

데이터분석 실습

경남대학교 전하용

01

파이썬이란 무엇인가?

01-1 파이썬이란?

01-2 파이썬의 특징

01-3 파이썬으로 무엇을 할 수 있을까?

01-4 파이썬 설치하기

01-5 파이썬 둘러보기

01-6 파이썬과 에디터

01-7 파이썬 개발툴

01-1 파이썬이란?

■ 파이썬(Python)

- 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터 언어
- 파이썬의 사전적 의미
 - 고대 신화에 나오는 파르나소스 산의 동굴에 살던 뱀
- 구글에서 만든 소프트웨어의 50% 이상이 파이썬으로 작성됨
- 드롭박스(Dropbox), 인스타그램(Instagram)
- 공동 작업과 유지 보수가 매우 쉽고 편



01-2 파이썬의 특징

- 파이썬은 인간다운 언어이다

- 파이썬은 사람이 생각하는 방식을 그대로 표현할 수 있는 언어

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

=

만약 4가 1, 2, 3, 4 중에 있으면 '4가 있습니다'를 출력한다.

01-2 파이썬의 특징

■ 파이썬은 문법이 쉬워 빠르게 배울 수 있다

- 문법 자체가 아주 쉽고 간결하며 사람의 사고 체계와 매우 닮아 있음
- 유명한 프로그래머인 에릭 레이먼드(Eric Raymond)는 공부한 지 단 하루 만에 자신이 원하는 프로그램을 작성!

■ 파이썬은 무료이지만 강력하다

- 오픈 소스 → 무료로 언제 어디서든 파이썬을 다운로드하여 사용 가능
- 파이썬과 C는 찰떡 궁합
 - 프로그램의 전반적인 뼈대는 파이썬으로 만들고,
빠른 실행 속도가 필요한 부분은 C로 만들어서 파이썬 프로그램 안에 포함

01-2 파이썬의 특징

■ 파이썬은 간결하다

```
# simple.py
languages = ['python', 'perl', 'c', 'java']

for lang in languages:
    if lang in ['python', 'perl']:
        print("%6s need interpreter" % lang)
    elif lang in ['c', 'java']:
        print("%6s need compiler" % lang)
    else:
        print("should not reach here")
```

- 프로그램이 실행 되게 하려면 줄(들여쓰기)을 반드시 맞추어야 함
 - 가독성 ↑

01-2 파이썬의 특징

- 파이썬은 개발 속도가 빠르다

"Life is too short, You need Python."
인생은 너무 짧으니 파이썬이 필요해.

- 파이썬의 빠른 개발 속도를 두고 유행처럼 퍼진 말

01-3 파이썬으로 무엇을 할 수 있을까?

■ 파이썬으로 할 수 있는 일

■ 시스템 유틸리티 제작

- 운영체제(윈도우, 리눅스 등)의 시스템 명령어를 사용하는 도구를 통한 시스템 유틸리티 제작

■ GUI(Graphic User Interface) 프로그래밍

- 화면에 윈도우 창을 만들고 프로그램을 동작시킬 수 있는 메뉴나 버튼, 그림 등을 추가하는 것
- GUI 프로그래밍을 위한 도구들을 갖추고 있어, GUI 프로그램을 만들기 쉬움
- Tkinter(티케이인터), PyQt

■ C/C++와의 결합

- C나 C++로 만든 프로그램을 파이썬에서, 파이썬으로 만든 프로그램을 C나 C++에서 사용 가능

■ 웹 프로그래밍

- Flask, Django

01-3 파이썬으로 무엇을 할 수 있을까?

■ 파이썬으로 할 수 있는 일

■ 수치 연산 프로그래밍

- C로 작성된 수치 연산 모듈 NumPy를 통해 빠른 수치 연산 가능

■ 데이터베이스 프로그래밍

- Sybase, Infomix, Oracle, MySQL, PostgreSQL 등의 데이터에 접근하기 위한 도구 제공
- 자료를 변형 없이 그대로 파일에 저장하고 불러오는 파이썬 모듈 피클(pickle)

■ 데이터분석, 사물 인터넷

- 판다스(Pandas) 모듈을 통한 데이터 분석
- 라즈베리파이를 제어하는 도구(GPIO(General Purpose Input/Output) 모듈)를 통한 사물 인터넷 구현

01-3 파이썬으로 무엇을 할 수 있을까?

■ 파이썬으로 할 수 없는 일

■ 시스템과 밀접한 프로그래밍 영역

- 운영체제, 엄청난 횟수의 반복과 연산이 필요한 프로그램, 데이터 압축 알고리즘 개발 프로그램 등 대단히 빠른 속도를 요구하거나 하드웨어를 직접 건드려야 하는 프로그램에는 어울리지 않음

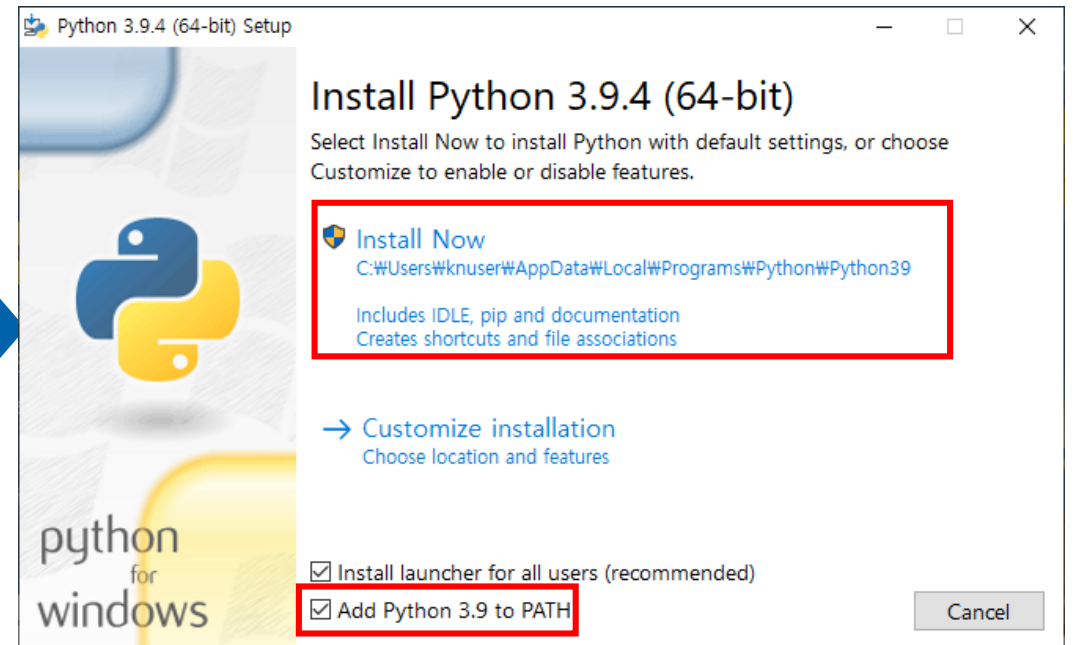
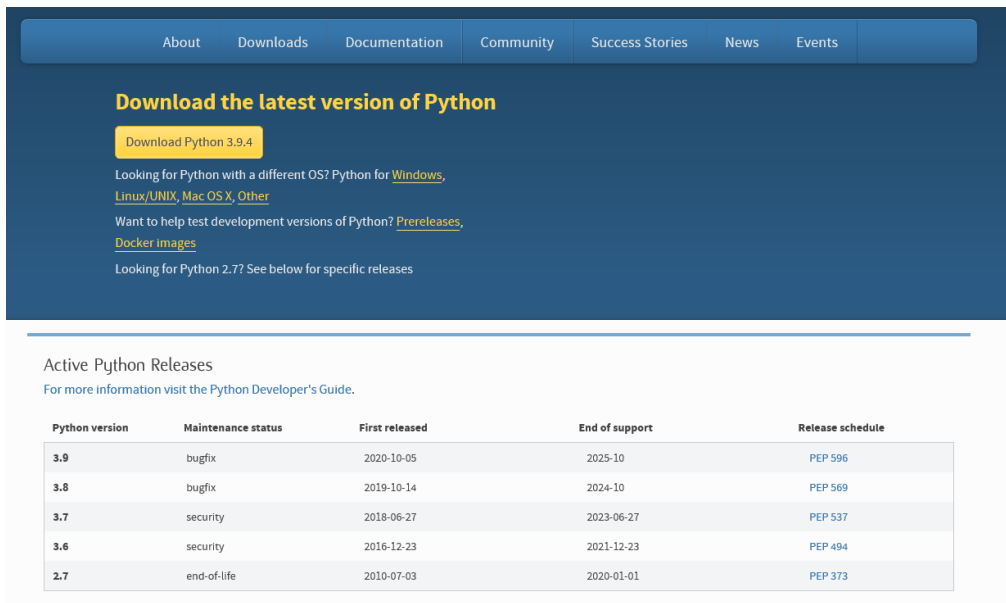
■ 모바일 프로그래밍

- 안드로이드 앱(App)을 개발하는 것은 아직 어려움

01-4 파이썬 설치하기

■ 윈도우에서 파이썬 설치하기

1. 파이썬 공식 홈페이지(www.python.org/downloads)에서 Python 3.x 최신 버전 다운로드
2. 파이썬이 어느 곳에서든지 실행될 수 있도록 '**Add Python 3.x to PATH**' 옵션 선택

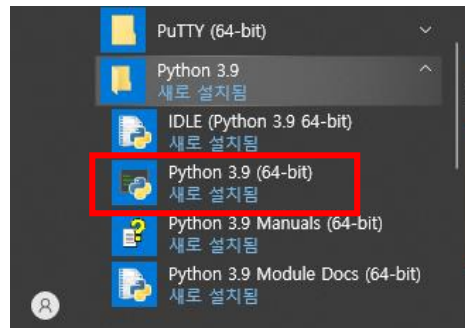


01-5 파이썬 둘러보기

■ 파이썬 기초 실습 준비하기

■ 파이썬 대화형 인터프리터 실행

- [시작] → [프로그램]



■ 인터프리터

- 사용자가 입력한 소스 코드를 실행하는 환경

```
Python 3.9 (64-bit)
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

■ 수식 입력

```
>>> 1 + 1
2
```

- 입력에 따른 결과값이 바로 출력됨

■ 대화형 인터프리터 종료

- 1) Ctrl + Z → Enter
- 2) sys 사용

```
>>> import sys
>>> sys.exit()
```

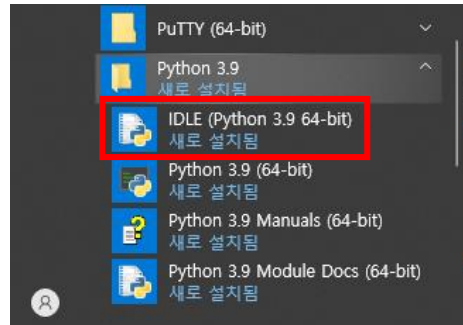
01-6 파이썬과 에디터

■ IDLE로 파이썬 프로그램 작성하기

■ 파이썬 IDLE 실행

- IDLE(Integrated Development and Learning Environment)
 - 파이썬 프로그램 작성을 도와주는 통합 개발 환경

■ [시작] → [프로그램]



■ 셸(Shell)

```
IDLE Shell 3.9.4
File Edit Shell Debug Options Window Help
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

01-6 파이썬과 에디터

■ IDLE로 파이썬 프로그램 작성하기

■ IDLE의 2가지 창

- **셸 창(Shell Window)**: IDLE 에디터에서 실행한 프로그램의 결과가 표시되는 창으로서 파이썬 셸과 동일한 기능을 수행한다. IDLE을 실행하면 가장 먼저 나타나는 창이다.
- **에디터 창(Editor Window)**: IDLE 에디터가 실행되는 창이다.

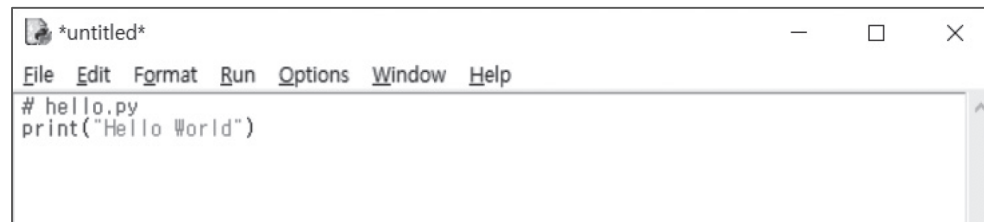
■ 에디터(Editor) 실행



01-6 파이썬과 에디터

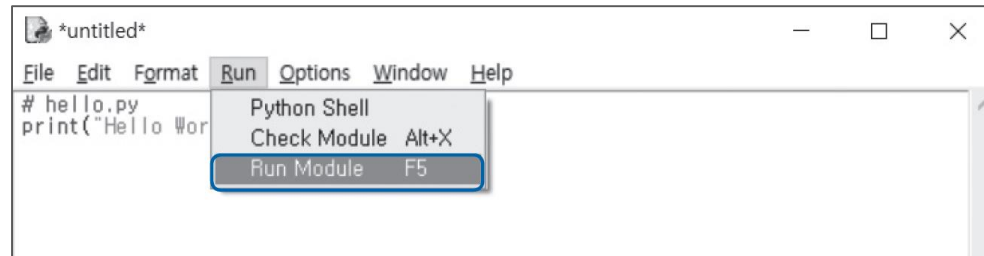
■ IDLE로 파이썬 프로그램 작성하기

1) 파이썬 프로그램 작성



```
*untitled*
File Edit Format Run Options Window Help
# hello.py
print('Hello World')
```

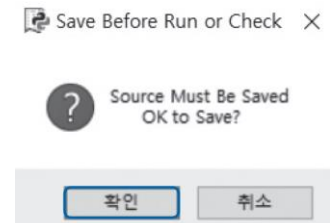
2) 작성한 프로그램 실행(F5)



```
*untitled*
File Edit Format Run Options Window Help
# hello.py
print('Hello Wor
Python Shell
Check Module Alt+X
Run Module F5
```

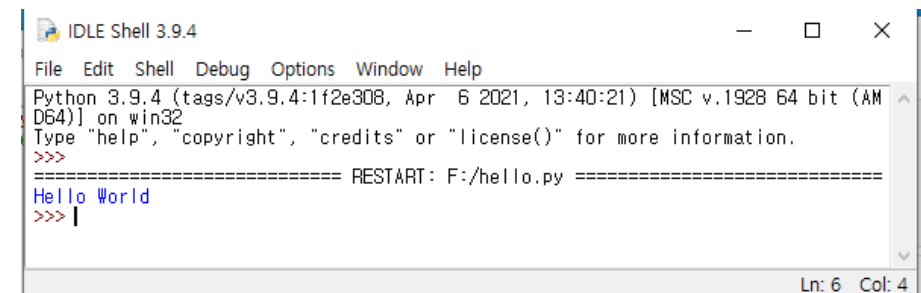
3) 파일 저장

- 예) F:\hello.py



4) 실행 결과

- IDLE 셸 창에 표시됨



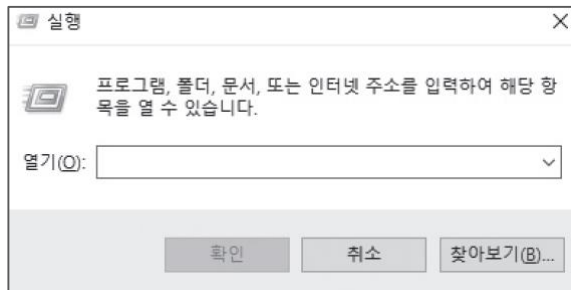
```
IDLE Shell 3.9.4
File Edit Shell Debug Options Window Help
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:/hello.py =====
Hello World
>>>
Ln: 6 Col: 4
```

01-6 파이썬과 에디터

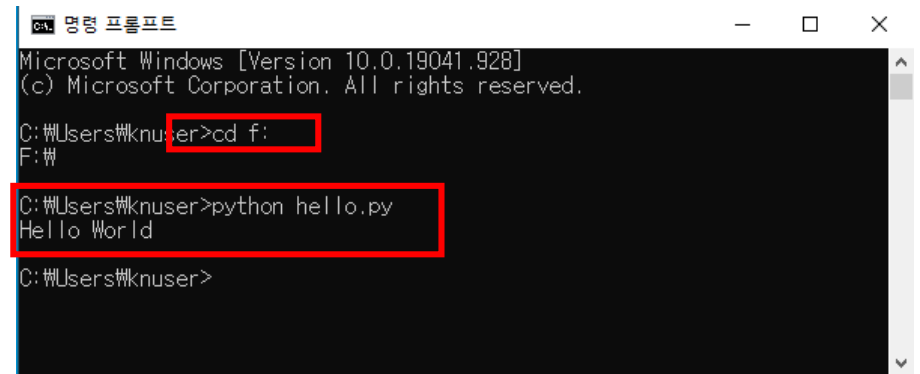
■ 명령 프롬프트 창에서 파이썬 프로그램 실행하기

1) 명령 프롬프트 창(command prompt) 열기

- 윈도우 키 + R → 'cmd' 입력 후 Enter



2) hello.py 저장한 경로로 이동하여 실행



01-6 파이썬 개발 툴

■ IDE(Integrated Development Environment)

- 효율적으로 소프트웨어를 개발하기 위한 통합개발환경 소프트웨어 어플리케이션 인터페이스이다. 코드 편집기, 디버거, 컴파일러, 인터프리터 등을 포함하고 개발자에게 제공한다
 - 네이버 지식 백과

■ 파이썬 IDE

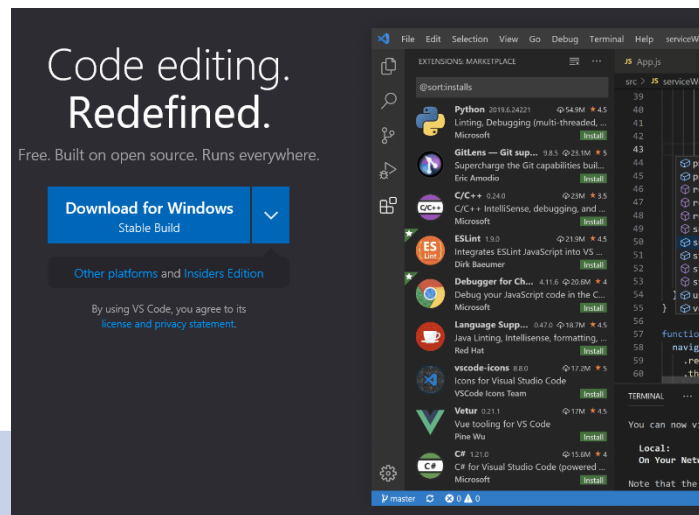
[파이참]

<https://www.jetbrains.com/ko-kr/pycharm/>



[VS code]

<https://code.visualstudio.com/>



[Jupyter Notebook]

<https://jupyter.org/>



01-6 파이썬 개발 툴(Jupyter Notebook)


- <https://www.anaconda.com/download>

Anaconda Distribution




Free Download

Everything you need to get started in data science on your workstation.

- ✓ Free distribution install
- ✓ Thousands of the most fundamental DS, AI, and ML packages
- ✓ Manage packages and environments from desktop application
- ✓ Deploy across hardware and software platforms

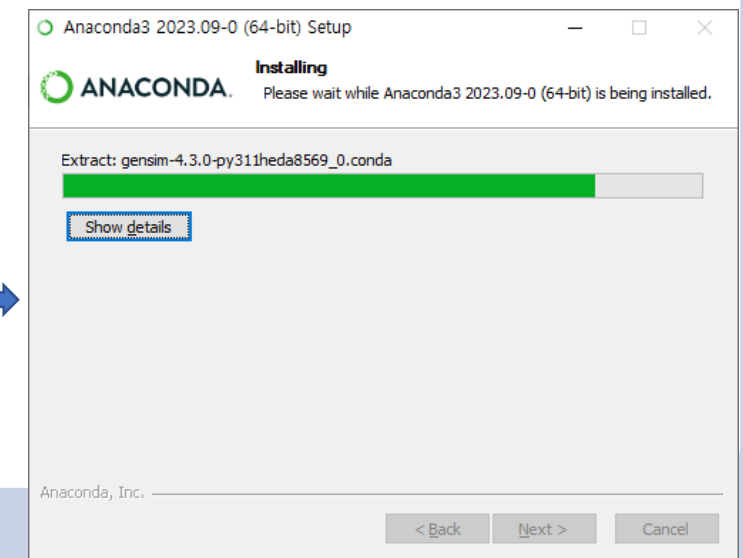
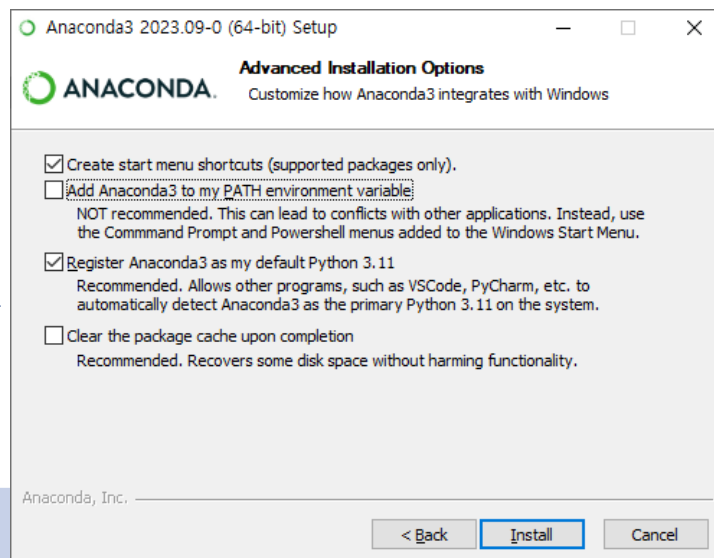
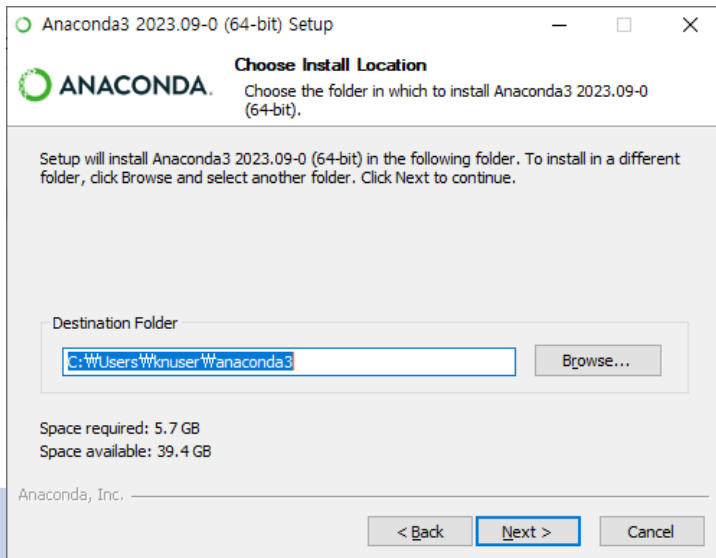
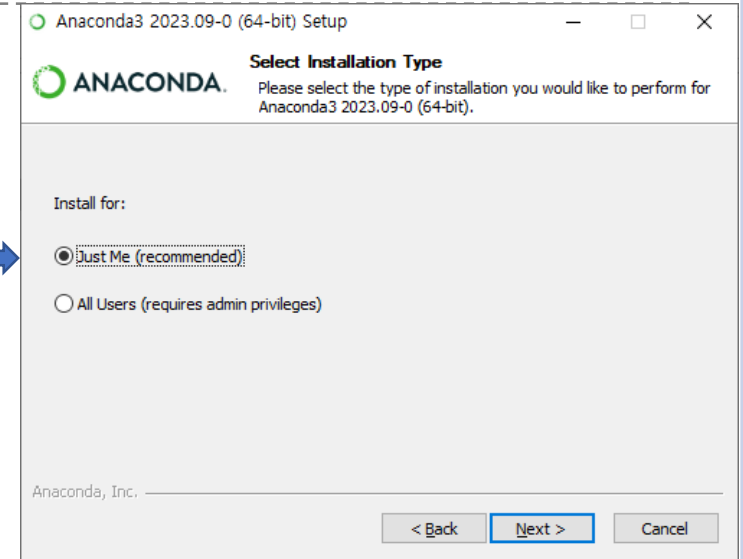
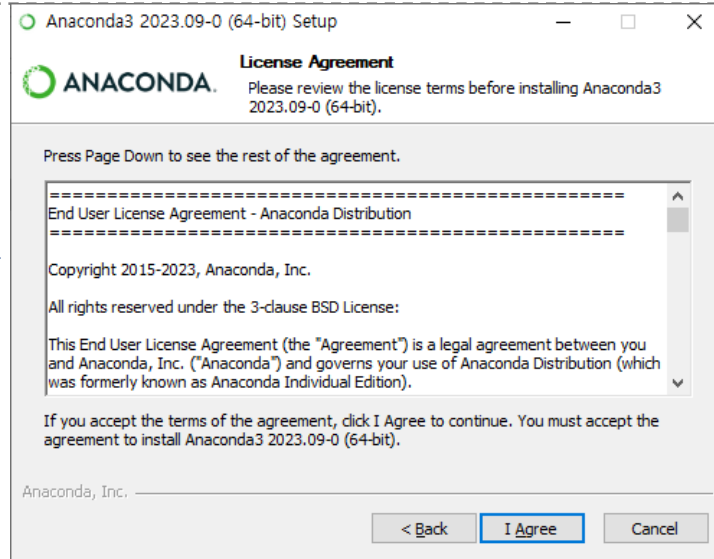
 Download

Get Additional Installers

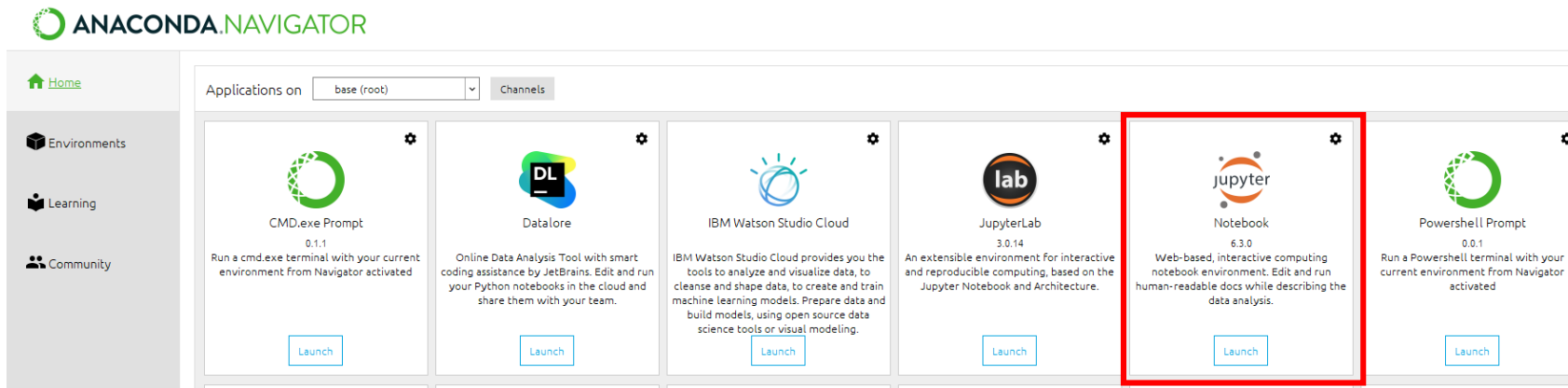
Anaconda Installers

 Windows	 Mac	 Linux
Python 3.11 ↓ 64-Bit Graphical Installer (898.6 MB)	Python 3.11 ↓ 64-Bit Graphical Installer (610.5 MB) ↓ 64-Bit Command Line Installer (612.1 MB) ↓ 64-Bit (M1) Graphical Installer (643.9 MB) ↓ 64-Bit (M1) Command Line Installer (645.6 MB)	Python 3.11 ↓ 64-Bit (x86) Installer (1015.6 MB) ↓ 64-Bit (Power8 and Power9) Installer (473.8 MB) ↓ 64-Bit (AWS Graviton2 / ARM64) Installer (727.4 MB) ↓ 64-bit (Linux on IBM Z & LinuxONE) Installer (340.8 MB)

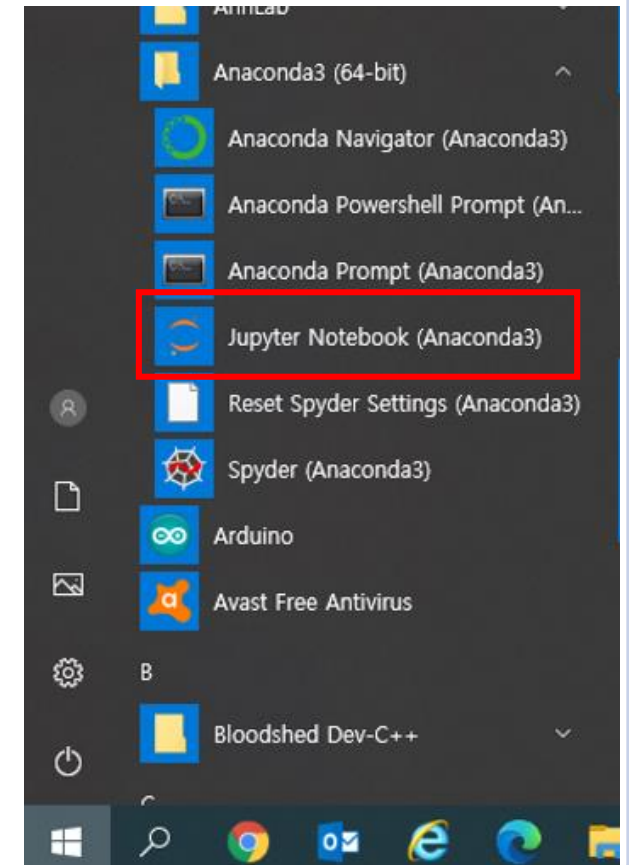


01-6 파이썬 개발 툴(Jupyter Notebook)

■ ANACONDA NAVIGATOR



■ 시작 메뉴



01-6 파이썬 개발 툴(Jupyter Notebook)

- 프로그램 코드를 브라우저에서 실행해주는 대화식 환경.
- 탐색적 데이터 분석에 적합.



The screenshot displays the Jupyter Notebook web interface. At the top, the 'jupyter' logo is on the left, and 'Quit' and 'Logout' buttons are on the right. Below the logo, there are tabs for 'Files', 'Running', and 'Clusters', with 'Files' being the active tab. A message 'Select items to perform actions on them.' is shown above a table of files. To the right of this message are buttons for 'Upload', 'New' (with a dropdown arrow), and a refresh icon. The table lists various system folders with checkboxes, names, last modified times, and file sizes.

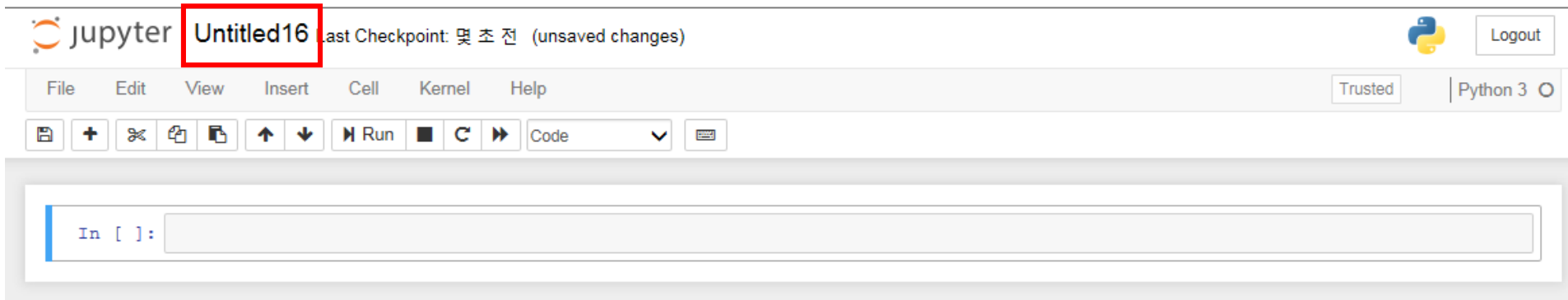
<input type="checkbox"/> 0 ▾	📁 /	Name ▾	Last Modified	File size
<input type="checkbox"/>	📁 Contacts		12일 전	
<input type="checkbox"/>	📁 Desktop		14분 전	
<input type="checkbox"/>	📁 Documents		2일 전	
<input type="checkbox"/>	📁 Downloads		12일 전	
<input type="checkbox"/>	📁 Favorites		5일 전	
<input type="checkbox"/>	📁 Links		12일 전	
<input type="checkbox"/>	📁 Music		12일 전	
<input type="checkbox"/>	📁 Pictures		12일 전	
<input type="checkbox"/>	📁 Saved Games		12일 전	
<input type="checkbox"/>	📁 Searches		12일 전	
<input type="checkbox"/>	📁 Videos		12일 전	

01-6 파이썬 개발 툴(Jupyter Notebook)

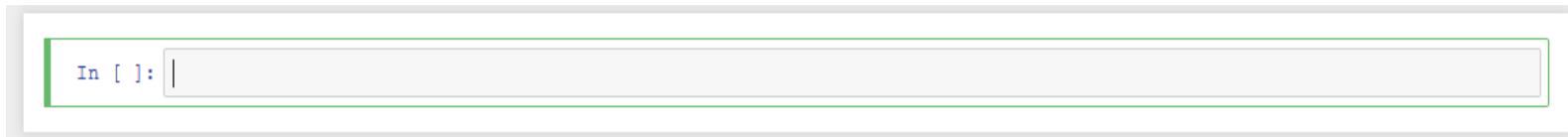
■ 실행

The screenshot displays the Jupyter Notebook web interface. At the top, the 'jupyter' logo is on the left, and 'Quit' and 'Logout' buttons are on the right. Below the logo, there are tabs for 'Files', 'Running', and 'Clusters'. A message 'Select items to perform actions on them.' is shown above a file list. The file list includes folders like 'Contacts', 'Desktop', 'Documents', 'Downloads', 'Favorites', 'Links', 'Music', 'Pictures', 'Saved Games', 'Searches', and 'Videos'. On the right side of the interface, there are 'Upload', 'New', and a refresh icon. A red box highlights the 'New' dropdown menu, which is open and shows 'Python 3' as the selected option under the 'Notebook:' section. Other options listed are 'Text File', 'Folder', and 'Terminal'. The date '5일 전' (5 days ago) is visible below the dropdown menu.

01-6 파이썬 개발 툴(Jupyter Notebook)



- 하나의 칸을 셀(cell)이라고 함
- 기본적으로 new를 실행하면 Untitled.ipynb로 생성
- 커맨드 모드(활성화)와 편집 모드(비활성화)로 구분 됨
 - 커맨드 모드 : enter



- 편집 모드 : esc



01-6 파이썬 개발 툴(Jupyter Notebook)

- **.ipynb라는 확장자 파일**

- Ipython Notebook 이라는 제품명이었을 때의 흔적

- **처음 작성시 Untitled.ipynb라는 파일명 생성 됨**

- **Auto Save 기능**

- 120 초

- **체크포인트**

- 코드의 입력이나 실행 유무와 같은 Notebook의 어느 시점에서의 논리적인 중단점이며, 그 때의 처리 상태를 완전하게 보존하고 후에 그 시점에서 처리를 재개할 수 있도록 한 포인트

01-6 파이썬 개발 툴(Jupyter Notebook)

■ Markdown과 수식의 이용

- Markdown은 문서 기술에 대한 서식을 정한 경량 마크업 언어의 한가지

[Markdown 예제]

```
# 이것이 표제이다.  
이것은 단락이다.  
- 조항 쓰기도  
- 이렇게 할 수 있다.  
테이블도 기술할 수 있다.  
Package | version  
----- | -----  
pandas | 0.19.2  
matplotlib | 2.0.0
```

이것이 표제이다.

이것은 단락이다.

- 조항 쓰기도
- 이렇게 할 수 있다.

테이블도 기술할 수 있다.

Package	version
pandas	0.19.2
matplotlib	2.0.0

01-6 파이썬 개발 툴(Jupyter Notebook)

- Notebook에서 Python 설명 코멘트로 Markdown 이용

표준 라이브러리 이용하기

이 Notebook에서는 Python의 표준 라이브러리 이용 방법을 설명한다.
표준 라이브러리를 이용할 때는 "import"문을 기술한다.

```
In [1]: import math
```

"import"한 "math" 모듈을 이용한다.

```
In [2]: print(math.sqrt(4))
```

표준 라이브러리 이용하기

이 Notebook에서는 Python의 표준 라이브러리 이용 방법을 설명한다.
표준 라이브러리를 이용할 때는 "import"문을 기술한다.

```
In [3]: import math
```

"import"한 "math" 모듈을 이용한다.

```
In [4]: print(math.sqrt(4))
```

2.0

- 수식 기술하기

- Markdown 타입으로 설정 후 입력

수식 표시

이 코드 셀에는 다음과 같이 서식을 표현할 수 있다.

```
%%begin{align}  
%sqrt{2x-1}+(3+x)^3$  
%%end{align}
```

```
$$ e=mc^2 $$
```

인라인에서도 오른쪽에 쓰여진 것처럼 $\sqrt{a^2+b^2}$ 기술할 수 있다.

수식 표시

이 코드 셀에는 다음과 같이 서식을 표현할 수 있다.

$$\sqrt{2x-1} + (3+x)^3$$$
$$e = mc^2$$

인라인에서도 오른쪽에 쓰여진 것처럼 $\sqrt{a^2 + b^2}$ 기술할 수 있다.

01-6 파이썬 개발 툴(Jupyter Notebook)

■ 이미지 첨부하기

- Markdown 타입 코드 셀에 이미지를 드래그 & 드롭하여 이미지 첨부 가능

```
## Jupyter 기동 후 홈 화면이다.
```

```

```

Jupyter 기동 후 홈 화면이다.

이것이 표제이다.

이것은 단락이다.

- 조항 쓰기도
- 이렇게 할 수 있다.

테이블도 기술할 수 있다.

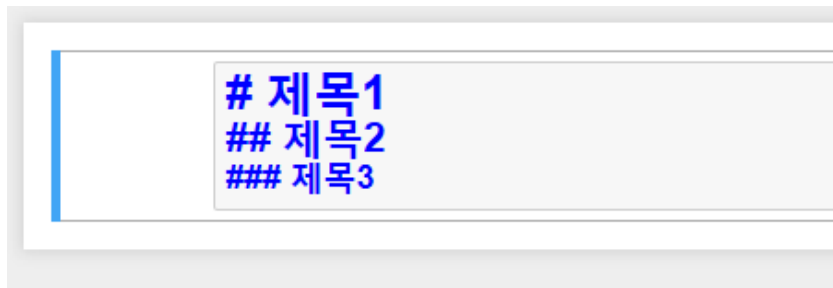
Package	version
pandas	0.19.2
matplotlib	2.0.0

■ 매직 명령어

- 유틸리티 기능 제공
- %pwd : 현재 디렉토리 출력
- %time : Python의 실행 시간 측정
- %timeit : 여러 번 시행한 결과의 측정 값을 요약해서 나타냄
- %%timeit : 코드 셀 전체에 적용
- %history : 코드 셀의 실행 이력을 목록으로 취득하는 명령
- %ls : 파일 목록 리스트
- %autosave : autosave 횟수 지정 ex)%autosave 60

01-6 파이썬 개발 툴(Jupyter Notebook)

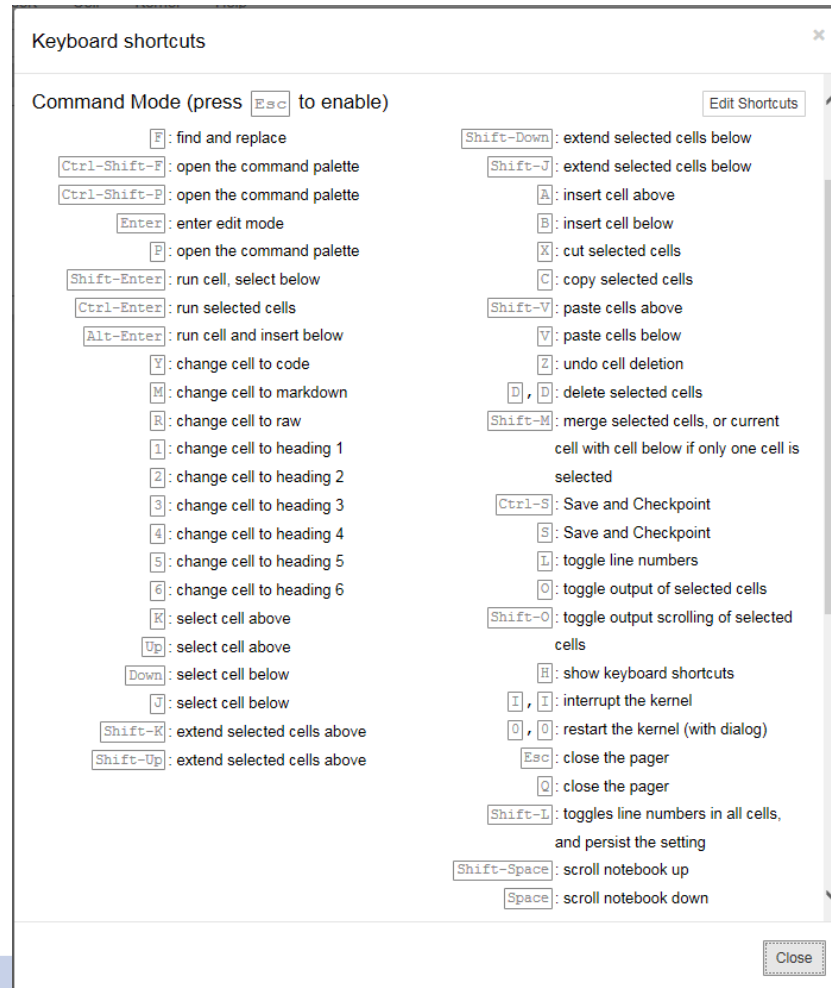
- **shit enter로 cell 실행**
- **다양한 편집 모드 가능**
 - code, markdown 등
- **Markdown으로 바꾸고 제목을 입력 (편집모드에서 단축키 M)**
 - 제목 입력시에는 # 삽입
 - #이 한개면 가장 큰 제목
 - ##이 두개면 두번째 제목
 - ### 이렇게 붙여가면서 크기 조절 가능



- **편집모드에서 M을 누르면 markdown으로 되고,**
- **Y를 누르면 code모드**

01-6 파이썬 개발 툴(Jupyter Notebook)

■ help –Keyboard Shorts 메뉴



01-6 파이썬 개발 툴(Jupyter Notebook)

- Jupyter Notebook 더 자세히 알고 싶다면...

<https://jupyter-notebook.readthedocs.io/en/latest/notebook.html>

파이썬 프로그래밍의 기초, 자료형

02-1 숫자형

02-2 문자열 자료형

02-3 리스트 자료형

02-4 딕셔너리 자료형

02-5 자료형의 값을 저장하는 공간, 변수

02-6 입/출력 함수

02-1 숫자형

■ 숫자형(Number)이란?

- 숫자 형태로 이루어진 자료형

항목	파이썬 사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
8진수	0o34, 0o25
16진수	0x2A, 0xFF

02-1 숫자형

■ 숫자형 사용법

■ 정수형(Integer)

- 정수를 뜻하는 자료형

```
>>> a = 123
>>> a = -178
>>> a = 0
```

■ 실수형(Floating-point)

- 소수점이 포함된 숫자

```
>>> a = 1.2
>>> a = -3.45
```

```
>>> a = 4.24E10
>>> a = 4.24e-10
```

※ 컴퓨터식 지수 표현 방식

■ 8진수(Octal)

- 숫자 0 + 알파벳 소문자 o 또는 대문자 O

```
>>> a = 0o177
```

■ 16진수(Hexadecimal)

- 숫자 0 + 알파벳 소문자 x

```
>>> a = 0x8ff
>>> b = 0xABC
```

02-1 숫자형

■ 숫자형 활용하기 위한 연산자

■ 사칙연산

```
>>> a = 3
>>> b = 4
>>> a + b
7
>>> a * b
12
>>> a / b
0.75
```

■ // 연산자 : 나눗셈 후 몫 반환

```
>>> 7 // 4
1
```

■ ** 연산자 : 제곱

```
>>> a = 3
>>> b = 4
>>> a ** b
81
```

■ % 연산자 : 나눗셈 후 나머지 반환

```
>>> 7 % 3
1
>>> 3 % 7
3
```

02-2 문자열 자료형

- 문자열(String)이란?

- 문자, 단어 등으로 구성된 문자들의 집합

```
"Life is too short, You need Python"
```

```
"a"
```

```
"123"
```

02-2 문자열 자료형

■ 문자열 사용법

■ 문자열 만드는 방법

1. 큰따옴표("")

```
"Hello World"
```

2. 작은따옴표('')

```
'Python is fun'
```

3. 큰따옴표 3개(""")

```
"""Life is too short, You need python"""
```

4. 작은따옴표 3개(''')

```
'''Life is too short, You need python'''
```

02-2 문자열 자료형

■ 문자열 사용법

■ 문자열 안에 작은따옴표나 큰따옴표를 포함시키기

1. 작은따옴표(')

- 큰따옴표(")로 둘러싸기

```
>>> food = "Python's favorite food is perl"
```

2. 큰따옴표(")

- 작은따옴표(')로 둘러싸기

```
>>> say = '"Python is very easy." he says.'
```

3. 백슬래시(\) 활용하기

- 백슬래시(\) 뒤의 작은따옴표(')나 큰따옴표(")는 문자열을 둘러싸는 기호의 의미가 아니라 문자 ('), (") 그 자체를 의미

```
>>> food = 'Python\'s favorite food is perl'
>>> say = "\"Python is very easy.\" he says."
```

02-2 문자열 자료형

■ 문자열 사용법

■ 여러 줄인 문자열을 변수에 대입하고 싶을 때

1. 이스케이프 코드 '\n' 삽입

```
>>> multiline = "Life is too short\nYou need python"
```

2. 작은따옴표 3개('')

```
>>> multiline = '''  
... Life is too short  
... You need python  
... '''
```

3. 큰따옴표 3개(""")

```
>>> multiline = """  
... Life is too short  
... You need python  
... """
```

02-2 문자열 자료형

■ 문자열 연산하기

1. 문자열 더해서 연결하기(Concatenation)

```
>>> head = "Python"
>>> tail = " is fun!"
>>> head + tail
'Python is fun!'
```

2. 문자열 곱하기

```
>>> a = "python"
>>> a * 2
'pythonpython'
```

3. 문자열 길이 구하기

■ 파이썬 기본 내장 함수 len()

```
>>> a = "Life is too short"
>>> len(a)
17
```

02-2 문자열 자료형

■ 문자열 인덱싱

■ 인덱싱(Indexing)

- '가리킨다'는 의미
- 파이썬은 0부터 숫자를 셈
- a[번호]
 - 문자열 안의 특정 값 뽑아냄
 - 마이너스(-)
 - 문자열 뒤부터 셈

L	i	f	e		i	s		t	o	o		s	h	o	r	t	,		Y	o	u		n	e	e	d		P	y	t	h	o	n
0									1										2										3				
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3

```
>>> a = "Life is too short, You need Python"
>>> a[0]
'L'
>>> a[12]
's'
>>> a[-1]
'n'
```

```
a[0]: 'L', a[1]: 'i', a[2]: 'f', a[3]: 'e', a[4]: ' ', ...
```


02-2 문자열 자료형

■ 문자열 슬라이싱

■ 슬라이싱(Slicing)

- '잘라낸다'는 의미
- a[시작 번호:끝 번호]
 - 시작 번호부터 끝 번호까지의 문자를 뽑아냄
 - 끝 번호에 해당하는 것은 포함하지 않음

L	i	f	e		i	s		t	o	o		s	h	o	r	t	,		Y	o	u		n	e	e	d		P	y	t	h	o	n
0										1										2										3			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3

```
>>> a = "Life is too short, You need Python"
>>> a[0:4]
'Life'
```

```
>>> a = "20010331Rainy"
>>> date = a[:8]
>>> weather = a[8:]
>>> date
'20010331'
>>> weather
'Rainy'
```

02-2 문자열 자료형

■ 문자열 포매팅

■ 포매팅(Formatting)

1. 숫자 바로 대입

- 문자열 포맷 코드 **%d**

```
>>> "I eat %d apples." % 3
'I eat 3 apples.'
```

```
>>> number = 3
>>> "I eat %d apples." % number
'I eat 3 apples.'
```

2. 문자열 바로 대입

- 문자열 포맷 코드 **%s**

```
>>> "I eat %s apples." % "five"
'I eat five apples.'
```

02-2 문자열 자료형

■ 문자열 포매팅

■ 문자열 포맷 코드

코드	설명
%s	문자열(String)
%c	문자 1개(Character)
%d	정수(Integer)
%f	부동 소수(Floating-point)
%o	8진수
%x	16진수
%%	Literal % (문자 '%' 자체)

```
>>> number = 10
>>> day = "three"
>>> "I ate %d apples. so I was sick for %s days." % (number, day)
'I ate 10 apples. so I was sick for three days.'
```

```
>>> "I have %s apples" % 3
'I have 3 apples'
>>> "rate is %s" % 3.234
'rate is 3.234'
```

02-2 문자열 자료형

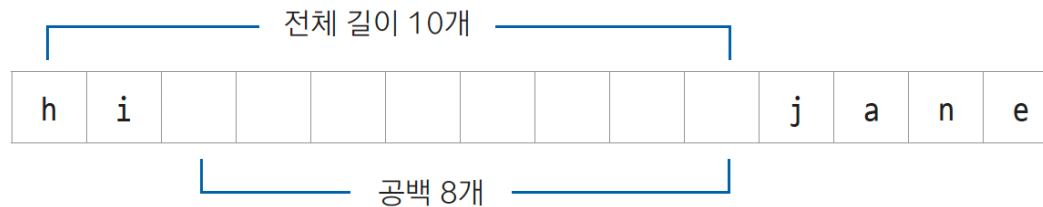
■ 문자열 포매팅

■ 문자열 포맷 코드 활용법

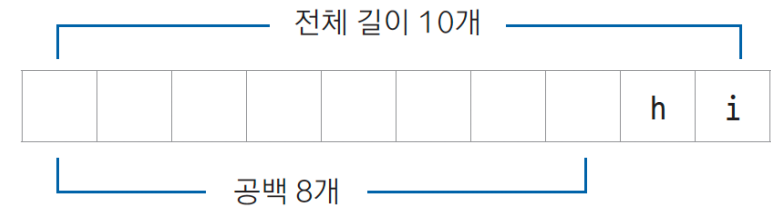
1. 정렬과 공백

- %s를 숫자와 함께 사용하면, 공백과 정렬 표현 가능

```
>>> "%10s" % "hi"
'          hi' ← hi가 오른쪽 정렬됨
```



```
>>> "%-10sjane" % 'hi'
'hi          jane' ← hi가 왼쪽 정렬됨
```



02-2 문자열 자료형

■ 문자열 포매팅

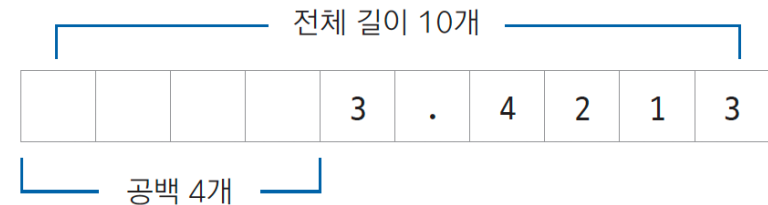
■ 문자열 포맷 코드 활용법

2. 소수점 표현하기

- %f를 숫자와 함께 사용하면, 소수점 뒤에 나올 숫자의 개수 조절 및 정렬 가능

```
>>> "%0.4f" % 3.42134234  
'3.4213'
```

```
>>> "%10.4f" % 3.42134234  
'      3.4213'
```



02-2 문자열 자료형

■ 문자열 포매팅

■ f 문자열 포매팅

- 파이썬 3.6 버전부터 f 문자열 포매팅 기능 제공
- 문자열 앞에 f 접두사를 붙이면, f 문자열 포매팅 기능 사용 가능

```
>>> name = '홍길동'
>>> age = 30
>>> f'나의 이름은 {name}입니다. 나이는 {age}입니다.'
'나의 이름은 홍길동입니다. 나이는 30입니다.'
```

```
>>> age = 30
>>> f'나는 내년이면 {age+1}살이 된다.'
'나는 내년이면 31살이 된다.'
```

02-2 문자열 자료형

■ 문자열 관련 함수

- 문자열 자료형이 가진 내장 함수

- **count()**

- 문자 개수 세는 함수

```
>>> a = "hobby"
>>> a.count('b')
2
```

- **find()**

- 찾는 문자열이 처음 나온 위치 반환
 - 없으면 -1 반환

```
>>> a = "Python is the best choice"
>>> a.find('b')
14 ← 문자열에서 b가 처음 나온 위치
>>> a.find('k')
-1
```

02-2 문자열 자료형

■ 문자열 관련 함수

■ index()

- find와 마찬가지로,
찾는 문자열이 처음 나온 위치 반환
- 단, 찾는 문자열이 없으면 오류 발생

```
>>> a = "Life is too short"
>>> a.index('t')
8
>>> a.index('k')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

— k가 없기 때문에 오류 발생

■ join()

- 문자열 삽입

```
>>> ",".join('abcd')
'a,b,c,d'
```

■ upper()

- 소문자를 대문자로 변환

```
>>> a = "hi"
>>> a.upper()
'HI'
```


02-2 문자열 자료형

■ 문자열 관련 함수

■ lower()

- 대문자를 소문자로 변환

```
>>> a = "HI"  
>>> a.lower()  
'hi'
```

■ lstrip()

- 가장 왼쪽에 있는 연속된 공백 삭제

```
>>> a = " hi "  
>>> a.lstrip()  
'hi '
```

■ rstrip()

- 가장 오른쪽에 있는 연속된 공백 삭제

```
>>> a= " hi "  
>>> a.rstrip()  
' hi'
```

■ strip()

- 양쪽에 있는 연속된 공백 삭제

```
>>> a = " hi "  
>>> a.strip()  
'hi'
```

02-2 문자열 자료형

■ 문자열 관련 함수

■ replace()

- replace(바뀌게 될 문자열, 바꿀 문자열)
- 문자열 안의 특정 값을 다른 값으로 치환

```
>>> a = "Life is too short"
>>> a.replace("Life", "Your leg")
'Your leg is too short'
```

■ split()

- 공백 또는 특정 문자열을 구분자로 해서 문자열 분리
분리된 문자열은 리스트로 반환됨

```
>>> a = "Life is too short"
>>> a.split() ← 공백을 기준으로 문자열 나눔
['Life', 'is', 'too', 'short']
>>> b = "a:b:c:d"
>>> b.split(':') ← : 기호를 기준으로 문자열 나눔
['a', 'b', 'c', 'd']
```

02-3 리스트 자료형

■ 리스트(List)란?

- 자료형의 집합을 표현할 수 있는 자료형

```
>>> odd = [1, 3, 5, 7, 9]
```

- 숫자와 문자열만으로 프로그래밍을 하기엔 부족한 점이 많음
 - 예) 1부터 10까지의 숫자 중 홀수 모음인 집합 {1, 3, 5, 7, 9}는 숫자나 문자열로 표현 불가능
 - 리스트로 해결 가능!

02-3 리스트 자료형

■ 리스트 사용법

- 대괄호([])로 감싸고 각 요소값은 쉼표(,)로 구분

```
리스트명 = [요소1, 요소2, 요소3, ...]
```

- 리스트 안에 어떠한 자료형도 포함 가능

```
>>> a = []  
>>> b = [1, 2, 3]  
>>> c = ['Life', 'is', 'too', 'short']  
>>> d = [1, 2, 'Life', 'is']  
>>> e = [1, 2, ['Life', 'is']]
```

02-3 리스트 자료형

■ 리스트 인덱싱

- 문자열과 같이 인덱싱 적용 가능

```
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
```

- 파이썬은 숫자를 0부터 세기 때문에 a[0]이 리스트 a의 첫 번째 요소

```
>>> a[0]
1
```

- a[-1]은 리스트 a의 마지막 요솟값

```
>>> a[-1]
3
```

- 요솟값 간의 덧셈

```
>>> a[0] + a[2] ← 1 + 3
4
```

02-3 리스트 자료형

■ 리스트 인덱싱

- 리스트 내에 리스트가 있는 경우

```
>>> a = [1, 2, 3, ['a', 'b', 'c']]
```

- a[-1]은 마지막 요솟값인 리스트 ['a', 'b', 'c'] 반환

```
>>> a[0]
1
>>> a[-1]
['a', 'b', 'c']
```

- 리스트 a에 포함된 ['a', 'b', 'c'] 리스트에서 'a' 값을 인덱싱을 사용해 반환할 방법은?

```
>>> a[-1][0]
'a'
```

- a[-1]로 리스트 ['a', 'b', 'c']에 접근하고, [0]으로 요소 'a'에 접근

02-3 리스트 자료형

■ 리스트 슬라이싱

- 문자열과 같이 슬라이싱 적용 가능

```
>>> a = [1, 2, 3, 4, 5]
>>> a[0:2]
[1, 2]
```

```
>>> a = [1, 2, 3, 4, 5]
>>> b = a[:2] ← 처음부터 a[1]까지
>>> c = a[2:] ← a[2]부터 마지막까지
>>> b
[1, 2]
>>> c
[3, 4, 5]
```

02-3 리스트 자료형

■ 리스트 연산하기

■ 더하기(+)

- + 기호는 2개의 리스트를 합치는 기능
- 문자열에서 "abc" + "def" = "abcdef"가 되는 것과 같은 의미

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
```

■ 반복하기(*)

- * 기호는 리스트의 반복을 의미
- 문자열에서 "abc" * 3 = "abccabccabc"가 되는 것과 같은 의미

```
>>> a = [1, 2, 3]
>>> a * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```


02-3 리스트 자료형

- 리스트 연산하기

- 리스트 길이 구하기

- len() 함수 사용
 - 문자열, 리스트 외에 앞으로 배울 튜플과 딕셔너리에서도 사용 가능한 내장 함수

```
>>> a = [1, 2, 3]
>>> len(a)
3
```

02-3 리스트 자료형

■ 리스트의 수정과 삭제

■ 리스트에서 값 수정하기

```
>>> a = [1, 2, 3]
>>> a[2] = 4
>>> a
[1, 2, 4]
```

■ 리스트 요소 삭제하기

■ [] 사용해 리스트 요소 삭제하기

```
>>> a = [1, 6, 7, 4]
>>> a[1:3] = []
>>> a
[1, 4]
```

■ del 키워드 사용

del 객체

```
>>> a = [1, 2, 3]
>>> del a[1]
>>> a
[1, 3]
```

```
>>> a = [1, 2, 3, 4, 5]
>>> del a[2:]
>>> a
[1, 2]
```

※ 슬라이싱 기법 활용 가능

02-3 리스트 자료형

■ 리스트 관련 함수

■ append()

- 리스트의 맨 마지막에 요소 추가

```
>>> a = [1, 2, 3]
>>> a.append(4) ← 리스트의 맨 마지막에 4를 추가
>>> a
[1, 2, 3, 4]
```

- 어떤 자료형도 추가 가능

```
>>> a.append([5,6]) ← 리스트의 맨 마지막에 [5,6]을 추가
>>> a
[1, 2, 3, 4, [5, 6]]
```

■ sort()

- 리스트의 요소를 순서대로 정렬

```
>>> a = [1, 4, 3, 2]
>>> a.sort()
>>> a
[1, 2, 3, 4]
```

- 문자의 경우 알파벳 순서로 정렬 가능

02-3 리스트 자료형

■ 리스트 관련 함수

■ reverse()

- 리스트를 역순으로 뒤집어 줌
- 요소를 역순으로 정렬하는 것이 아닌, 현재의 리스트 그대로 뒤집음

```
>>> a = ['a', 'c', 'b']
>>> a.reverse()
>>> a
['b', 'c', 'a']
```

■ index()

- 요소를 검색하여 위치 값 반환

```
>>> a = [1,2,3]
>>> a.index(3) ← 3은 리스트 a의 세 번째(a[2]) 요소
2
```

```
>>> lst = ['red', 'blue', 'black', 'blue']
>>> lst.index('blue')
1
>>> lst.index('blue', 2)
3
>>> lst.index('blue', 1, 3)
```

- 값이 존재하지 않으면, 값 오류 발생

```
>>> a.index(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: 0 is not in list
```

오류 메시지

02-3 리스트 자료형

■ 리스트 관련 함수

■ insert()

- 리스트에 요소 삽입
- insert(a, b)
 - a번째 위치에 b를 삽입하는 함수

```
>>> a = [1, 2, 3]
>>> a.insert(0, 4) ← a[0] 위치에 4 삽입
[4, 1, 2, 3]
```

■ remove()

- remove(x)
 - 리스트에서 첫 번째로 나오는 x를 삭제

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
[1, 2, 1, 2, 3]
```

- 값이 여러 개인 경우 첫 번째 것만 삭제

```
>>> a.remove(3)
[1, 2, 1, 2]
```

02-3 리스트 자료형

■ 리스트 관련 함수

■ pop()

- 리스트의 맨 마지막 요소를 돌려주고 해당 요소 삭제
- pop(x)
 - 리스트의 x번째 요소를 돌려주고 해당 요소 삭제

```
>>> a = [1, 2, 3]
>>> a.pop()
3
>>> a
[1, 2]
```

```
>>> a = [1, 2, 3]
>>> a.pop(1)
2
>>> a
[1, 3]
```

■ count()

- 리스트에 포함된 요소의 개수 반환
- count(x)
 - 리스트 안에 x가 몇 개 있는지 조사하여 그 개수를 돌려주는 함수

```
>>> a = [1, 2, 3, 1]
>>> a.count(1)
2
```

02-3 리스트 자료형

■ 리스트 관련 함수

■ extend()

- 리스트에 리스트를 더하는 함수
- extend(x)
 - x에는 리스트만 올 수 있음

a.extend([4, 5])

=

a += [4, 5]

```
>>> a = [1, 2, 3]
>>> a.extend([4, 5])
>>> a
[1, 2, 3, 4, 5]
>>> b = [6, 7]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6, 7]
```

02-4 딕셔너리 자료형

■ 딕셔너리(Dictionary)란?

- 대응 관계를 나타내는 자료형
- 연관 배열(Associative array) 또는 해시(Hash)
- Key와 Value를 한 쌍으로 갖는 자료형
- 순차적으로 해당 요솟값을 구하지 않고, Key를 통해 Value를 바로 얻는 특징
- key는 중복 및 변경 불가

02-4 딕셔너리 자료형

■ 딕셔너리의 모습

```
{Key1:Value1, Key2:Value2, Key3:Value3, ...}
```

- Key와 Value의 쌍 여러 개 (Key : Value)
- Key : 변하지 않는 값(숫자, 문자열, 튜플)을 사용
- Value : 변하는 값과 변하지 않는 값 모두 사용
- 순서이 개념이 존재하지 않아 인덱싱 및 슬라이싱 불가능
- {}로 둘러싸임
- 각 요소는 쉼표(,)로 구분됨

```
>>> dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
```

```
>>> a = {1: 'hi'}
```

```
>>> a = {'a': [1,2,3]}
```

02-4 딕셔너리 자료형

■ 딕셔너리 쌍 추가, 삭제하기

■ 딕셔너리 쌍 추가

```
>>> a = {1: 'a'}
>>> a[2] = 'b' ← {2: 'b'} 쌍 추가
>>> a
{1: 'a', 2: 'b'}
```

■ 딕셔너리 요소 삭제

```
>>> del a[1] ← key가 1인 key:value 쌍 삭제
>>> a
{'name': 'pey', 3: [1, 2, 3], 2: 'b'}
```

■ 딕셔너리에서 Key 사용해 Value 얻기

```
>>> grade = {'pey': 10, 'julliet': 99}
>>> grade['pey'] ← Key가 'pey'인 딕셔너리의 Value를 반환
10
>>> grade['julliet'] ← Key가 'julliet'인 딕셔너리의 Value를 반환
99
```

■ 리스트나 튜플, 문자열은 요솟값 접근 시 인덱싱이나 슬라이싱 기법을 사용

■ 딕셔너리는 Key를 사용해 Value 접근

02-4 딕셔너리 자료형

- 딕셔너리 만들 때 주의할 사항

- key 중복 금지

```
>>> a = {1:'a', 1:'b'}  
>>> a  
{1: 'b'}
```

- key에 리스트 사용 금지

```
>>> a = {[1,2] : 'hi'}  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: unhashable type: 'list'
```

02-4 딕셔너리 자료형

■ 딕셔너리 관련 함수

■ keys()

- Key만을 모아서 dict_keys 객체 반환

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
>>> a.keys()
dict_keys(['name', 'phone', 'birth'])
```

- 리스트처럼 사용할 수 있지만,
리스트 관련 함수(append, insert, pop 등)는
사용 불가능

```
>>> for k in a.keys():
...     print(k)
...
name
phone
birth
```

- dict_keys 객체를 리스트로 변환하는 방법

```
>>> list(a.keys())
['name', 'phone', 'birth']
```

02-4 딕셔너리 자료형

■ 딕셔너리 관련 함수

■ `values()`

- Value만을 모아서 `dict_values` 객체 반환

```
>>> a.values()  
dict_values(['pey', '0119993323', '1118'])
```

■ `items()`

- Key와 Value의 쌍을 튜플로 묶은 값을 모아서 `dic_items` 객체 반환

```
>>> a.items()  
dict_items([('name', 'pey'), ('phone', '0119993323'), ('birth', '1118')])
```

02-4 딕셔너리 자료형

■ 딕셔너리 관련 함수

■ clear()

- 딕셔너리 내의 모든 요소 삭제
- 빈 딕셔너리는 {}로 표현

```
>>> a.clear()
>>> a
{}

```

■ in

- Key가 딕셔너리 안에 있는지 조사
- Key가 딕셔너리 안에 존재하면 True, 존재하지 않으면 False 반환

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
>>> 'name' in a
True
>>> 'email' in a
False

```

02-4 딕셔너리 자료형

■ 딕셔너리 관련 함수

■ get()

- Key에 대응되는 Value 반환
- 존재하지 않는 키 사용 시 None 반환
 - 오류를 발생시키는 list와 차이가 있음
- Key 값이 없을 경우
디폴트 값을 대신 반환하도록 지정 가능

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
>>> a.get('name')
'pey'
>>> a.get('phone')
'0119993323'
```

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
>>> print(a.get('nokey')) ← None을 리턴함
None
>>> print(a['nokey'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'nokey'
```

```
>>> a.get('foo', 'bar')
'bar'
```

02-5 자료형의 값을 저장하는 공간, 변수

■ 변수란?

- 파이썬에서 사용하는 변수는 객체를 가리키는 것이라고 할 수 있음
 - 객체 = 자료형과 같은 것을 의미하는 말

```
>>> a = 1
>>> b = "python"
>>> c = [1,2,3]
```

- a, b, c를 변수라고 함

- 변수 선언 방법
 - =(assignment) 기호 사용

변수 이름 = 변수에 저장할 값

```
>>> 4pple = 10
>>> admin = 50
>>> Apple = 60
>>> apple = 70
>>> print(admin, Apple, apple)
>>> _4pple = 10
```


02-5 자료형의 값을 저장하는 공간, 변수

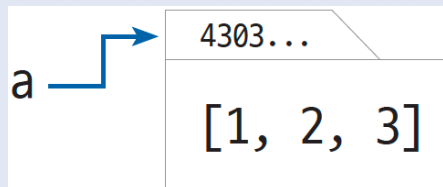
■ 변수란?

■ 변수의 예시

```
>>> a = [1, 2, 3]
```

- [1, 2, 3] 값을 가지는 리스트 자료형(객체)이 자동으로 메모리에 생성됨
- 변수 a는 [1, 2, 3] 리스트가 저장된 메모리의 주소를 가리킴
 - id() 함수를 사용하여 메모리 주소 확인

```
>>> a = [1, 2, 3]
>>> id(a)
4303029896
```



```
>>> a = [1,2,3]
```

```
>>> b = a
```

```
>>> id(a)
```

```
4303029896
```

```
>>> id(b)
```

```
4303029896
```

```
>>> a is b # a와 b가 가리키는 객체는 동일한가?
True
```

02-5 자료형의 값을 저장하는 공간, 변수

```
>>> a[1] = 4
>>> a
[1, 4, 3]
>>> b
[1, 4, 3]
```

- b 변수를 생성할 때 a 변수의 값을 가져오면서 a와는 다른 주소를 가리키도록 만들수는 없을까?

```
>>> a = [1, 2, 3]
>>> b = a[:]
>>> a[1] = 4
>>> a
[1, 4, 3]
>>> b
[1, 2, 3]
```

```
>>> from copy import copy
>>> b = copy(a)
```

```
>>> b is a
False
```

02-5 자료형의 값을 저장하는 공간, 변수

■ 변수명 작성시 유의사항

- 대소문자 구별
- 변수의 형(type)을 선언할 필요도 없음
- 변수명은 숫자, 알파벳, 밑줄 기호(_), 한글 등을 사용할 수 있음
- 변수명은 숫자로 시작하면 안되고, 공백을 포함하면 안됨
- 키워드(예약어) 사용 금지

False, None, True, and, as, assert, break, class, continue, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield

02-6 입/출력 함수

■ 입력함수(input()) 기본 활용법

```
print("Enter your name:")  
x = input()  
print("Hello, " + x)
```

■ 2개 이상의 값 입력

```
변수1, 변수2 = input().split()  
변수1, 변수2 = input().split('기준문자열')  
변수1, 변수2 = input('문자열').split()  
변수1, 변수2 = input('문자열').split('기준문자열')
```

02-6 입/출력 함수

- `input()` 함수 입력 값은 문자열임
- 사칙 연산 계산을 위해서는 형변환(casting) 필요

```
x = int(input())
```

- `map` 함수 사용하여 형 변환하기

```
변수1, 변수2 = map(자료형, input().split())  
변수1, 변수2 = map(자료형, input().split('기준문자열'))  
변수1, 변수2 = map(자료형, input('문자열').split())  
변수1, 변수2 = map(자료형, input('문자열').split('기준문자열'))
```

02-6 입/출력 함수

▪ print() 함수 기본 문법

```
print(값1, 값2, 값3)  
print(변수1, 변수2, 변수3)
```

```
>>> print(1, 2, 3)  
1 2 3
```

▪ 출력값 사이에 문자 넣기

```
print(값1, 값2, sep='문자 또는 문자열')  
print(변수1, 변수2, sep='문자 또는 문자열')
```

```
>>> print(1, 2, 3, sep=', ')  
>>> print(4, 5, 6, sep=',')  
>>> print('Hello', 'Python', sep='')  
>>> print(1920, 1080, sep='x')
```

02-6 입/출력 함수

```
>>> print(1, 2, 3, sep='\n')
1
2
3
```

```
>>> print(1, end=' ')
>>> print(2, end=' ')
>>> print(3)
```

▪ format 함수 사용 출력하기

```
>>> print("I eat %d apples" %3)
>>> print("I eat {} apples" .format(3))
```

```
>>> print("I eats %d apples and %d bananas" %(3, 5))
>>> print("I eats {} apples and {} bananas" .format(3, 5))
```

```
>>> print(" π is %.1f " %(3.14))
>>> print(" π is {} " .format(3.14))
```

02-6 입/출력 함수

- # 한줄 주석
- """ ~""" 여러줄 주석

```
#This is a comment.  
print("Hello, World!")
```

```
print("Hello, World!") #This is a comment.
```

```
"""This is a  
multiline  
docstring. """  
print("Hello, World!")
```

감사합니다

**“Life is too short,
You need Python!”**

인생은 너무 짧으니,
파이썬이 필요해!