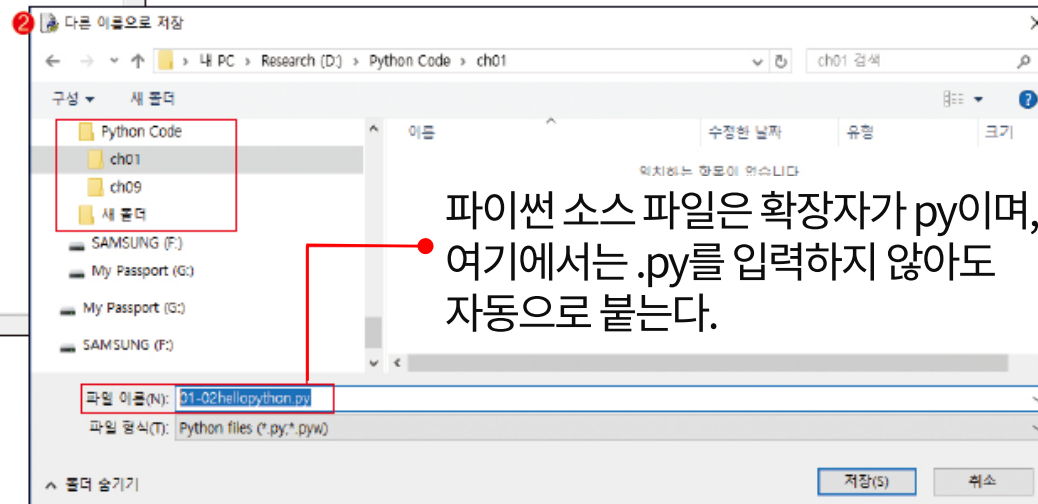
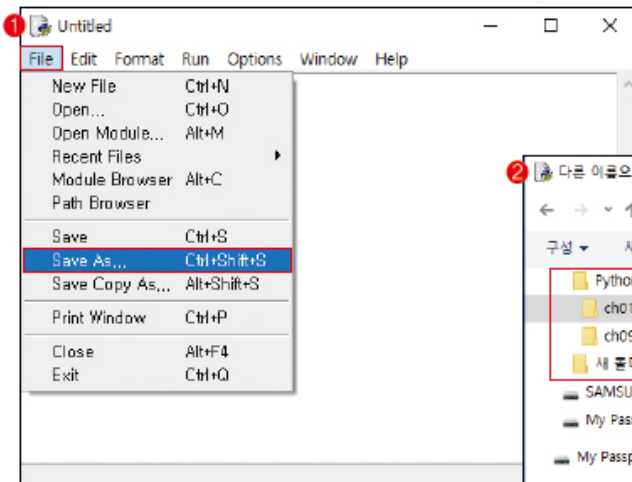
! 편집기에서 첫 파일 프로그래밍

+ 파이썬 코드 작성과 저장

- 폴더: Python Code\ch01

- 소스 파일 이름: 01-02hellopython.py

파이썬 명령어인 문장의 모임이 저장되는 파일을 소스 파일이라 하며, 중간에 공백도 가능하나 주로 공백없이 확장자는 py로 파일이름.py로 저장한다.



파이썬 소스 파일은 확장자가 py이며, 여기에서는 .py를 입력하지 않아도 자동으로 붙는다.

[그림2-7] 파이썬 소스 파일 저장

⚠ 파이썬 두 번째 프로그램 작성과 실행

✦ 편집기 창의 제목 부분에 소스 파일 이름 앞뒤에 별표가 붙어 “*○○○○○*”처럼 표시

- 소스가 수정된 이후에 저장되지 않았음을 표시

[코딩실습] 두 줄에 글자 Hello World!와 Hi, python 출력

난이도 기본

```
1. print( ' Hello World! ' )  
2. print( ' Hi, python ' )
```

결과

```
Hello World!  
Hi, python
```


⚠ 파이썬 두 번째 프로그램 작성과 실행

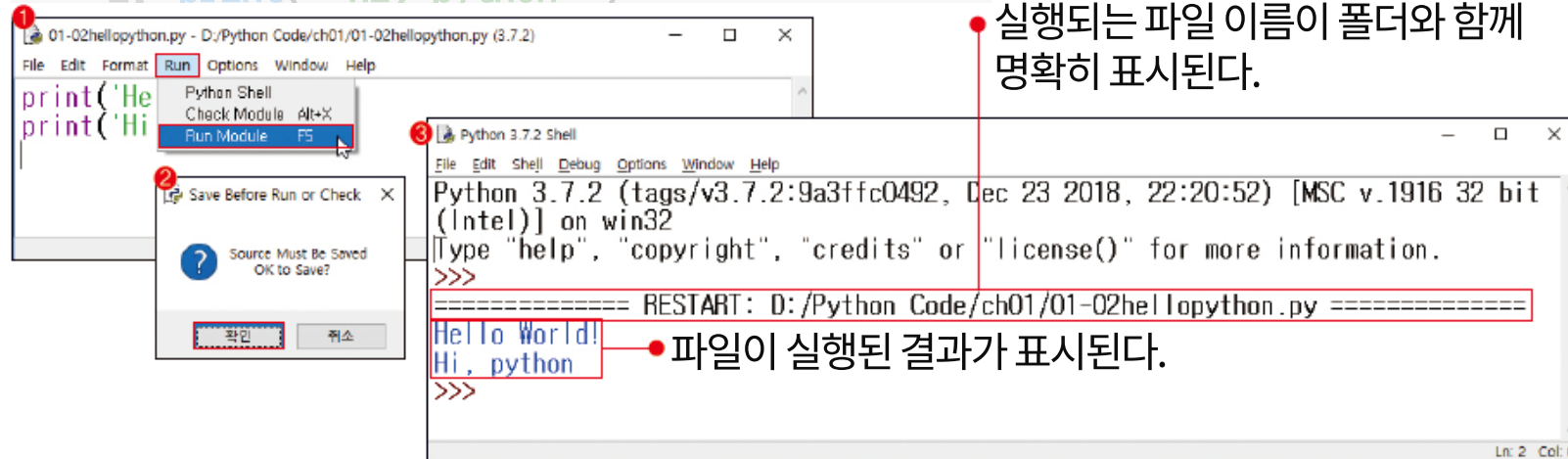
✦ 편집기 창의 제목 부분에 소스 파일 이름 앞뒤에 별표가 붙어 “*○○○○○*”처럼 표시

- 소스가 수정된 이후에 저장되지 않았음을 표시

[코딩실습] 두 줄에 글자 Hello World!와 Hi, python 출력

난이도 기본

```
1. print( ' Hello World! ' )  
2. print( ' Hi. python ' )
```



[그림2-8] 파이썬 쉘(①)과 저장 확인 창(②) 그리고 실행된 결과가 보이는 파이썬 쉘(③)



좀 더 알아봅시다!

+ 오류와 디버깅 과정

- 처음부터 문제 없이 파란색의 결과만 표시되지 않을 수 있다. 아니 당연하다. 여러 가지 이유로 파이썬 쉘이 인식할 수 없는 명령어가 전달되면 다음과 같이 한 줄에서 네 줄 정도의 빨간색 결과 메시지가 표시된다. 이것이 바로 파이썬 쉘이 개발자에게 오류(error)를 알리는 메시지다. 이러한 오류 메시지는 개발자에게 오류를 알려 수정을 하는데 도움을 주는 하나의 방법이다. 오류 메시지에는 소스 파일 이름, 추정되는 오류가 있는 줄 번호, 추정 오류 문장, 오류 이름: 오류 메시지 등이 표시된다. 바로 마지막 줄의 `SyntaxError: EOL while scanning string literal`에서 `SyntaxError`가 오류 이름이며, `EOL while scanning string literal`은 오류 메시지다. 오류 중에서 `SyntaxError`는 앞으로 자주 표시될 오류로, 구문 오류라 부르며 주로 한 줄로 표시된다.
 - 특히 오류 메시지의 마지막 줄에 있는 콜론(:) 이후, 오류 원인이 표시되는 오류 메시지를 읽어 보면 무엇이 문제인지 어느 정도 인지할 수 있다. 아직은 몰라도 상관 없으니 무조건 읽어보자.



좀 더 알아봅시다!

+ 오류와 디버깅 과정

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
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!')
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    print('Hello World!')
NameError: name 'prit' is not defined
>>> print('Hi, python)
SyntaxError: EOL while scanning string literal
>>>
```

출력 글자 Hi, python의 앞에는
홀따옴표가 있지만 맨 뒤에는 홀따옴표가
없어 발생하는 오류임을 표시한다.

‘문자열을 그대로 스캔하는 동안 EOL’이라는
메시지로, EOL(End of Line)이 없는 구문 오류다.

[그림2-9] 오류 메시지의 두 가지 예

```
IDLE Shell 3.10.1
File Edit Shell Debug Options Window Help
>>> print('Hi, python)
...
SyntaxError: unterminated string literal (detected at line 1)
>>>
```



실습

+ 표준 파이썬 설치: www.python.org

실습구간입니다.



실습

[코딩실습1] 글자Hello World! 출력

난이도 기본

1. `>>> print(' Hello World! ')`
2. Hello world!

실습구간입니다.



실습

[코딩실습2] 두 줄에 글자 Hello World!와 Hi, python 출력

난이도 기본

1. `print(' Hello World! ')`
2. `print(' Hi, python ')`

결과

Hello World!
Hi, python

실습구간입니다.

⚠ 파이썬 철학 [The Zen of Python, by Tim Peters]

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one--and preferably only one--obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea--let's do more of those!

⚠ 파이썬 설치와 쉘 실행

- ... 파이썬 홈페이지(www.python.org)
- ... 쉘 실행과 종료

⚠ 파이썬 쉘(IDLE) 사용 방법

- ... 명령 프롬프트: `>>>`
- ... 구문 작성: `print('python')`
- ... 쉘, REPL의 이해: `Read, Evaluate, Print, Loop`

⚠ 첫 코딩

- ... `print('Hello, World!')`
- ... 소스 파일로 저장과 실행: *.py 파일로 저장, 메뉴 Run / Run module

⚠ 오류와 디버깅

- ... 붉은 색 메시지 확인
- ... 수정 후 다시 실행