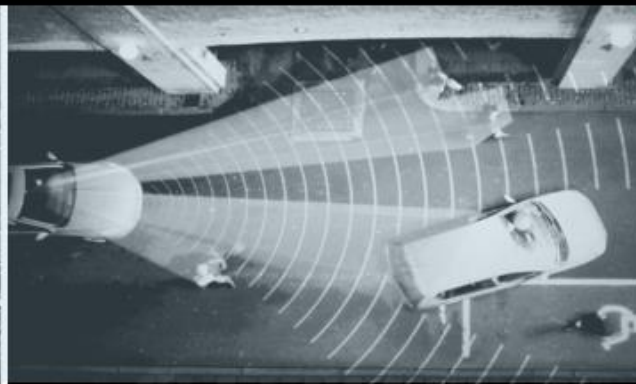


파이썬을 이용한 데이터 분석



2023. 8. 23



1. 데이터 사이언스

데이터 중심산업

앞으로(현재) 데이터 기술(data technology)의 중요성이 새롭게 부각

- 데이터 사이언스는 정보통신기술(ICT)을 기반으로 하는 최신 기술
- ICT 기술은 전자공학, 컴퓨터공학, 통신공학을 기본으로 발전
- (기존) 음악, 영화, 블로그, 홈쇼핑은 음원, 비디오, 텍스트, 상품 거래, SNS에 관한 데이터를 효과적으로 안전하게 다루는 것이 목적
- (앞으로) 데이터 사이언스에서는 데이터를 효과적으로 다루고 분석하는 기술을 포함, 데이터를 수집하고 변환하고 알고리즘을 적용하는 분석 과정 포함
- 이제 모든 산업이 데이터 기반으로 동작

1. 데이터사이언스

4차 산업혁명

“우리는 지금까지 우리가 살아오고 일하던 방식을
송두리째 바꿀 기술 혁명 직전에 와 있다.
제 4차 산업혁명은 그 속도와 파급효과 측면에서
이전의 혁명과 비교도 안 될 정도로
빠르고 광범위하게 일어날 것이다”

— 클라우드 슈밥, 세계경제포럼(WEF · 다보스 포럼) 의장 —

Copyright © 2017, Fabmonster co., Ltd. · LEE Gyeongjin All rights reserved.

— 출처, 4차 산업혁명 위기인가 기회인가 뉴시스 2017. 02. 07. —

1. 데이터사이언스

4차 산업혁명

4차 산업혁명이란 무엇인가?

파괴적 기술과
역사적 산업혁명의 전개



4차 산업혁명



1차 산업혁명

증기기관
18세기



2차 산업혁명

전기·내연기관
19-20세기



3차 산업혁명

컴퓨터·인터넷
20세기 후반

IoT

로봇

AI

3D
프린팅

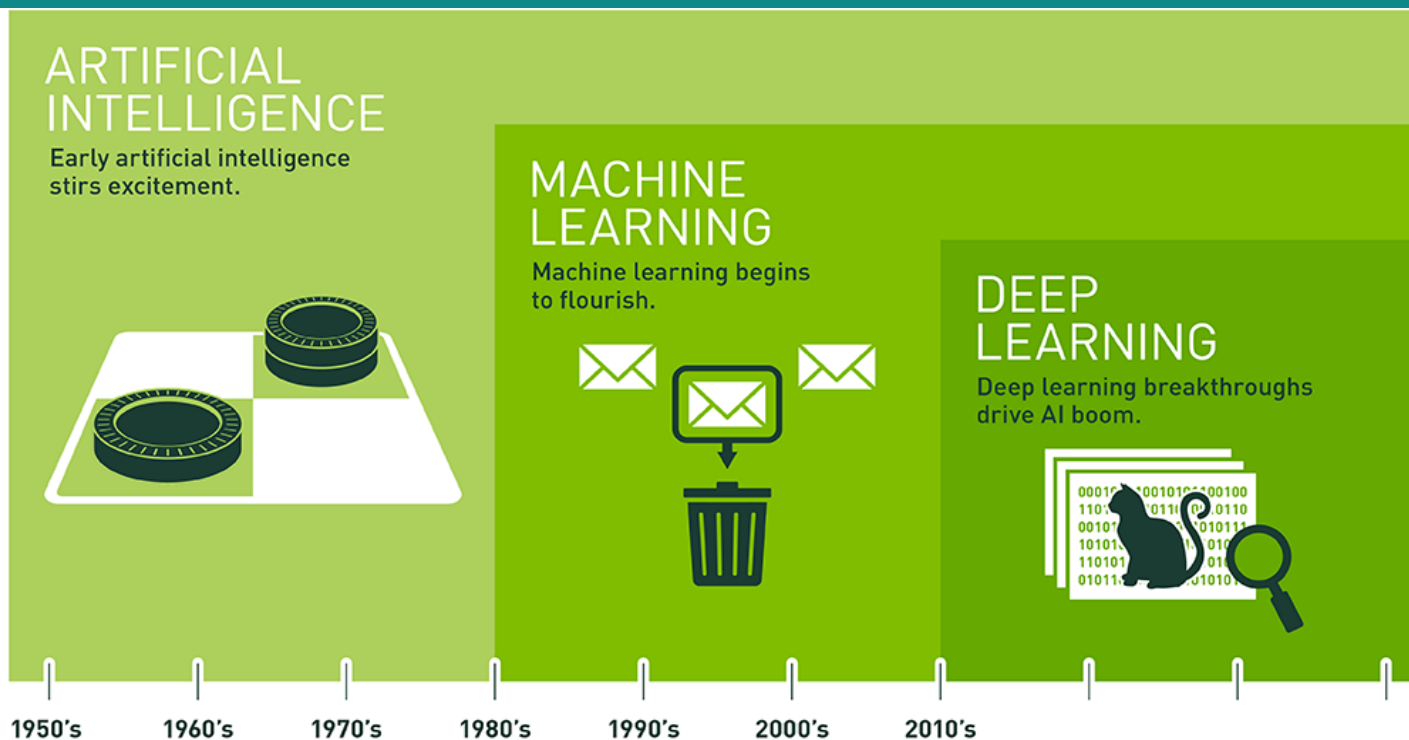
AR/VR

AI 기술을 핵심동인으로
상품·서비스의 생산·유통·소비 전 과정에서
모든 것이 연결되고 지능화

1. 데이터 사이언스

AI의 확대

인공지능 분야는 컴퓨팅, 대용량 스토리지, 인터넷과 같은 핵심기술 발전으로 크게 성장



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

1. 데이터 사이언스

AI의 확대

인공지능은 인간의 지능을 기계로 구현한 것



1. 데이터 사이언스

AI의 확대

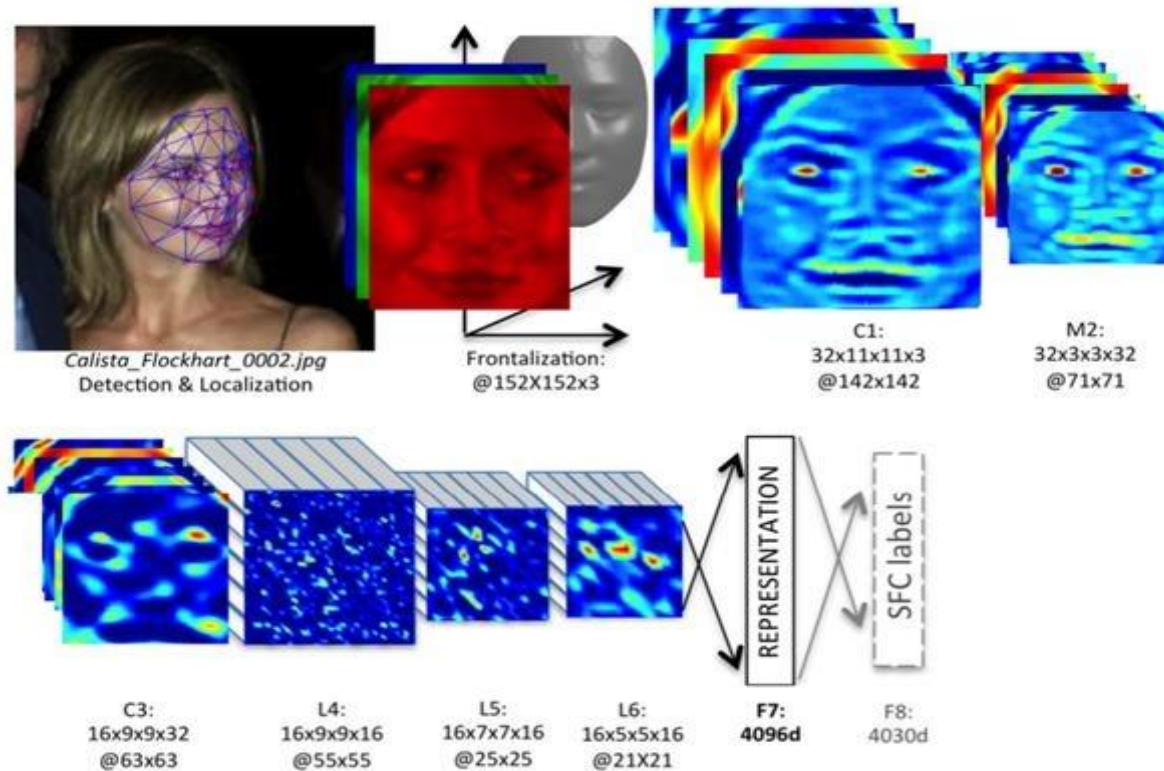
딥 러닝은 완전한 머신러닝을 실현하는 기술



1. 데이터 사이언스

AI의 확대

딥러닝을 이용한 이미지 인식 프로그램은 이미 실생활에 접목되어 사용

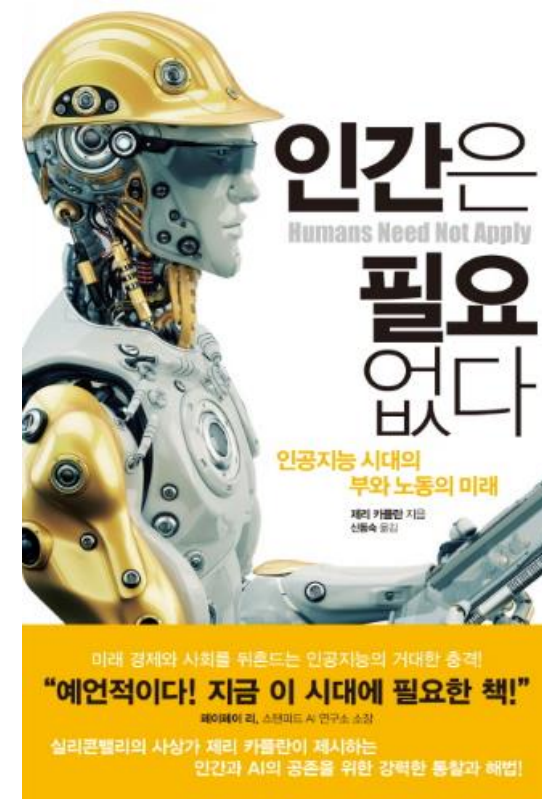


1. 데이터사이언스

AI의 미래

다가올 미래에 대비해야 내 일자리가 생긴다.

- 우리는 AI의 미래를 잘 예측하고 준비해서 불필요한 경쟁을 피해야
- 사람 VS 기계, 엔진
- AI로 인해 직업 변화가 크게 나타날 것
- 자율주행차는 아직 불안하다? 배달용 무인 자동차 운영 및 도입



2. 데이터 사이언스 도구

파이썬(Python)



2. 데이터 사이언스 도구

파이썬(Python)

- 일반 programming 언어이면서 자바, C에 비해 문법이 간단, 배우기 쉽고 강력한 기능
- 초보자 교육에서부터 웹 개발, 과학 계산, 데이터 분석, 인공지능 등 다양한 분야에 활용
- 데이터 사이언스에서 사용할 수 있는 도구는 자바, C와 같은 일반 프로그래밍 언어와 R과 같은 통계 분석에 특화된 언어, 그리고 Matlab, SPSS, SAS와 같은 데이터 분석 도구
- R과 유사하게 데이터를 다루는데 편리한 기능을 제공한다. 최근에는 파이썬(python) 언어가 데이터 분석에 가장 많이 사용
- 딥러닝을 이용한 프로그램 작성에서 파이썬을 기본 언어로 채택함으로써 이제 머신러닝 영역뿐만 아니라 일반 데이터 분석에서 폭넓게 파이썬을 도입

2. 데이터사이언스 도구

파이썬(Python)이란?

- 오픈 소스
- 크로스 플랫폼(Windows, Mac OS X, Linux 등)
- Interpreter형 컴파일 방식을 채택하여 동적 변환이 쉽고, 여러 분야의 어플리케이션 개발과 스크립팅에 이상적인 언어
- 대화형 언어
- 파이썬은 기능 위주의 프로그래밍, 절차지향적 프로그래밍, 객체지향적 프로그래밍 등을 채택하여 다중 프로그래밍 패턴을 지원
- 방대한 사용자와 활발한 커뮤니티 : 파이썬은 풍부한 라이브러리를 보유, 파이썬 웹사이트에서 무료로 소스를 얻을 수 있고 자유로운 배포가 가능한 장점이 있음
- 구글, 유튜브, 인스타그램, 드롭박스, NASA 등에서 사용

2. 데이터 사이언스 도구

○ 파이썬(Python)으로 할 수 있는 것

- 웹 개발
- 게임 개발
- 업무 자동화
- 데이터 분석
- 기계 학습
- 과학 계산
- 텍스트 처리
- 이미지 처리
- 멀티 미디어
- 데이터 베이스
- 기타

2. 데이터 사이언스 도구

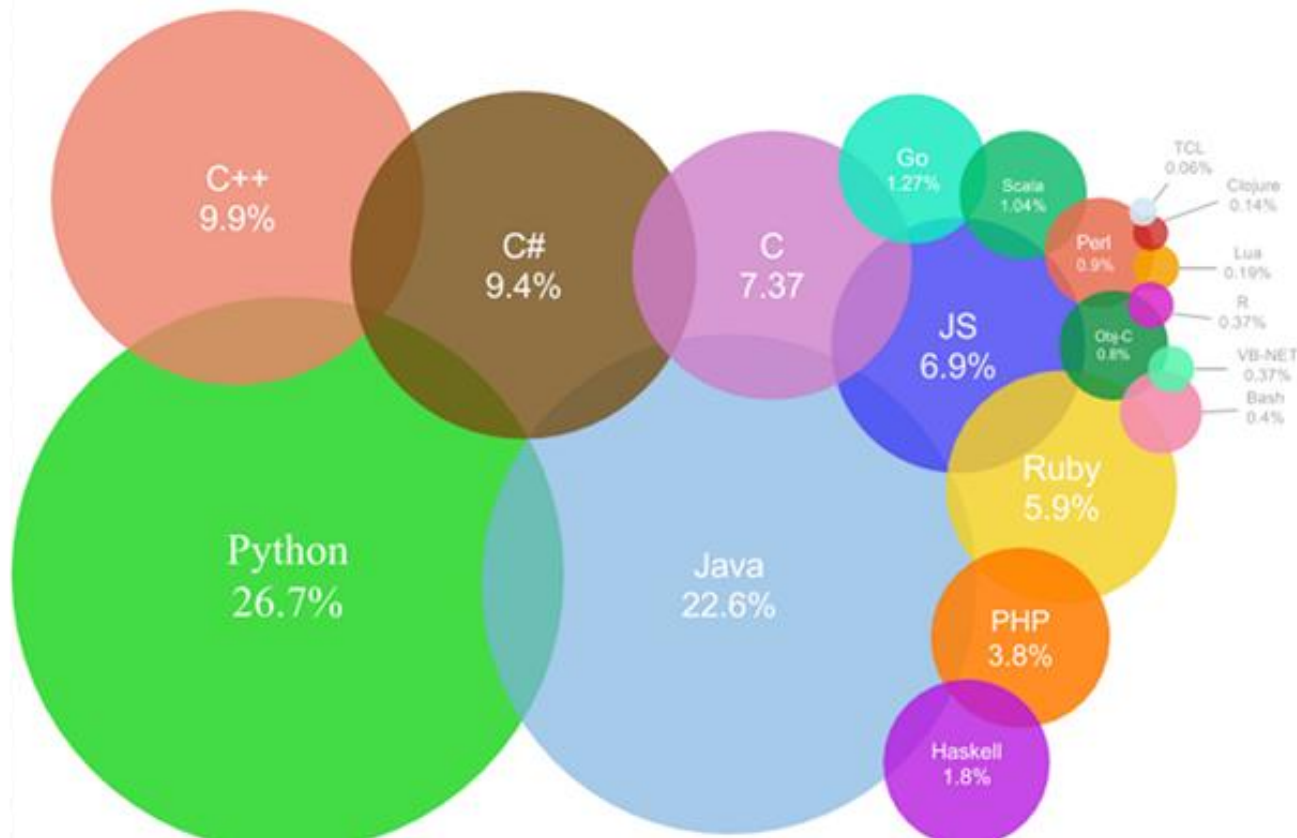
파이썬(Python)의 단점

- 느린 속도
- 모바일 앱 개발에서 사용하기 어렵다.

2. 데이터 사이언스 도구

파이썬(Python)이란?

Most Popular Coding Languages of 2016

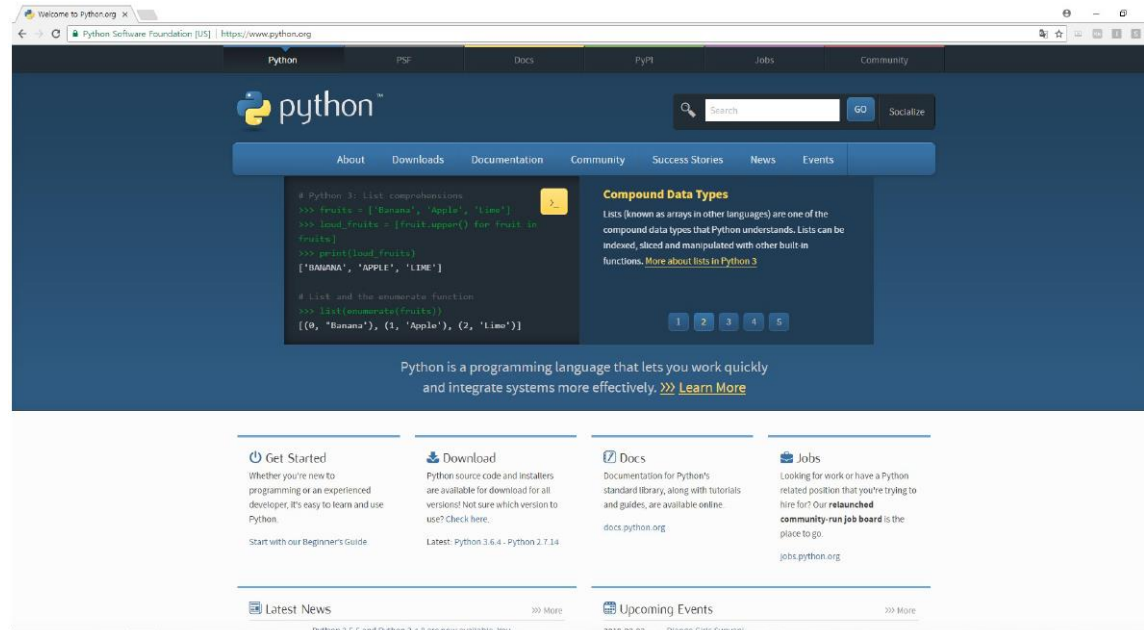


2. 데이터 사이언스 도구: 파이썬

환경 설정

python 은 공식 홈페이지에서 다운 받을 수 있다.

- <http://www.python.org>



2. 데이터 사이언스 도구: 파이썬

○ 환경 설정

- python 에서는 pip 명령으로 사용하고 싶은 패키지들을 직접 설치해 줘야
- 필요한 모듈을 개별적으로 직접 설치하는 것은 상당히 불편
 - 번거롭다
 - 버전 의존성 문제 발생 가능성
- (불편 해소) 주로 아나콘다(Anaconda)로 설치
 - 데이터 분석에 자주 쓰이는 모듈과 함께 python 을 패키지로 제공

2. 데이터 사이언스 도구 : anaconda

○ 환경 설정



Virtual environment



2. 데이터 사이언스 도구: 파이썬

환경 설정

- 주피터 노트북(Jupyter notebook) : 파이썬 통합 개발 환경으로 가장 널리 사용
 - Program code 작성
 - code block 단위의 동작 확인
 - 실행결과 출력 화면 보관
 - 문서 작성

등을 편리하게 할 수 있다.

- 주피터 노트북 : 웹 기반으로 동작 (자신의 PC가 local server 역할)
 - 즉, 임의의 컴퓨터에서 웹 브라우저를 통해서 서버의 개발 환경을 이용할 수 있다.
- 구글이 제공하는 colab을 주피터 노트북으로 무료로 사용할 수도 있다.

2. 데이터 사이언스 도구 : anaconda

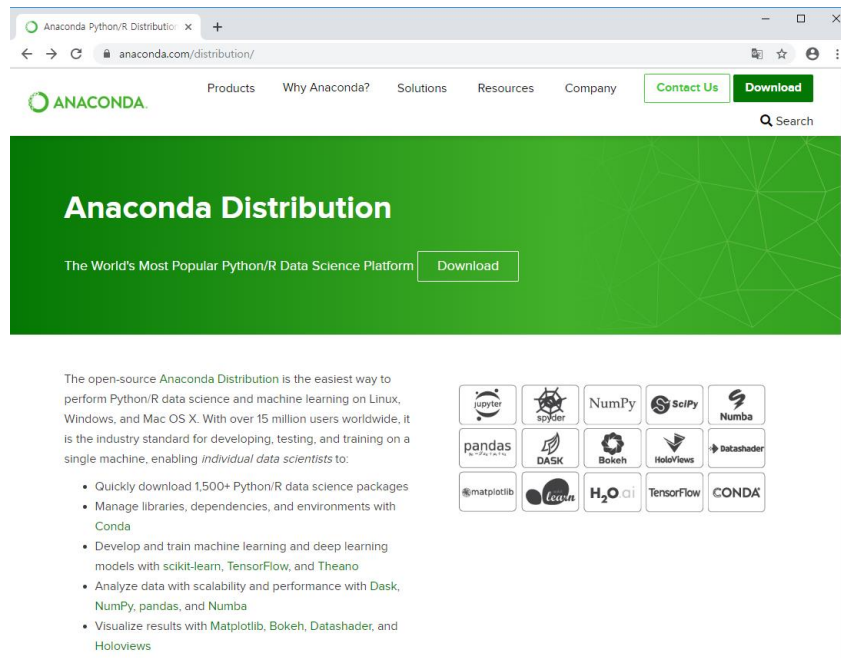
환경 설정

- Anaconda
 - Continuum Analytics 에서 만든 python 배포판
 - 무료로 설치할 수 있다
 - 패키지 관리나 환경설정을 손쉽게 할 수 있다
 - 여러 분야에 사용할 수 있는 다양한 패키지를 가지고 있다.
 - 기본적으로 445개 정도의 python 관련 패키지를 포함
- Anaconda 설치
 - anaconda 웹사이트에 접속하여 down
 - <https://www.anaconda.com/downloads>

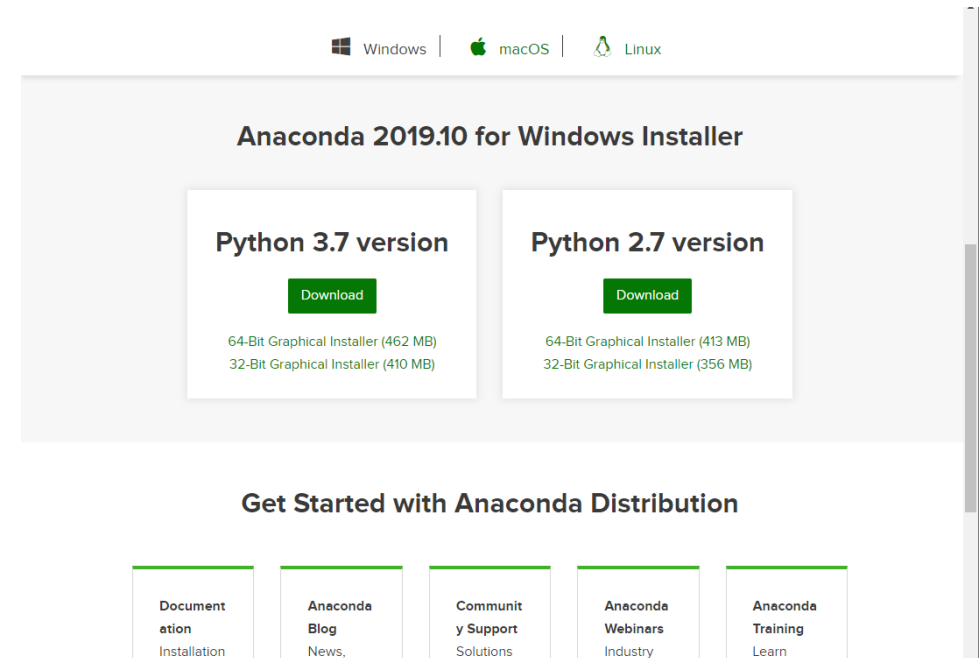
2. 데이터 사이언스 도구 : anaconda

환경 설정

- 윈도우, 맥, 리눅스 버전을 각각 제공
- 현재 파이썬 버전 3.7을 제공(2.7 버전은 이전 프로그램과의 호환 시 필요)
- 윈도우 사용자의 경우 사용하는 PC가 32/64 비트 버전 확인 필요



The screenshot shows the Anaconda Distribution website. The header includes the Anaconda logo and navigation links: Products, Why Anaconda?, Solutions, Resources, Company, Contact Us, and Download. A search bar is also present. The main content area features a green banner with the text "Anaconda Distribution" and "The World's Most Popular Python/R Data Science Platform". Below this, there is a list of supported data science packages: NumPy, SciPy, Numba, pandas, Dask, Bokeh, HoloViews, Datashader, matplotlib, and others. A list of features is provided, including quick download of 1,500+ Python/R data science packages, management of libraries, dependencies, and environments with Conda, development and training of machine learning and deep learning models with scikit-learn, TensorFlow, and Theano, analysis of data with scalability and performance with Dask, NumPy, pandas, and Numba, and visualization of results with Matplotlib, Bokeh, Datashader, and HoloViews.



The screenshot shows the Anaconda 2019.10 for Windows Installer page. The header includes links for Windows, macOS, and Linux. The main content area features a section titled "Anaconda 2019.10 for Windows Installer". Below this, there are two boxes for "Python 3.7 version" and "Python 2.7 version". Each box contains a "Download" button and links to the 64-Bit Graphical Installer (462 MB) and 32-Bit Graphical Installer (410 MB) for Python 3.7, and the 64-Bit Graphical Installer (413 MB) and 32-Bit Graphical Installer (356 MB) for Python 2.7. At the bottom, there is a section titled "Get Started with Anaconda Distribution" with links to Documentation, Anaconda Blog, Community Support, Anaconda Webinars, and Anaconda Training.

2. 데이터 사이언스 도구: 파이썬

내 PC 환경 확인

컴퓨터에 대한 기본 정보 보기

Windows 버전

Windows 10 Pro

© 2018 Microsoft Corporation. All rights reserved.



시스템

프로세서: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz

설치된 메모리(RAM): 8.00GB(7.88GB 사용 가능)

시스템 종류: 64비트 운영 체제, x64 기반 프로세서

펜 및 터치: 이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.

컴퓨터 이름, 도메인 및 작업 그룹 설정

컴퓨터 이름: DESKTOP-U7MNMU1

전체 컴퓨터 이름: DESKTOP-U7MNMU1

컴퓨터 설명:

작업 그룹: WORKGROUP

 설정 변경

Windows 정품 인증

Windows 정품 인증을 받았습니다. [Microsoft 소프트웨어 사용 조건 읽기](#)

2. 데이터 사이언스 도구 : 파이썬

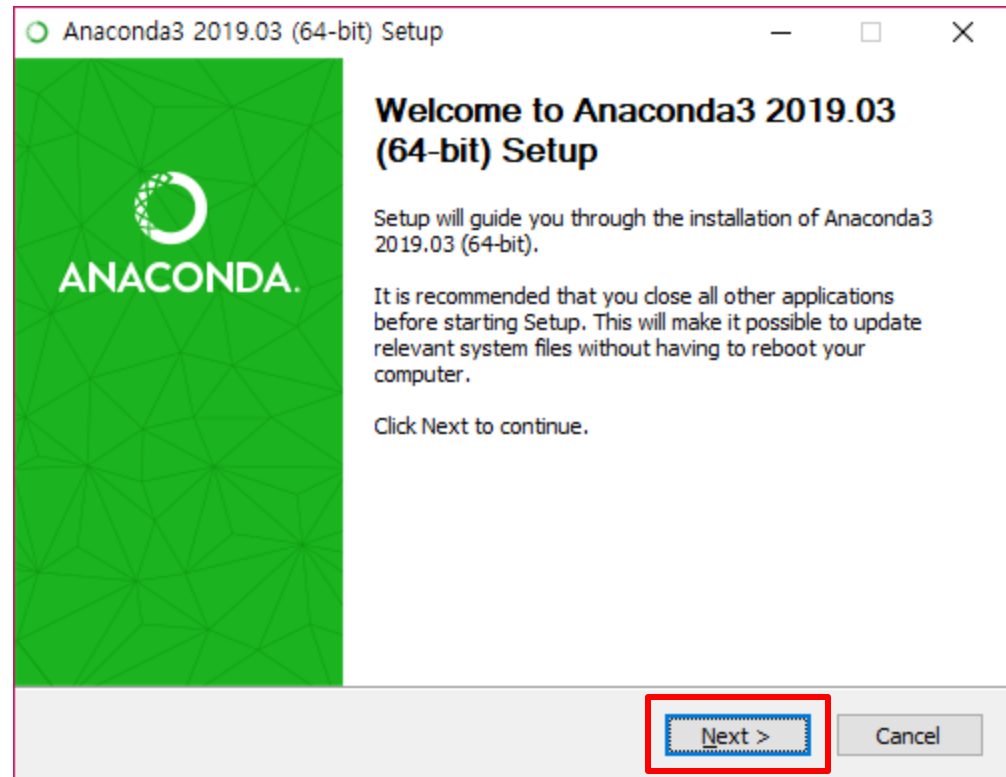
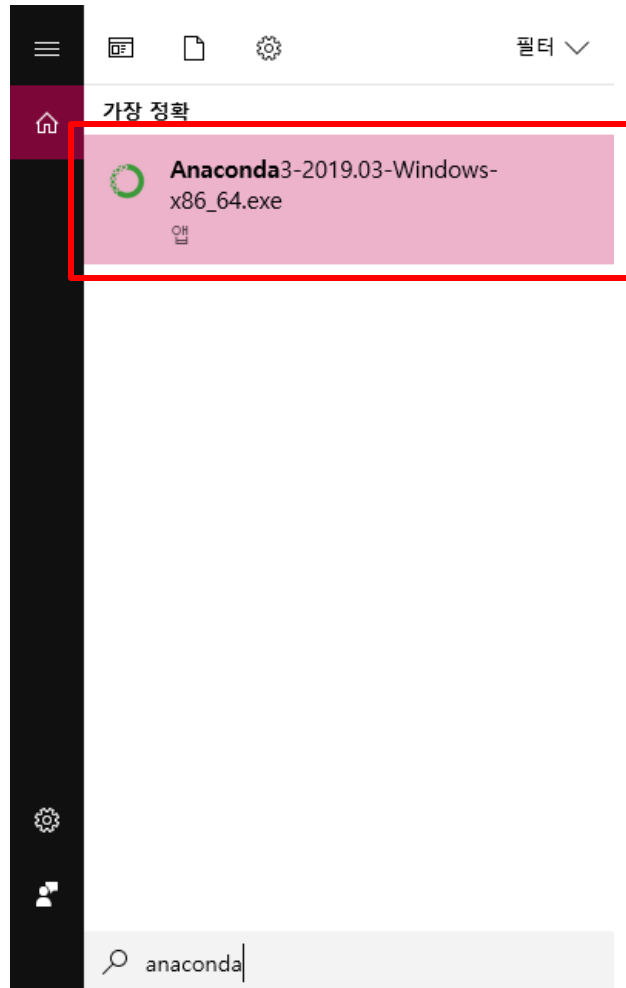
○ 다운로드



- Anaconda Windows installer (<https://www.anaconda.com/distribution/>)
- jupyter notebook 실행을 위한 구글 크롬(브라우저) 설치

2. 데이터 사이언스 도구: 파이썬

설치(setup)



2. 데이터 사이언스 도구: 파이썬

설치(setup)

Anaconda3 2019.03 (64-bit) Setup

License Agreement
Please review the license terms before installing Anaconda3 2019.03 (64-bit).

Press Page Down to see the rest of the agreement.

=====

Anaconda End User License Agreement

=====

Copyright 2015, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2019.03 (64-bit).

Anaconda, Inc.

< Back I Agree Cancel

Anaconda3 2019.03 (64-bit) Setup

Select Installation Type
Please select the type of installation you would like to perform for Anaconda3 2019.03 (64-bit).

Install for:

☒ Just Me (recommended)

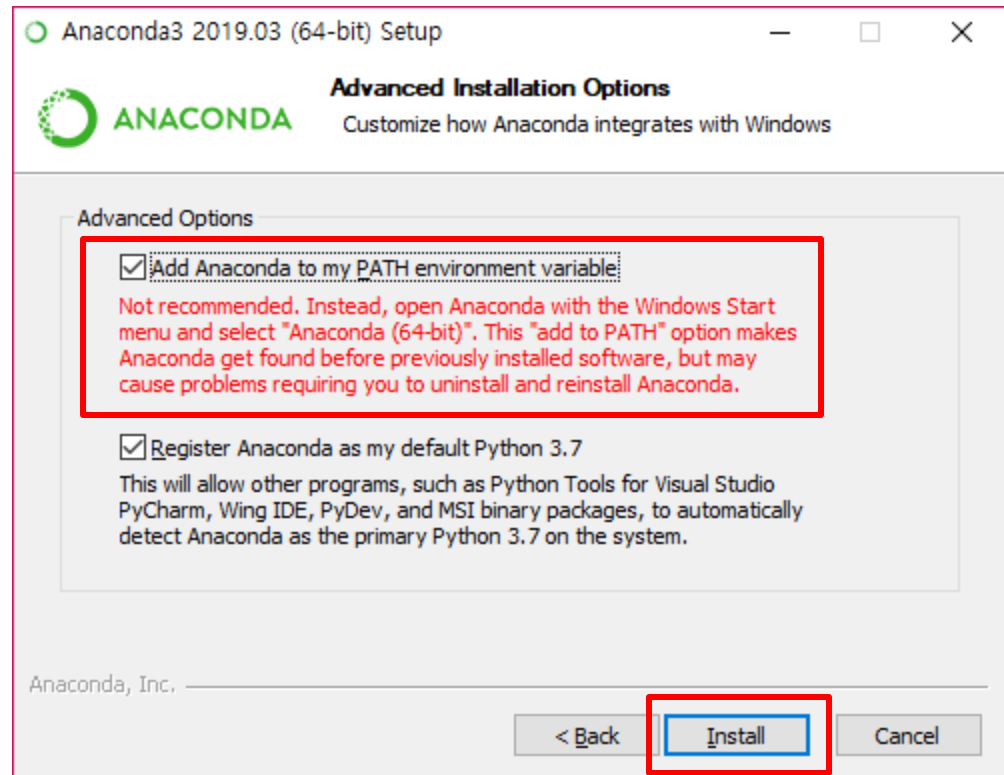
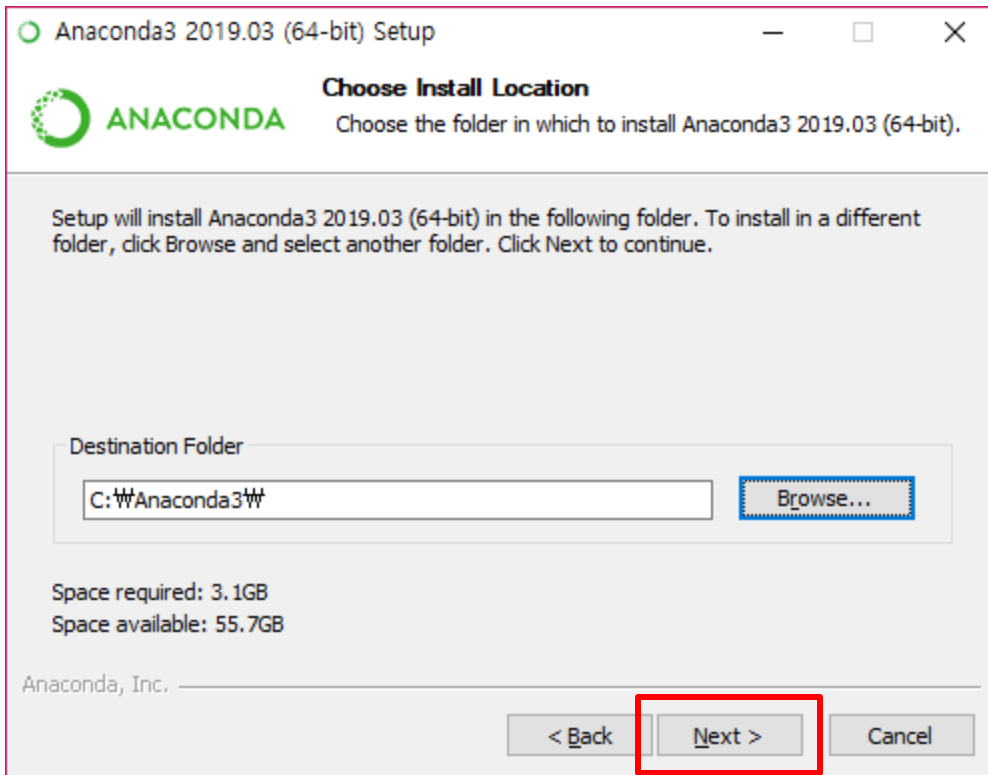
☐ All Users (requires admin privileges)

Anaconda, Inc.

< Back Next > Cancel

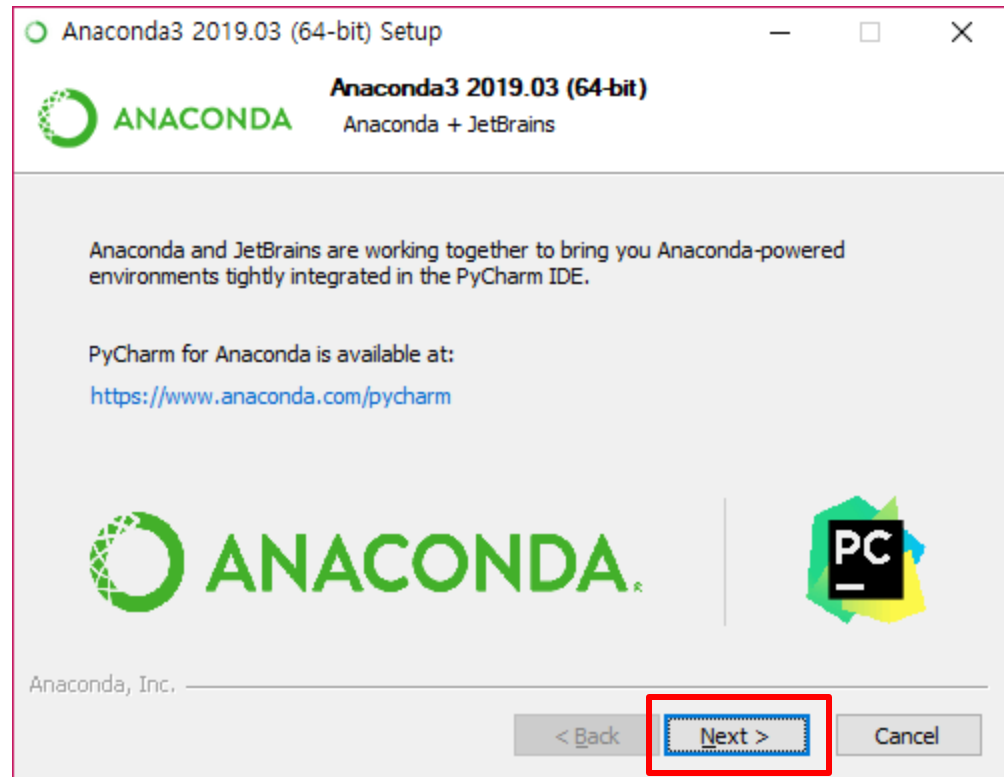
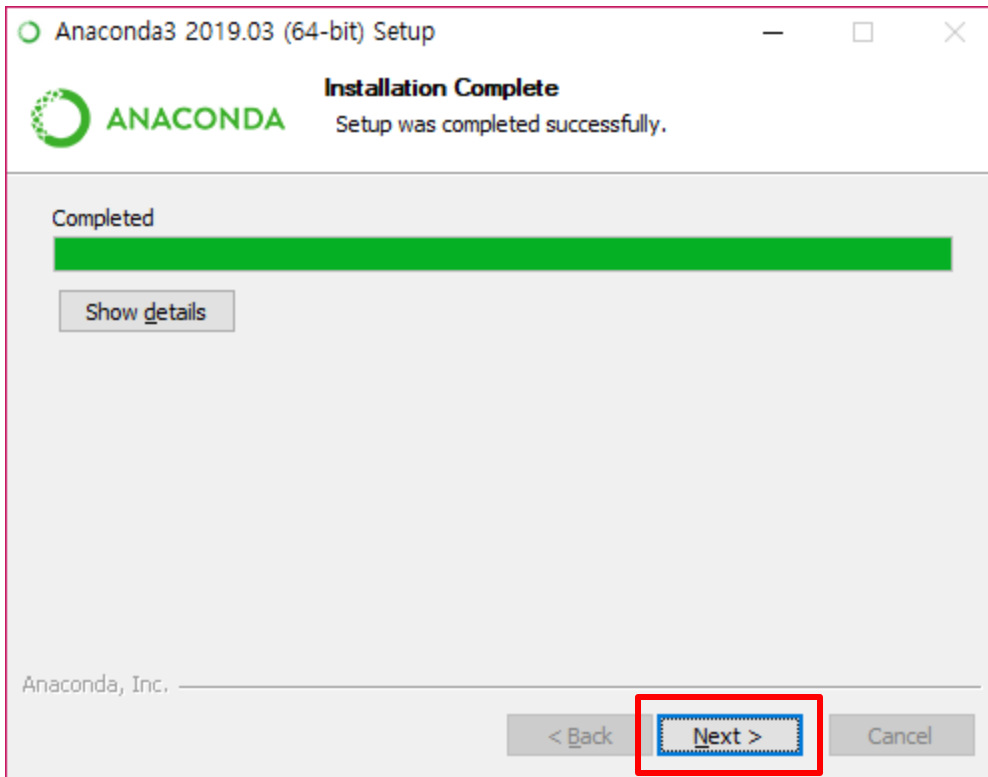
2. 데이터 사이언스 도구: 파이썬

설치(setup)



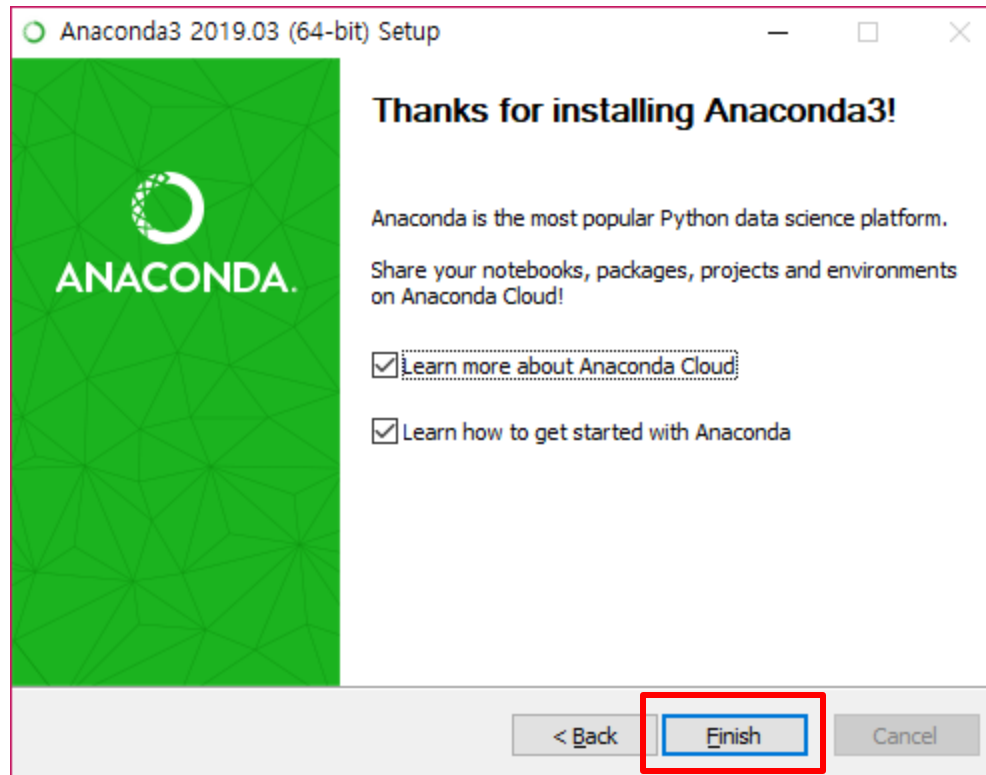
2. 데이터 사이언스 도구: 파이썬

설치(setup)



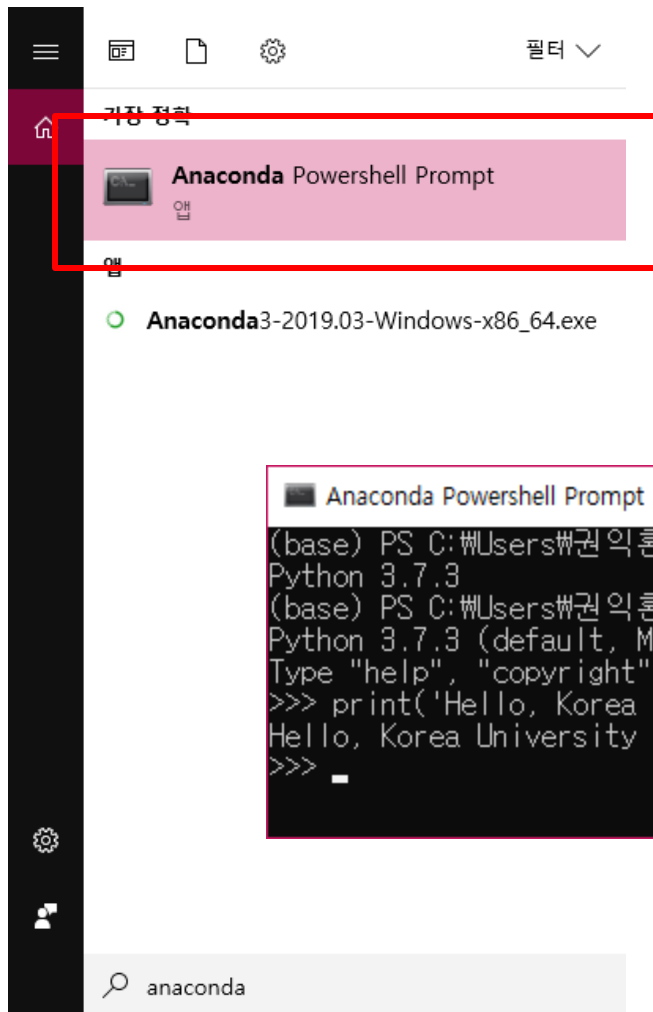
2. 데이터 사이언스 도구 : 파이썬

설치(setup)



2. 데이터 사이언스 도구: 파이썬

test



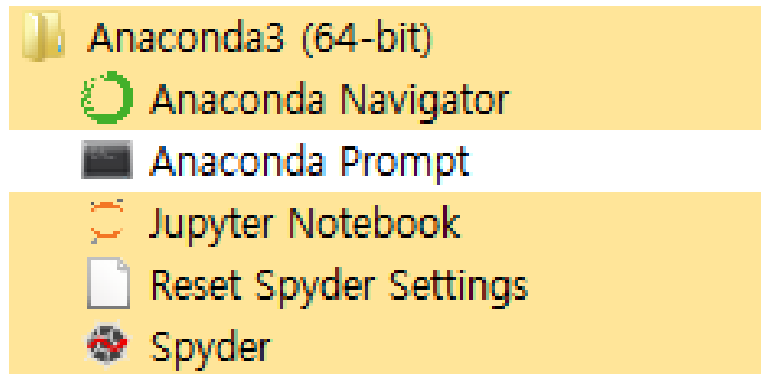
A screenshot of the Anaconda Powershell Prompt terminal window. The window title is 'Anaconda Powershell Prompt'. The terminal shows the following commands and output:

```
(base) PS C:\Users\권익환> python --version
Python 3.7.3
(base) PS C:\Users\권익환> python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello, Korea University')
Hello, Korea University
>>> _
```

2. 데이터 사이언스 도구: 파이썬

○ 아나콘다 환경에서 파이썬 구동

- 설치된 아나콘다 폴더에서 **anaconda prompt**를 실행 할 수 있음



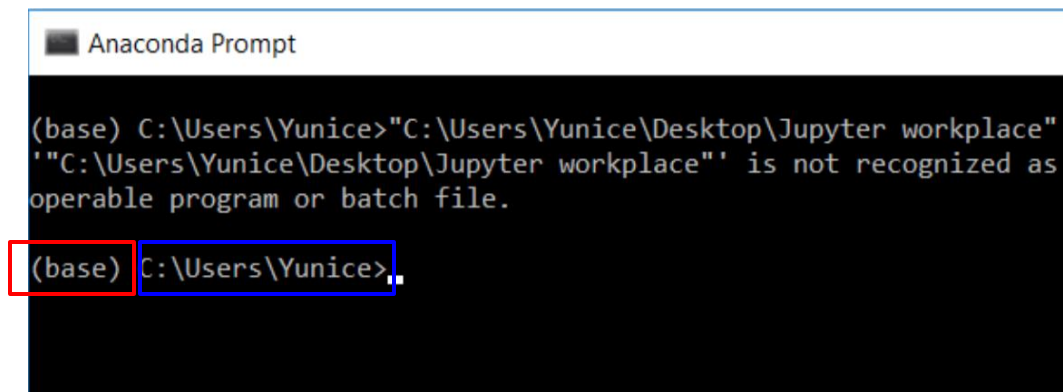
- 프롬프트에서 python을 입력하면 실행 가능
- print('Hello, Korea University') 를 입력하면 명령어가 실행 됨

종료 시, exit() 또는 ctrl + z 입력

2. 데이터 사이언스 도구: 파이썬

Anaconda 환경에서 파이썬 구동

- **anaconda prompt (windows 경우)**
 - **(base)** : python 가상 환경이 기본(base)로 설정되어 있다는 의미
- **가상 환경**
 - 개발 환경을 독립적으로 구성하는 것이 필요할 때 사용
 - 예를 들어 수행하는 프로젝트마다 추가로 필요한 패키지의 종류나 버전이 다른 경우 이들을 각각 다른 가상환경에서 작업하면 마치 서로 다른 컴퓨터를 사용하듯이 환경 조건을 다르게 만들 수 있다.



```
Anaconda Prompt

(base) C:\Users\Yunice>"C:\Users\Yunice\Desktop\Jupyter workplace"
'"C:\Users\Yunice\Desktop\Jupyter workplace"' is not recognized as
operable program or batch file.

(base) C:\Users\Yunice>.
```

2. 데이터 사이언스 도구: 파이썬

○ 아나콘다 환경에서 파이썬 구동

- Home 디렉토리
 - (base) 다음에 나타나는 폴더 경로 위치
 - Jupyter notebook이 처음 시작되는 위치 (기억 필요)
 - Jupyter notebook은 홈 디렉토리보다 상위의 폴더에 있는 파일은 검색이 안 된다.
 - 이 폴더의 하위에 작업 폴더를 만들고 필요한 파일을 여기에 저장

2. 데이터 사이언스 도구: 파이썬

아나콘다 환경에서 파이썬 구동

- **anaconda prompt**를 통해 패키지 관리 및 작업 환경 설정을 실행

(base) python -V

설치된 python의 버전 확인

- Python에서는 anaconda의 기본 제공 library 외의 새로운 library 를 추가 설치
 - **anaconda prompt**에서 pip (python install program)을 이용
- Python 버전 3에서는 pip대신 pip3를 사용하기도 한다. (pip의 사용 예)

pip --version

pip이 설치되었는지 확인

pip install pkg

특정 패키지 "pkg"를 설치

pip install -- upgrade pip

패키지 pip 자체를 upgrade

- Mac 이나 Linux에서는
 - **anaconda prompt**가 아니라 일반 명령창에서 pip 명령을 사용

2. 데이터 사이언스 도구: 파이썬

○ 아나콘다 환경에서 파이썬 구동

- **anaconda prompt**를 통해 새로운 폴더를 생성하여 코드를 관리

mkdir 명령어를 통해 수행

- 간단한 DOS 명령어 수행

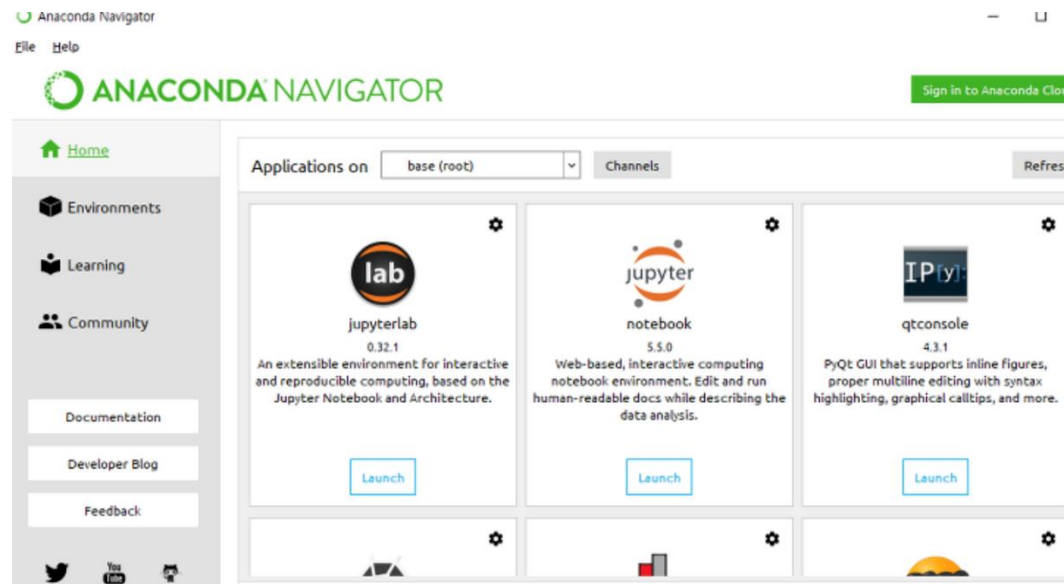
cd, chdir
md, mkdir
rd, rmdir
Dir

- **anaconda navigator**를 사용할 수도 있다. (속도가 느리다)

2. 데이터 사이언스 도구 : Jupyter notebook

Jupyter notebook 실행

- anaconda navigator 사용



2. 데이터 사이언스 도구 : Jupyter notebook

○ Jupyter notebook 실행

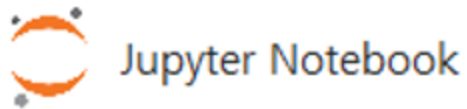
- **anaconda prompt 사용**

(base) jupyter notebook

- Mac이나 Linux에서는 일반 명령창에서 수행

\$ jupyter notebook

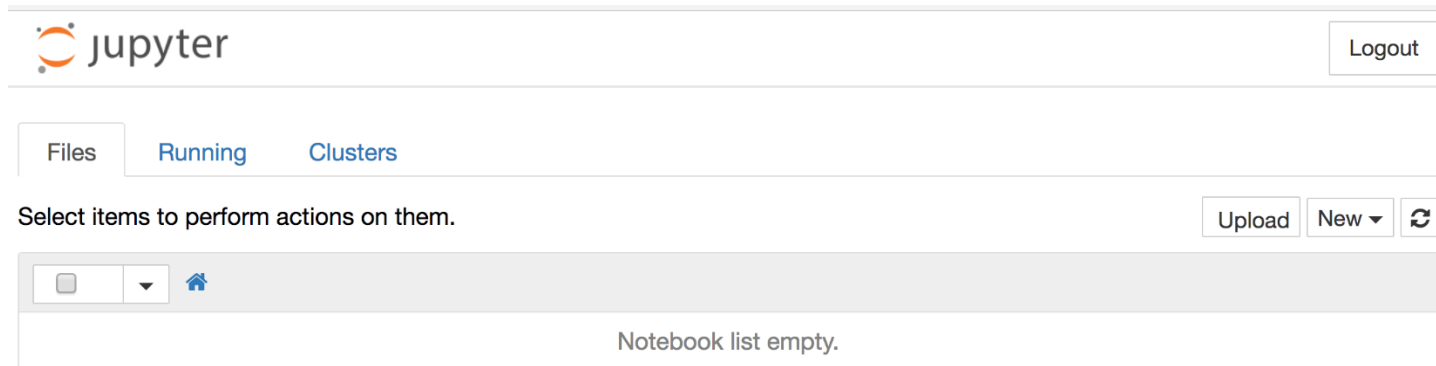
- Program 메뉴 : Jupyter notebook 바로 실행



2. 데이터 사이언스 도구 : Jupyter notebook

○ 현재 폴더에 있는 모든 파일 보기

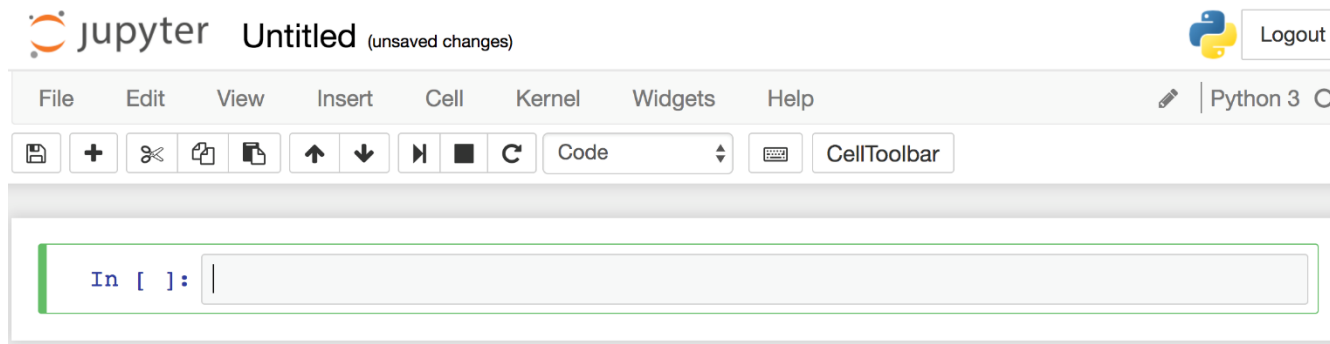
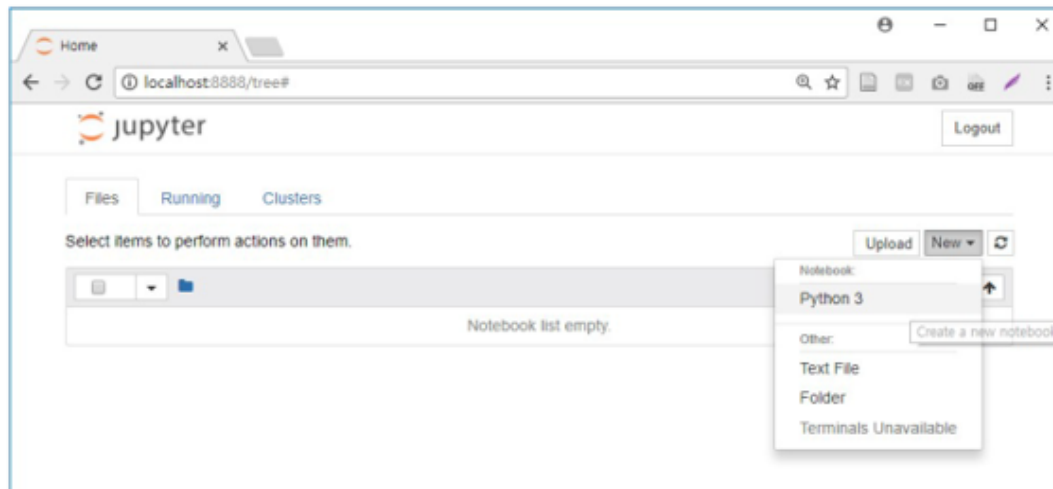
- Files 메뉴 : 현재 폴더에 있는 모든 파일을 보여준다.
- 쥬피터에서 다루는 프로그램 파일 : 노트북(notebook)이라 호칭
- 확장자 : .ipynb



2. 데이터 사이언스 도구: Jupyter notebook

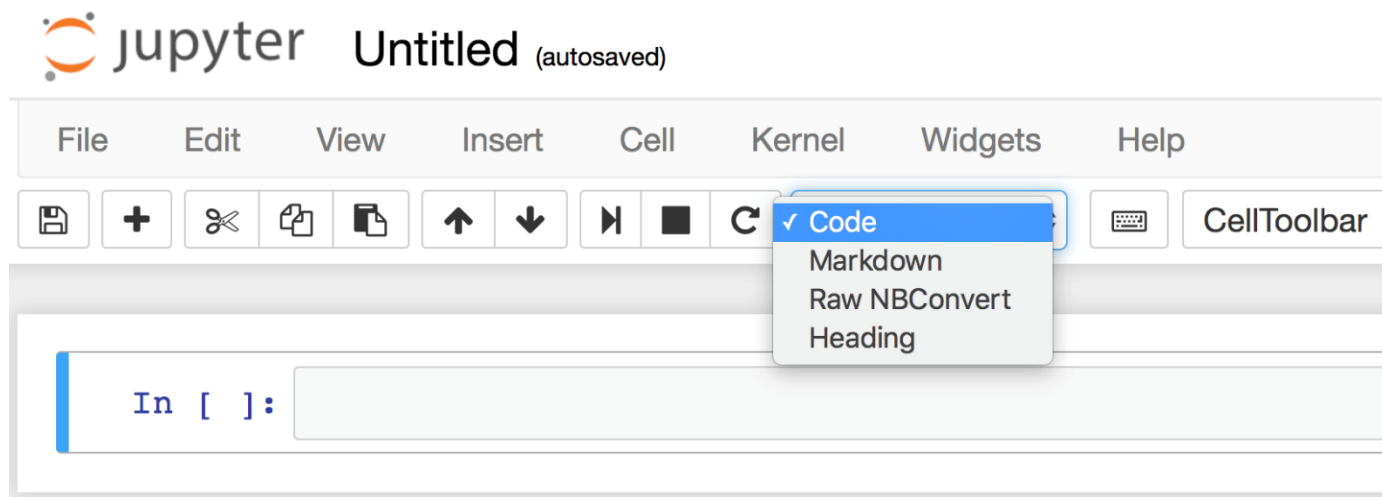
새 파일 만들기

- New 의 Python 3 를 클릭하면 새로운 파일을 만들 수 있다.



2. 데이터 사이언스 도구 : Jupyter notebook

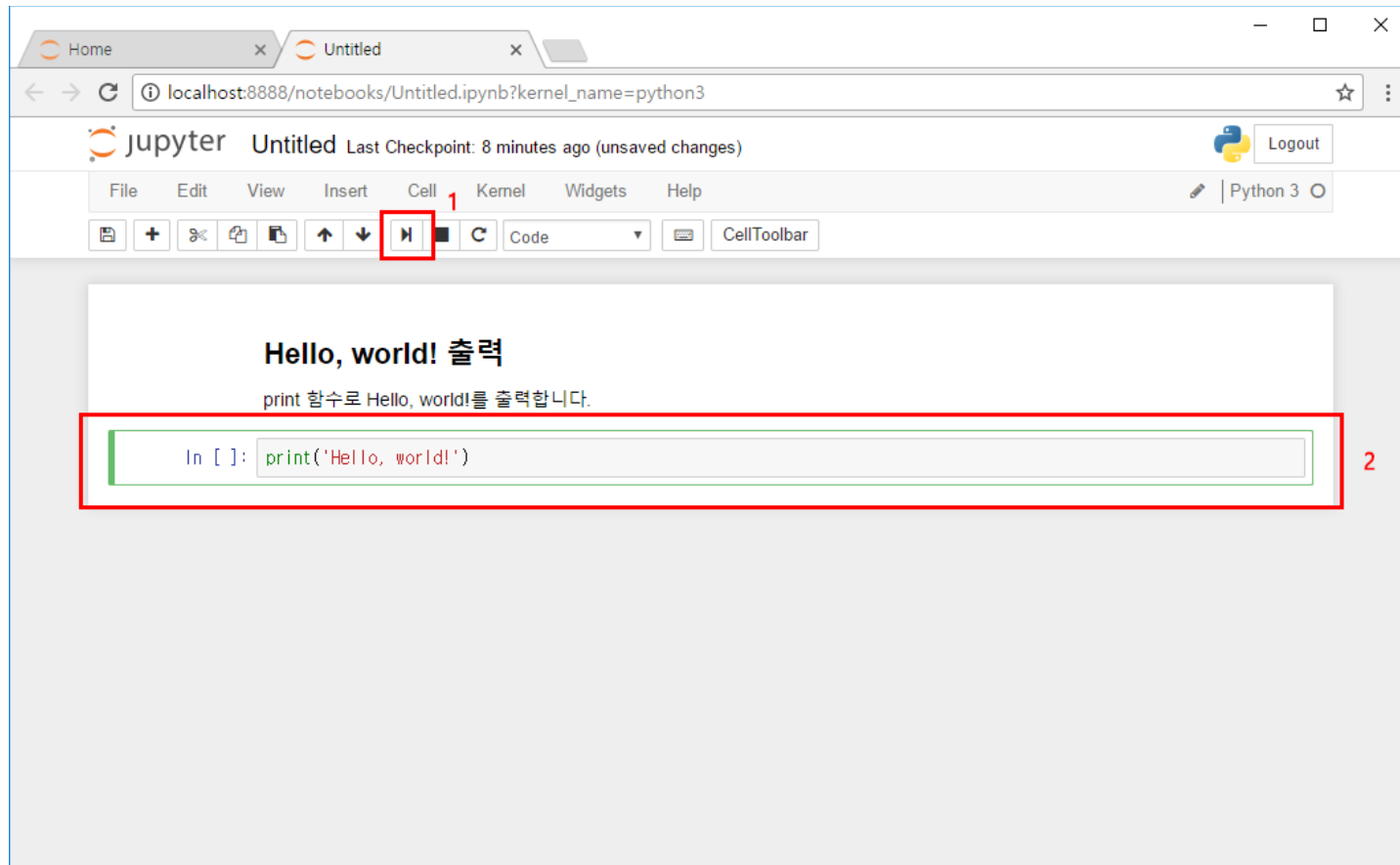
○ 새 파일 만들기



- 새로 만들어진 노트북의 이름 : "Untitled"
 - 이 부분을 클릭하여 이름을 변경할 수 있다.
- In [] 부분 : 셀(Cell)
 - Code Cell : 파이썬 코드를 입력
 - Markdown Cell : 문서 작성

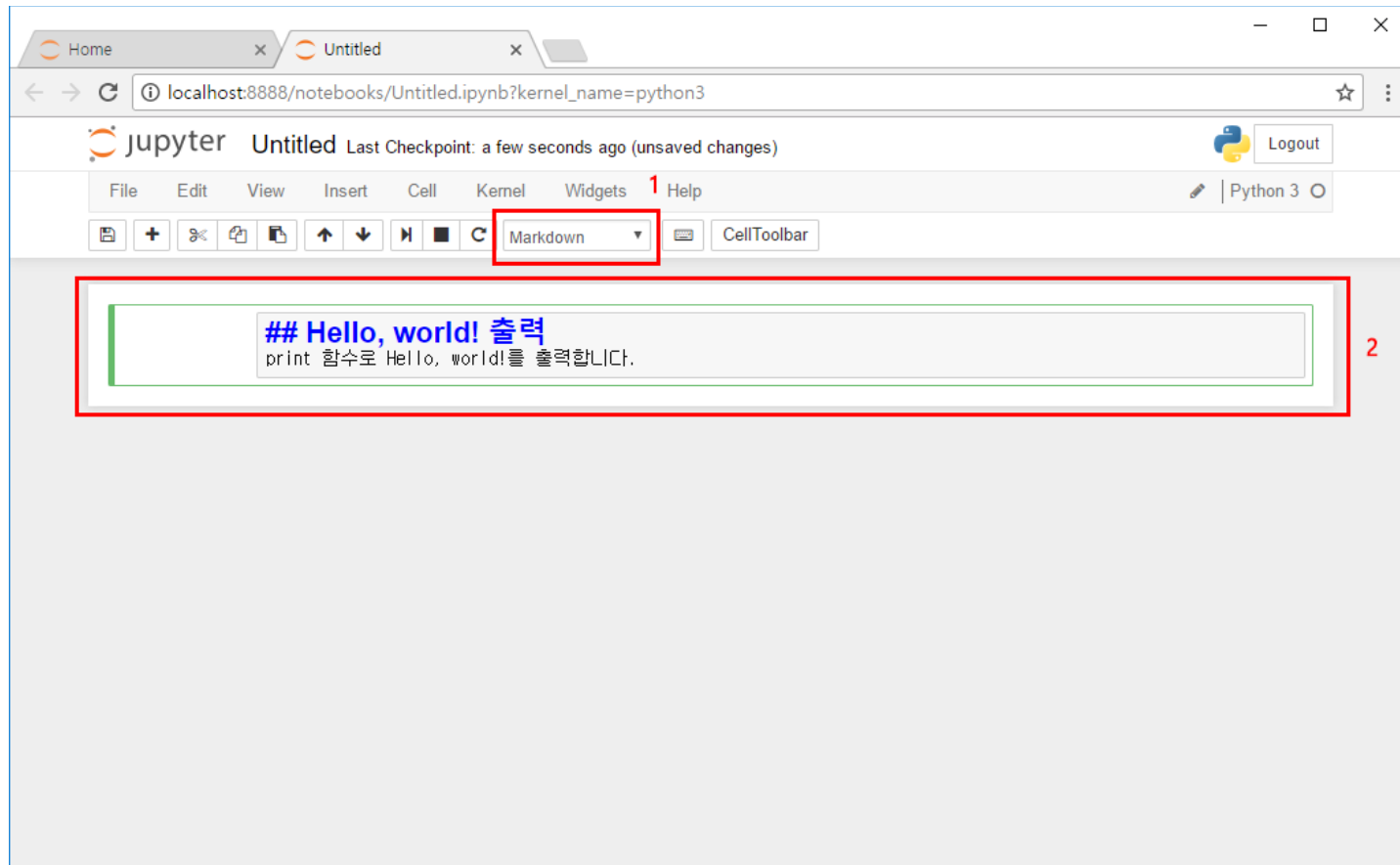
2. 데이터 사이언스 도구: Jupyter notebook

파이썬 code 작성, 실행 Code



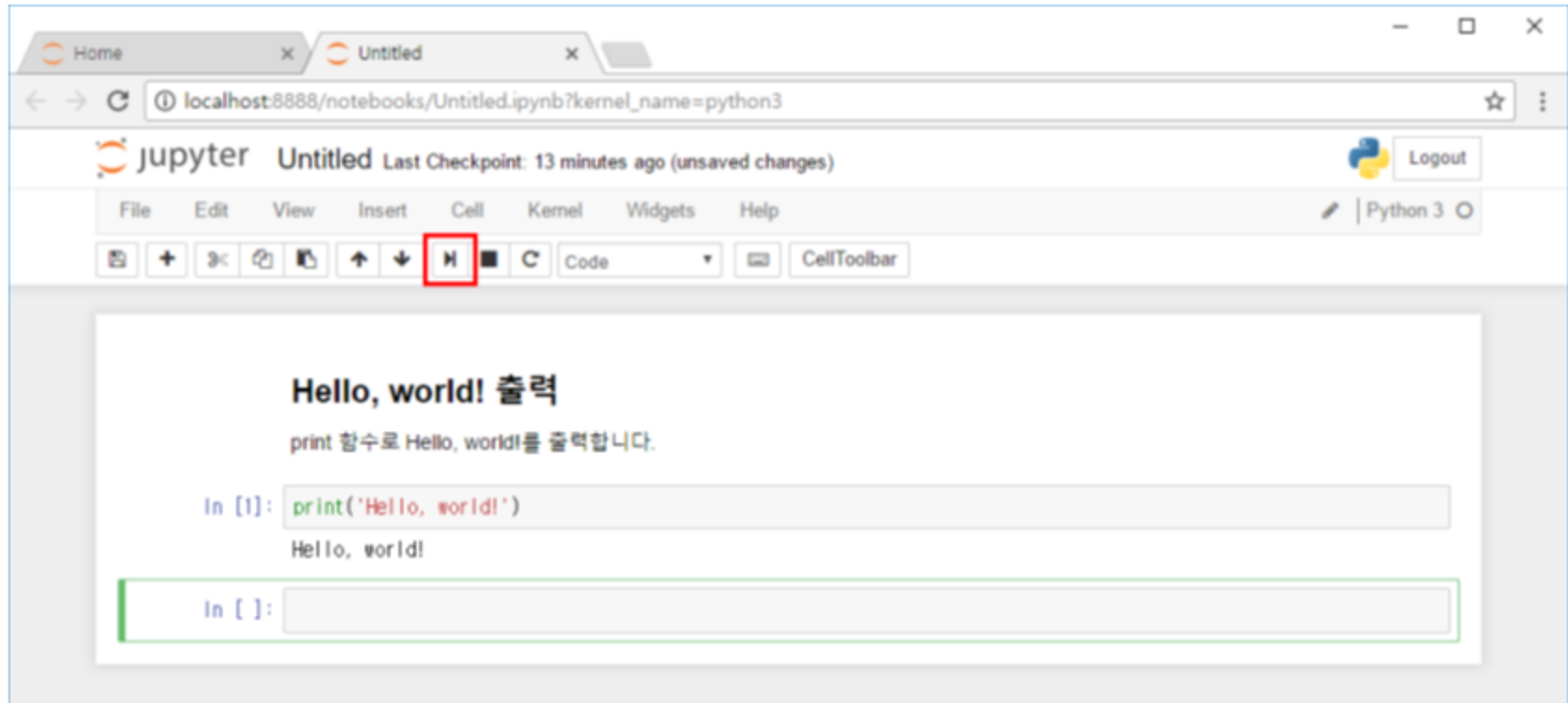
2. 데이터 사이언스 도구 : Jupyter notebook

설명(문서) 작성 Markdown



2. 데이터 사이언스 도구: Jupyter notebook

Code/Markdown 실행



- 코드 삽입 후 ▶| 버튼 클릭하면 코드가 실행되고 결과가 출력 됨
- 이때 In [1]로 괄호 안이 변경되는데 이는 첫 번째로 실행된 코드라는 의미

2. 데이터 사이언스 도구 : Jupyter notebook

Code/Markdown 실행

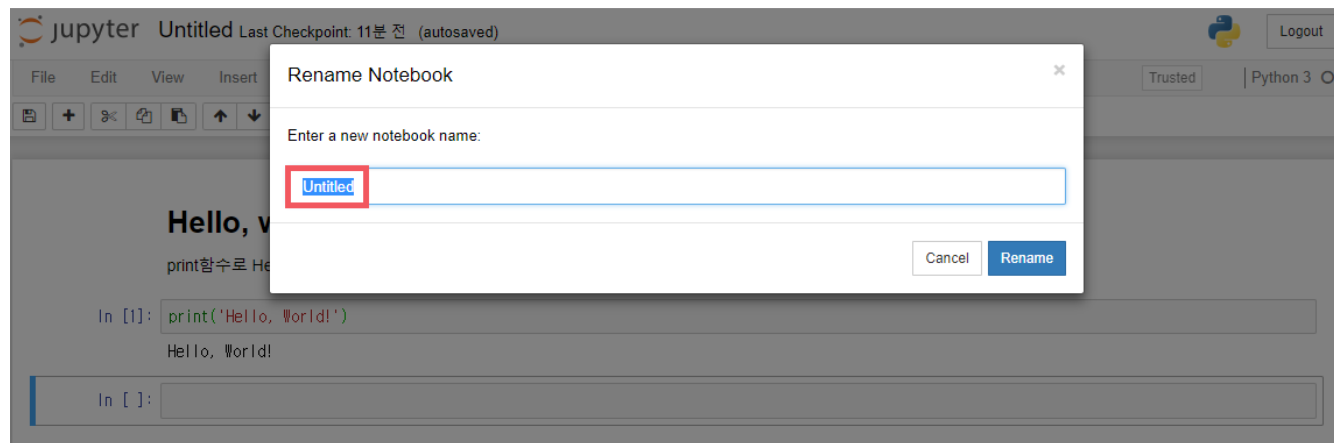
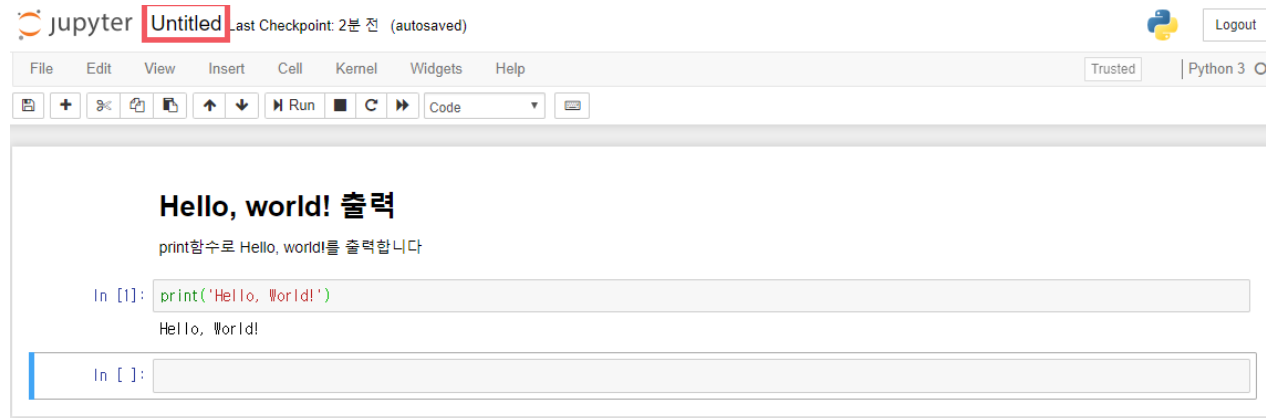
- Ctrl + Enter
 - 현재 cell의 내용을 실행

- ❖ In place: Ctrl + Enter
- ❖ To execute cell and move to next cell: Shift + Enter
 - Create new cell if necessary
- ❖ To execute and insert new cell: Alt + Enter

- Shift + Enter
 - 현재 cell의 내용을 실행 & 다음 cell로 이동
 - 다음 cell이 없으면 새로 만들어서 이동
- Alt + Enter
 - 현재 cell의 내용을 실행 & 아래에 새로 cell을 만들고 이동
- 전체 코드 실행
 - Cell의 Run All 클릭

2. 데이터 사이언스 도구 : Jupyter notebook

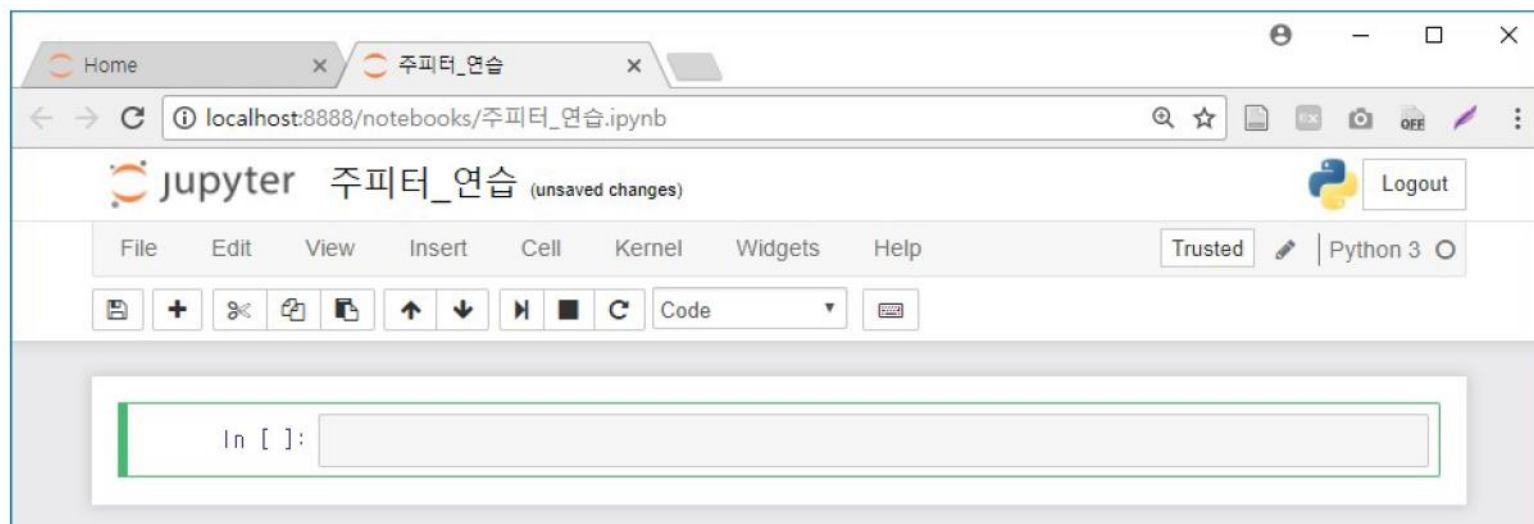
제목 변경, 코드 저장



2. 데이터 사이언스 도구 : Jupyter notebook

Cell 모드

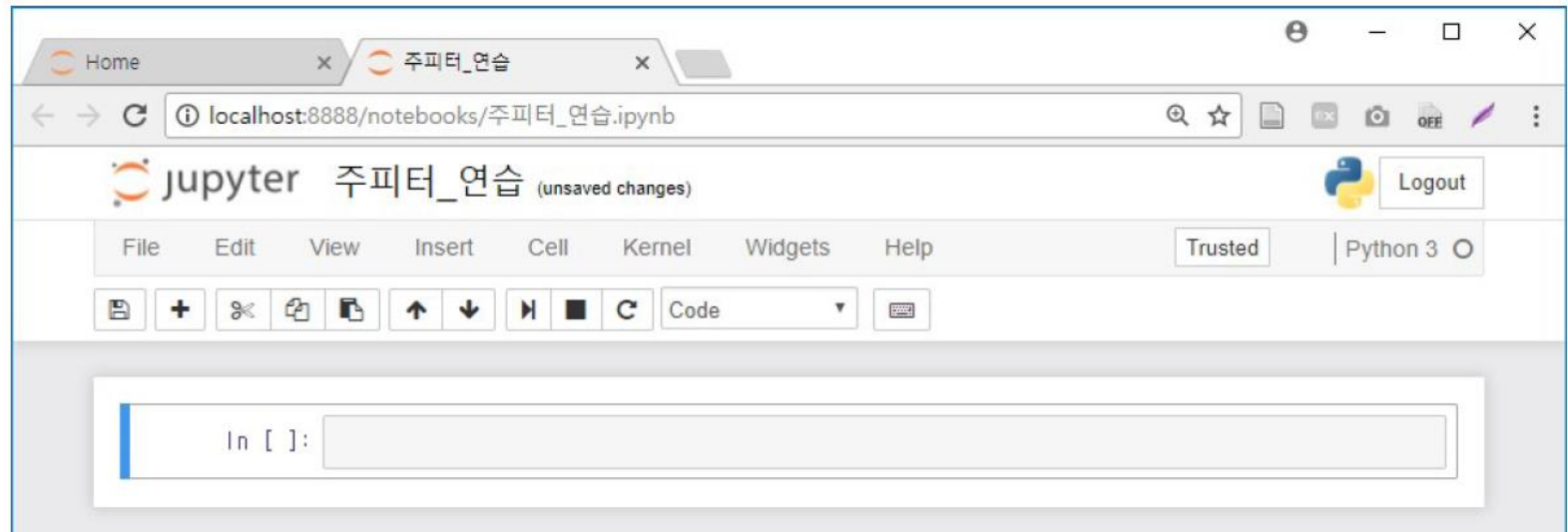
- 편집(Edit) / 명령(Command) 모드
- 편집(Edit) 모드
 - Cell 내에 내용을 입력
 - 테두리 색 : 초록색(green)
 - 단축 키 : Enter



2. 데이터 사이언스 도구 : Jupyter notebook

Cell 모드

- 명령(Command) 모드
 - Cell에 어떤 명령을 내리고 싶을 때 사용
 - 테두리 색 : 파란색(blue)
 - 단축 키 : Esc



2. 데이터 사이언스 도구 : Jupyter notebook

Cell 조작하기

- Cell 추가

(명령 모드에서)

- 현재 cell 위에 추가 : a
- 현재 cell 아래에 추가 : b

```
In [1]: a = 1
```

```
In [ ]:
```

- Cell 제거

- 현재 cell 제거 : x
- cell 제거 취소(undo) : z

2. 데이터 사이언스 도구 : Jupyter notebook

Cell 조작하기

- Cell 복사
 - C
- Cell 붙여넣기
 - 현재 cell 아래에 붙여넣기 : v
 - 현재 cell 위에 붙여넣기 : shift + v

2. 데이터 사이언스 도구 : Jupyter notebook

Cell 조작하기

- Cell 이동
 - 화살표 버튼(상/하)
- Cell 합치기
 - 현재 cell + 아래 cell 합치기 : shift + m

2. 데이터 사이언스 도구 : Jupyter notebook

단축 키(short cuts)

- anaconda prompt를 통해 생성한 python_class 폴더로 작업환경을 설정하기 위한 작업
- Jupyter notebook 상의 New → Python3 클릭을 통해 작업 시행

▪ Jupyter notebook 단축키

단축키	동작
Shift + Enter	현재 셀을 수행하고 아래 셀을 선택, 셀이 없을 경우 추가
Alt + Enter	현재 셀을 수행하고 아래에 새로운 셀을 추가
Ctrl + Enter	현재 셀을 수행, 아래 셀 추가나 및 선택 없음
Ctrl + S	Jupyter notebook 저장
Enter	명령 모드에서 입력 모드로 전환
Esc	입력 모드에서 명령 모드로 전환
M	명령 모드에서 셀 타입을 마크다운으로 전환
Y	명령 모드에서 셀 타입을 코드로 전환
Ctrl + /	셀 편집 영역에서 선택된 코드를 주석/비주석으로
상·하 방향키	명령 모드에서 셀 간의 이동

2. 데이터 사이언스 도구 : Jupyter notebook

○ 편리한 특징

- ❖ Syntax Highlighting
 - Automatically highlights standard functions (e.g. **for**, **range**), keywords (e.g. **in**, **and**), special characters (e.g. #)
- ❖ Auto Indent
 - Primarily driven by the colon operator (:)
 - Automatically indents blocks after if, for, while, etc.
 - Helps with debugging
- ❖ Parentheses Matching
 - Helps with debugging

2. 데이터 사이언스 도구 : Jupyter notebook

○ Tab Completion

- ❖ Type part of the name and then press <Tab> to see options
 - The more you type, the more specific the matches are
 - Auto-completes if there's only one option
 - Use to learn about useful attributes and methods
- ❖ Use as much as possible... saves time !
 - Variable names
 - Function names, keywords, and descriptions
 - File directions and names
 - Objects, attributes, and methods
 - Need to assign to variables first in this case
- ❖ Insert ? after a variable, function, or object to find out more information (e.g. sum?)

2. 데이터 사이언스 도구: 파이썬 라이브러리

- 파이썬의 장점은 유용한 라이브러리가 많다.
- 기본적인 라이브러리는 numpy, pandas, matplotlib, scikit-learn 등
- 딥러닝 모델을 이용하려면 텐서플로우, 케라스 등을 추가로 설치해야 한다.

numpy

- numpy는 Numerical Python의 의미
- 데이터 처리에 많이 사용되는 다차원 어레이(매트릭스)를 제공하며 매트릭스를 사용하면 계산 속도가 빠르다.
- 내부적으로 C 언어로 작성

pandas

- pandas 패키지는 데이터를 microsoft의 excel을 사용하듯이 테이블 구조의 데이터를 편리하게 조작하는데 쓰이는 패키지
- numpy가 매트릭스의 연산 (곱셈, 덧셈 등)을 빠르게 처리하기 위한 것이라면 pandas는 테이블 구조의 데이터의 컬럼 추가, 컬럼 삭제, 조건에 맞는 행 추출 등을 편리하게 수행
- pandas는 테이블 구조의 데이터를 담기 위해서 DataFrame 타입을 제공

2. 데이터 사이언스 도구: 파이썬 라이브러리

matplotlib

- 데이터 시각화를 위해 사용되는 라이브러리
- histogram, boxplot, 직선 그리기, 산포도(scatter plot) 등을 그리는 함수를 제공

scikit-learn

- 간단히 sklearn이라고 부르기도
- 선형회귀, 결정트리, 랜덤포레스트 등 machine learning 알고리즘들을 포함하는 라이브러리

Importing Modules and Scripts

- ❖ Modules and Python scripts are loaded in the same manner. For a module or Python script `P` (.py):
 - (ex) **`import P [as p]`**
 - Loads the module or script into the workspace, with an optional shorter name
 - Can use any functionality in an OOP fashion (e.g., `P.method()`)
 - (ex) **`from python_module import *`**
 - Imports all of the functionality directly into workspace
 - (ex) **`from python_module import f, g, h`**
 - Imports specific functions

2. 데이터사이언스 도구: 프로그램 개발 환경

구글 colab

<https://colab.research.google.com>

- 구글이 파이썬 주피터 노트북 환경을 무료로 제공
- colab에서는 GPU도 사용할 수 있어 딥러닝을 구동할 때 속도가 빠르다.
- 프로그램 환경설정 내용과 업로드한 데이터를 12시간만 유지해준다는 단점이 있다. 즉, 12 시간이 지나면 서버에서 데이터가 사라진다.
- colab에 파일을 업로드하거나 colab에서 작성한 프로그램을 저장하는 방법은 여러 가지(자신의 컴퓨터, 구글 드라이브, 또는 github) 가능

github

- 프로그램 공유 및 버전 관리 사이트
- 대부분의 프로그램 개발자가 사용
- 편리하게 프로그램 source code를 배포할 수 있다.

파이썬 기초

기본 변수

- 정수(int), 실수(float), 불리언(bool), 문자열(str)
- 변수명, 변수 타입을 미리 정의할 필요없이 임의로 즉시 만들어 사용
- 예약어(if, for, True 등)는 변수로 사용할 수 없다.

```
In [1]: type(4)
```

```
Out[1]: int
```

```
In [2]: type(3.3)
```

```
Out[2]: float
```

Python을 활용한 계산

사칙연산

- Python에서의 사칙연산 기호는 더하기(+), 빼기(-), 곱하기(*), 나누기(/)로 구성

```
In [1]: 2+2
```

```
Out[1]: 4
```

```
In [2]: 3-2
```

```
Out[2]: 1
```

```
In [3]: 13*3
```

```
Out[3]: 39
```

```
In [4]: 15/3
```

```
Out[4]: 5.0
```

Python을 활용한 계산

거듭제곱 및 나머지

- Python에서의 거듭제곱 연산자는 **로 활용되고 있음(공백 주의)

```
In [1]: 5 ** 2
```

```
Out[1]: 25
```

```
In [2]: 2.5 ** 2
```

```
Out[2]: 6.25
```

```
In [3]: 4 ** (1/2)
```

```
Out[3]: 2.0
```

Python을 활용한 계산

몫과 나머지

- 예를 들어 13 나누기 3을 진행 할 경우 몫은 3이고 나머지는 2임
- 몫은 // 연산자로 구할 수 있음
- 나머지는 % 연산자로 구할 수 있음

```
In [1]: 13 // 3
```

```
Out[1]: 4
```

```
In [2]: 13 % 3
```

```
Out[2]: 1
```

Python을 활용한 계산

몫과 나머지

- 예를 들어 13 나누기 3을 진행 할 경우 몫은 3이고 나머지는 2임
- 몫은 // 연산자로 구할 수 있음
- 나머지는 % 연산자로 구할 수 있음

```
In [1]: 13 // 3
```

```
Out[1]: 4
```

```
In [2]: 13 % 3
```

```
Out[2]: 1
```

변수

- 변수 이름 작명 규칙
 - 알파벳 대/소문자, 숫자, _ 그리고 한글, 한자 등 사용 가능
 - 단, 첫 글자는 숫자로 시작할 수 없다.
 - 공백, 특수문자, 문장 부호는 변수의 이름으로 쓸 수 없다.
- 덮어쓰기
 - 중복되는 변수가 있다면, 나중에 정의된 변수가 선택된다.
- 여러 변수 지정하기
 - 한번에 여러 변수를 지정할 수 있다.
 - $a = b = 1$
 - $a, b = 1, 2$
 - $a, b = b, a$

변수

- 변수 삭제
 - del 함수 사용
- Name Error
 - 변수명을 정의하지 않고 실행시키면 , Name Error 발생

```
In [6]: 1 aa # 정의하지 않은 변수
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-6-a18dbf8aafdf> in <module>
----> 1 aa # 정의하지 않은 변수

NameError: name 'aa' is not defined
```

문자열

- 따옴표(' ', " ") 사용

```
a = '문자열입니다'
b = "문자열입니다"
```

- 여러 줄(line)으로 된 문자열은 """ """ 으로 묶어서 표시

```
"""
```

```
영희가 물었다. "철수야, 숙제했어?"
```

```
철수는 생각했다. '영희는 숙제를 안하나?'
```

```
"""
```

- str() 함수를 사용하여 변환

```
In [7]: str(123)
```

```
Out[7]: '123'
```


문자열

- 문자열 표현

'\n' # 줄바꿈
'\t' # 탭

- 문자열 연산

```
1 '1 더하기 1은 ' + '2 입니다'
```

'1 더하기 1은 2 입니다'

```
1 '하' * 3
```

'하하하'

불리언(bool)

- 논리값 : True, False (대소문자를 구분)
- 파이썬에서는 논리값이 마치 숫자인 것처럼 연산에 바로 사용 가능
 - True : 1
 - False : 0
 - 숫자(정수&실수)와 연산 시, 자동 변환

```
x = 3  
y = True  
print (x + y)                # 출력: 4
```

리스트(list)

- 여러 데이터를 하나의 변수에 저장하는 방법
- 리스트는 정수, 소수, 문자, 논리값 등 임의의 데이터 타입을 담을 수 있다.
- [] 사용

```
x = 3  
y = True  
z = '홍길동'
```

```
list_1 = [x, y, z]  
print(list_1)                # [3, True, '홍길동']
```

- len() 함수 : 리스트 내에 들어있는 항목의 갯수

```
print(len(list_1))           # 출력: 3
```



- 리스트 내에 어떤 항목이 들어 있는지를 확인
- 해당 항목이 들어 있으면 True를 반환

```
print(3 in list_1)          # True
print(4 in list_1)          # False
```

- `append()` 함수 : 리스트에 항목을 추가

```
list_1.append(100)
print(list_1)                # [3, True, '홍길동', 100]
```

(주의) `append()`를 실행하면 리스트 `list_1`의 내용이 변경된다

range

- 일정 간격(default=1)으로 일정한 범위의 숫자 리스트를 만든다

```
x2 = range(10)           # 리스트 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- for 문의 반복 수행되는 영역 앞에는 반드시 ":"가 있어야 한다
- block은 반드시 동일한 크기로 들여쓰기 (indent)를 해주어야 한다

```
x3 = range(10, 20, 2)
for i in x3:
    print(i)               # 10, 12, 14, 16, 18
```

- 리스트 내의 특정 위치의 값을 얻으려면 인덱싱(indexing)을 사용

```
print(x3[0])      # 10
print(x3[2])      # 14
print(x3[-1])     # 18
print(x3[-2])     # 16
```

List Comprehension

- 리스트 내부에서 for문을 사용

```
list_2 = [x*10 for x in range(5)]  
print(list_2)                                # [0, 10, 20, 30, 40]
```

Tuple

- 리스트와 유사하게 여러 항목의 집합체
- 튜플은 항목의 값을 바꿀 수 없다(즉, 튜플은 상수화된 리스트)
- 튜플을 사용하는 이유
 - 항목의 내용을 더 이상 내용을 변경하는 것을 방지
 - 처리속도를 빠른 장점
- 튜플을 만들려면 ()을 사용하거나, 아무 괄호를 넣지 않아도 된다.

```
t1 = (1, 2, 3)
print(t1)                # (1, 2, 3)
```

```
t2 = 4, 5, 6
print(t2)                # (4, 5, 6)
```

○ 딕셔너리(dictionary, 사전)

- 데이터 형식
 - 모든 항목이 "키(key)", "값(value)" 의 pair로 구성
- {}를 사용
- 항목의 값은 키 값을 인자로 하여 얻을 수 있다.

```
name_age = {"kim": 20, "lee": 25}
name_age["kim"]          # 20
```

- 딕셔너리 내에 특정한 키 값이 들어 있는지 아닌지를 in으로 확인

```
print("kim" in name_age)    # True
print("park" in name_age)   # False
```

- 새로운 항목을 추가 : 새로운 "키" 값을 인자로 주면서 새로운 값을 배정

```
name_age["song"] = 35
print(name_age)    # {'kim': 20, 'lee': 25, 'song': 35}
```


○ 딕셔너리(dictionary, 사전)

- 딕셔너리에 어떤 키(Key)가 있는지 보거나 또는 값(Value)들만을 출력
 - 각각 `keys()`, `values()` 함수를 사용

```
name_age.keys()          # dict_keys(['kim', 'lee', 'song'])  
name_age.values()        # dict_values([20, 25, 35])
```

- `items()` 함수
 - 딕셔너리 각 항목을 튜플로 만들고 이 항목들을 모두 리스트로 리턴
 - 딕셔너리를 튜플로 바꾸는 이유
 - tuple로 만들어서 검색 속도를 빠르게 하기 위해

```
name_age.items()          # dict_items([('song', 35), ('lee', 25), ('kim', 20)])
```

if-else 문

- if-else 문을 사용하면 조건에 따라 프로그램의 흐름을 정할 수 있다
- elif을 사용하면 조건을 두 가지 이상의 여러 가지로 적용 가능

```
x=3
```

```
if x > 10:
```

```
    print("x > 10")
```

```
elif x > 3:
```

```
    print("10 >= x < 3")
```

```
else:
```

```
    print(" x < 3")
```

for 문

- 어떤 조건이 만족되는 동안 작업을 수행

```
for x in (1,3,5,10):  
    print(x, "'s sqaure = ", x*x)
```

출력

1 's sqaure = 1

3 's sqaure = 9

5 's sqaure = 25

10 's sqaure = 100

○ sort()

- 리스트 항목의 내용들을 값의 크기 순으로 정렬
- 기본적으로 오름차순으로 정렬
- sort()를 실행하고 나면 리스트의 내용이 변경

```
x = [2, 1, 3, 5, 4]
x.sort()
print(x)           # [1, 2, 3, 4, 5]
```

- 만일 원래 리스트의 내용이 변경되지 않게 하려면 sorted() 메소드를 사용하고 새로운 변수에 지정하면 된다.

```
x = [2, 1, 3, 5, 4]
y = sorted(x)
print(y)           # y = [1, 2, 3, 4, 5]
print(x)           # x = [2, 1, 3, 5, 4], 원래 값을 그대로 유지
```

○ sort()

- 내림차순으로 정렬하려면, 속성값 reverse를 True로 설정

```
x = [2, 1, 3, 5, 4]
x.sort(reverse=True)
print(x)                # x = [5, 4, 3, 2, 1]
```

○ sort_values()

- 데이터프레임을 정렬하는 명령어, 특정 열을 기준으로 정렬
- 변수 (by : 기준 열, ascending : 오름차순, 내림차순 정렬을 선택)

```
df = pd.DataFrame({'순서': [1, 3, 2],  
                  '이름': ['park', 'lee', 'choi'],  
                  '나이': [30, 20, 40]})  
df.sort_values(by=['순서'], ascending=True)
```

```
##  
나이 순서 이름  
0 30 1 park  
2 40 2 choi  
1 20 3 lee
```

들여쓰기(indentation)

- if, elif, else, for, def 문 내부에 해당하는 블록을 구분하기 위해서 사용
 - 보통 탭을 사용하지만 탭의 크기는 블록 내에서 통일시켜야
- (), { } 블록 내부에서는 space나 return이 무시된다
 - 코드의 길이가 긴 경우 보기 쉽게, 여러 줄로 표시해도 동작은 동일

```
x = [[1,2,3],  
      [4,5,6],  
      [7,8,9]]  
print(x)
```

```
##  
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

- (), { } 괄호가 없는 경우에 코드를 여러 줄로 나누어 쓰려면 아래와 같이 문장 끝에 역 슬래시 "\w"를 사용해서 줄이 연속되는 것을 나타낼 수 있다.

```
x = 3 + 4 + 5 + 6 \w  
    + 7
```

함수 정의

- def : 임의의 기능을 함수로 정의

```
def double(x):  
    return x*2
```

```
double(5)          # 10
```

- 함수의 인자 값으로 디폴트 값을 지정할 수 있다

```
def double(x=100):  
    return x*2
```

```
double()           # 200
```


함수 정의

- 함수 실행 결과로 두 개 이상의 값을 리턴할 때
 - 리턴 값들을 tuple로 처리하면 편리

```
def double_triple(x):  
    return x*2, x*3
```

```
x, y = double_triple(4)  
print(x, y)                # x = 8, y = 12
```

Indexing & Slicing

- Indexing
 - 원하는 자료에 접근
 - 0 부터 시작
 - - 부호 : 뒤에서부터 센다는 의미
 - * -1 : 제일 마지막 자료
- Slicing
 - 자료의 범위를 선택

Slicing: list and array

❖ 1-D array slicing (quite often used)

```
a = np.arange(10)    # a = array([0,1,2,3,4,5,6,7,8,9])
a[start:end]         # items start through end-1
a[start:]            # items start through the rest of the array
a[:end]              # items from the beginning through end-1
a[:]                 # a copy of the whole array
a[start:end:step]    # start through not past end, by step

a[-1]                # last item in the array
a[-2:]               # last two items in the array
a[:-2]               # everything except the last two item
a[::-1]             # all items in the array, reversed
a[1::-1]             # the first two items, reversed
a[:-3:-1]            # the last two items, reversed
a[-3::-1]            # everything except the last two items, reversed
```

Slicing: list and array

❖ 2-D array slicing (to split loaded data into input(X) and the output(y))

```
X =[:, :-1]    # select all the rows and all columns except the last one  
y =[:, -1]     # select all rows again, and index just the last column
```

Python을 활용한 계산

논리 연산

- 참(True), 거짓(False)를 이용하여 조건을 만족하거나 아닐 경우를 이용해 연산에 활용하는 방법
- 논리 연산은 'boolean 연산' 이라고도 하며 파이썬의 논리 연산을 위한 데이터 타입은 'bool' 이다
- 논리 연산자인 논리곱(and), 논리합(or), 논리 부정(not)이 존재함

논리 연산자	활용 예	내용
논리곱 (and)	A and B	A와 B 둘다 참일 때 참, 나머지는 거짓
논리합 (or)	A or B	A와 B중 한 개라도 참일 경우 참, 둘다 거짓일 때 거짓
논리 부정(not)	not A	A가 참이면 거짓이고, 거짓일 경우 참

Python을 활용한 계산

논리 연산

- 참(True), 거짓(False)를 이용하여 조건을 만족하거나 아닐 경우를 이용해 연산에 활용하는 방법
- 논리 연산은 'boolean 연산' 이라고도 하며 파이썬의 논리 연산을 위한 데이터 타입은 'bool' 이다
- 논리 연산자인 논리곱(and), 논리합(or), 논리 부정(not)이 존재함

```
In [1]: print(True)
        print(False)
```

```
True
False
```

```
In [2]: type(True)
```

```
Out[2]: bool
```

```
In [1]: print(True and False)
        print(True or False)
        print(not True)
        print(not False)
```

```
False
True
False
True
```

Python을 활용한 계산

비교 연산

- 숫자를 비교하는 연산으로 결과는 bool 데이터로 출력됨
- 논리 연산과 함께 이용하는 경우가 많음

비교연산자	의미	예	내용
<	작다	a<b	a는 b보다 작다
>	크다	a>b	a는 b보다 크다
<=	작거나 같다	a<=b	a는 b보다 작거나 같다
>=	크거나 같다	a>=b	a는 b보다 크거나 같다
==	같다	a==b	a와 b는 같다
!=	같지 않다	a!=b	a와 b는 같지 않다

```
In [1]: print(4 < 3)
print(4 > 3)
print(4 <= 3)
print(4 >= 3)
print(4 == 3)
print(4 != 3)
```

```
False
True
False
True
False
True
```