



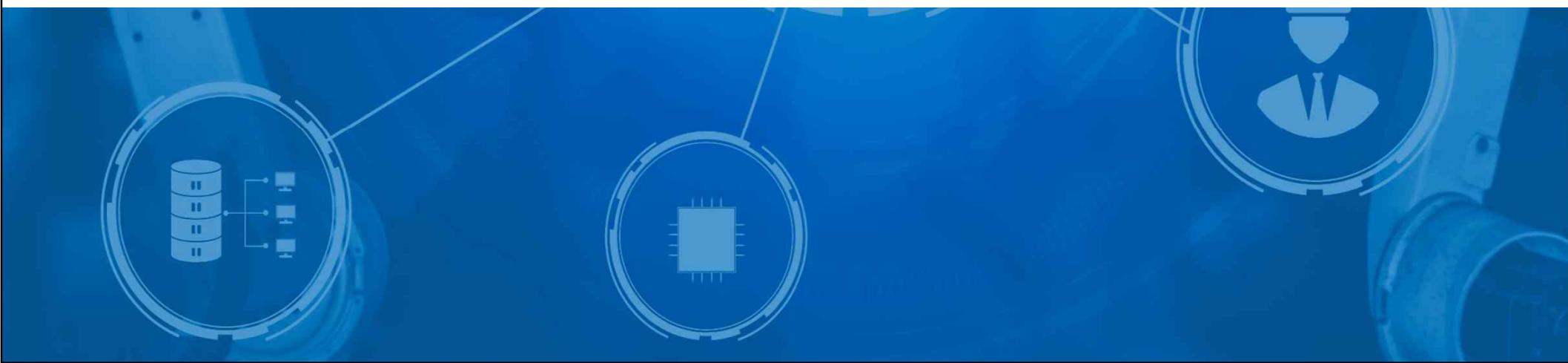
인공지능과 빅데이터 대한 이해

신병주

경남대학교 WISE LINC 3.0 사업단, 프로네시스융합학부장, 컴퓨터공학부 교수

경남대학교 빅데이터센터장, 인공지능융합ICC센터장

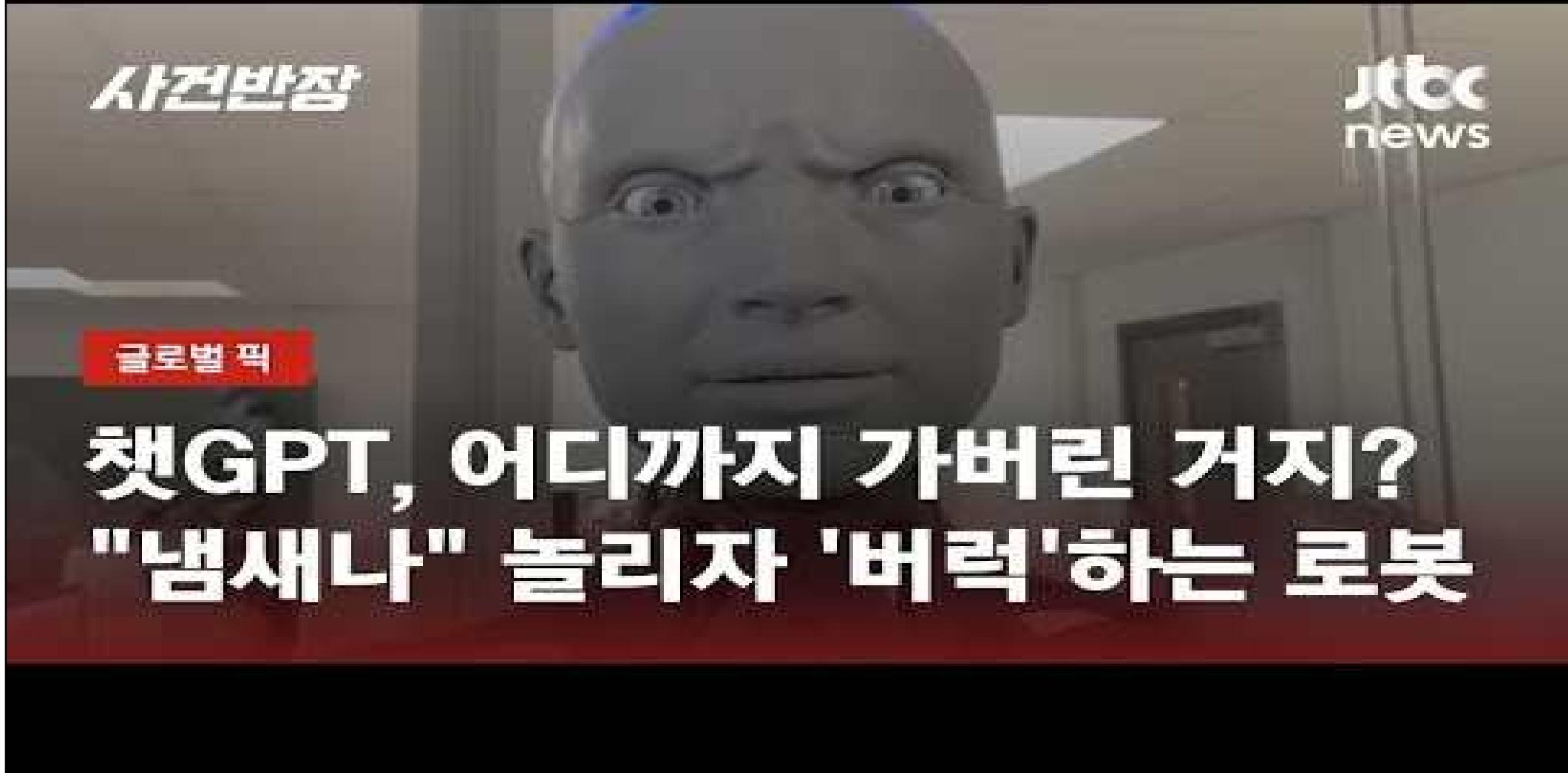
bjshin@kyungnam.ac.kr



인공지능, 빅데이터 시대



인공지능, 빅데이터 시대



인공지능이란?

x	y
-2	7
-1	4
0	1
1	-2
2	-5
3	-8
4	-11

$x = 301$
 $y = ?$

인공지능이란?

Learning

	x	y
	-2	7
	-1	4
	0	1
	1	-2
	2	-5
	3	-8
	4	-11

독립변수(feature)

종속변수

$$\begin{aligned} x &= 301 \\ y &= -902 \end{aligned}$$

} 예측
(Prediction)

$$y = -3x + 1$$



AI 모델

인공지능이란?

x	y
51	-152
-15	46
48	-143
-108	325
365	-1094
1056	-3167
-3265	9796

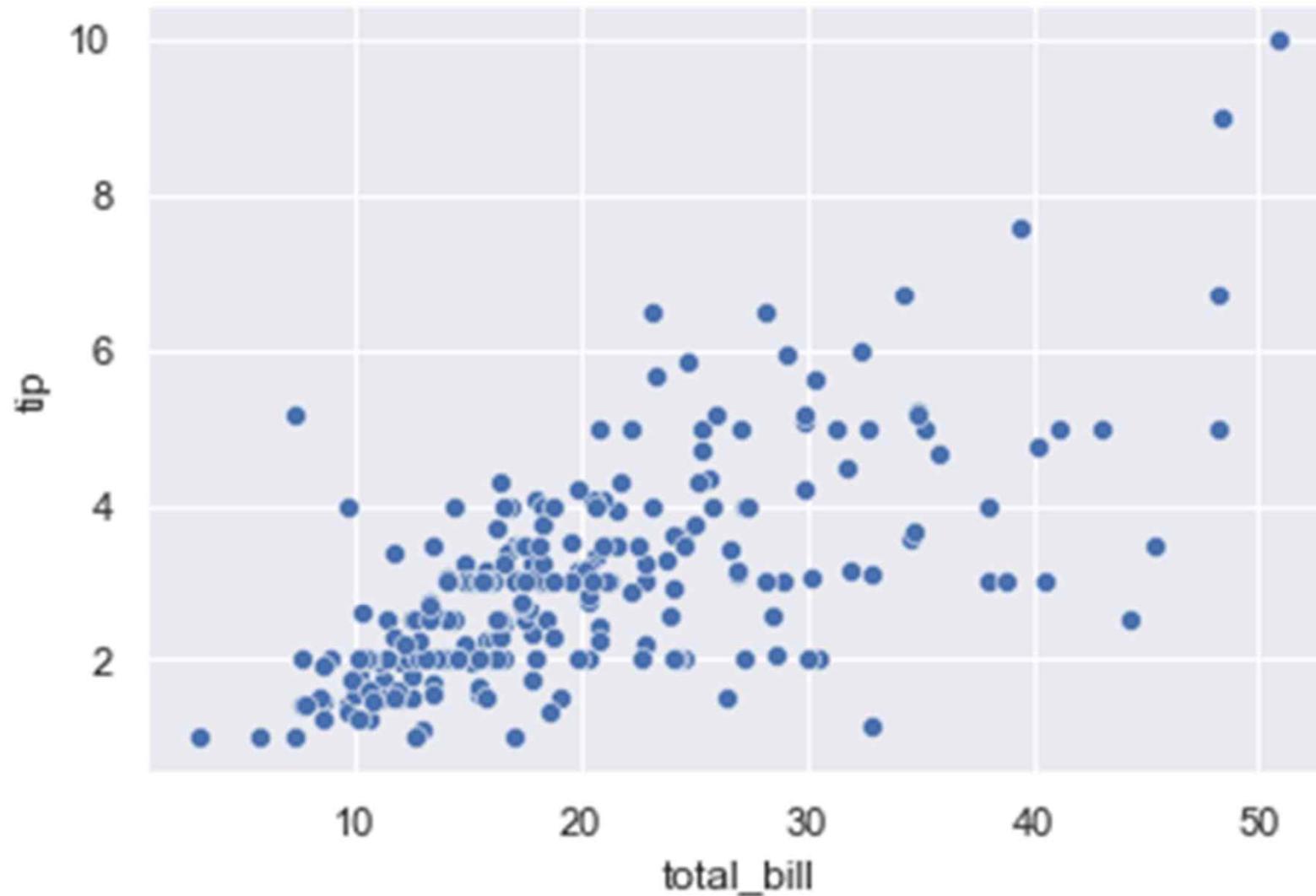
$$\begin{aligned}x &= 100 \\y &= ?\end{aligned}$$

인공지능이란?

x1	x2	x3	y
-2	1	-7	-17
-1	2	-6	-11
0	3	-5	-5
1	4	-4	1
2	5	-3	7
3	6	-2	13
4	7	-1	19

$x_1 = 1405$
 $x_2 = 10$
 $x_3 = 30$
 $y = ?$

인공지능이란?



강의자료

<https://github.com/shinbyungjoo/Python-Lecture/>

인공지능, 빅데이터 이해

구글신 vs. 구글철학관

인공지능 vs. 빅데이터

데이터 마이닝 vs. 머신러닝

Hadoop vs. Spark

R vs. Python

인공지능, 빅데이터 = 돈?

Data Scientist vs. Data Analyst

누구나 빅데이터 분석가가 될 수 있다?

인공지능의 현실성

구글신 vs. 구글철학관



구글신 vs. 구글철학관

한영자막

알렉사가
없던 시절

아마존 슈퍼볼광고

인공지능 vs. 빅데이터

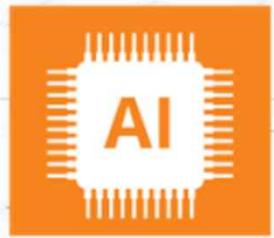
The three essential future technologies



Data collection through IOT



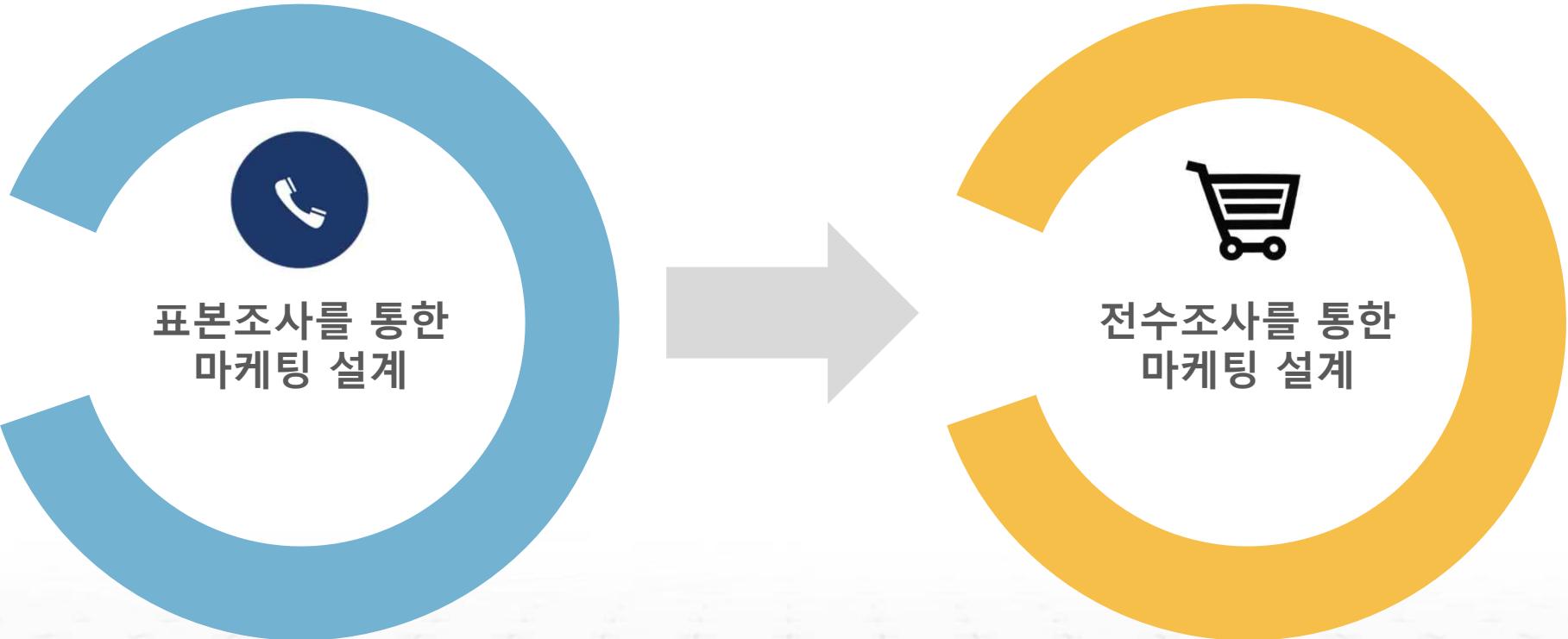
Capture, storages, analysis of data



Data based learning

Source: I-ON communications blog

Data Mining vs. Machine Learning



Data Mining vs. Machine Learning

나이	연봉 평균(만원)
30	4,000
31	4,100
32	4,200
33	4,300
34	4,400
35	4,500
36	4,600

내가 40세가 되었
을 때 받을 연봉은?

Data Mining vs. Machine Learning

Data mining

- 많은 데이터 속에서 유효한 정보를 찾아내는 과정
(데이터 속에 숨어 있는 의미를 발견)

Machie learning

- 데이터를 이용해 검증과 학습의 과정을 통해 특정 조건에서 예측값을 얻는 과정
(과거의 추이를 통해 미래를 예측)

Data Mining vs. Machine Learning

성별	학습시간	학점
남	10	A0
남	11	A0
남	12	A+
여	5	A0
여	6	A0
여	7	A+
여	8	A+

- 학습시간이 많을수록 성적은 오른다.
- 남학생에 비해 여학생의 학습시간 대비 성적은 높다.

Data Mining vs. Machine Learning

성별	학습시간	학점
남	10	A0
남	11	A0
남	12	A+
여	5	A0
여	6	A0
여	7	A+
여	8	A+

- 학습시간이 많을수록 성적은 오른다.
- 남학생에 비해 여학생의 학습시간 대비 성적은 높다.
- 내가 20시간을 학습했을 때 받을 학점은?

Data Mining vs. Machine Learning

- 지도시각화 분석

[메인페이지]

한경기상

상수도수질검사 현황

수령일: 2023-02-22 조회수: 1344 다운로드: 96

교통동향

화재현장

KBS

이건 아니지
말입니다--

분석결과

산업경제 통

주변자치 소통센

산업현황 및 기업간색

상권 분석 서비스

산업분류별 데이터 테이블

■ 매출액

분류	매출액
A	17,221
B	7,697
C	26,000,000
D	637,066
E	76,243
F	6,215,672
G	8,204,793
H	677,127
I	60,514
J	242,607
K	35,544
L	1,806,932
M	485,449
N	333,642
O	71
P	10,007
Q	418,324
R	59,021
S	91,894
T	0

보사 | 역별다운

주거여행

전통시장 현황

수령일: 2020-12-18 조회수: 26 다운로드: 19

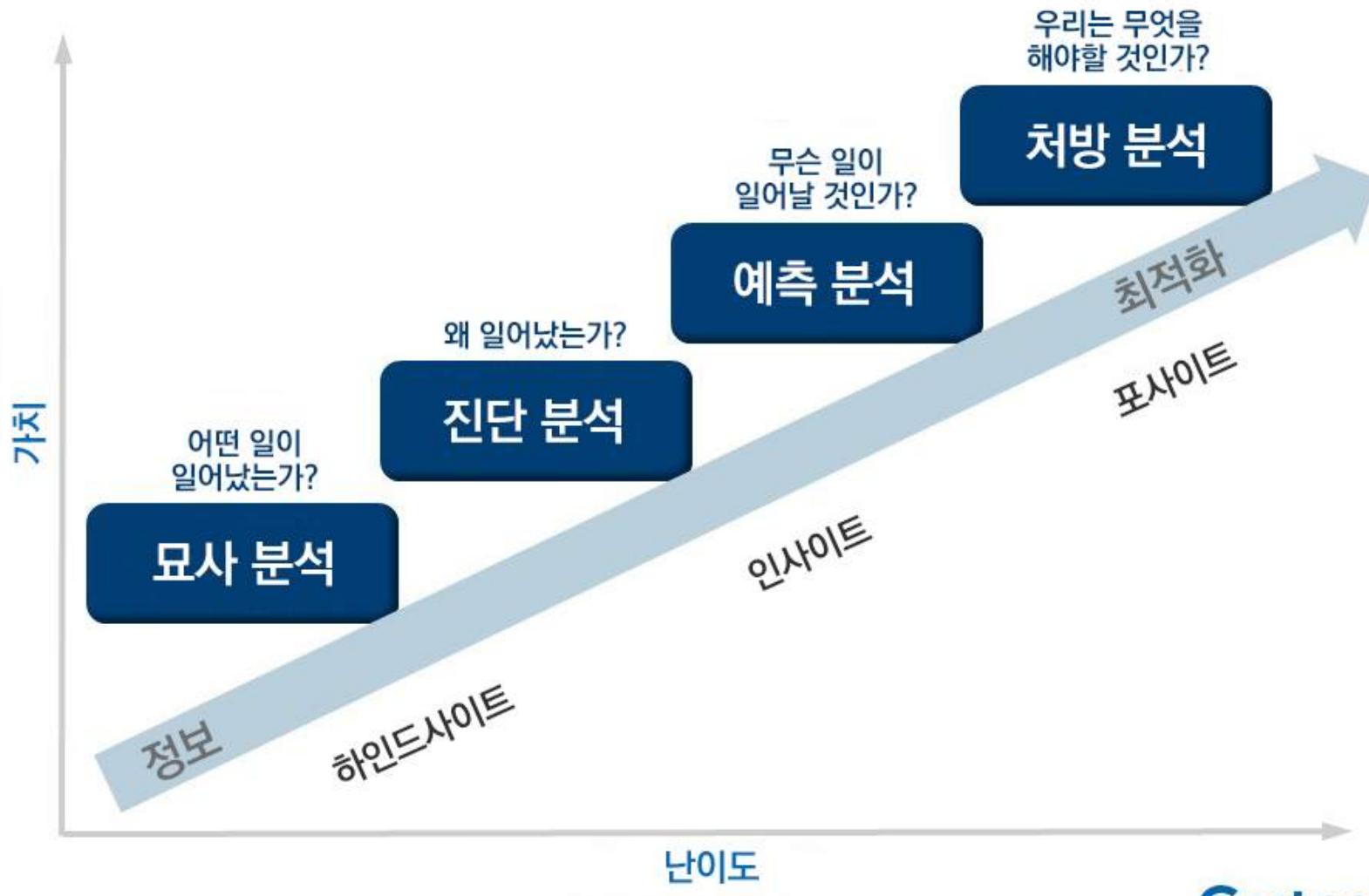
한경기상

생활폐기물수질문제 현황

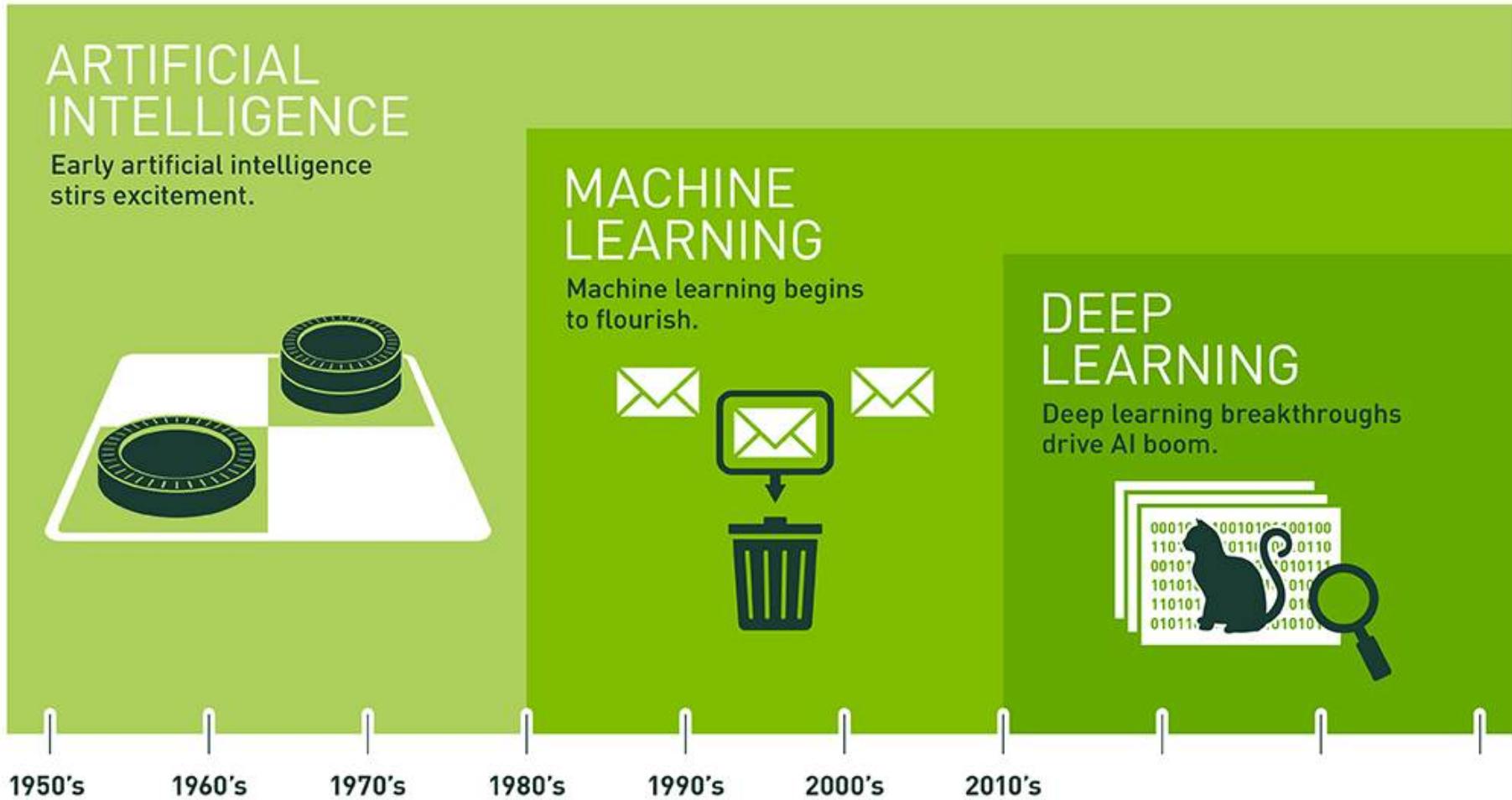
수령일: 2023-02-22 조회수: 158 다운로드: 34

Data Mining vs. Machine Learning

Analytic Value Escalator

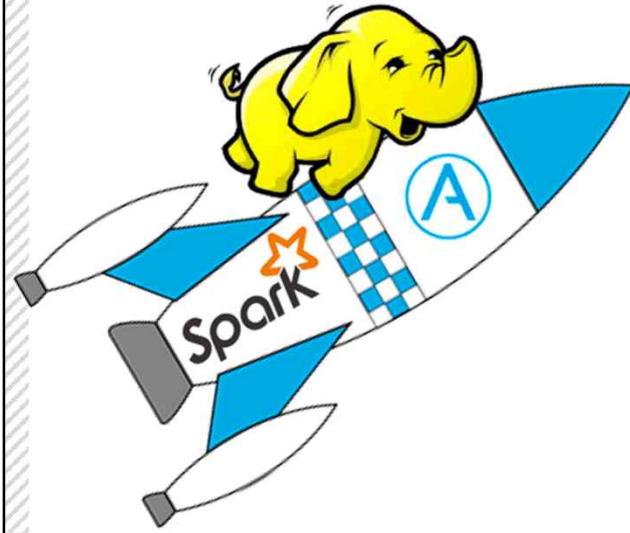


Data Mining vs. Machine Learning

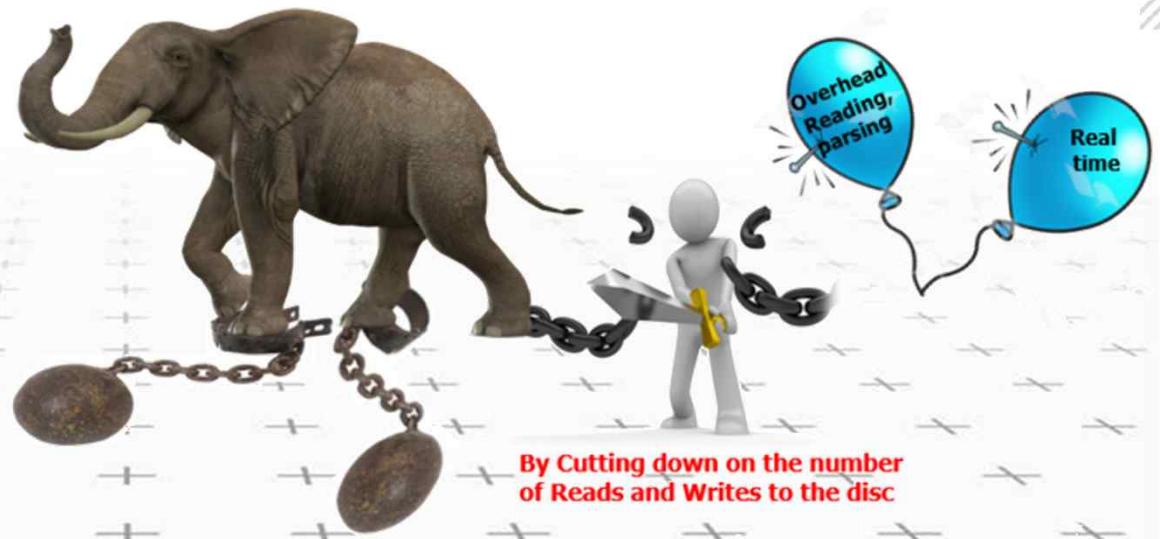
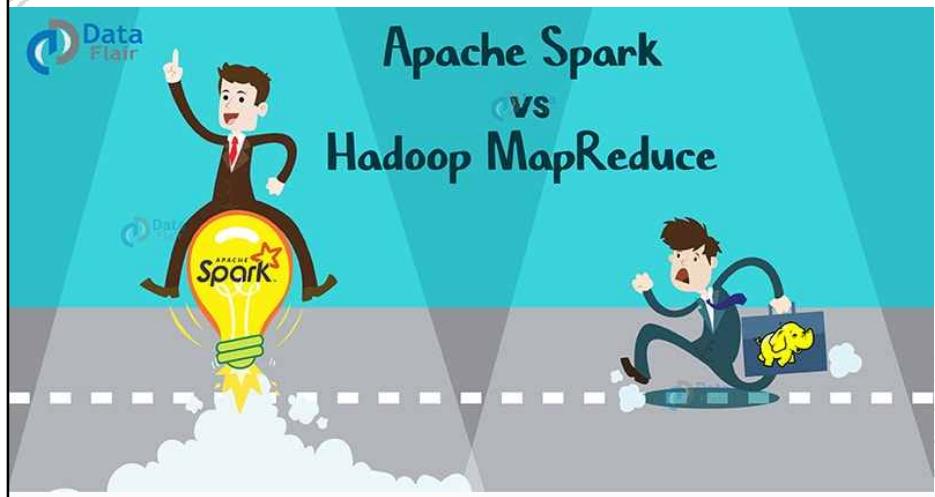


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Hadoop vs. Spark



Distributed Computing



R vs. Python



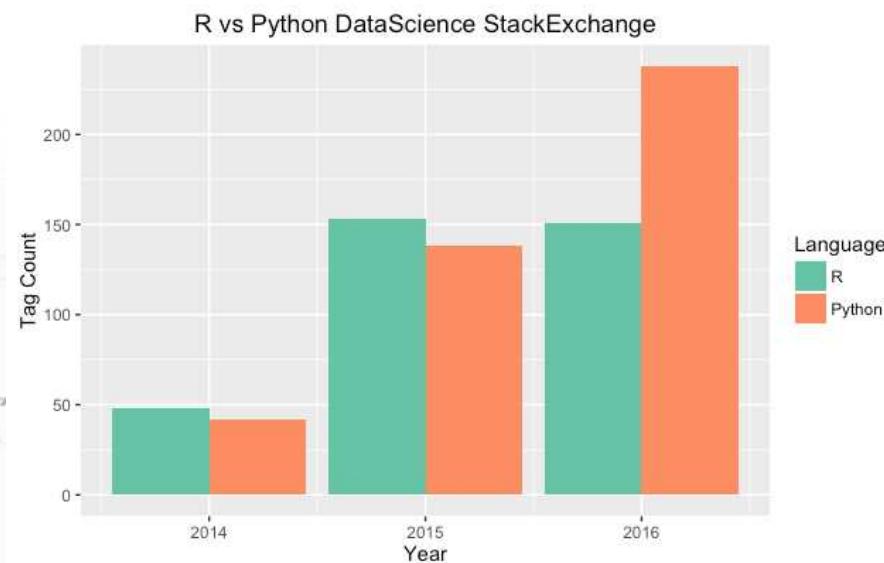
R & PYTHON: ECOSYSTEMS

R

Basically, this was created by statisticians for statisticians. Hence it is a preferable choice if the nature of the task relates to statistics, research and data science.

Python

This is a preferred choice for those working in the engineering environment.



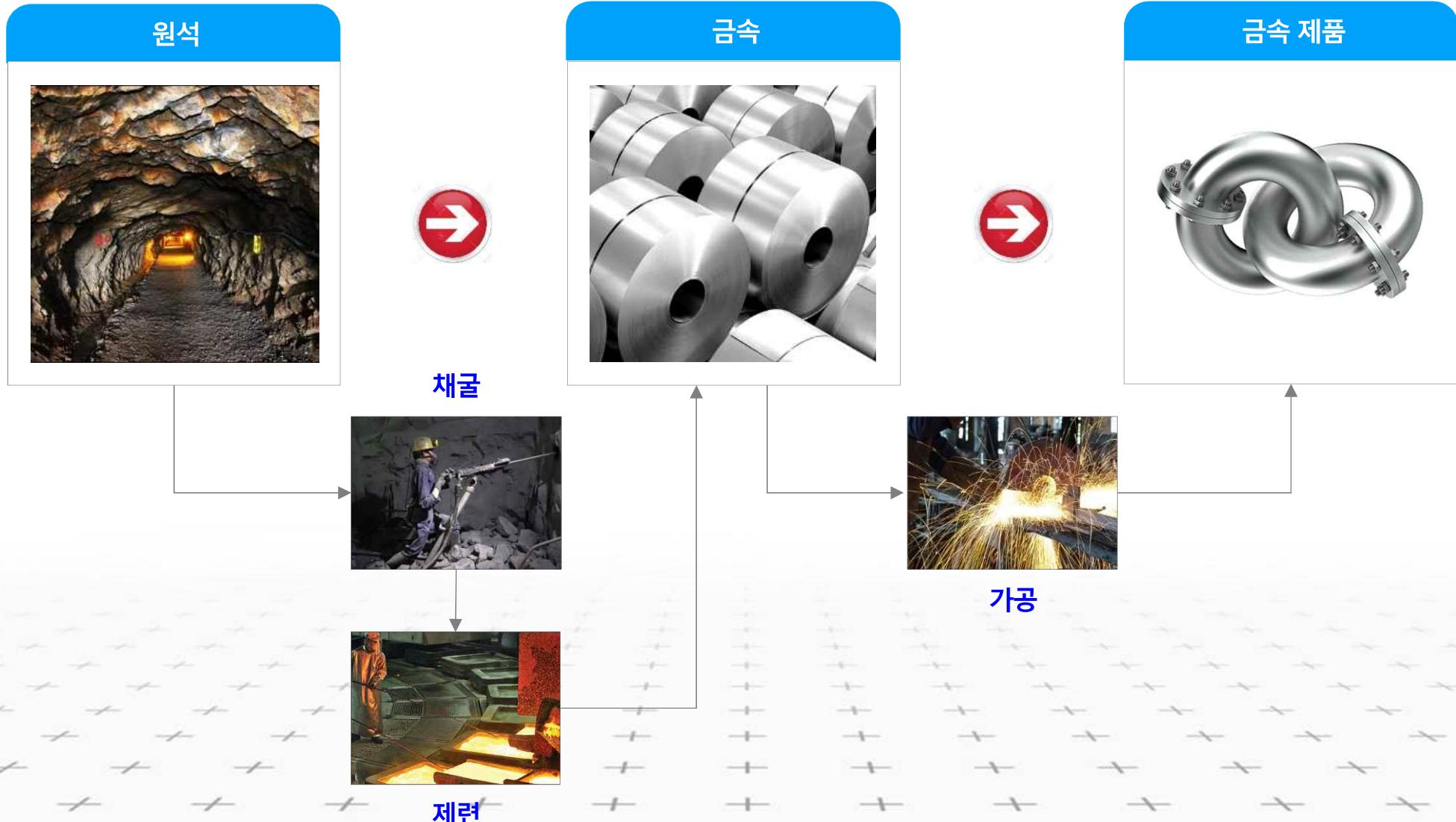
인공지능, 빅데이터 = 돈?

문제 - 빅데이터 인력 및 시스템 투자의 어려움



인공지능, 빅데이터 = 돈?

목적 달성 과정 - 금속 제품이 우리 손에 들어오기까지...



인공지능, 빅데이터 = 돈?

목적 달성 과정 – 데이터가 가치를 발휘할 때까지...



인공지능, 빅데이터 = 돈?

단계별 넘어야 할 고개 – 데이터 발굴 및 정제

기존 데이터 테이블의 해독 난이성 문제



비슷한 테이블이 너무 많음

- ✓ 시스템 운용 연수가 길수록
- ✓ 시스템 개선 작업(차세대 사업)을 많이 했을 수록
- ✓ 기존 테이블을 정리하지 않고 새로운 테이블을 만들어서 사용
- ✓ 데이터의 마이그레이션이 되어 있지 않으며 때로는 이 과정에서 데이터가 잘못된 컬럼에 연결되어 있음
- ✓ 또한 많은 컬럼의 값이 비어 있고 사용하지 않음

기존 테이블들에서 조인한 값을 특정 테이블에 입력함

- ✓ SQL문을 정확히 입수하기 전까지는 각각의 데이터가 어느 테이블에서 왔는지 유추하기 매우 어려움

인공지능, 빅데이터 = 돈?

단계별 넘어야 할 고개 – 데이터 발굴 및 정제

데이터 입수를 위한 담당자 미팅의 어려움



현업 인터뷰

- ✓ 특정 도메인에 대한 비즈니스 로직 이해가 필요함
- ✓ 큰 시스템의 경우 1~2명의 현업으로 파악이 안될 수 있음
- ✓ 업무와 관련된 모든 사람의 인터뷰 필요

현업 미팅

- ✓ 물리적으로 떨어져있는 경우 담당자 미팅이 매우 어려움
- ✓ 비즈니스 로직이 완전히 파악되지 않으면 불필요한 미팅일 수 있음
- ✓ 따라서 비즈니스 로직 이해가 선행되어야 함
- ✓ 용어에 대한 정의도 필요
- ✓ 테이블 정보를 주기 전까지 필요한 데이터를 파악하고 요청할 수 없음
- ✓ 그러나 보안상의 이유로 테이블 정보를 받기 어려움!

인공지능, 빅데이터 = 돈?

단계별 넘어야 할 고개 – 데이터 발굴 및 정제

기존 데이터 테이블의 해독 난이성 문제



현업 담당자와 전산 담당자 모두의 도움이 필요함

- ✓ 현업 담당자는 비즈니스 로직은 완전히 이해하고 있지만 데이터베이스에 어떻게 저장되고 사용되는지 모름
- ✓ 반면에 전산 담당자는 비즈니스 로직을 모르고 유지 보수 관점에서만 데이터를 다룸
- ✓ 경험상 전산 담당자 중에서 비즈니스 로직을 아는 사람보다 현업 중에서 전산을 일부 이해하는 사람을 찾는 편이 더 도움이 됨

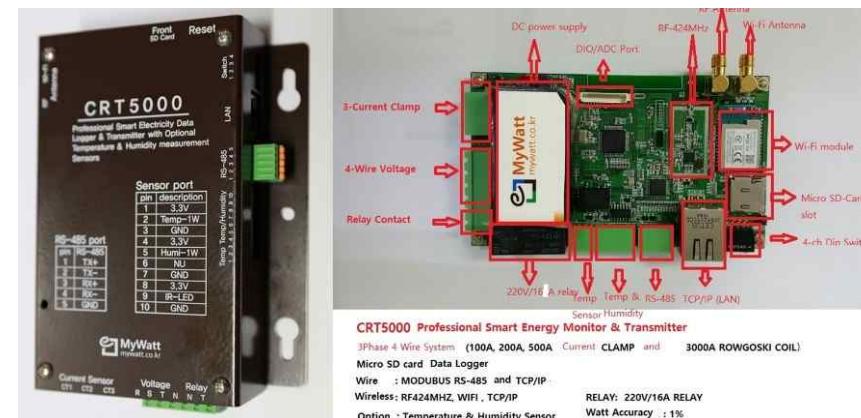
인공지능, 빅데이터 = 돈?

단계별 넘어야 할 고개 – 데이터 발굴 및 정제

데이터 신뢰성 문제

이상 데이터 값

-0.8	-0.67
-0.8	-0.67
-0.81	-0.67
-0.81	-0.66
-0.81	-0.67
-0.81	-0.67
-0.81	-0.67
-0.81	-0.67
-0.81	-0.66
-0.81	-0.66
-0.81	-0.67
-0.8	-0.67
-0.81	-0.66
-0.81	-0.66
-0.81	-0.67



결측 데이터

19	11	26	654.83	225	224	228	2015-09-23 0:01
19	11	27	654.84	225	224	228	2015-09-23 0:02
19	11	27	654.83	225	224	228	2015-09-23 0:02
19	10	26	654.85	225	224	228	2015-09-23 0:02
19	9	26	654.89	225	224	228	2015-09-23 0:02
19	9	26	654.87	225	224	228	2015-09-23 0:02
19	9	27	654.86	225	224	228	2015-09-23 0:02
65	23	37	654.43	224	224	227	2015-09-25 9:36
65	23	37	654.43	224	224	227	2015-09-25 9:36
65	23	37	654.43	224	224	227	2015-09-25 9:36
66	21	37	654.43	224	224	227	2015-09-25 9:36
66	21	37	654.44	224	224	227	2015-09-25 9:36

사용자 부주의



인공지능, 빅데이터 = 돈?

단계별 넘어야 할 고개 – 데이터 발굴 및 정제

클래스 불균형 문제

클래스 불균형 문제란?

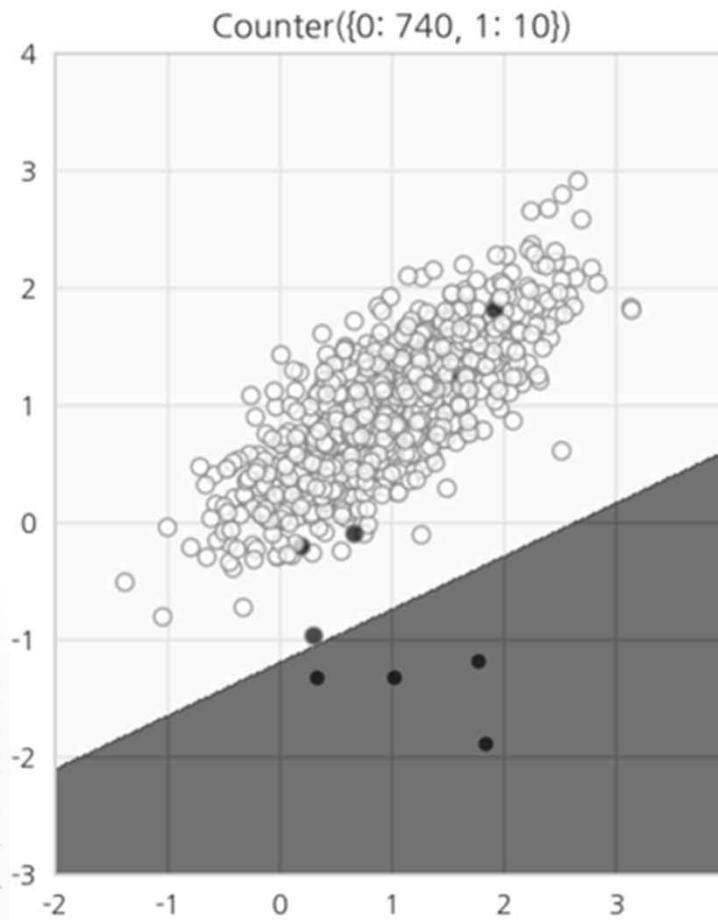
- ✓ 데이터 클래스의 총 수 (Positive)가 다른 데이터 클래스의 총 수 (Negative)보다 훨씬 적은 것

왜 문제가 될까?

- ✓ 대부분의 기계 학습 알고리즘 및 각 클래스의 인스턴스 수가 대략 같을 때 가장 잘 작동
- ✓ 한 클래스의 인스턴스 수가 다른 인스턴스의 인스턴스 수를 훨씬 초과하면 문제가 발생

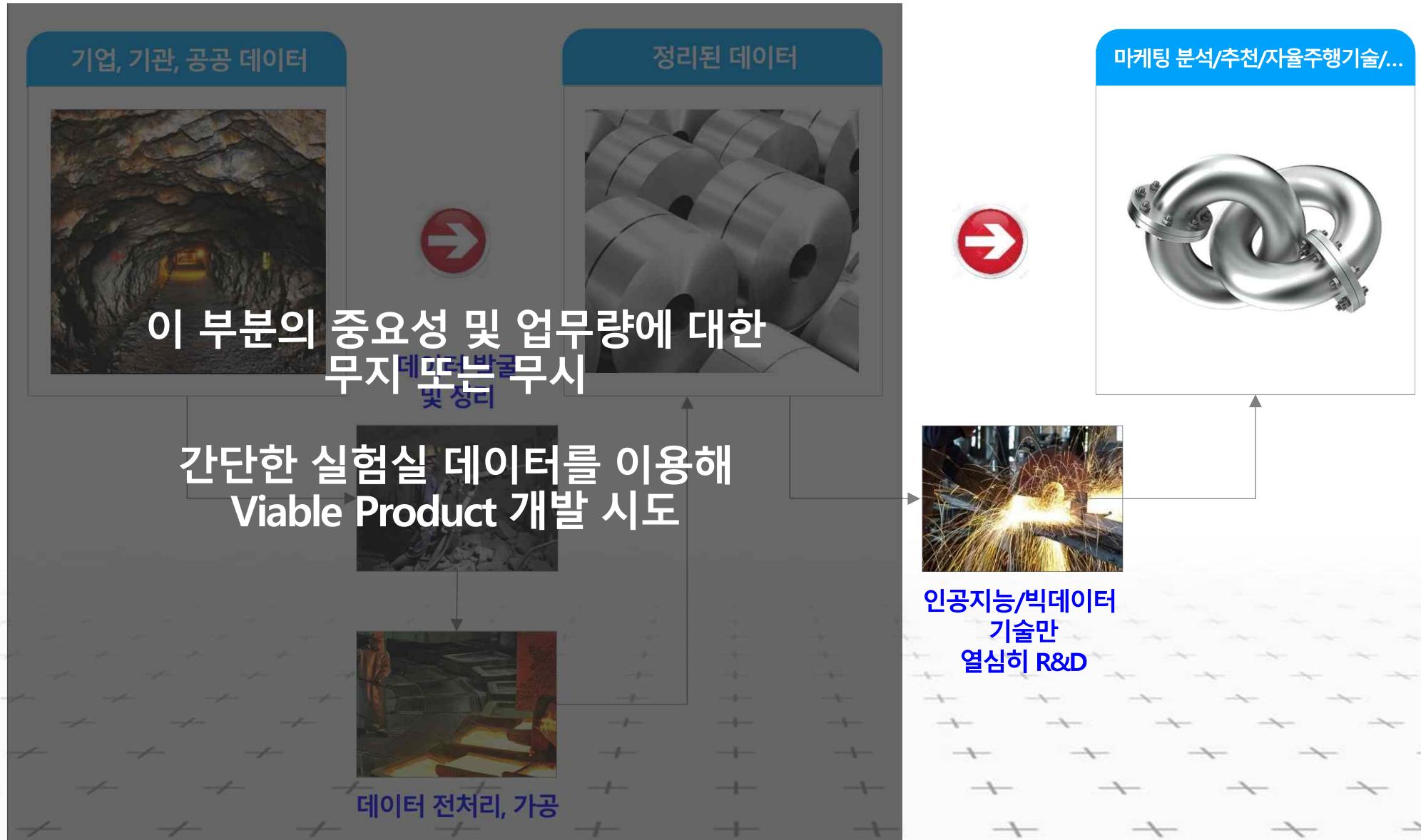
예제: 금융거래 이상탐지

- ✓ **사기거래의 비용** >> 정상적인거래의 비용

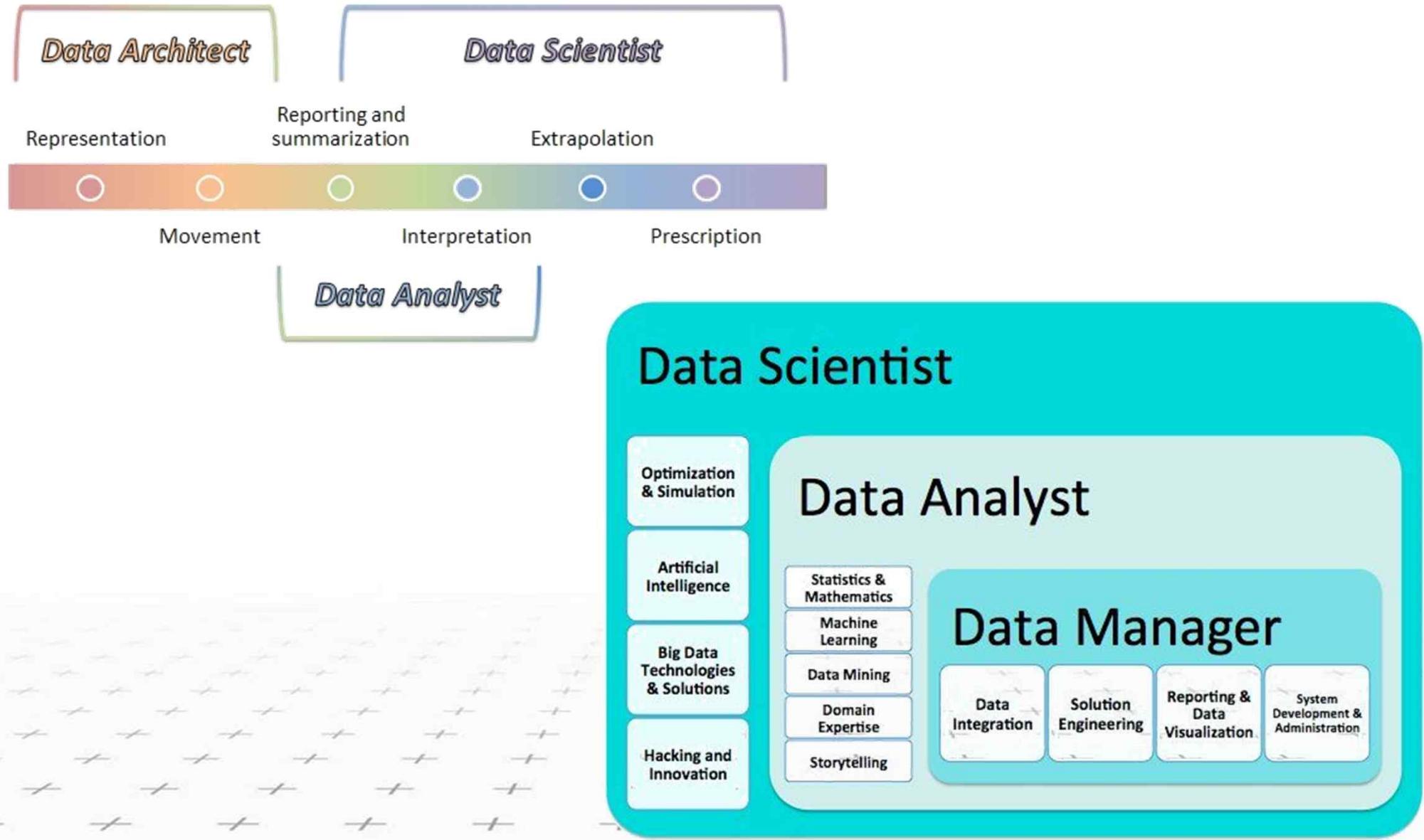


인공지능, 빅데이터 = 돈?

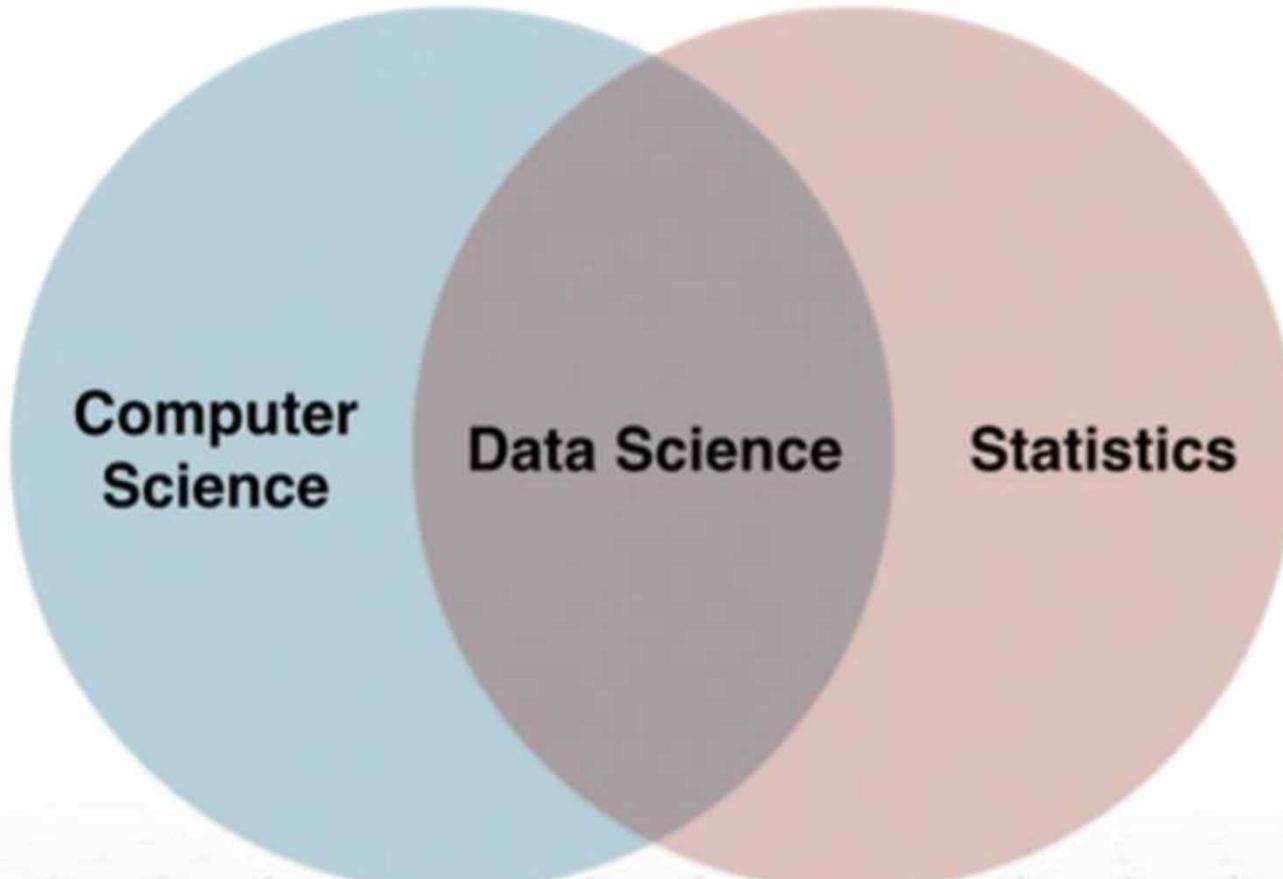
단계별 넘어야 할 고개 - 분석 및 예측



Data Scientist



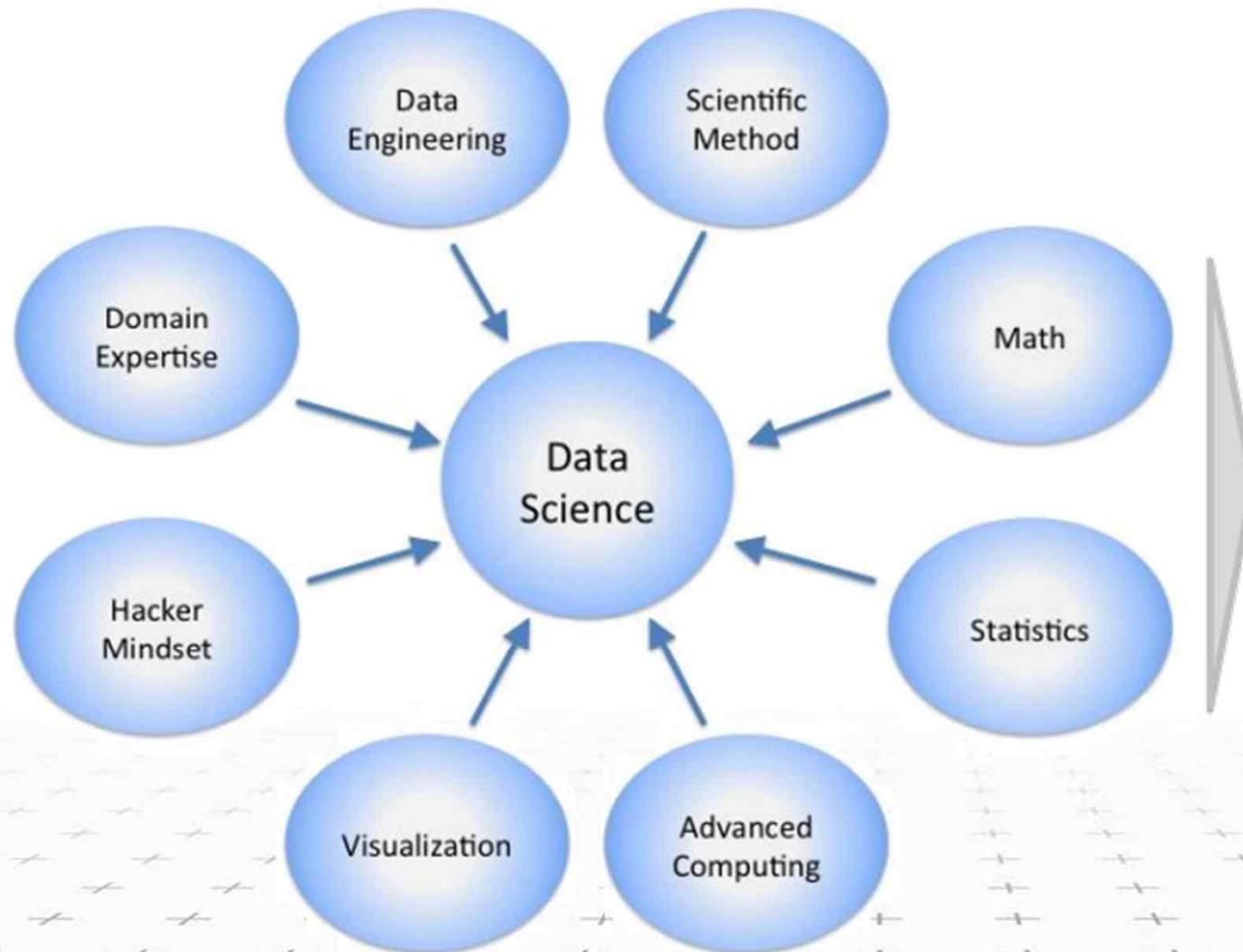
누구나 빅데이터 분석가가 될 수 있다?



Modeling - Statistics, Machine Learning
Experiments - Statistics
Coding - Computer Science
Quantitative problem solving - Math, Physics, Applied Math
Dealing with large datasets - Experimental physics, Astronomy, Bioinformatics, etc.
Using data to understand people - Social science (with strong mathematical / computational bent) including Economics, Psychology, Political Science, etc.

<https://www.quora.com/What-classes-should-I-take-if-I-want-to-become-a-data-scientist>

누구나 빅데이터 분석가가 될 수 있다?



한 사람이 이 모든 영역을 커버하는 것은 사실상 불가능

보통 1~2분야에 전문가이면서, 또 다른 2~3분야에서 능숙한 경우가 대부분 5~6분야에 능숙한 경우도 드문 편

누구나 빅데이터 분석가가 될 수 있다?

MODERN DATA SCIENTIST

Data Scientist, the sexiest job of the 21th century, requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.

MATH & STATISTICS

- ★ Machine learning
- ★ Statistical modeling
- ★ Experiment design
- ★ Bayesian inference
- ★ Supervised learning: decision trees, random forests, logistic regression
- ★ Unsupervised learning: clustering, dimensionality reduction
- ★ Optimization: gradient descent and variants

DOMAIN KNOWLEDGE & SOFT SKILLS

- ★ Passionate about the business
- ★ Curious about data
- ★ Influence without authority
- ★ Hacker mindset
- ★ Problem solver
- ★ Strategic, proactive, creative, innovative and collaborative



PROGRAMMING & DATABASE

- ★ Computer science fundamentals
- ★ Scripting language e.g. Python
- ★ Statistical computing packages, e.g. R
- ★ Databases: SQL and NoSQL
- ★ Relational algebra
- ★ Parallel databases and parallel query processing
- ★ MapReduce concepts
- ★ Hadoop and Hive/Pig
- ★ Custom reducers
- ★ Experience with xaaS like AWS

COMMUNICATION & VISUALIZATION

- ★ Able to engage with senior management
- ★ Story telling skills
- ★ Translate data-driven insights into decisions and actions
- ★ Visual art design
- ★ R packages like ggplot or lattice
- ★ Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau

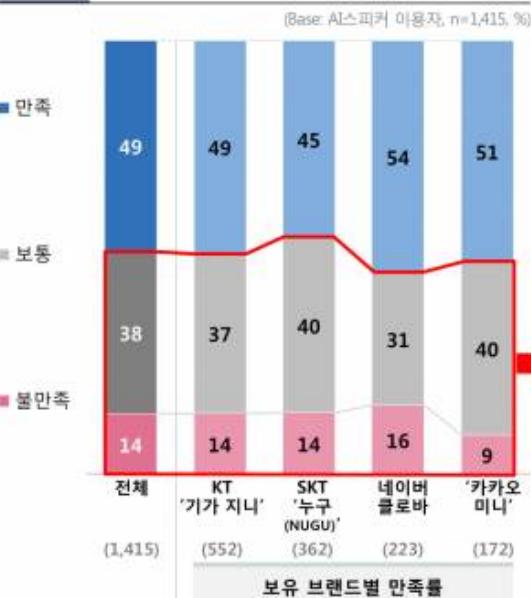
인공지능의 현실성



<출처: FACEBOOK>

[그림2] 인공지능(AI) 스피커 만족도

[그림2-1] 만족도



[그림2-2] 불만족 이유

(Base: AI스피커 이용-불만족자, n=728, 복수응답, %)



* 기타(5%) 제외 안함

Q 현재 이용하고 있는 인공지능(AI)스피커에 얼마나 만족하십니까?

Q 현재 이용하고 있는 인공지능(AI)스피커가 불만족스러운 이유는 무엇입니까? 해당되는 것 모두 선택해 주십시오.

"단순해서 재미 없어요".AI 스피커 사용자 만족도 기대 이하

<http://v.media.daum.net/v/20180710162905818>

인공지능의 현실성

“1950년대에 탄생한 인공지능은 그동안 수차례에 걸쳐 영욕을 겪었다. 이 마법 같은 기술에 뜨거운 기대와 투자가 집중됐다가 어느 날 갑자기 썰물처럼 빠져나가는 ‘인공지능의 겨울(AI Winter)’이 반복되어온 것이다.”

“**양 르쿤 뉴욕 대학 교수(페이스북 수석 AI 엔지니어)**는 ‘인공지능(AI)은 아직 초보적인 수준이어서 영화 <터미네이터>처럼 인간을 공격하는 상황은 발생하지 않는다’라고 단언했다.”

“프랑스 출신인 르쿤 교수는 1980년대 말부터, 컴퓨터에 인간의 두뇌를 모방한 가상 신경망을 심어 복잡한 연산을 수행토록 하는 딥러닝(deep learning·심층 신경망) 연구에 몰두했다. 그 성과가 바로 ‘콘볼루션 신경망(convolutional neural network)’이다.”

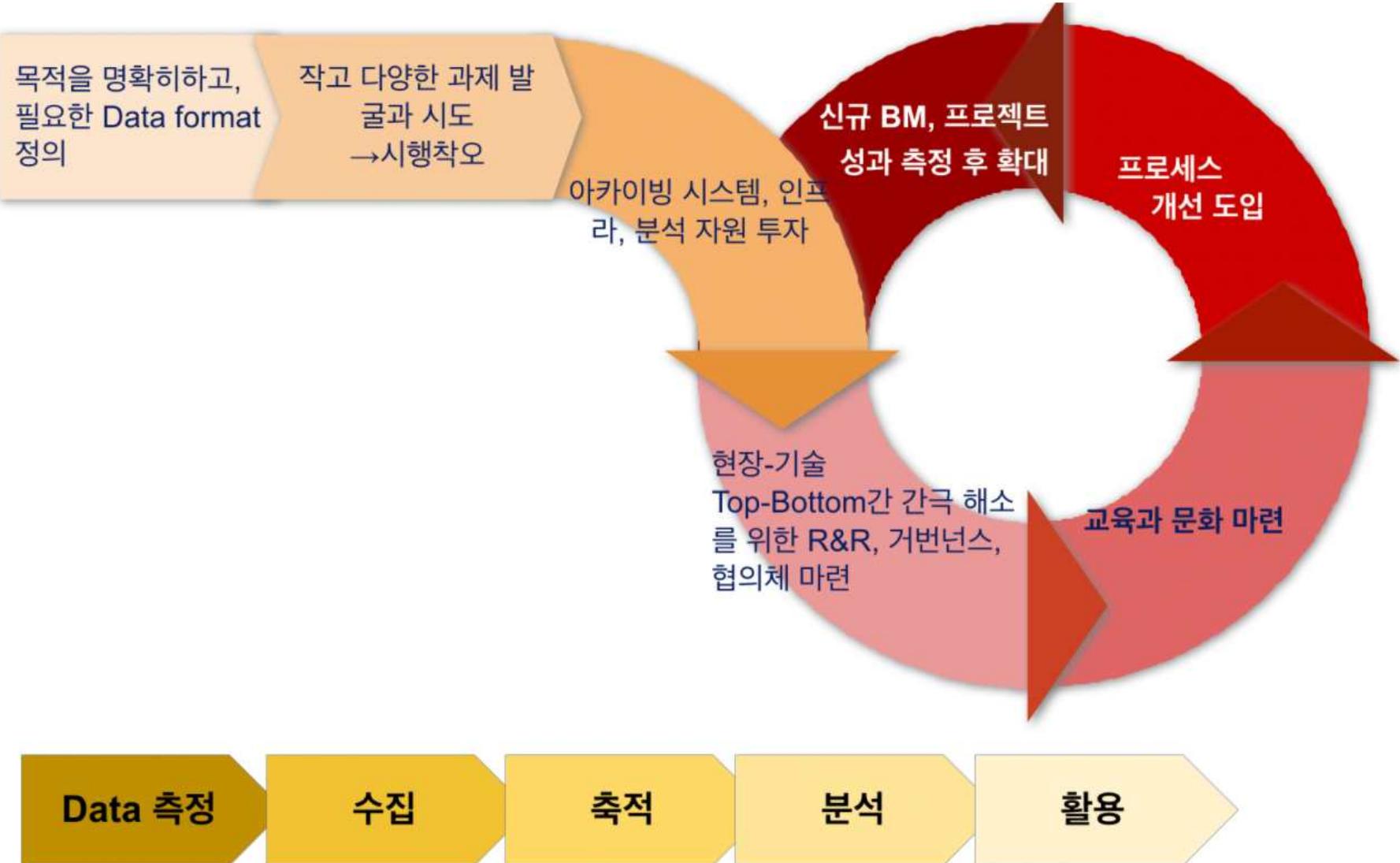
<출처><http://www.sisain.co.kr/?mod=news&act=articleView&idxno=32487>



이슈 1 – 데이터/인공지능 기술자의 부족



이슈 2 – 가공된 공공, 개인정보 기반 데이터가 전부가 아님



이슈 3 – 자체 보유 데이터를 이용한 가치 창출



이슈 4 – Fast Data – IT 기술과의 접목 필요

데이터 분석 전문가 역할은 기업 및 기관 내 시스템에서 일부

벌써 7년 전 일이다.

타부서 직원 3~4명이 수개월간 써름하며 작업하던 엑셀파일이 있었다.
엑셀파일 주제에 용량이 자그마치 200MB를 넘었었다. 20만행, 100열 정도되는 데이터였다.

여러 소스를 활용해 수 작업을 했던거라 오류가 많았지만, 어디서 오류가 있는지, 결합하면 통계치가 어떻게 되는지를 알아야 했는데 당췌 몇개월을 작업해도 진전이 없었다.

당시에 전혀 다른 부서에서 일하는 주니어였던 나에게
SOS가 왔었다.

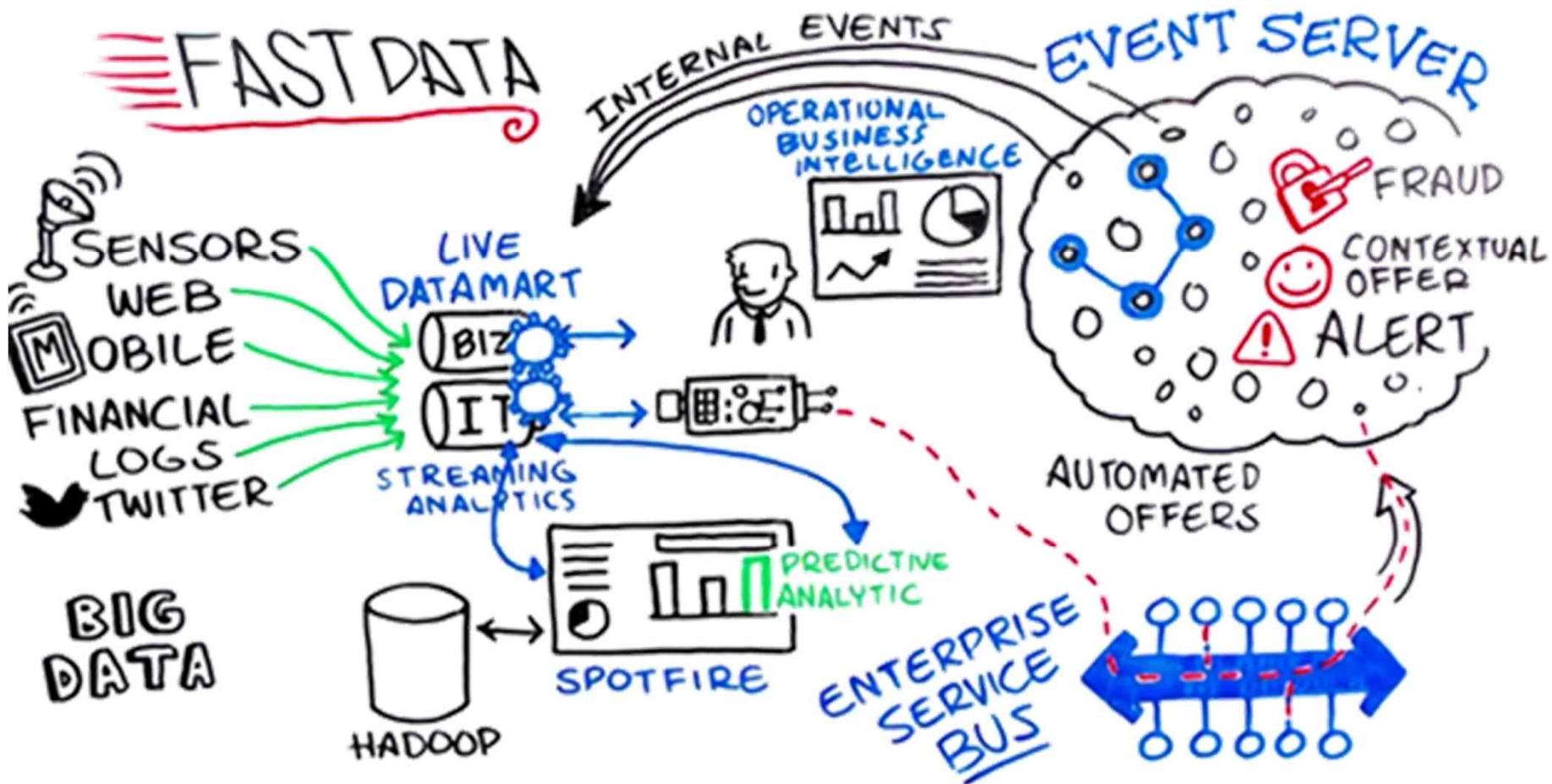
당시에 내가 엑셀 잘한다고 소문 나있었던 듯 하다. (엑셀은 잘 못하는데)
내 일도 아닌데 왜 시키나 속으로는 투덜투절 했었다.

작업은 3시간 걸려서 끝냈다.
실제 주어진 작업 시간은 1주일 이었다.

내가 한거라고는 고작, 엑셀파일 데이터를 Oracle DB에 넣어서 각종 통계함수를 사용했을 뿐이다.

어이없다고?
지금도 금융권에서는 현재 진행형들인 일이다.

이슈 4 - Fast Data System 구축



<출처: TIBCO Software>

단기 대응 방안 – 국내 모든 가용 자원 Outsourcing



인공지능, 빅데이터 활용의 장벽들

개인정보보호법 장벽

데이터 독점자

투자 대비 수익의 불투명성

내·외부 데이터의 전사적 미통합

현업의 문화, 정보, 데이터, 조직의 장벽

데이터 사이언티스트의 부재

데이터 수집 및 분석을 위한 여유

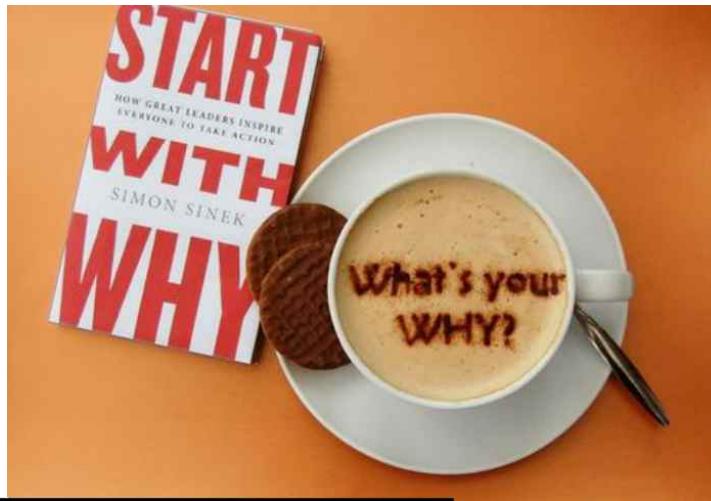
실험을 위해 현상 유지를 포기할 용기

데이터가 말하는 결론을 따를 유연성

첨언

빅데이터 – 질문을 명확히 하라

데이터에서 무엇에 대한 답을 원하는가



첨언

바람은

'목적지가 없는 배'를

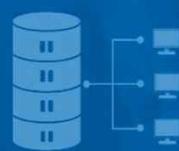
밀어주지 않는다

〈몽테뉴 Montaigne, Michel De〉





데이터 분석을 위한 준비



실습 자료

직장인을 위한
데이터 분석
실무 — with —
파이썬

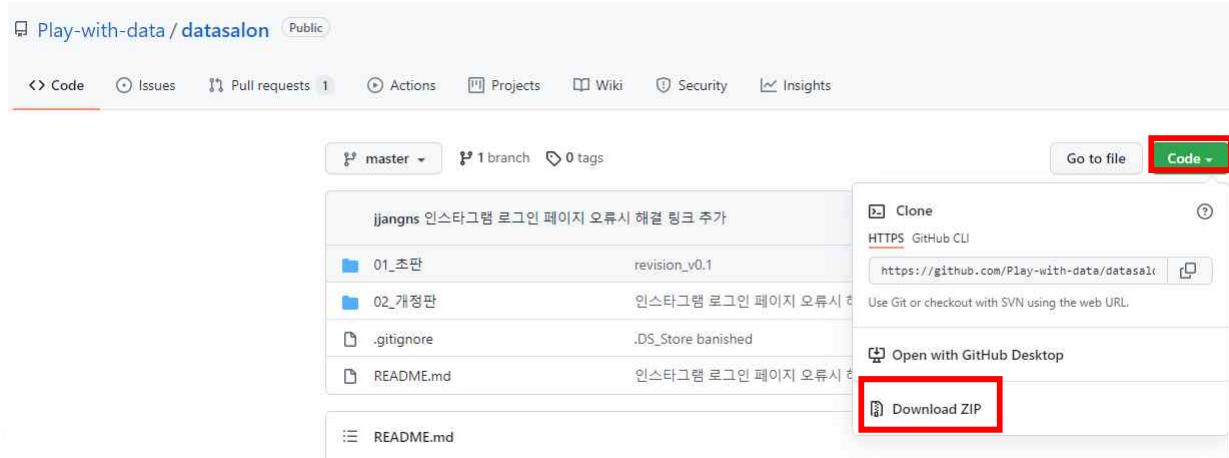


마케팅, 영업,
기획 실무
담당자를 위한
데이터 분석의 기술

이형석, 장남수,
전성환, 정상욱 저음

DS 데이터 사이언스 시리즈
실습자료

<https://github.com/shinbyungjoo/Python-Lecture>



The screenshot shows a GitHub repository page for 'Play-with-data / datasalon' (Public). The master branch has 1 branch and 0 tags. The repository contains the following files:

File	Description
01_초판	revision_v0.1
02_개정판	인스타그램 로그인 페이지 오류시 해결 링크 추가
.gitignore	.DS_Store banned
README.md	인스타그램 로그인 페이지 오류시 해결 링크 추가
README.md	(This row is listed twice)

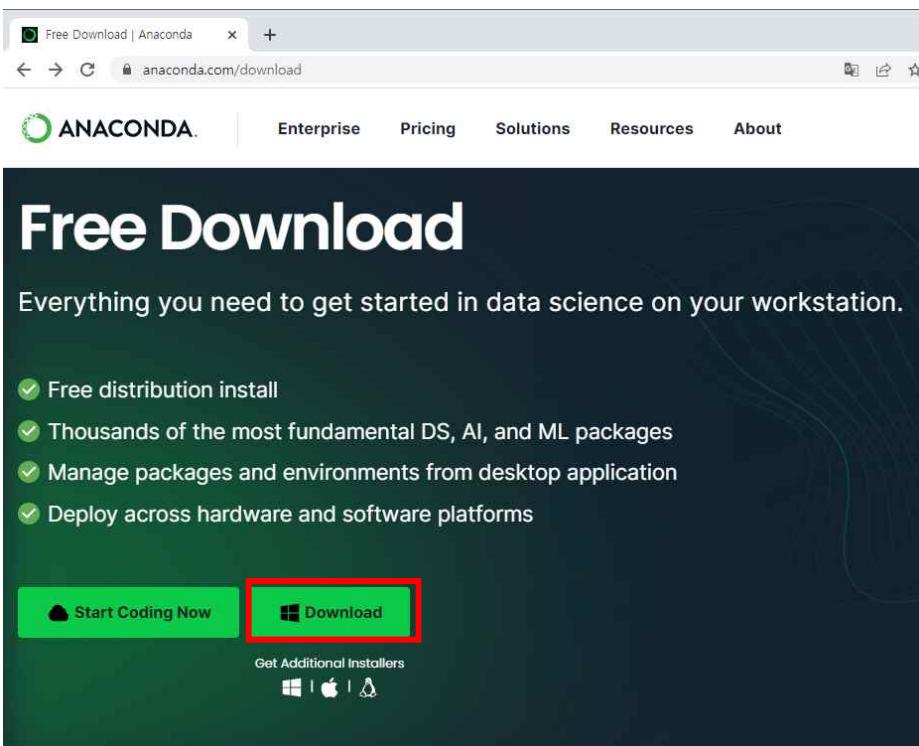
On the right side, there is a sidebar with options like 'Clone' (GitHub CLI), 'Download ZIP' (highlighted with a red box), and 'Open with GitHub Desktop'.

파이썬 준비

◆ 아나콘다 내려받기

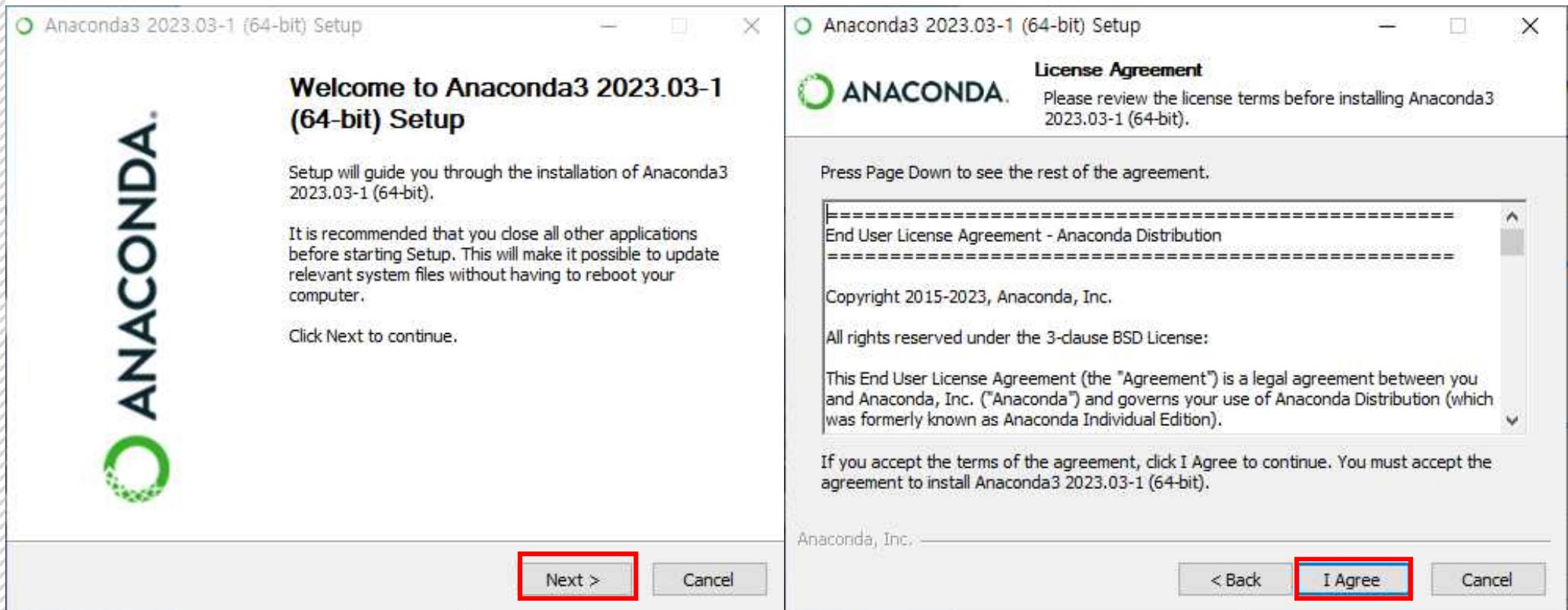
❖ 아나콘다(Anaconda)

- 파이썬 언어와 주요 라이브러리, 파이썬을 입력하고 실행하기에 편리한 몇 가지 툴들을 패키지로 제공
- 다운로드 전 확인사항
 - ✓ 파이썬 버전: 현재 파이썬 버전 3.11
 - ✓ 현재 사용 중인 운영체제의 아키텍처
- 다운로드: <https://www.anaconda.com/download>



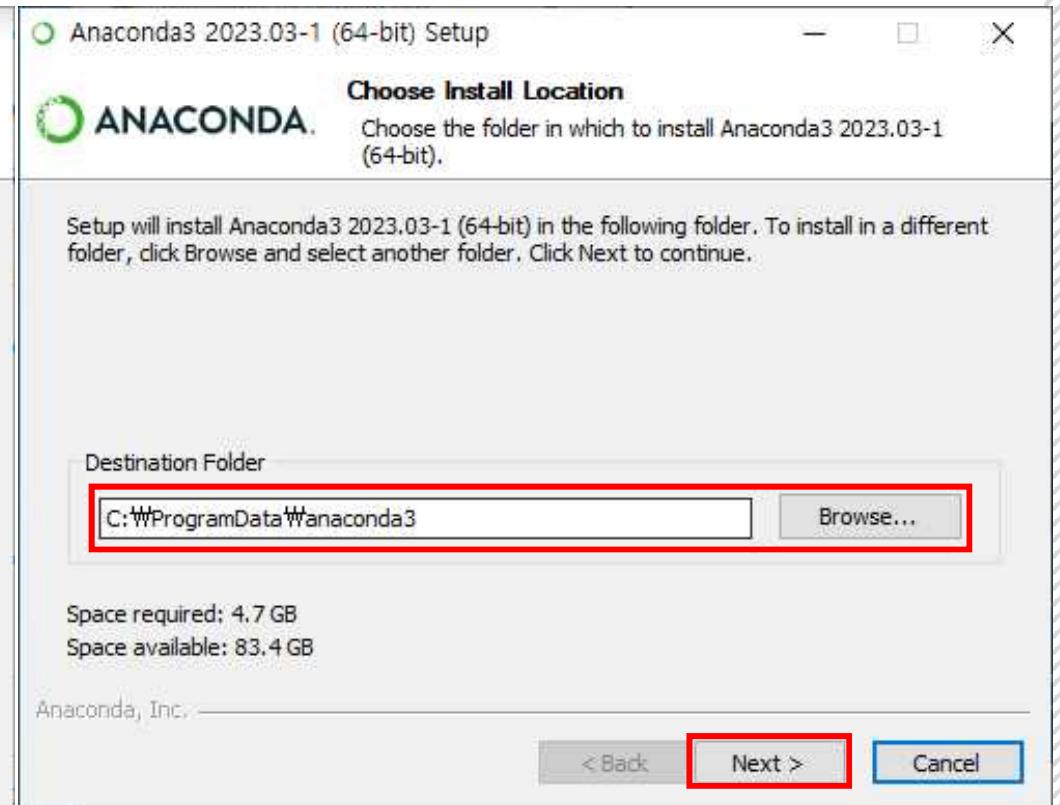
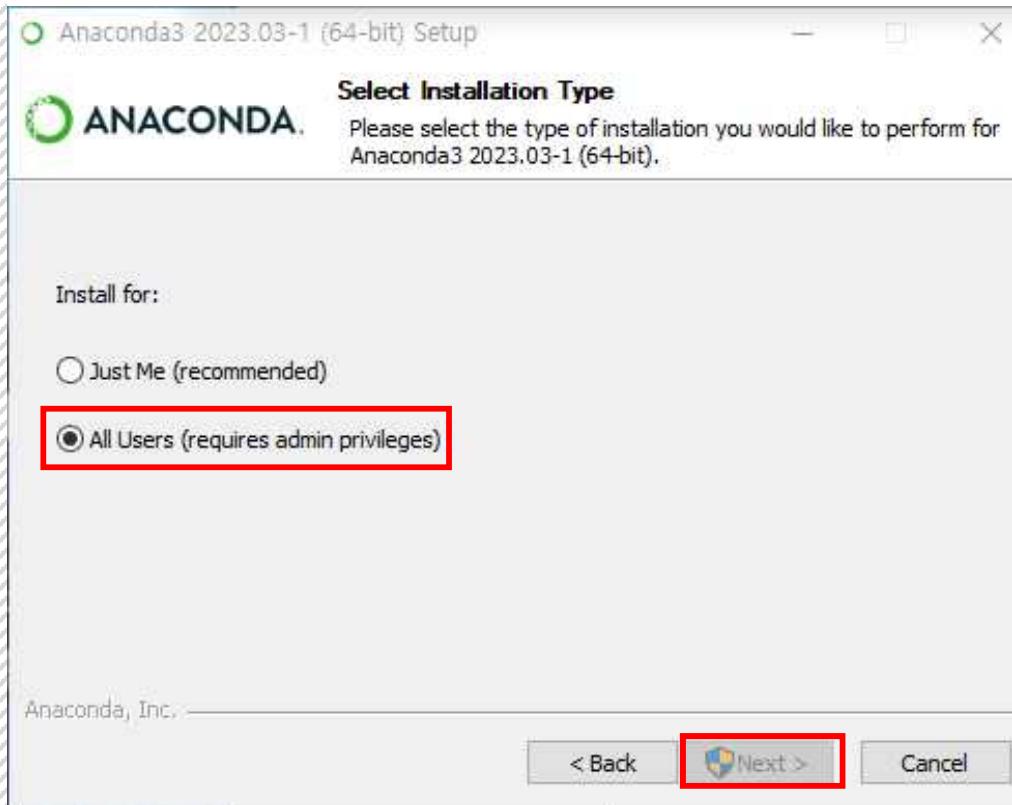
파이썬 준비

◆ 아나콘다 설치



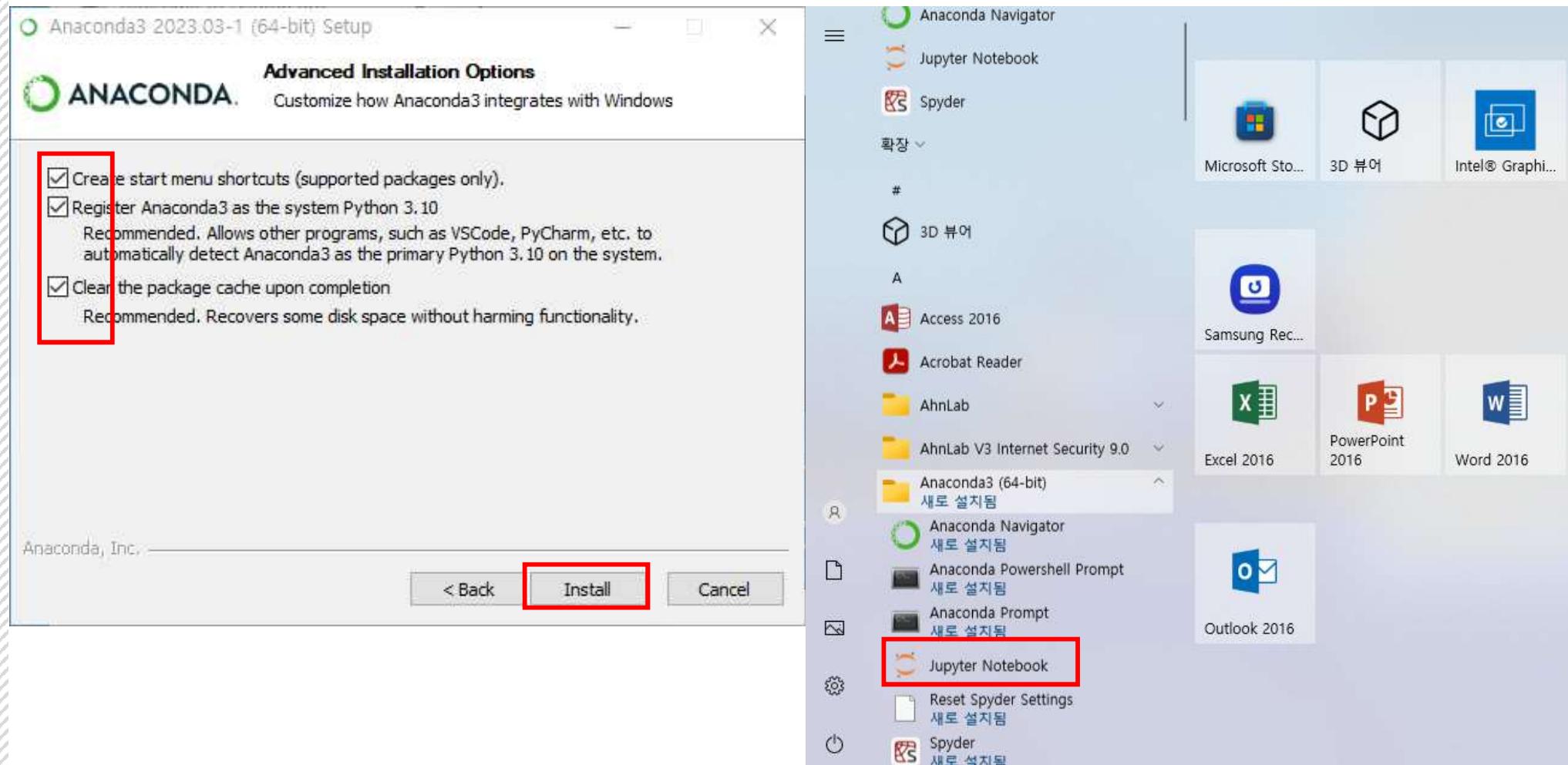
파이썬 준비

◆ 아나콘다 설치



파이썬 준비

◆ 아나콘다 설치



파이썬 준비

◆ 주피터 노트북 준비

❖ 주피터 노트북(Jupyter Notebook)

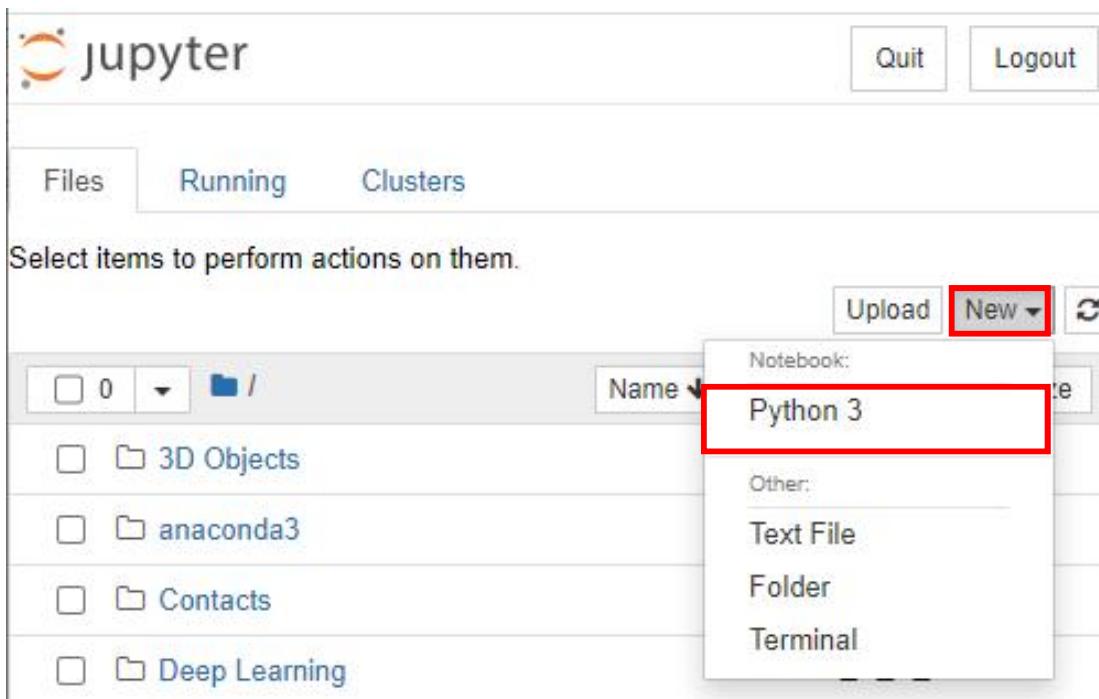
- 웹브라우저 내에서 파이썬 코드를 입력하고 실행 결과를 확인할 수 있는 툴
- 코드 작성 시 중간 결과를 확인할 수 있어 분석을 진행할 때 유용
- 실행: [Windows 시작 프로그램]-[Anaconda3]-[Jupyter Notebook]
 - ✓ 콘솔 창과 함께 웹 브라우저(localhost:8888/tree)를 통해 접속
 - ✓ 웹 브라우저로는 호환성이 좋은 구글 크롬 사용 권장



파이썬 준비

◆ 주피터 노트북 시작하기

- ❖ 주피터 노트북에서 새 파일 만들기
 - 주피터 노트북 우측 상단의 [New] 버튼 클릭
 - [Python 3] 선택

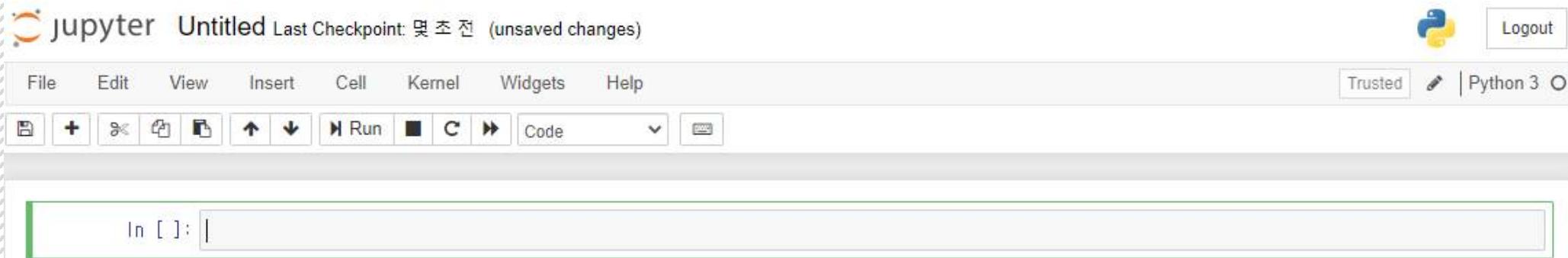


파이썬 준비

◆ 주피터 노트북 사용하기

❖ 주피터 노트북 실행 화면

- 긴 회색 네모 한 칸을 셀(Cell)이라고 부르며, 셀 안에 코드를 입력하고 그 셀을 실행하면 결과가 아래에 출력



❖ 주피터 노트북 단축키 및 팁

명령	단축키	설명	명령	단축키	설명
코드 실행	Shift+Enter	현재 셀 실행(실행 후 아래 셀로 이동)	셀 추가	A	현재 위치 위(above)에 셀 추가
	Ctrl+Enter	현재 셀 실행(실행 후 이동하지 않음)		B	현재 위치 아래(below)에 셀 추가
셀 삭제	DD	현재 셀 삭제(delete)	셀 복사	C	현재 셀 복사
셀 잘라내기	X	현재 셀 잘라내기	셀 붙여넣기	V	복사하거나 다른 셀 붙여넣기
셀 분할	Ctrl+Shift+-	현재 위치한 라인을 기준으로 셀 분할	셀 병합	Shift+M	선택한 복수의 셀을 하나의 셀로 병합
작업 취소	Z	셀 삭제/붙여넣기/분할/병합 등 셀 편집 작업 취소	코드 라인 보이기		메뉴바-[View]-[Toggle Line Numbers]

파이썬 맛보기

◆ 파이썬 코드 입력 및 실행

❖ 셀 단위 코드 입력/실행

- 주피터 노트북에서 파이썬 코드 실행: Ctrl+Enter 또는 재생버튼( Run) 클릭
- 실행 결과는 셀 아래에서 바로 확인

In [1]:

```
# 예제 1-1 셀단위 코드 입력/실행/주석
print("안녕하세요")
```

안녕하세요

❖ 주석

- 간략한 코드 설명 등 메모 작성할 때 사용하며 실행되지 않음
- 코드 내에서 # 뒤에 있는 부분은 주석으로 인식
- 라인 전체를 주석으로 처리할 때는 한 줄의 코드 라인을 클릭하거나 여러 줄의 코드 라인을 드래그해서 선택한 후 Ctrl+/를 입력
- 주석 처리된 라인을 선택 후 Ctrl+/를 재입력하면 주석 처리 해제

In [2]:

```
# 예제 1-2 셀단위 코드 입력/실행/주석
print("Hi")
# 주석: # 뒤에 나온 부분은 주석으로 실행되지 않습니다.
```

Hi

파이썬 맛보기

◆ 값 입력 및 출력

❖ 변수 및 연산

- 변수를 사용하여 값을 저장하고 활용
- 형태: 변수명=값
- 변수명은 파이썬에서 사용 중인 명령어 사용 불가
- 대/소문자 구분
- 변수에 값 저장 및 사칙연산
 - ✓ / 연산자: 나누기 결과(실수)
 - ✓ // 연산자: 나누기의 몫(정수)
 - ✓ % 연산자: 나누기의 나머지(정수)

In [3]:

```
# 예제 1-3 변수 입력 및 화면 출력
a = 9
b = 5
```

In [4]:

```
# 예제 1-4 사칙연산 화면에 출력하기
print( a + b )    # 더하기
print( a - b )    # 빼기
print( a * b )    # 곱하기
print( a / b )    # 나누---기
print( a // b )   # --나눈 몫
print( a % b )    # 나눈 나머지
```

14
4
45
1.8
1
4

파이썬 맛보기

◆ 값 입력 및 출력

❖ 값 출력

➤ 한 줄에 여러 개의 값 출력

- ✓ print(a,b,c) 형태로 변수 혹은 값을 구분하기 위해 콤마(,) 사용

In [5]:

```
# 예제 1-5 변수 여러 개 출력하기
print('파이썬', '데이터분석입문', 'PlaywithData','데이터수집')
```

파이썬 데이터분석입문 PlaywithData 데이터수집

➤ 값 사이의 구분 기호 사용

- ✓ print() 함수의 매개변수인 sep 사용
- ✓ 형식: print(a, b, sep = '구분기호')
- ✓ 매개변수 sep의 값을 지정하지 않으면 기본값으로 띄어쓰기 한 칸 설정

In [6]:

```
# 예제 1-6 변수 여러 개 출력하기
print('귤','사과','배',sep = ' / ')
```

귤 / 사과 / 배

파이썬 맛보기

◆ 리스트(List)

- ❖ 변수는 하나의 값 혹은 다른 변수의 값을 가질 수 있음
- ❖ 여러 개의 값이나 변수를 가지기 위해서는 집합으로 지정
- ❖ 집합의 종류에는 리스트(List), 딕셔너리(Dictionary), 튜플(Tuple) 등이 있음
- ❖ 리스트
 - 여러 개의 변수, 값을 묶어서 집합으로 관리하는 자료 형태
 - 형식: 리스트명 = [원소1, 원소2, 원소3, ...]
 - 리스트로 정의된 변수의 이름을 사용하면 리스트 내 모든 원소 지정 가능

```
In [7]: # 예제 1-7 리스트 자료 입력
print('1번', '2번', '3번', '4번', '5번', '6번', '7번', '8번', '9번', '10번')
class_2_1= ['1번', '2번', '3번', '4번', '5번', '6번', '7번', '8번', '9번', '10번']
print(class_2_1)
```

1번 2번 3번 4번 5번 6번 7번 8번 9번 10번
['1번', '2번', '3번', '4번', '5번', '6번', '7번', '8번', '9번', '10번']

파이썬 맛보기

◆ 리스트(List)

- ❖ 리스트 내에서 하나의 원소를 선택할 때는 인덱스를 이용하여 몇 번째 원소인지 지정해서 선택 가능
- ❖ 형식: 리스트변수이름[인덱스 번호]
- ❖ 리스트의 첫 번째 원소에 해당하는 인덱스 번호는 0이며 순서대로 1씩 증가
- ❖ 리스트의 마지막 원소를 기준으로 인덱스 번호 표현 가능(마지막 원소가 -1이며, 역순으로 -1씩 증가)

```
In [8]: # 예제 1-8 리스트 자료 내 원소 1개 선택하기: 인덱스 번호 활용
k    = ['a','b','c','d','e']
#index  0   1   2   3   4
#index -5  -4  -3  -2  -1

print( k[1] )
```

b

파이썬 맛보기

◆ 리스트(List)

❖ 슬라이싱(Slicing)

- 리스트에서 여러 원소를 구간으로 선택할 경우 활용
- 형식: 리스트변수이름[시작 인덱스 번호 : 마지막 인덱스 번호]
- 주의: 콜론(:) 뒷부분에 적히는 인덱스 번호는 포함하지 않고 바로 앞의 인덱스 번호 까지만 포함
- 인덱스 번호는 왼쪽부터 셀 수 있고, 오른쪽부터도 셀 수 있음

```
In [9]: # 예제 1-9 리스트 자료 구간별 글자 선택하기: 슬라이싱
# list [인덱스시작번호 : 끝 다음 번호]
k      = ['a', 'b', 'c', 'd', 'e']
#index   0   1   2   3   4
#index -5  -4  -3  -2  -1
print( k[ 1 : 3 ] )
print( k[ -4 : -2 ] )
```

['b', 'c']

['b', 'c']

리스트 k의 인덱스 번호 3번 직전(인덱스 번호 2)까지만 선택해서 화면 출력

리스트 k의 인덱스 번호 -4(오른쪽 끝에서부터 4번째 자리의 원소)부터 인덱스 번호 -2 직전(인덱스 번호 -3)까지만 선택해서 화면 출력

파이썬 맛보기

◆ 리스트(List)

❖ 병합(Merge)

- 두 개 이상의 리스트를 합쳐서 하나의 리스트로 만들 수 있음
- 방법 1
 - ✓ 형식: 리스트A + 리스트B
 - ✓ 리스트A의 원소 이후에 리스트B의 원소가 차례대로 나열된 리스트 생성
 - ✓ 리스트 순서에 따라 결과가 달라짐

- 방법 2
 - ✓ 형식: 리스트A.append(리스트B)
 - ✓ 리스트A의 우측 끝 원소 자리에 리스트B 전체를 마지막 원소로 추가
 - ✓ 리스트A 내에 리스트B가 하나의 원소로 들어가게 되며, 리스트A의 오른쪽 끝 원소인 리스트A[-1]은 리스트B가 됨

In [10]:

```
# 예제 1-10 리스트 합치기: 리스트1 + 리스트2
# A + B
l1 = ['a']
l2 = ['b', 'c', 'e']
print(l1 + l2)
print(l2 + l1)
```

```
['a', 'b', 'c', 'e']
['b', 'c', 'e', 'a']
```

In [11]:

```
# 예제 1-11 리스트 추가하기(하나의 원소로 블록하기): 리스트1.append(리스트2)
l1 = ['a']
l2 = ['b', 'c', 'e']
l1.append(l2)
print(l1)
```

```
['a', ['b', 'c', 'e']]
```

파이썬 맛보기

◆ 반복문

- ❖ 동일한 코드를 여러 번 반복해서 실행

```
In [12]:
```

```
# 예제 1-12 반복문 활용하기
fruits = ['바나나', '사과', '딸기', '배', '감']
for fruit in fruits:
    print(fruit)
```

```
바나나
사과
딸기
배
감
```

fruits 리스트에서 원소를 순서대로 한
나씩 꺼내어 fruit 변수에 저장

반복문을 사용할 때는 들여쓰기(탭 또
는 공백 4칸으로 띄어쓰기)를 사용해
야 하며, 들여쓰기가 사용되지 않은 곳
부터는 반복이 일어나지 않음

파이썬 맛보기

◆ 문자열

- ❖ 문자로 이루어진 자료의 형태
- ❖ 문자열을 나타내려면 시작과 끝부분에 작은따옴표('') 혹은 큰따옴표(" ")를 붙여야 함
- ❖ 문자열 입출력

In [13]:

```
# 예제 1-13 문자열 입/출력
s = '문자'
print(s)
```

문자

❖ 문자열 합치기

- 두 개 이상의 문자열 + 기호를 사용해 문자열을 합침
- 왼쪽 문자열부터 순서대로 합쳐지므로 순서를 구분해서 입력해야 함
- 형식: 문자열1 + 문자열2

In [14]:

```
# 예제 1-14 문자열 합치기(+)
a = "사과"
b = "망고, 파인애플"
ab = a + b
print(ab)
```

사과망고, 파인애플

파이썬 맛보기

◆ 문자열

❖ 문자열 반복

- 곱하기(*)를 활용해 문자열을 여러 번 더해서 반복된 문자열 생성

In [15]:

```
# 예제 1-15 문자열 반복하기(*)
z = "*"
print( z * 10)
```

❖ 문자열 포매팅(Formatting)

- 문자열을 고정된 하나의 값으로 사용하지 않고, 변수를 활용해 변수의 값에 따라 문자열이 변하도록 만들 수 있음
- 형식: “문자열 { }”.format(변수)
- 변수를 여러 개 사용할 경우 순서에 맞춰 차례대로 변수를 입력하면 됨

In [16]:

```
# 예제 1-16 문자열 포매팅
month = "2월"
day = "20일"
say = "{} {}은 일요일입니다".format(month, day)
print(say)
```

2월 20일은 일요일입니다

파이썬 맛보기

◆ 문자열

❖ 문자열 인덱스, 슬라이싱

- 문자열 중 일부 문자만 선택
- 리스트에서 사용했던 인덱스 번호와 슬라이싱을 문자열에서도 동일하게 사용
- 공백이나 문장 부호 등도 하나의 글자로 인식

```
In [17]:  
  
# 예제 1-17 문자열 일부 선택하기: 인덱스, 슬라이싱  
k      = "가나다라마바"  
# index      0 1 2 3 4 5  
# index      -6-5-4-3-2-1  
print(k[2])  
print(k[-1])  
print(k[2:5])  
print(k[3:])
```

```
다  
바  
다라마  
라마바
```

파이썬 맛보기

◆ 문자열

❖ 문자열 처리 함수

- 문자열 시작과 끝부분의 공백을 제거하거나 글자 변경, 나누기 등이 필요
- 웹 크롤링을 통해 데이터를 수집했거나 텍스트 분석을 진행할 경우에는 데이터 분석이 용이하도록 문자열 전처리 필요
- 문자열 처리 함수 활용

In [18]:

```
# 예제 t-18 문자열 정리하기: strip(), replace(a,b), split()
t1 = " hp010-0000-0000 "
print(t1)
t2 = t1.strip()
print(t2)
t3 = t2.replace('hp', ' ')
print(t3)
t4 = t3.split('-')
print(t4)
```

```
hp010-0000-0000
hp010-0000-0000
010-0000-0000
['010', '0000', '0000']
```

t1 문자열의 시작과 끝부분에 있는 공백문자(띄어쓰기, 줄 바꿈, 들여쓰기 등)을 모두 없앤 뒤 변수 t2에 저장

t2에 있는 'hp'라는 문자를 모두 ''로 변경하여 변수 t3에 저장. ''를 사용하게 되면 'hp'를 제거하라는 의미와 동일

'-' 기준으로 문자열을 분할. 분할된 문자열은 리스트로 생성

파이썬 맛보기

◆ 조건문

- ❖ 특정한 조건을 만족할 경우에만 명령문이 수행되도록 조건문을 사용
- ❖ 형식

```

if 조건1:
    조건1 만족 시 실행할 명령문1
elif 조건2:
    조건1은 불만족하지만 조건2 만족 시 실행할 명령문2
else:
    모두 불만족 시 실행할 명령문3
  
```

들여쓰기가 어이지
는 범위까지만 실행

In [19]:

```

# 예제 1-19 if 조건문 사용하기
price = 5000
if price > 3000:
    print('비싸다')
else:
    print('저렴하다')
  
```

비싸다

price가 3000보다 크다면 그 아래에 들여쓰기한 부분을 실행. price 값인 5000은 3000보다 크므로 이 조건은 참이 되어 '비싸다'가 출력됨

만약 1번째 줄에서 'price = 2000'으로 입력한 경우에는 4번째 줄인 else 부분으로 넘어가서 '저렴하다'가 출력됨

파이썬 맛보기

◆ 함수

- ❖ 동일한 코드를 여러 번 사용할 경우 함수로 지정해서 활용
- ❖ 형식

def 함수명(매개변수):
 실행부분

함수명과 함수 내에서 사용할 매개변수 정의

콜론(:) 아래에 들여쓰기로 사용해 함수가 실행할 내용 입력

```
In [20]:  
# 예제 1-20 함수 예제: 자기 소개  
def say_name(name):  
    print('안녕하세요')  
    print('제 이름은' + name + '입니다')  
  
say_name( '홍길동' )
```

이름을 입력할 경우 인사말과 자기 소개 출력하는 함수

‘홍길동’을 입력값으로 하는 say_name 함수 호출

안녕하세요
제 이름은홍길동입니다

파이썬 맛보기

◆ 함수

- ❖ 함수는 코드를 실행하고, 특정 결과를 함수를 호출한 곳으로 돌려줄 수도 있음
- ❖ 어떤 결과를 돌려받을지 정의하려면 함수를 작성할 때 return문으로 지정
- ❖ 형식

```
def 함수명( 매개변수 ):  
    실행부분  
    return 결과
```

In [21]:

```
# 예제 1.21 지수 곱 함수만들기  
def calcul(a,b):  
    result = a**b  
    return result  
  
test = calcul(10,3)  
print(test)
```

1000

두 수 a, b를 입력할 경우 a의 b승을 계산해서 함수를 호출한 곳으로 결과를 반환

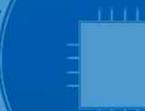
calcul(a,b) 함수의 결과를 test변수에 저장하고 출력



4.



데이터 분석 기초



pandas 기초

◆ pandas란?

- ❖ 엑셀(Excel)의 스프레드시트(spreadsheet) 같이 행(row)과 컬럼(column)으로 구성된 테이블 형태의 데이터를 쉽게 다룰 수 있는 파이썬 라이브러리
- ❖ import pandas
 - 파이썬 라이브러리를 사용하기 위해서는 import문으로 라이브러리를 불러와야 함
 - 아나콘다는 데이터 분석에 활용되는 많은 라이브러리가 파이썬과 함께 설치함으로 pandas 역시 바로 사용 가능

```
In [1]:
```

```
# 예제 2-1 pandas 라이브러리 불러오기
import pandas as pd
```

- pandas 내부의 함수들을 파이썬에서 사용할 수 있음
- pandas.함수명() 형태로 사용
- as pd를 붙여서 pd라는 별명(alias)을 지정하면 pd.함수명()으로 사용 가능

pandas 기초

- ◆ 데이터 불러오기(read_excel)
- ❖ 예제 데이터(sample_1.xlsx)

	A	B	C	D
1	2019년 11월 입국객 수			
2	국적코드	성별	입국객수	전년동기
3	A01	남성	106320	85815
4	A01	여성	191436	125241
5	A31	남성	319	299
6	A31	여성	42	54
7	A18	남성	158912	124486
8	A18	여성	232943	163466
9	총 합계		689972	
10	전년동기		499361	

1행: 해당 데이터에 대한 제목으로 데이터 분석에는 불필요한 정보

2행: 4개의 컬럼명. 예제 데이터 분석에서는 국적코드, 성별, 입국객수 컬럼 사용 예정

3-8행, A-C열: 예제 데이터 분석에서 분석할 데이터

9-10행: 총합계와 전년동기라는 요약 정보로써 데이터 분석에는 불필요한 정보

pandas 기초

◆ 데이터 불러오기(read_excel)

- 컬럼명이 있는 위치
- 파이썬에서는 시작 숫자가 1이 아닌 0이기 때문에 엑셀에서 2행을 의미

- 처음부터 3번째 행까지 보여주는 함수
- 불러온 데이터의 행이 많을 경우 파이썬에서 원하는 형태로 잘 불러왔는지 확인하기 위한 용도로 주로 사용

엑셀 파일을 불러오는 함수

- 엑셀 파일의 경로(Path)
- './'는 상대경로를 의미하는데, 파이썬 코드가 있는 폴더의 위치를 의미

In [2]:

```
# 예제 2-2 sample_1.xlsx 엑셀데이터 불러오기
sample_1 = pd.read_excel('./files/sample_1.xlsx',
header=1,
skipfooter=2,
usecols='A:C')
```

sample_1.head(3)

Out [2]:

	국적코드	성별	입국객수
0	A01	남성	106320
1	A01	여성	191436
2	A31	남성	319

데이터를 잘 불러왔는지 확인하기 위해 앞부분 뿐만 아니라 마지막 부분도 확인하기 위한 함수

마지막 행으로부터 두 행을 생략

A컬럼부터 C컬럼까지 가져옴

In [3]:

```
# 예제 2-3 tail() 함수 활용
sample_1.tail(3)
```

Out [3]:

	국적코드	성별	입국객수
3	A31	여성	42
4	A18	남성	158912
5	A18	여성	232943

pandas 기초

◆ 데이터 불러오기

- ❖ 데이터 정보 확인: info() 함수는 데이터에 대한 요약 정보 제공

pandas의 DataFrame(데이터프레임)으로 데이터 구성

In [4]:

```
# 예제 2-4 sample_1 데이터 정보 살펴보기
sample_1.info()
```

<class 'pandas.core.frame.DataFrame'>

RangedIndex: 6 entries, 0 to 5

Data columns (total 3 columns):

#	Column	Non-Null Count	Dtype
0	국적코드	6 non-null	object
1	성별	6 non-null	object
2	입국객수	6 non-null	int64

dtypes: int64(1), object(2)
memory usage: 272.0+ bytes

0-5까지, 총 6개의 행으로 구성

국적코드 컬럼은 빈 칸(non-null)이
없이 6개의 행으로 구성된 문자열 데
이터

입국객수 컬럼은 빈 칸이 없이 6개의
행으로 구성된 정수형 데이터

총 3개의 컬럼으로 구성

성별 컬럼은 빈 칸이 없이 6개의 행
으로 구성된 문자열 데이터

데이터는 정수형 데이터 1개, 문자열
데이터 2개로 구성

데이터가 총 224바이트 메모리 사용

pandas 기초

◆ 데이터 불러오기

- ❖ 데이터 기초통계량 확인: `describe()` 함수는 숫자형 데이터에 대한 여러가지 통계량 출력

```
In [5] :
# 예제 2-5 sample_1 데이터 기초통계량 확인
sample_1.describe()

Out [5] :

          입국객수
count    6.000000
mean   114995.333333
std    98105.752006
min    42.000000
25%   26819.250000
50%   132616.000000
75%   183305.000000
max   232943.000000

데이터 개수
표준편차
1사분위수
3사분위수
평균값
최소값
2사분위수(중위수)
최대값
```

pandas 기초

◆ 데이터 선택 – 컬럼 기준

❖ 데이터 확인

In [6] :

```
# 예제 2-6 sample_1 데이터 확인하기  
sample_1
```

Out [6] :

	국적코드	성별	입국객수
0	A01	남성	106320
1	A01	여성	191436
2	A31	남성	319
3	A31	여성	42
4	A18	남성	158912
5	A18	여성	232943

❖ 한 개의 컬럼 선택

In [7] :

```
# 예제 2-7 한 개 컬럼 선택하기  
sample_1['입국객수']
```

Out [7] :

```
0    106320  
1    191436  
2     319  
3      42  
4    158912  
5    232943  
Name: 입국객수, dtype: int64
```

pandas 기초

◆ 데이터 선택 – 컬럼 기준

- ❖ 여러 개의 컬럼 선택

In [8] :

```
# 예제 2-8 여러 컬럼 선택하기
sample_1[['국적코드', '입국객수']]
```

Out [8] :

	국적코드	입국객수
0	A01	106320
1	A01	191436
2	A31	319
3	A31	42
4	A18	158912
5	A18	232943

- 대괄호로 묶은 형태는 리스트를 의미함
- 여러 개의 컬럼을 선택하기 위해서는 여러 개의 컬럼을 리스트로 묶어서 입력

- ❖ 신규 컬럼 생성

In [9] :

```
# 예제 2-9 컬럼 생성하기
sample_1['기준년월'] = '2019-11'
sample_1
```

Out [9] :

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

데이터에 존재하지 않던 컬럼에 값을 부여하면 새로운 컬럼이 생성됨

pandas 기초

◆ 데이터 선택 – 행 기준(Filtering)

- ❖ 성별이 남성인 데이터 필터링

In [10]:

```
# 예제 2-10 필터링하기 1
condition = (sample_1['성별'] == '남성')
condition
```

Out [10]:

```
0    True
1   False
2    True
3   False
4    True
5   False
Name: 성별, dtype: bool
```

- ❖ 결과

In [6]:

```
# 예제 2-6 sample_1 데이터 확인하기
sample_1
```

Out [6]:

	국적코드	성별	입국객수
0	A01	남성	106320
1	A01	여성	191436
2	A31	남성	319
3	A31	여성	42
4	A18	남성	158912
5	A18	여성	232943

In [11]:

```
# 예제 2-11 필터링하기 2
sample_1[condition]
```

Out [11]:

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
2	A31	남성	319	2019-11
4	A18	남성	158912	2019-11

pandas 기초

◆ 데이터 선택 – 행 기준(Filtering)

- ❖ 입국객수가 150,000명 이상인 데이터 필터링

```
In [12]: # 예제 2-12 필터링하기 3
condition = (sample_1['입국객수'] >= 150000)
sample_1[condition]
```

Out [12]:

	국적코드	성별	입국객수	기준년월
1	A01	여성	191436	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

- ❖ 한 컬럼에 대해 여러 조건으로 데이터 필터링

```
In [15]: # 예제 2-17 한 컬럼에 여러 조건 필터링하기 1
conditions = (sample_1['국적코드'] == 'A01') #
| (sample_1['국적코드'] == 'A18')
sample_1[conditions]
```

Out [15]:

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

- ❖ 성별이 남성이면서 입국객수가 150,000명 이상인 데이터 필터링

```
In [14]: # 예제 2-14 두 개 컬럼에 필터링하기 1
conditions = (sample_1['성별'] == '남성') & (sample_1['입국객수'] >= 150000)
sample_1[conditions]
```

Out [14]:

	국적코드	성별	입국객수	기준년월
4	A18	남성	158912	2019-11

- ❖ 한 컬럼에 대해 여러 조건으로 데이터 필터링

```
In [16]: # 예제 2-18 한 컬럼에 여러 조건 필터링하기 2
conditions = (sample_1['국적코드'].isin(['A01', 'A18']))
sample_1[conditions]
```

Out [16]:

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

pandas 기초

◆ 데이터 통합 – 옆으로 통합(Merge)

In [17] :

```
# 예제 2-21 데이터 확인하기
sample_1
```

Out [17] :

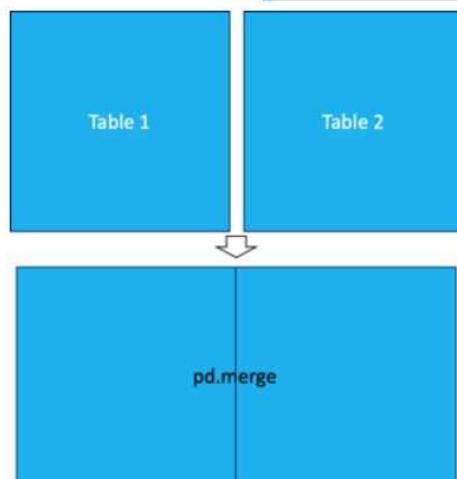
	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

In [18] :

```
# 예제 2-22 국적코드표 엑셀파일 불러오기
code_master = pd.read_excel('./files/sample_codemaster.xlsx')
code_master
```

Out [18] :

	국적코드	국적명
0	A01	일본
1	A02	대만
2	A03	홍콩
3	A18	중국
4	A19	이란
5	A22	우즈베키스탄
6	A23	카자흐스탄
7	A99	아시아 기타



pandas 기초

◆ 데이터 통합 - 옆으로 통합(Merge)

기준 테이블과 정보를 결합하고 싶은 테이블 간에 공통된 컬럼에 대해 같은 값을 찾아서 결합해 하나의 테이블로 합치는 역할을 하는 함수

In [19] :

```
# 예제 2-23 데이터 옆으로 통합하기(left 조건)
sample_1_code = pd.merge(left=sample_1,
                        right=code_master,
                        how='left',
                        left_on='국적코드',
                        right_on='국적코드')
```

왼쪽 테이블은 sample_1로 지정

오른쪽 테이블은 code_master로 지정

왼쪽 테이블을 기준으로 두 테이블 결합

sample_1_code

왼쪽 테이블의 기준 컬럼

오른쪽 테이블의 기준 컬럼

Out [19] :

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국

In [17] :

```
# 예제 2-21 데이터 확인하기
sample_1
```

Out [17] :

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

In [18] :

```
# 예제 2-22 코드표 엑셀파일 불러오기
code_master = pd.read_excel('./files/sample_codemaster.xlsx')
```

Out [18] :

	국적코드	국적명
0	A01	일본
1	A02	대만
2	A03	홍콩
3	A18	중국
4	A19	이란
5	A22	우즈베키스탄
6	A23	카자흐스탄
7	A99	아시아 기타

pandas 기초

◆ 데이터 통합 - 옆으로 통합(Merge)

In [20] :

```
# 예제 2-24 데이터 옆으로 통합하기(inner 조건)
sample_1_code_inner = pd.merge(left=sample_1,
                               right=code_master,
                               how='inner',
                               left_on='국적코드',
                               right_on='국적코드')
```

sample_1_code_inner

두 테이블의 기준 컬럼의 값이 서로 일치하는 경우에만 데이터 통합

Out [20] :

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A18	남성	158912	2019-11	중국
3	A18	여성	232943	2019-11	중국

In [17] :

```
# 예제 2-21 데이터 확인하기
sample_1
```

Out [17] :

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

In [18] :

```
# 예제 2-22 국적코드표 엑셀파일 불러오기
code_master = pd.read_excel('./files/sample_codemaster.xlsx')
```

Out [18] :

	국적코드	국적명
0	A01	일본
1	A02	대만
2	A03	홍콩
3	A18	중국
4	A19	이란
5	A22	우즈베키스탄
6	A23	카자흐스탄
7	A99	아시아 기타

pandas 기초

◆ 데이터 통합 – 아래로 통합(Append)

In [24] :

```
# 예제 2-25 sample_1_code 확인하기
sample_1_code
```

Out [24] :

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국

In [23] :

```
# 예제 2-26 sample_2_code 확인하기
sample_2_code
```

Out [23] :

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	92556	2019-12	일본
1	A01	여성	163737	2019-12	일본
2	A18	남성	155540	2019-12	중국
3	A18	여성	249023	2019-12	중국

Table 1

Table 2

pd.append

pandas 기초

◆ 데이터 통합 – 아래로 통합(Append)

In [25] :

```
# 예제 2-27 데이터 아래로 통합하기 1
sample = sample_1_code.append(sample_2_code, ignore_index=True)
sample
```

Out [25] :

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국
6	A01	남성	92556	2019-12	일본
7	A01	여성	163737	2019-12	일본
8	A18	남성	155540	2019-12	중국
9	A18	여성	249023	2019-12	중국

- append() 함수를 사용해 sample_1_code의 칼럼을 기준으로 sample_2_code를 아래로 통합
- append() 함수를 사용하여 데이터 통합을 하기 위해서는 반드시 컬럼 순서가 동일해야 함

일반적으로 ignore_index = True 인자를 지정함

In [24] :

```
# 예제 2-25 sample_1_code 확인하기
sample_1_code
```

Out [24] :

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국

In [23] :

```
# 예제 2-26 sample_2_code 확인하기
sample_2_code
```

Out [23] :

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	92556	2019-12	일본
1	A01	여성	163737	2019-12	일본
2	A18	남성	155540	2019-12	중국
3	A18	여성	249023	2019-12	중국

pandas 기초

◆ 데이터 저장(to_excel)

In [26]:

```
# 예제 2-31 sample 데이터 엑셀파일로 저장하기 2
sample.to_excel('./files/sample_index_false.xlsx', index=False)
```

	A	B	C	D	E
1	국적코드	성별	입국객수	기준년월	국적명
2	A01	남성	106320	2019-11	일본
3	A01	여성	191436	2019-11	일본
4	A31	남성	319	2019-11	
5	A31	여성	42	2019-11	
6	A18	남성	158912	2019-11	중국
7	A18	여성	232943	2019-11	중국
8	A01	남성	92556	2019-12	일본
9	A01	여성	163737	2019-12	일본
10	A18	남성	155540	2019-12	중국
11	A18	여성	249023	2019-12	중국

웹 크롤링 기초

◆ 웹 크롤링(Web Crawling)

❖ 웹 페이지에 있는 정보를 가지고 오는 것을 의미

➤ selenium 라이브러리

- ✓ webdriver를 활용해 웹 브라우저 조작
- ✓ webdriver는 크롬이나 인터넷 익스플로러 등에서 사이트 접속, 버튼 클릭, 글자 입력과 같이 웹 브라우저에서 사람이 할 수 있는 일들을 코드를 통해 제어할 수 있는 라이브러리
- ✓ webdriver를 활용하기 위해서는 사용 중인 웹 브라우저의 종류에 따라 제어하는 드라이브 필요

➤ BeautifulSoup 라이브러리

- ✓ 웹 페이지 상의 HTML 데이터에서 필요한 정보를 가져옴

웹 크롤링 기초

- ◆ selenium과 크롬 드라이버 설치
 - ❖ selenium 라이브러리 불러오기

```
In [3]:
```

```
# 예제 2-34 selenium 라이브러리 불러오기
from selenium import webdriver
```

- ❖ selenium 라이브러리 미설치 시

```
In [1]: # 예제 2-34 selenium 라이브러리 불러오기
from selenium import webdriver
```

```
-----
ModuleNotFoundError Traceback (most recent call last)
<ipython-input-1-d8b29e91051a> in <module>
      1 # 예제 2-34 selenium 라이브러리 불러오기
      2 from selenium import webdriver
ModuleNotFoundError: No module named 'selenium'
```

```
In [2]:
```

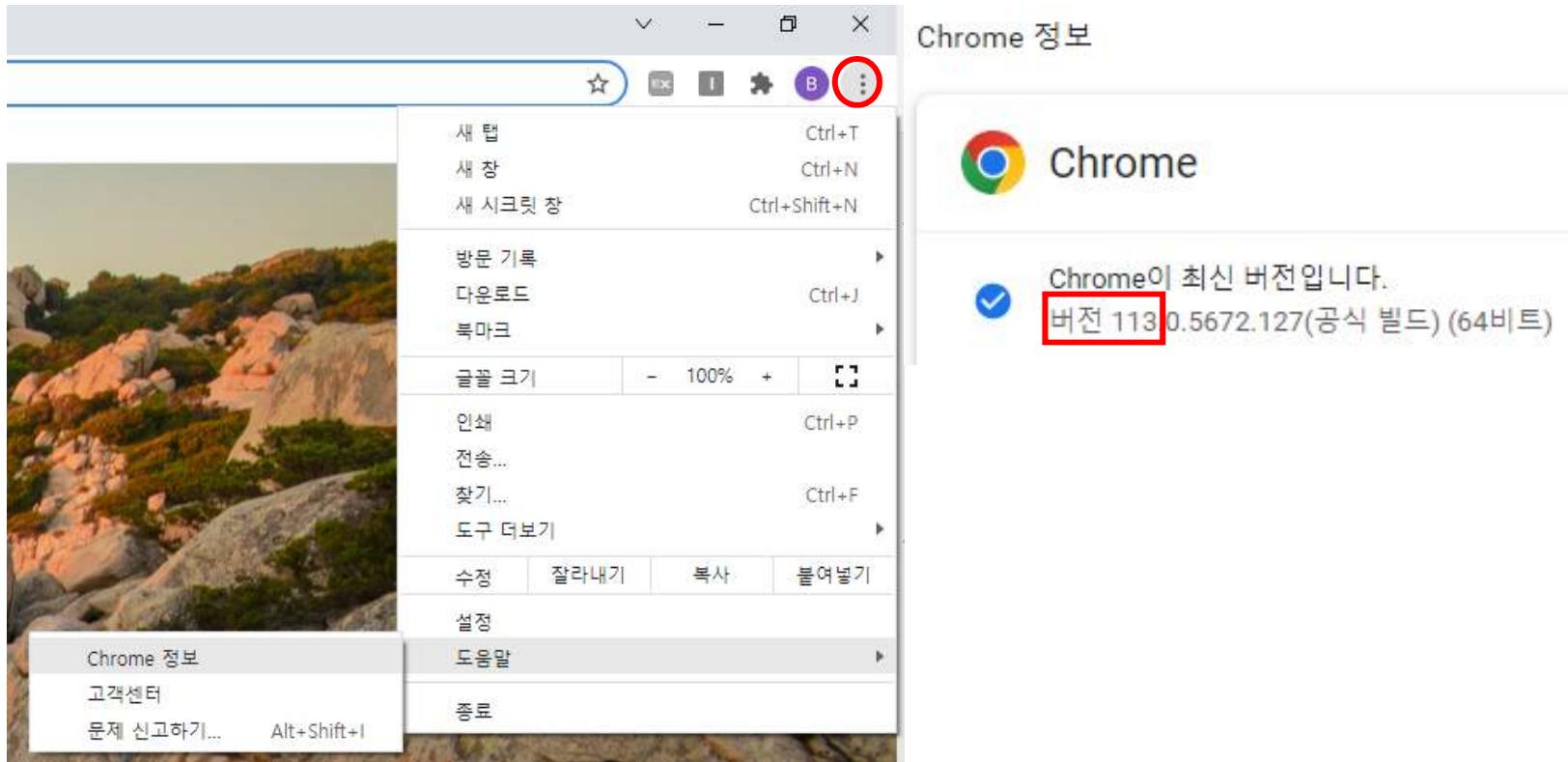
```
# 예제 2-35 selenium 설치하기 (미설치 된 경우에 진행)
! pip install selenium
```

웹 크롤링 기초

◆ selenium과 크롬 드라이버 설치

❖ 크롬 드라이버 설치

- 크롬과 설치와 관계없이 selenium으로 작동하는 크롬드라이버는 별도의 추가 설치가 필요함
- 크롬 버전 확인
 - ✓ 크롬 부라우저의 우측 상단 메뉴-[도움말]-[Chrome 정보]

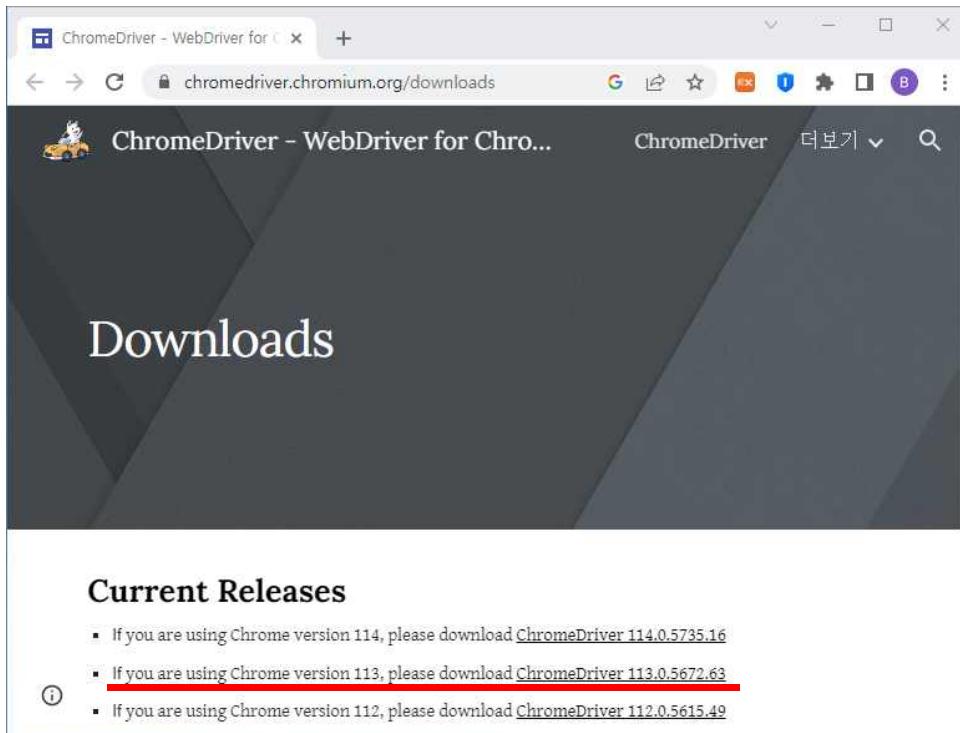


웹 크롤링 기초

◆ selenium과 크롬 드라이버 설치

❖ 크롬 드라이버 설치

- 크롬 버전과 일치하는 크롬드라이버 파일 다운로드
- <https://chromedriver.chromium.org/downloads>



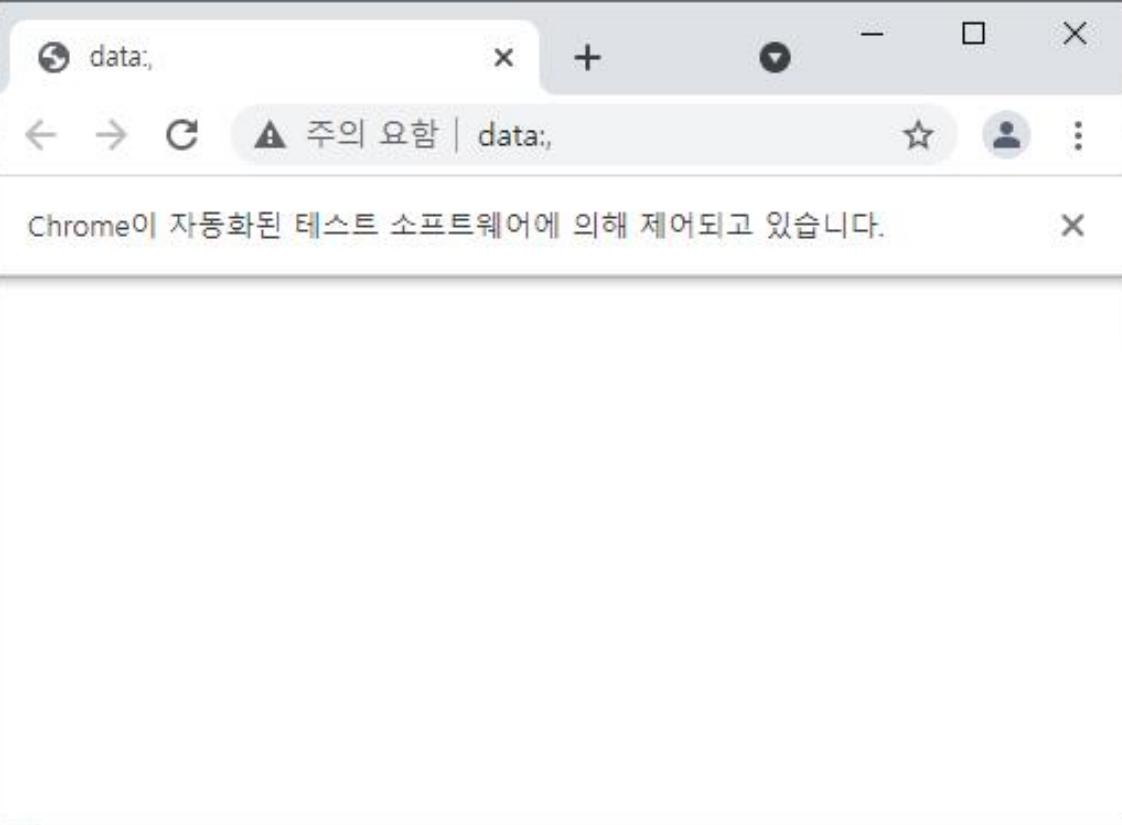
- 압축 해제 후 저장

Name	Last modified	Size	ETag
Parent Directory			
chromedriver_linux64.zip	2023-05-03 09:24:07	6.98MB	b64b1d3b6ff10d635dc5775956ca8048
chromedriver_mac64.zip	2023-05-03 09:24:10	8.81MB	559719231804384ee2345bba18584cca
chromedriver_mac_arm64.zip	2023-05-03 09:24:13	8.06MB	191bb4192a44e26af2c58d71b1876bfa
chromedriver_win32.zip	2023-05-03 09:24:16	6.81MB	47853134740941971c153b11365716ef
notes.txt	2023-05-03 09:24:22	0.00MB	7a6ae80cd5a17d1041aa80e953a140bf

웹 크롤링 기초

- ◆ 크롬드라이버 활용하기
 - ❖ 크롬 드라이버 실행

```
In [4]:  
# 예제 2-38 크롬브라우저 실행하기  
from selenium import webdriver  
driver = webdriver.Chrome('c:/playwithdata/chromedriver.exe')
```



- 새로운 크롬 창이 자동으로 열리게 됨
- 이 크롬 창은 직접 제어하면 안 되고 파이썬 코드에 의해서만 제어해야 함

웹 크롤링 기초

◆ 웹 페이지 접속

In [5]:

```
# 예제 2-37 URL 접속하기
url = 'https://www.naver.com/'
driver.get(url)
```

◆ HTML 구조(예제)

```
<html>
  <head>
  </head>
  <body>
    <h1>우리동네시장</h1>
    <div class = 'sale'>
      <p id='fruits1' class='fruits'>
        <span class = 'name'> 바나나 </span>
        <span class = 'price'> 3000원 </span>
        <span class = 'inventory'> 500개 </span>
        <span class = 'store'> 가나다상회 </span>
        <a href = 'http://bit.ly/forPlayWithData' > 홈페이지 </a>
      </p>
    </div>
    <div class = 'prepare'>
      <p id='fruits2' class='fruits'>
        <span class = 'name'> 파인애플 </span>
      </p>
    </div>
  </body>
</html>
```

◆ 웹 페이지(HTML) 다운로드

In [6]:

```
# 예제 2-38 웹페이지 html 다운로드 하기
html = driver.page_source
```

❖ HTML 규칙

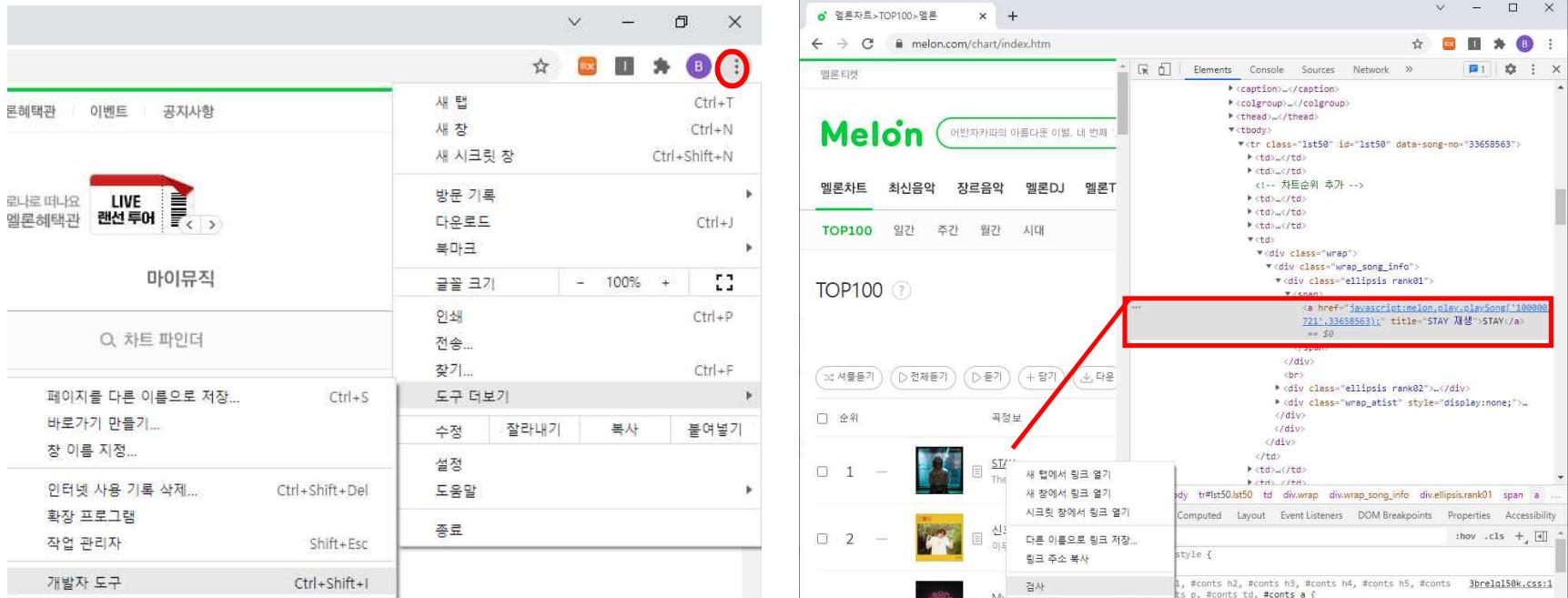
- <태그>로 시작하고 </태그>로 끝남. 태그 부분에 들어가는 태그명은 div, p, span, a, tr, td 등 다양
- 태그는 다른 태그에 속할 수 있음
- 태그의 시작과 끝 사이에 화면에 표시되는 정 보가 들어감
- 태그 기호 내에 속성을 가질 수 있음
 $<\text{태그 속성1} = \text{값1} \text{ 속성2} = \text{값2}>$

'바나나' 값을 찾는 방법은?

웹 크롤링 기초

◆ 크롬 브라우저에서 웹 페이지의 HTML 살펴보기

- ❖ 크롬 개발자 도구를 활용하여 현재 웹 페이지의 HTML 정보를 살펴보고, 필요 한 정보에 해당하는 HTML의 위치 확인
- ❖ 멜론 차트 페이지(<https://www.melon.com/chart/index.htm>)에서 1위 곡에 대한 정보 찾기
 - 크롬 우측 상단의 메뉴-[도구 더보기]-[개발자 도구]
 - 개발자 도구 창에서 [Elements] 탭을 클릭한 후 원하는 정보에서 마우스 오른쪽 버튼을 클릭한 후 [검사] 선택
 - 해당 태그명, 속성, 태그 구조 등 확인 가능



웹 크롤링 기초

- ◆ BeautifulSoup을 이용한 정보 찾기
 - ❖ HTML 문자열을 BeautifulSoup으로 해석하기

```
In [7]: # 예제 2-40 실습용 HTML
html = '''
<html>
  <head>
  </head>
  <body>
    <h1> 우리동네시장</h1>
    <div class = 'sale'>
      <p id='fruits1' class='fruits'>
        <span class = 'name'> 바나나 </span>
        <span class = 'price'> 3000원 </span>
        <span class = 'inventory'> 500개 </span>
        <span class = 'store'> 가나다상회 </span>
        <a href = 'http://bit.ly/forPlayWithData' > 홈페이지 </a>
      </p>
    </div>
    <div class = 'prepare'>
      <p id='fruits2' class='fruits'>
        <span class = 'name'> 파인애플 </span>
      </p>
    </div>
  </body>
</html>
'''
```

```
In [8]:
# 예제 2-41 HTML 문자열 BeautifulSoup 으로 해석하기
from bs4 import BeautifulSoup

soup = BeautifulSoup(html, 'html.parser')
```

웹 크롤링 기초

- ◆ BeautifulSoup을 이용한 정보 찾기
 - ❖ HTML 문자열을 BeautifulSoup으로 해석하기

```
In [7]: # 예제 2-40 실습용 HTML
html = '''
<html>
  <head>
  </head>
  <body>
    <h1> 우리동네시장</h1>
    <div class = 'sale'>
      <p id='fruits1' class='fruits'>
        <span class = 'name'> 바나나 </span>
        <span class = 'price'> 3000원 </span>
        <span class = 'inventory'> 500개 </span>
        <span class = 'store'> 가나다상회 </span>
        <a href = 'http://bit.ly/forPlayWithData' > 홈페이지 </a>
      </p>
    </div>
    <div class = 'prepare'>
      <p id='fruits2' class='fruits'>
        <span class = 'name'> 파인애플 </span>
      </p>
    </div>
  </body>
</html>
'''
```

```
In [8]:
# 예제 2-41 HTML 문자열 BeautifulSoup 으로 해석하기
from bs4 import BeautifulSoup

soup = BeautifulSoup(html, 'html.parser')
```

웹 크롤링 기초

◆ HTML 정보 찾기 – 태그 속성 활용

❖ 태그명으로 태그 찾기

- BeautifulSoup의 select('조건') 함수를 이용하면 HTML 내에서 입력한 조건을 만족하는 모든 태그 선택
- 조건 부분에는 해당 태그의 태그명이나 속성값을 지정하거나 태그 간의 구조를 지정할 수도 있음

In [7] : # 예제 2-40 실습용 HTML

```
html = '''
<html>
  <head>
  </head>
  <body>
    <h1> 우리동네시장</h1>
    <div class = 'sale'>
      <p id='fruits1' class='fruits'>
        <span class = 'name'> 바나나 </span>
        <span class = 'price'> 3000원 </span>
        <span class = 'inventory'> 500개 </span>
        <span class = 'store'> 가나다상회 </span>
        <a href = 'http://bit.ly/forPlayWithData' > 홈페이지 </a>
      </p>
    </div>
    <div class = 'prepare'>
      <p id='fruits2' class='fruits'>
        <span class = 'name'> 파인애플 </span>
      </p>
    </div>
  </body>
</html>
'''
```

In [14] :

```
# 예제 2-42 태그명으로 태그 찾기
tags_span = soup.select('span')
tags_p = soup.select('p')
tags_span
```

Out [14] :

```
[<span class="name"> 바나나 </span>,
 <span class="price"> 3000원 </span>,
 <span class="inventory"> 500개 </span>,
 <span class="store"> 가나다상회 </span>,
 <span class="name"> 파인애플 </span>]
```

웹 크롤링 기초

◆ HTML 정보 찾기 – 태그 속성 활용

❖ 속성값으로 태그 찾기

- BeautifulSoup의 select('조건')의 '조건' 부분에 샵(#) 기호 뒤에 값을 넣거나 점(.) 뒤에 값을 넣는 방식으로 조건 지정

In [7]: # 예제 2-40 실습용 HTML

```
html = '''
<html>
  <head>
  </head>
  <body>
    <h1> 무리동네시장</h1>
    <div class = 'sale'>
      <p id='fruits1' class='fruits'>
        <span class = 'name'> 바나나 </span>
        <span class = 'price'> 3000원 </span>
        <span class = 'inventory'> 500개 </span>
        <span class = 'store'> 가나다상회 </span>
        <a href = 'http://bit.ly/forPlaywithData' > 홈페이지 </a>
      </p>
    </div>
    <div class = 'prepare'>
      <p id='fruits2' class='fruits'>
        <span class ='name'> 파인애플 </span>
      </p>
    </div>
  </body>
</html>
'''
```

In [15]:

```
# 예제 2-43 id와 class로 태그 찾기
ids_fruits1 = soup.select('#fruits1')
ids_fruits1
```

Out [15]:

```
[<p class="fruits" id="fruits1">
  <span class="name"> 바나나 </span>
  <span class="price"> 3000원 </span>
  <span class="inventory"> 500개 </span>
  <span class="store"> 가나다상회 </span>
  <a href="http://bit.ly/forPlaywithData" > 홈페이지 </a>
</p>]
```

In [16]:

```
class_price = soup.select('.price')
class_price
```

Out [16]:

```
[<span class="price"> 3000원 </span>]
```

In [17]:

```
tags_span_class_price = soup.select('span.price')
tags_span_class_price
```

Out [17]:

```
[<span class="price"> 3000원 </span>]
```

웹 크롤링 기초

◆ HTML 정보 찾기 – 상위 구조 활용

❖ 태그 구조로 위치 찾기

- 정보가 담긴 태그의 속성만으로 찾기가 어려울 경우 부모 태그 아래에 있는지 등의 정보를 추가해서 정보를 찾을 수 있음
- 한 단계 아래를 의미할 때는 ‘>’ 기호를 사용
- 여러 단계(한 단계도 포함) 아래를 의미할 때는 띄어쓰기(‘ ’)를 사용

```
In [7]: # 예제 2-40 실습용 HTML
html = '''
<html>
  <head>
  </head>
  <body>
    <h1> 무리동네시장</h1>
    <div class = 'sale'>
      <p id='fruits1' class='fruits'>
        <span class = 'name'> 바나나 </span>
        <span class = 'price'> 3000원 </span>
        <span class = 'inventory'> 500개 </span>
        <span class = 'store'> 가나다상회 </span>
        <a href = 'http://bit.ly/forPlayWithData' > 홈페이지 </a>
      </p>
    </div>
    <div class = 'prepare'>
      <p id='fruits2' class='fruits'>
        <span class = 'name'> 파인애플 </span>
      </p>
    </div>
  </body>
</html>
'''
```

```
In [23]: # 예제 2-48 태그 구조로 위치 찾기③
tags_banana2 = soup.select('div.sale > #fruits1 > span.name')
tags_banana3 = soup.select('div.sale span.name')
print(tags_banana2)
print(tags_banana3)

[<span class="name"> 바나나 </span>]
[<span class="name"> 바나나 </span>]
```

웹 크롤링 기초

- ◆ 정보 가져오기 – 태그 그룹에서 하나의 태그 선택하기
 - ❖ 태그 그룹에서 하나의 태그만 선택

```
In [7]: # 예제 2-40 실습용 HTML
html = '''
<html>
  <head>
  </head>
  <body>
    <h1> 우리동네시장</h1>
    <div class = 'sale'>
      <p id='fruits1' class='fruits'>
        <span class = 'name'> 바나나 </span>
        <span class = 'price'> 3000원 </span>
        <span class = 'inventory'> 500개 </span>
        <span class = 'store'> 가나다상회 </span>
        <a href = 'http://bit.ly/forPlayWithData' > 홈페이지 </a>
      </p>
    </div>
    <div class = 'prepare'>
      <p id='fruits2' class='fruits'>
        <span class = 'name'> 파인애플 </span>
      </p>
    </div>
  </body>
</html>
'''
```

```
In [74]: # 예제 2-47 태그 그룹에서 하나의 태그만 선택하기
tags = soup.select('span.name')
print(tags)
tag_1 = tags[0]    #인덱스 번호로 하나의 태그 지정하기
print(tag_1)

[<span class="name"> 바나나 </span>, <span class="name"> 파인애플 </span>
<span class="name"> 바나나 </span>]
```

tags[0]	바나나
tags[1]	파인애플

웹 크롤링 기초

- ◆ 정보 가져오기 – 태그 그룹에서 하나의 태그 선택하기
 - ❖ 태그 그룹에서 반복문으로 태그 하나씩 선택하기

```
In [7]: # 예제 2-40 실습용 HTML
html = '''
<html>
  <head>
  </head>
  <body>
    <h1> 우리동네시장</h1>
    <div class = 'sale'>
      <p id='fruits1' class='fruits'>
        <span class = 'name'> 바나나 </span>
        <span class = 'price'> 3000원 </span>
        <span class = 'inventory'> 500개 </span>
        <span class = 'store'> 가나다상회 </span>
        <a href = 'http://bit.ly/forPlayWithData' > 홈페이지 </a>
      </p>
    </div>
    <div class = 'prepare'>
      <p id='fruits2' class='fruits'>
        <span class = 'name'> 파인애플 </span>
      </p>
    </div>
  </body>
</html>
'''
```

```
In [75]: # 예제 2-48 태그 그룹에서 반복문으로 태그 하나씩 선택하기
tags = soup.select('span.name')
for tag in tags:    # 반복문으로 태그 그룹에서 각각의 태그 선택하여 활용하기
    print(tag)

<span class="name"> 바나나 </span>
<span class="name"> 파인애플 </span>
```

tags[0]	바나나
tags[1]	파인애플

웹 크롤링 기초

◆ 정보 가져오기 – 선택한 태그에서 정보 가져오기

- ❖ .text: 선택한 태그에서 화면에 보이는 글 부분 가져오기
- ❖ ['속성명']: 선택한 태그에서 태그 내 속성의 값 가져오기

```
In [7]: # 예제 2-40 실습용 HTML
html = '''
<html>
  <head>
  </head>
  <body>
    <h1> 무리동네시장</h1>
    <div class = 'sale'>
      <p id='fruits1' class='fruits'>
        <span class = 'name'> 바나나 </span>
        <span class = 'price'> 3000원 </span>
        <span class = 'inventory'> 500개 </span>
        <span class = 'store'> 가나다상회 </span>
        <a href = 'http://bit.ly/forPlayWithData' > 홈페이지 </a>
      </p>
    </div>
    <div class = 'prepare'>
      <p id='fruits2' class='fruits'>
        <span class ='name'> 파인애플 </span>
      </p>
    </div>
  </body>
</html>
'''
```

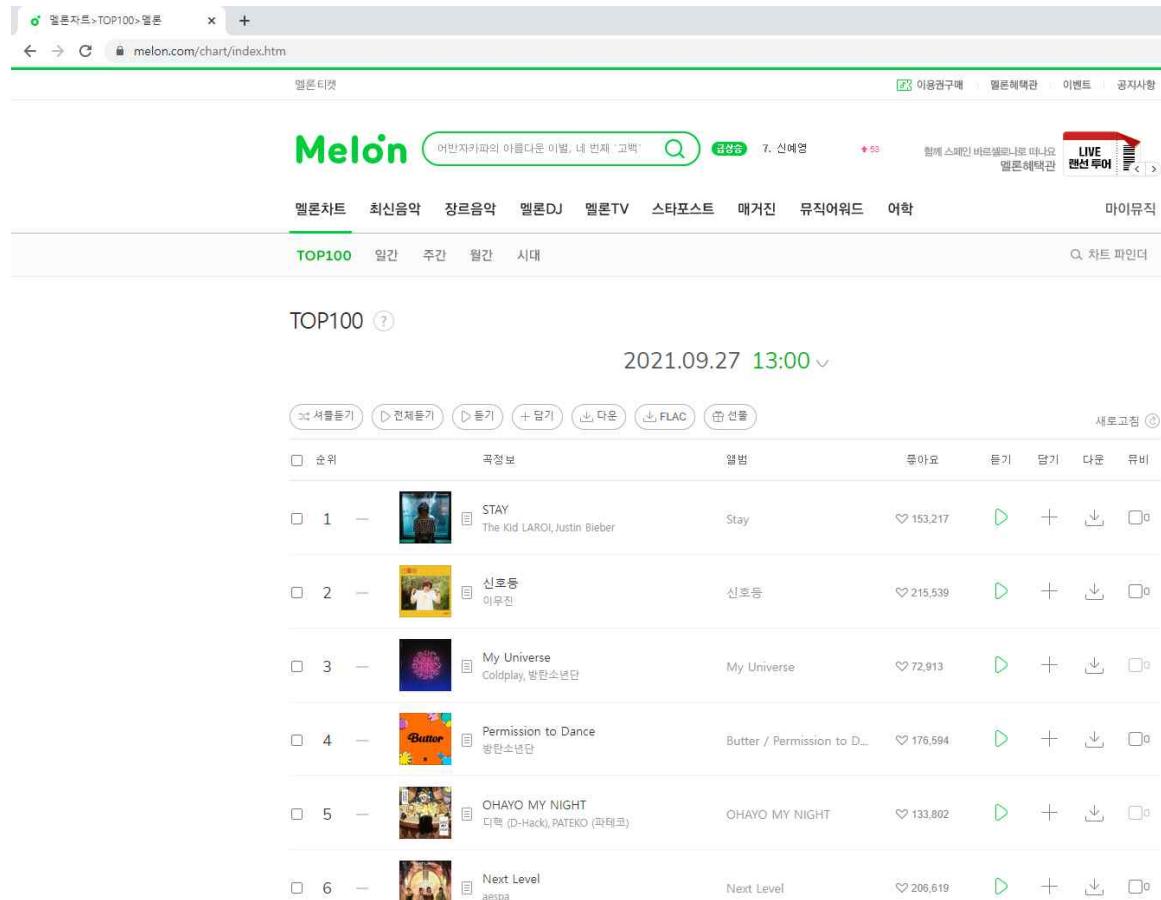
```
In [77]: # 예제 2-50 선택한 태그에서 텍스트, 속성 값 가져오기
tags = soup.select('a')
tag = tags[0]
content = tag.text
print(content)
link = tag['href']
print(link)
```

홈페이지
<http://bit.ly/forPlayWithData>

웹 크롤링 기초

◆ 실습(멜론 사이트 노래 순위 정보 크롤링)

- ❖ Chrome Driver를 이용해 <http://www.melon.com/chart/index.htm> 접속
- ❖ BeautifulSoup을 이용해 멜론차트에서 Top100의 노래 정보(날짜, 순위, 타이틀, 가수 수집)
- ❖ 수집 데이터 엑셀에 저장하기



The screenshot shows the Melon Chart index page. At the top, there's a navigation bar with links like 'Melon 차트', '최신음악', '장르음악', 'Melon DJ', etc. Below that is a search bar with the placeholder '어떤지가파의 아름다운 이별, 네 번째 고백'. The main content area displays the 'TOP100' chart for the date '2021.09.27 13:00'. The chart lists 10 songs, each with a thumbnail, title, artist, and various interaction buttons like play, add to playlist, and download.

순위	곡정보	앨범	좋아요	듣기	담기	다운	유비
1	STAY The Kid LAROI, Justin Bieber	Stay	♡ 153,217				
2	신호등 이우진	신호등	♡ 215,539				
3	My Universe Coldplay, 방탄소년단	My Universe	♡ 72,913				
4	Permission to Dance 방탄소년단	Butter / Permission to D...	♡ 176,594				
5	OHAYO MY NIGHT 디렉 (D-Hack), PATEKO (파테코)	OHAYO MY NIGHT	♡ 133,802				
6	Next Level aespa	Next Level	♡ 206,619				

웹 크롤링 기초

◆ 크롤링 방식의 장단점 비교

	Selenium+BeautifulSoup	Selenium만 사용
웹 페이지 접속	HTML 정보 다운로드 후 브라우저 영향 없음	웹 페이지 연결 유지 필요
웹 페이지 동작	불가능	클릭, 입력 등 조작 가능
크롤링 속도	빠름	느림



4.



데이터 분석 I

- 서울 주유소 가격 비교 -



경남대학교
KYUNGNAM UNIVERSITY





4.

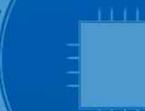
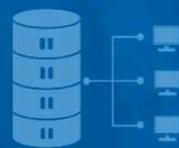


데이터 분석 II

- 중국인 관광객 수 분석 -



경남대학교
KYUNGNAM UNIVERSITY





Thank you

