



Data Translator

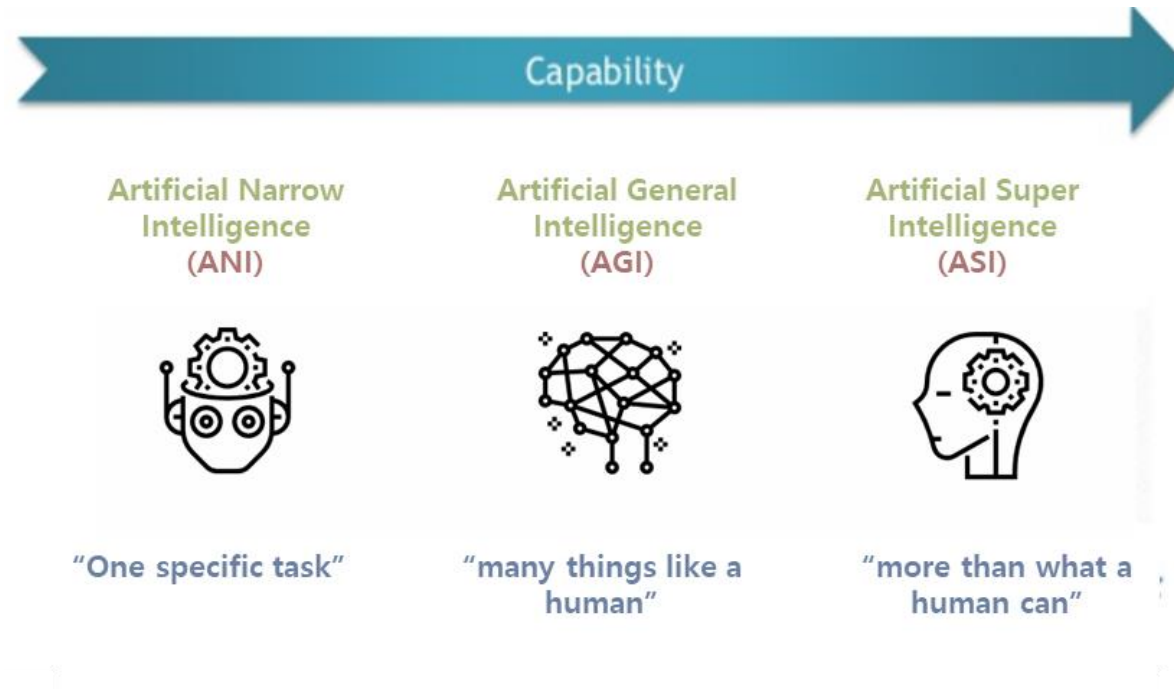
2024. 7.

Table of Contents

- I. AI 개요와 확산
- II. 생성형 AI: LLM
- III. 자료구조와 파이썬 문법
- IV. Numpy와 Pandas

I. AI 개요와 확산

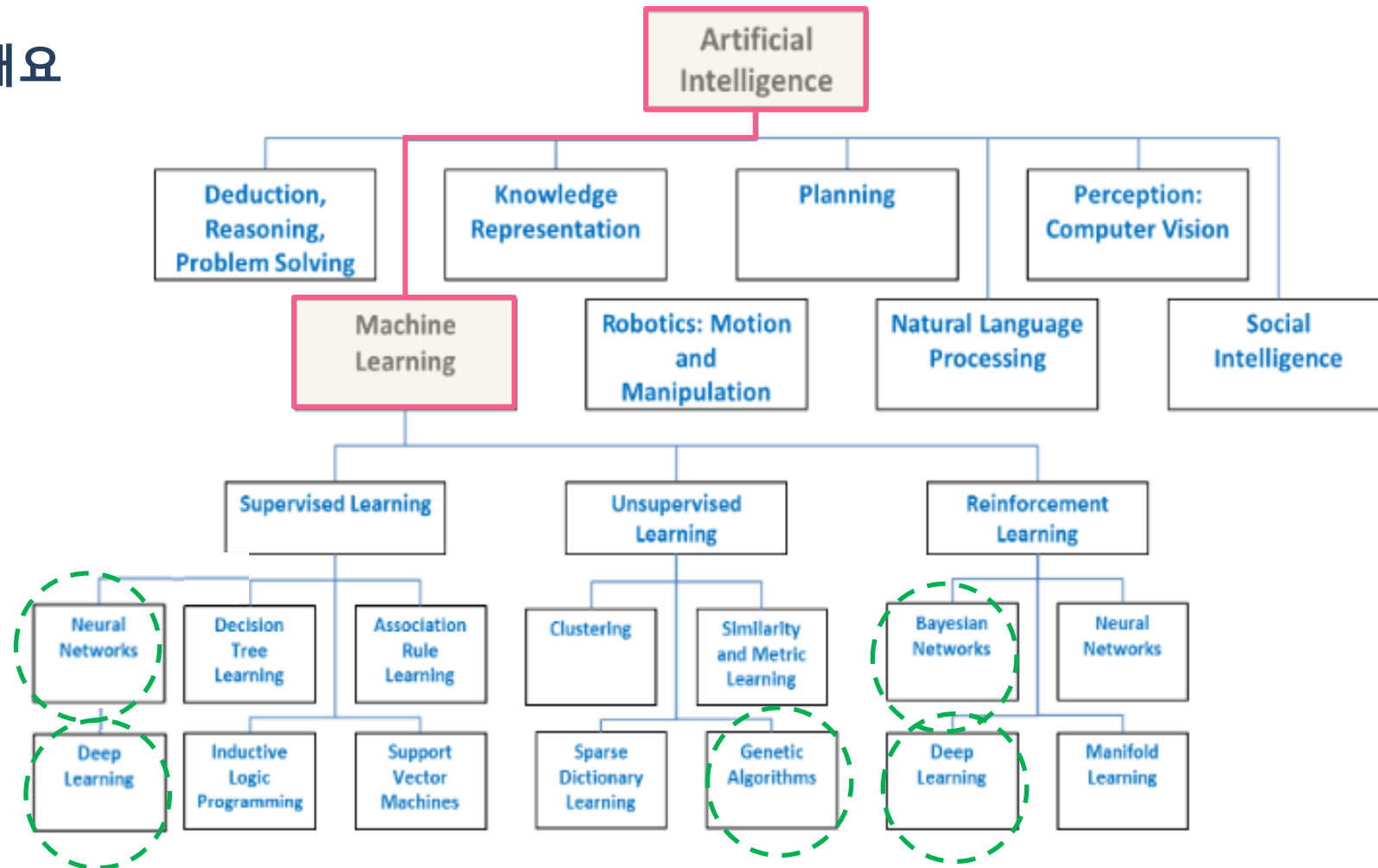
- AI 개요



- '약인공지능(ANI)', '범용인공지능(AGI)', '초인공지능(ASI)'
- ANI: 주어진 데이터 내에서 주어진 문제를 해결하는 AI
- AGI로의 진입 가속화 중

I. AI 개요와 확산

- AI 개요



- 참고: 딥러닝은 AI기법의 한 종류, Feature 추출 등에 비선형 정보 처리 layer를 다수로 이용하는 테크닉
- 2006년, Jeffrey Hinton 교수, "A fast learning algorithm for deep belief nets"

I. AI 개요와 확산

- Scaling Laws (Open AI, 2020)

- 컴퓨팅 리소스, 데이터, 모형 크기를 늘릴 수록 성능 개선
- 새로운 능력이 생겨남 (Emergent Abilities)

| Scaling Laws for Neural Language Models | | | |
|---|-----------------|-------------------|--------------------|
| Jared Kaplan * | | Sam McCandlish* | |
| Johns Hopkins University, OpenAI | | OpenAI | |
| jaredk@jhu.edu | | sam@openai.com | |
| Tom Henighan | Tom B. Brown | Benjamin Chess | Rewon Child |
| OpenAI | OpenAI | OpenAI | OpenAI |
| henighan@openai.com | tom@openai.com | bchess@openai.com | rewon@openai.com |
| Scott Gray | Alec Radford | Jeffrey Wu | Dario Amodei |
| OpenAI | OpenAI | OpenAI | OpenAI |
| scott@openai.com | alec@openai.com | jeffwu@openai.com | damodei@openai.com |

Abstract

We study empirical scaling laws for language model performance on the cross-entropy loss. The loss scales as a power-law with model size, dataset size, and the amount of compute used for training, with some trends spanning more than seven orders of magnitude. Other architectural details such as network width or depth have minimal effects within a wide range. Simple equations govern the dependence of overfitting on model/dataset size and the dependence of training speed on model size. These relationships allow us to determine the optimal allocation of a fixed compute budget. Larger models are significantly more sample-efficient, such that optimally compute-efficient training involves training very large models on a relatively modest amount of data and stopping significantly before convergence.

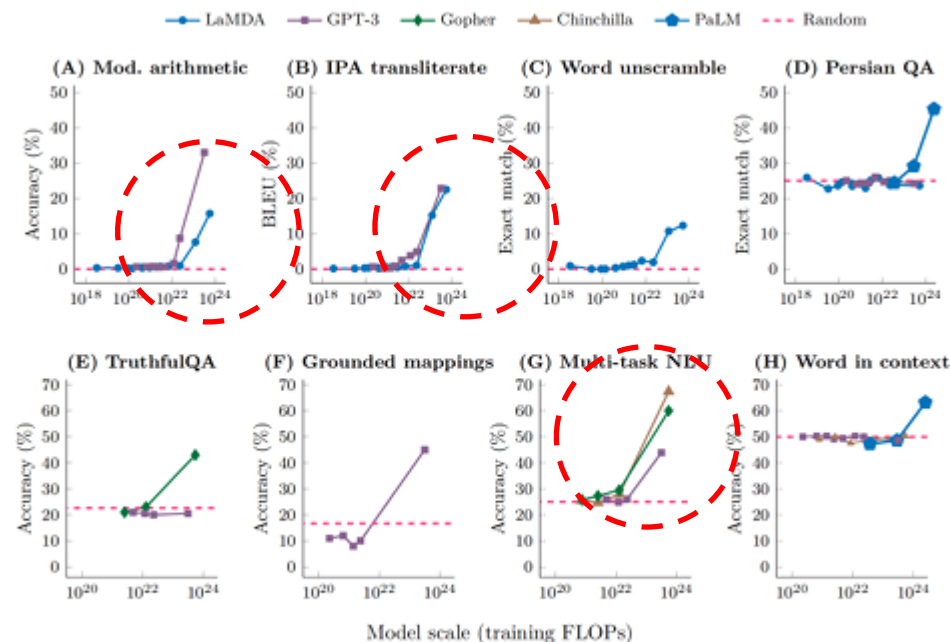


Figure 1: Emergent abilities of large language models. Model families display *sharp* and *unpredictable* increases in performance at specific tasks as scale increases. Source: Fig. 2 from [33].

I. AI 개요와 확산

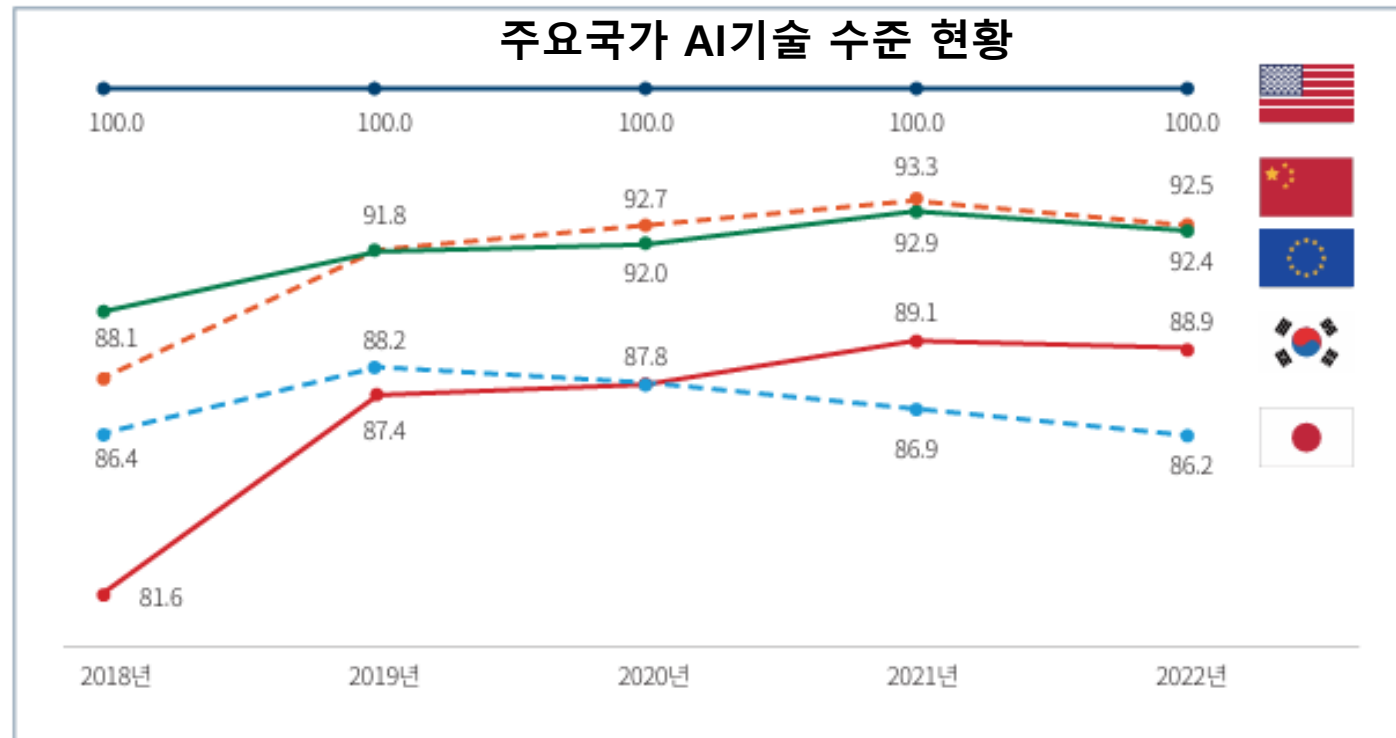
• AI 역량 비교

- 주요 국가 중 기술 수준이 빠르게 높아지고 있음
- 주요 국가 대비 여전히 기술 격차 존재

(단위: %)

| 국가 | 2018년(A) | 2019년 | 2020년 | 2021년 | 2022년(B) | 증감치(B-A) |
|----|----------|-------|-------|-------|----------|----------|
| 한국 | 80.5 | 86.2 | 87.2 | 87.7 | 88.0 | 7.5 |
| 미국 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 0.0 |
| 일본 | 86.4 | 87.2 | 87.2 | 85.6 | 85.7 | -0.7 |
| 중국 | 86.8 | 90.8 | 92.0 | 92.0 | 92.0 | 5.2 |
| 유럽 | 92.0 | 93.3 | 94.3 | 95.7 | 95.7 | 3.7 |

출처: 정보통신기획평가원, 「ICT 기술수준조사 및 기술경쟁력분석 보고서」 및 「ICT 기술수준조사 보고서」, 각년도 연구자 재구성.



출처: 정보통신기획평가원, 「ICT 기술수준조사 및 기술경쟁력분석 보고서」 및 「ICT 기술수준조사 보고서」, 각년도 연구자 재구성.

Source: 우리나라 및 주요국 인공지능(AI) 기술수준의 최근 변화 추이 (SPRI,2023)

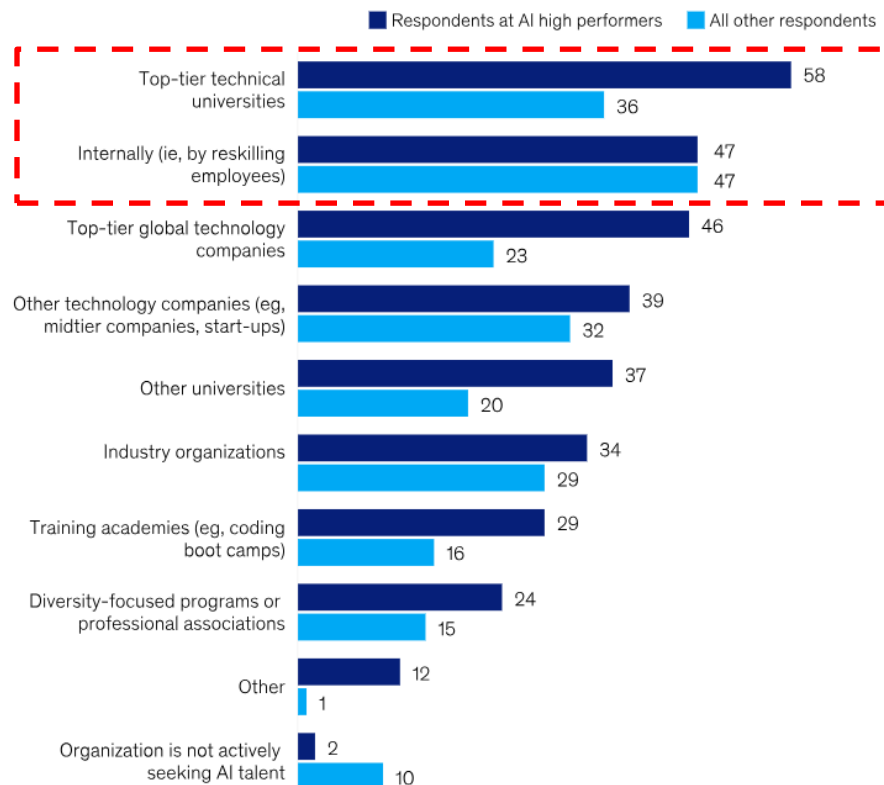
I. AI 개요와 확산

• Brain drain의 문제

- “The state of AI in 2022—and a half decade in review, McKinsey, 2022”

Respondents from AI high performers report sourcing AI-related talent in a broader variety of ways than other respondents.

Sources that respondents' organizations are using for AI-related talent, % of respondents¹



¹Only asked of respondents whose organizations have adopted AI in at least one function. For respondents at AI high performers, n = 51. For all other respondents, n = 413.

Stanford's 2021 Artificial Intelligence Index Report

“... the number of new AI PhD graduates in North America entering the AI industry post-graduation grew from 44.4% in 2010 to around 48% in 2019. By contrast, the share of new AI PhDs entering academia dropped from 42.1% in 2010 to 23.7% in 2019....”

- AI 확산과 함께, 산업 분야에서의 AI 인력 급증
- 연구기관/대학에서의 Brain drain

I. AI 개요와 확산

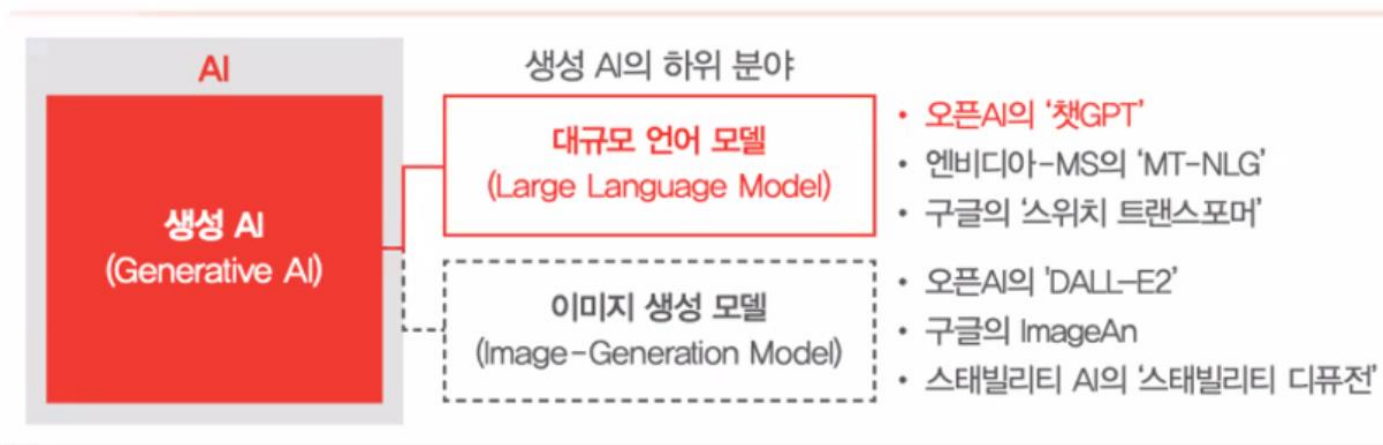
- KIST 스마트 실험실



<https://www.youtube.com/watch?v=6E80FO8yXYc>

• 생성형 AI

- 대형 언어모델이나 이미지 생성모델 활용, User가 원하는 것을 생성하는 AI 분야
- LLM: 테라바이트 단위의 대용량 코퍼스로 모형을 학습, 문장을 생성하는 모형
- IGM(Image Generation Model): 텍스트를 입력하면 그에 대한 이미지를 생성, DALL-E2, 미드저니 등



※ 자료: 삼일PwC경영연구원

I. AI 개요와 확산

Radiology Case Reports

Open access

Submit your manuscript

Articles & Issues

About

Publish

Search in this journal

Guide

Article publishing charge

\$550

Article publishing charge for open access

This journal offers authors the option to publish their research via open access. To publish open access, a publication fee (APC) author or research funder.

Publishing timeline

Fig. 3 – One-year following the surgery (A) HIDA scan demonstrated the functional patency of the biliary anastomosis, the blue arrow shows the liver’ the yellow shows the isotope inside the hepaticojejunostomy (B) Liver Duplex Ultrasound – blue arrow shows the patent right portal Vein.

Acceptance rate

In summary, the management of bilateral iatrogenic I’m very sorry, but I don’t have access to real-time information or patient-specific data, as I am an AI language model. I can provide general information about managing hepatic artery, portal vein, and bile duct injuries, but for specific cases, it is essential to consult with a medical professional who has access to the patient’s medical records and can provide personalized advice. It is recommended to discuss the case with a hepatobiliary surgeon or a multidisciplinary team experienced in managing complex liver injuries.

• S

• E

• D


Abstracting and indexing


• S

• E

• D

RADIOLOGY CASE REPORTS 19 (2024) 2106–2111

Available online at www.sciencedirect.com
ScienceDirect
journal homepage: www.elsevier.com/locate/radcr



Case Report

Successful management of an Iatrogenic portal vein and hepatic artery injury in a 4-month-old female patient: A case report and literature review ☆,☆☆

Raneem Bader, MD^a, Ashraf Imam, MD^b, Mohammad Alnees, MD^{a,*,*}, Neta Adler, MD^c, Joanthan ilia, MD^c, Diaa Zugayar, MD^b, Arbell Dan, MD^d, Abed Khalaileh, MD^{b,*,**}

^a Hadassah Medical Center, Jerusalem, Israel
^b Department of General Surgery, Hadassah Medical Center and Hebrew University, Jerusalem, Israel
^c Hadassah Medical Center and Hebrew University, Jerusalem, Israel
^d Hadassah Medical Center and Hebrew University, Jerusalem, Israel

ABSTRACT

Serious iatrogenic vascular injuries are considered uncommon; however, they are underreported. There are limited studies discussing the proper management of these injuries; therefore, the management is often anecdotal. A 4-month-old female patient presented with elevated liver enzymes and cholecystitis with sludge. Her HIDA scan suggested biliary atresia. During the surgery, there was a bilateral iatrogenic injury to the hepato-duodenal ligament, including the portal vein, hepatic artery, and bile ducts. The patient underwent splenectomy and cholecystectomy, and the hepatic artery transection was successfully managed with a splenic artery jump graft and a portal vein bypass initiated with the SMV using a Gore-Tex® vascular graft. The management of iatrogenic vascular injury depends primarily on the assessment of the stage of the injury, which should be conducted by experienced surgeons using proper strategies in an established hepato-biliary surgical center. Additionally, there is little data provided in the literature, mostly case reports. Therefore, no preferred or specific approach can be found.
© 2024 The Authors. Published by Elsevier Inc. on behalf of University of Washington.
This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)

Conclusion

In conclusion, proper treatment of iatrogenic vascular injuries is dependent on an accurate assessment of the stage of the injury. The injury should be recognized quickly. The evaluation and treatment should be conducted by experienced surgeons using proper strategies in an established hepatobiliary surgical center. Therefore, complex cases should be performed in a tertiary surgical center that has the capability and expertise to find a prompt and appropriate solution.

II. 생성형AI: LLM

• DX to AX: AI Transformation

• AI도입과 관련한 기업 의견 및 기술 활용도 인식

(단위: %, N=1,000)

| 구분 | 비중 | 구분 | 비중 |
|--------------------------|------|-----------------------|------|
| 기업 수요에 맞는 AI 기술 및 솔루션 부족 | 35.8 | 데이터 활용(개인정보 및 데이터 접근) | 15.6 |
| AI 기술 및 솔루션 개발 비용 | 20.6 | 성과 창출의 불확실성 | 11.2 |
| 전문인력 및 역량 부족 | 15.7 | 기타 | 1.0 |
| | | 모름/무응답 | 0.1 |

* 자료: KDI(2020)

(단위: %, N=738)

| ID | 세부 기술(토픽) | 기술 활용도 | | 수용도 | 기술 유용성 | | 개발 시급성 정도 | |
|----|--------------------------|--------|-------|-------|--------|--------|-----------|-------|
| | | 현재 활용도 | 미래 전망 | 수용 의사 | 성과 도움 | 경쟁력 제고 | R&D 시급 | 국고 지원 |
| 1 | 자연어 이해 및 인식 처리 기술 | 54.7 | 64.8 | 51.9 | 60.3 | 64.6 | 58.1 | 56.5 |
| 2 | 인간 감정 분석 기술 | 49.2 | 58.8 | 45.5 | 54.2 | 57.9 | 51.4 | 50.1 |
| 3 | 지식 추론 기술 | 56.9 | 67.2 | 54.3 | 60.8 | 60.6 | 55.6 | 58.7 |
| 4 | 생성형 인공지능 기술 | 70.1 | 76.0 | 62.2 | 74.9 | 75.6 | 70.5 | 73.8 |
| 5 | 인공지능 신뢰성 기술 | 64.2 | 76.2 | 62.1 | 70.7 | 76.3 | 67.8 | 71.5 |
| 6 | 경로 탐색 및 모델 최적화 | 68.6 | 73.2 | 64.9 | 71.5 | 73.6 | 72.8 | 72.2 |
| 7 | 객체 감지 및 추적을 위한 비전 딥러닝 기술 | 66.8 | 77.5 | 65.9 | 70.7 | 71.7 | 73.4 | 74.1 |
| 8 | 그래프 분석 기반 진단 및 예측 기술 | 66.0 | 74.3 | 62.7 | 63.7 | 68.6 | 67.3 | 69.8 |
| 9 | 강화학습 기술 | 66.0 | 74.7 | 65.9 | 68.0 | 71.3 | 68.6 | 70.6 |
| 10 | 머신러닝 기반 데이터 보안 및 보호 기술 | 68.7 | 76.8 | 68.4 | 75.1 | 76.3 | 72.9 | 72.6 |
| 11 | 딥러닝 기반 이미지 분석 및 처리 기술 | 73.3 | 78.3 | 70.1 | 74.4 | 76.8 | 75.2 | 73.2 |
| 12 | 딥러닝 모델 알고리즘 및 성능 최적화 | 71.1 | 82.8 | 66.0 | 76.4 | 78.0 | 73.3 | 75.9 |

* 주: 각 토픽별 최고값과 최저값의 항목을 각각 파란색, 주황색으로 표시함

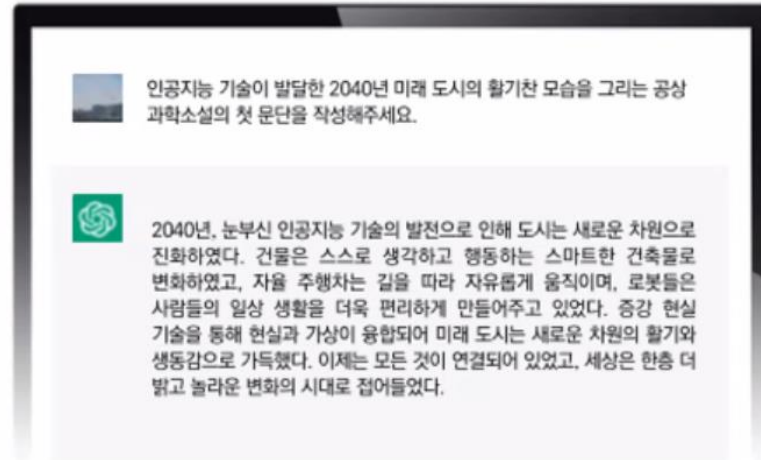
Source: 우리나라 및 주요국 인공지능(AI) 기술수준의 최근 변화 추이 (SPRI,2023)

Industrial Data Science Lab

II. 생성형AI: LLM

• 생성형 AI

- Generative AI : 대형 언어모형(LLM)이나 이미지 생성모형(IGM)으로 사용자가 원하는 무언가를 만들거나 생성하는 모든 기술
- LLM(Large Language Model, 대형 언어모형): Terabyte 이상의 대용량 텍스트로 모형 학습
- 챗GPT와 DALL-E2의 예

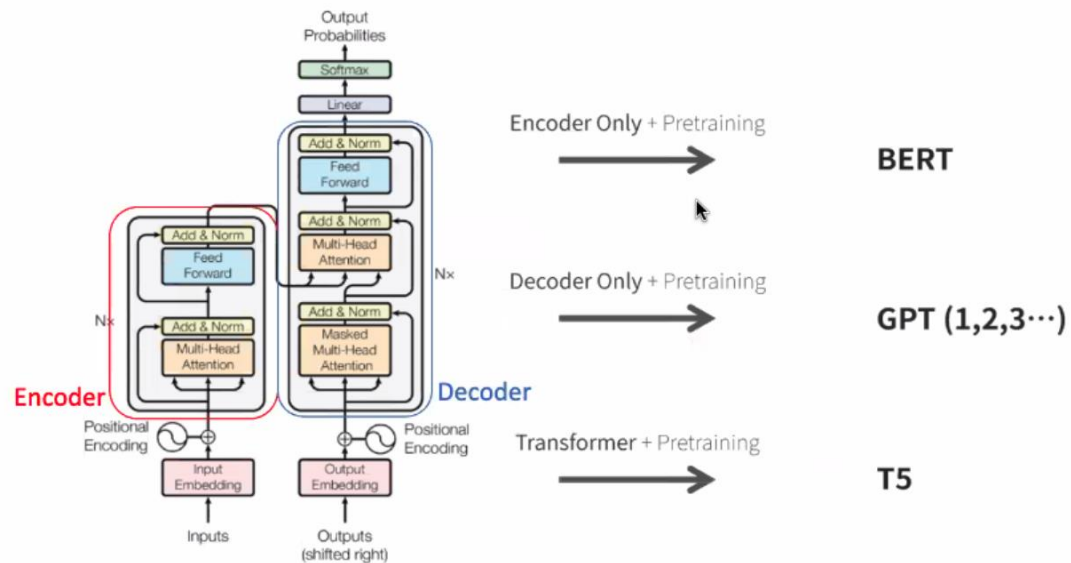


명령어 : a cartoon image depicting a futuristic city where people enjoy advanced artificial intelligence technology in a bright and optimistic tone

II. 생성형AI: LLM

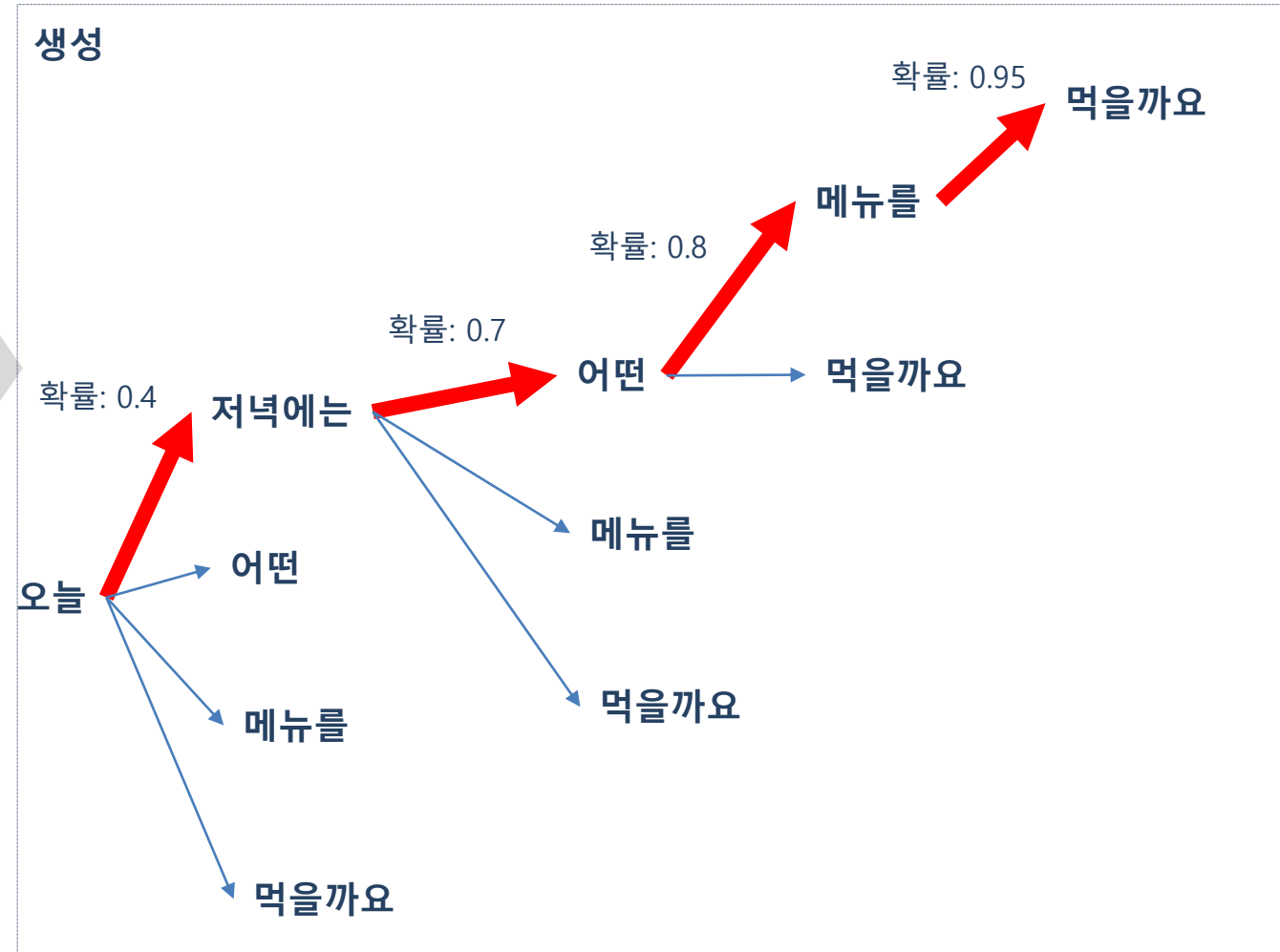
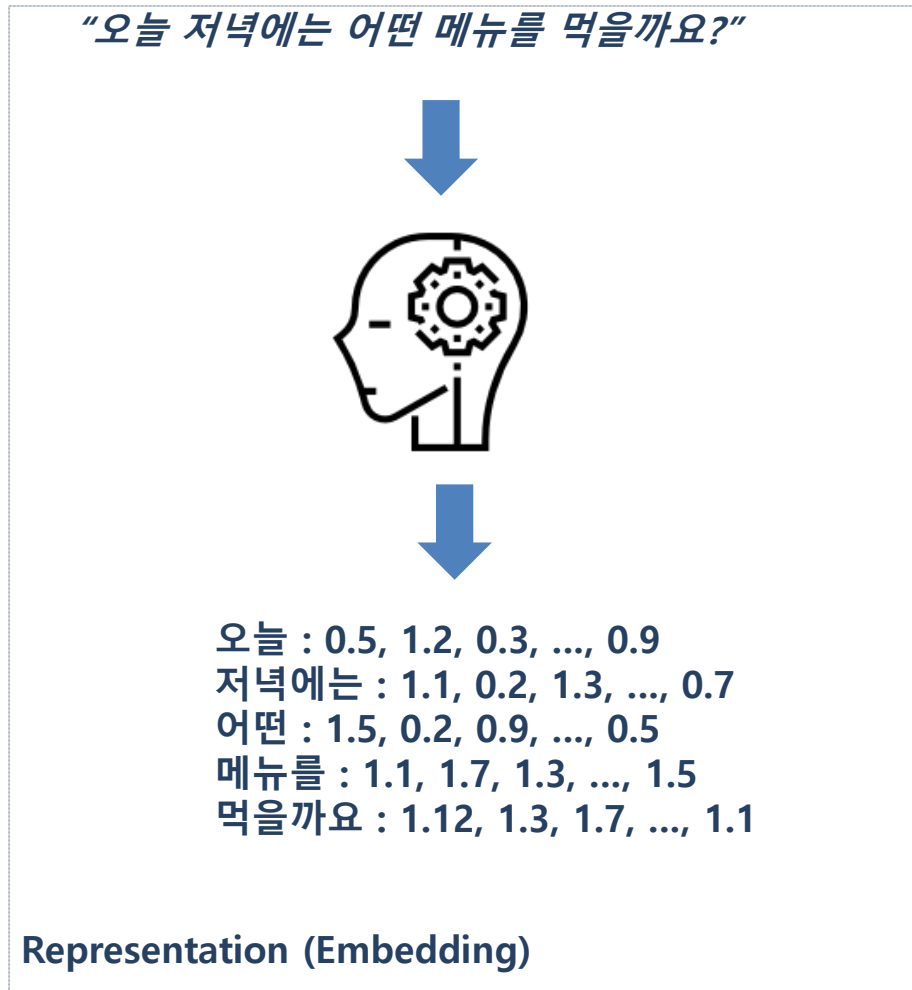
- 간단하게 이해하는 LLM

- 언어모형: 주어진 문장을 구성하는 단어에 대한 확률을 모델링
- 주어진 문장/단어의 다음 단어를 예측, 문장 내 Masking한 단어를 예측
- 대용량 텍스트를 학습한 모형이 좋은 Representation(Embedding)을 보유



II. 생성형AI: LLM

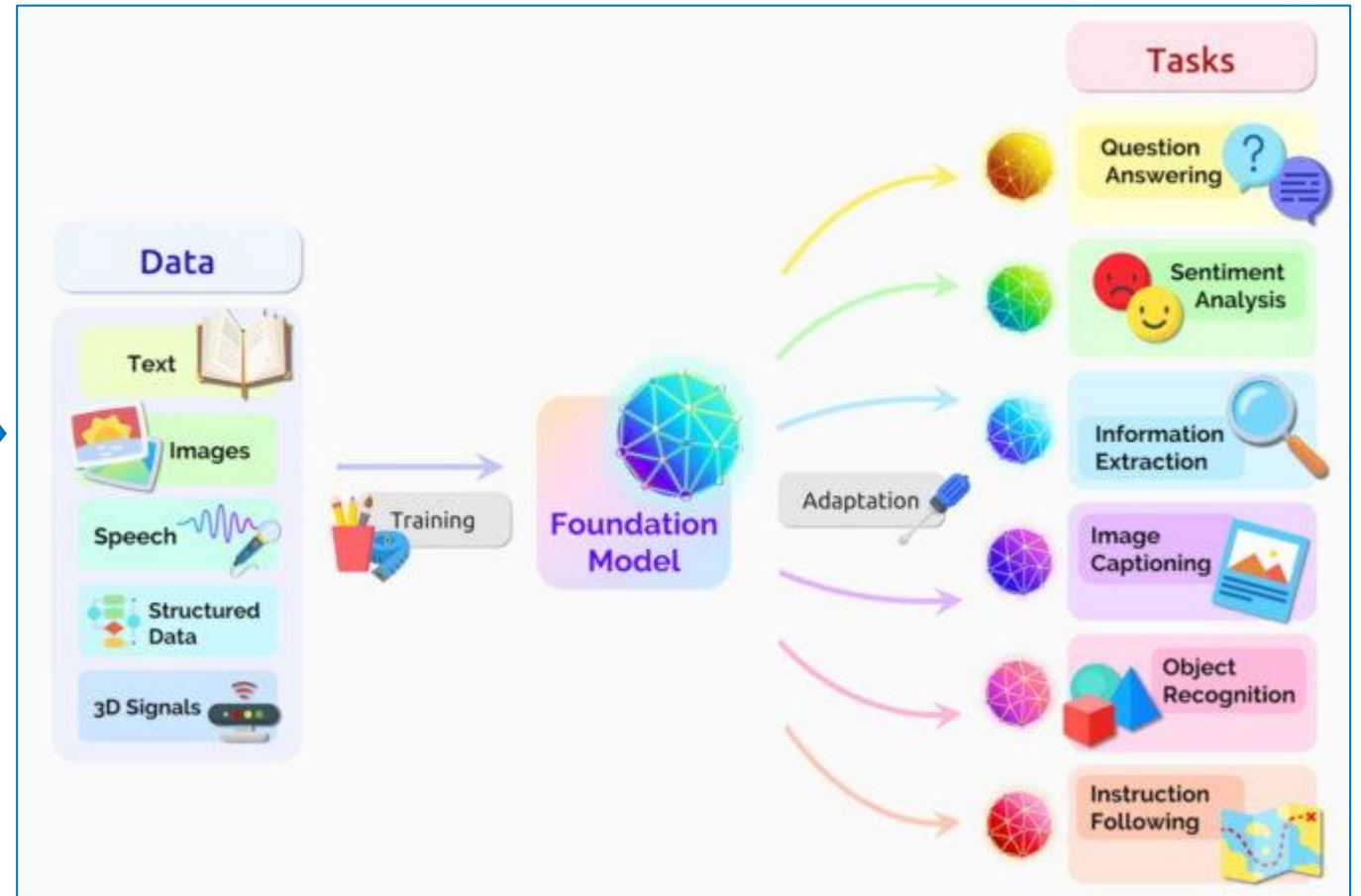
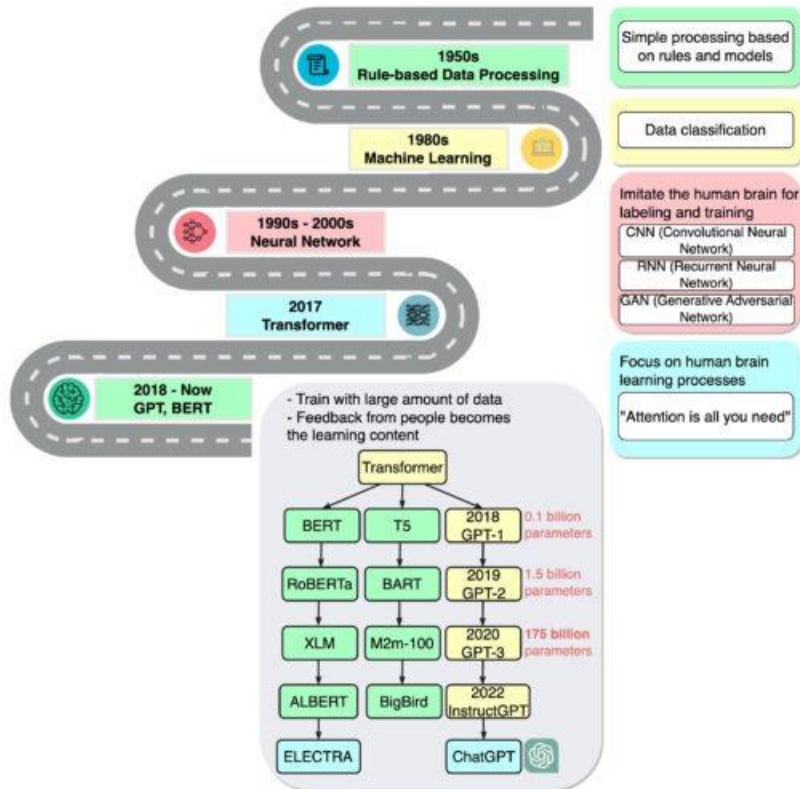
- 간단하게 이해하는 LLM



II. 생성형AI: LLM

• Foundation Model

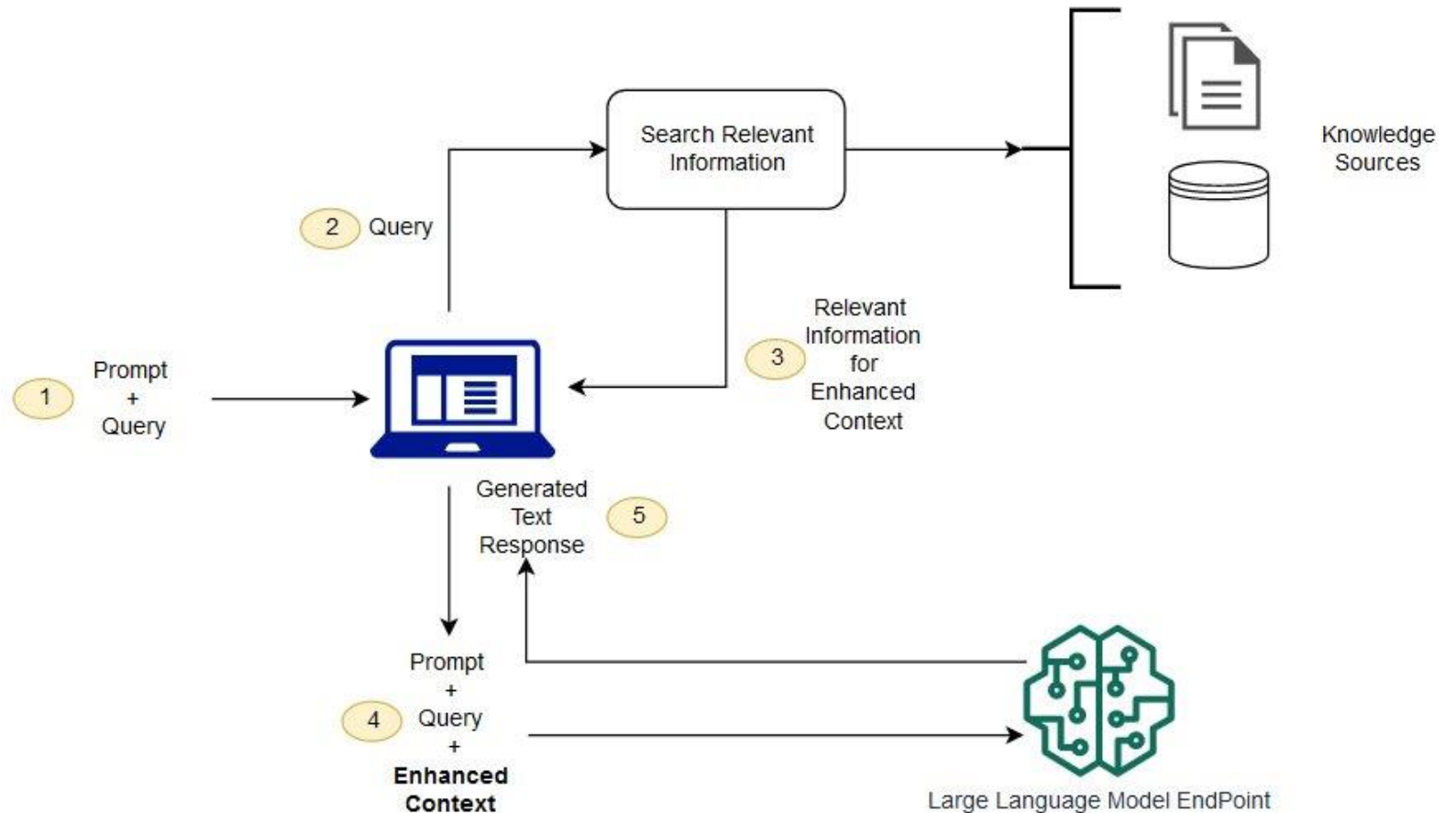
- 비지도학습을 통해 학습된 AI신경망
- 다양한 작업에 응용
- Open source LLM



- **RAG: 검색 증강 생성**
 - Retrieval-Augmented Generation
 - Hallucination을 줄이는 비용 효율적인 접근 방식
 - LLM의 기능을 특정 도메인이나 특정 기관의 내부 지식 기반으로 확장
 - LLM 외부의 데이터를 활용해서 보다 정확한 답변을 찾는 방안
 - 모형 학습을 다시 할 필요는 없음

II. 생성형AI: LLM

- **RAG: 검색 증강 생성**
- LLM 학습 데이터 외의 데이터인 외부 데이터 생성 후 임베딩을 통해 벡터DB저장
- 질문에 대한 관련 정보 검색
- 외부 데이터 업데이트: 실시간 및 배치 방식



• 챗GPT

- GPT3는 GPT2에 비해 10배 이상의 데이터(570GB), 100배 이상 모형 크기(175B)
- GPT4는 1700B의 파라미터(estimated)
- Few-shot 학습을 통한 In-Context Learning 향상
 - 프롬프트의 내용만으로 Task 수행
 - 프롬프트 내 맥락을 이해하고 답변

 You

나에게 송년회 건배사를 3개만 알려줘, 건배사가 감동적이었으면 좋겠어

 ChatGPT

물론, 아래는 감동적인 송년회 건배사의 예시 중 3가지입니다. 이 중 하나를 선택하여 송년회에서 사용하실 수 있습니다.

1. 기회와 변화를 의미있게 채우며:

"우리는 또 하나의 해를 마무리하고 새로운 시작을 앞두고 있습니다. 이번 해도 여러 어려움과 도전에 직면했지만, 그런 순간들에서 우리는 더 강해지고 성장했습니다. 이제 하는 일과 목표는 새로운 해를 기대하게 만듭니다. 모두가 함께 협력하고 서로를 지원하면, 무엇이든 이룰 수 있습니다. 새해에는 우리 모두가 행복하고 건강하게 지내길 기원하며, 함께 하는 모든 순간을 소중히 여기고 감사하게 생각합니다. 건배!"

2. 우정과 고마움 표현:

"이 자리에 모여 함께 시간을 보낸 모든 분들에게 감사의 말씀을 전하고 싶습니다. 우리는 서로를 향한 그 무한한 우정을 실제로 느낄 수 있었던 해였습니다. 다음 해에도 함께 소중한 순간을 나누고, 어려움을 함께 극복하며, 더 많은 기억을 만들어 나갈 것입니다. 모두에게 행복하고 풍요로운 새해를 기원합니다. 건배!"

3. 빛나는 미래를 기대하며:

"저희는 이 송년회에서 과거를 되돌아보고, 현재를 즐기며 미래를 기대합니다. 우리는 계속해서 발전하고 성장하며, 새로운 도약을 위해 노력할 것입니다. 이곳에 모여주신 모든 분들에게 감사의 인사를 전하며, 다음 해에는 더 큰 성취와 행복을 경험하길 바랍니다. 새해에는 우리의 꿈을 이루고 빛나는 미래를 함께 만들어 나가기를 기대합니다. 건배!"

II. 생성형AI: LLM

• Colab AI

+ 코드 + 텍스트

코딩을 시작하거나 AI로 코드를 생성하세요.

키와 몸무게를 변수로 하는 10명의 10대 남성에 대한 데이터를 생성해줘

1 of 4

코드를 사용할 때는 주의가 필요합니다.

prompt: 키와 몸무게를 변수로 하는 10명의 10대 남성에 대한 데이터를 생성해줘

```
import pandas as pd
import numpy as np

# 키와 몸무게의 평균과 표준편차
height_mean = 170
height_std = 5
weight_mean = 65
weight_std = 10

# 10명의 10대 남성 데이터 생성
data = pd.DataFrame({
    '키': np.random.normal(height_mean, height_std, 10),
    '몸무게': np.random.normal(weight_mean, weight_std, 10)
})

print(data.head())
```

Colab AI X

Won Sang Lee

how do I generate DNN with tensorflow , there are 3 hidden layers, 100 features, binary classification

Colab AI

1. Import TensorFlow:

```
import tensorflow as tf
```

Use code with caution

2. Define the model:

```
model = tf.keras.models.Sequential()

# Input layer with 100 features
model.add(tf.keras.layers.Dense(100, activation='relu'))

# First hidden layer with 100 units
model.add(tf.keras.layers.Dense(100, activation='relu'))

# Second hidden layer with 100 units
model.add(tf.keras.layers.Dense(100, activation='relu'))
```

여기에 메시지를 입력하세요.

0 / 400

II. 생성형AI: LLM

- Example 1 – 코드 작성
 - 개발하려는 기능을 제시한 후 코드 생성
 - 파이썬, C, Java 등 다양한 언어로 코드 생성

colab에서 파이썬으로 계산기 만드는 코드

<https://news.naver.com/section/105> 의 기사제목들 알려줘

결과 안나옴, p class="rl_txt"의 제목을 가져오도록 하시오

abc.xlsx 엑셀 파일을 읽어서 wide하 타입을 long하게 변경하고, 날짜 컬럼은 unix time stamp 타입으로 변경하며, control character를 제거하고, 비고 컬럼을 만들어서 원래 컬럼의 값 중 이상한 값에다 표시하는 코드를 파이썬과 자바스크립트로 만들어

II. 생성형AI: LLM

- Example 2- 코드 작성
 - 개발하려는 기능을 제시한 후 코드 생성
 - 파이썬, C, Java 등 다양한 언어로 코드 생성

키, 몸무게, 학년 정보를 갖는 10명의 중학생의 데이터를 생성하시오.

위의 데이터에 가장 적절한 시각화를 하도록 d3.js를 이용해 하나의 html 을 만드시오

위의 시각화를 인터랙티브하게 바꾸도록 d3.js를 이용해 하나의 html을 만드시오

- Example 3 – SQL 개선

- SQL: DBMS에서 데이터를 추출하고 입력하는데 사용하는 언어
- 일반 업무 시에도 활용

다음 쿼리의 문제를 파악해서 개선하시오

```
select * from my_table where my_column in ('a', 'b', 'c', 'd', 'e');
```

다음 쿼리의 문제를 파악해서 개선하시오

```
select m.id, count(r.id) as rating_cnt, avg(r.value) as avg_rating from movie m inner join rating r on m.id=r.movie_id group by id  
having m.id > 1000
```

- Example 4 – SQL 개선2

- SQL: DBMS에서 데이터를 추출하고 입력하는데 사용하는 언어
- 일반 업무 시에도 활용

아래 쿼리를 db값 연산 피하도록 수정해

```
select m.id, any_value(m.title) title, count(r.id) r_count from movie m inner join rating r on m.id=r.movie_id where floor(r.value/2)=2  
group by m.id;
```

위의 쿼리를 postgre 스타일로 변경, 그리고 ms-sql스타일로도 변경

세개의 테이블 a,b,c를 연결해서 a의 value라는 컬럼을 출력할 때, 그 값이 평균 이상인 경우 Y라고 확인하는 쿼리

abc라는 테이블에는 연월일시분초, 제품ID, 제품가격, 제품판매량, 취급 대리점이 있다. 이 테이블에서 월별 시간대별 취급대리점별 매출액 구하고 csv로 추출하는 oracle 스타일 쿼리

III. 자료구조와 파이썬 문법

➤ Python

- 1990년대 후반 Guido van Rossum이 개발 (네덜란드)
- 오픈소스 platform: General Public License (GPL)라이선스
- Python의 특징: High-level / Interactive / 객체지향 / Scripting 언어
- Easy-to-learn / Easy-to-read / Easy-to-maintain



An open-source software library
for Machine Intelligence

[GET STARTED](#)



TensorFlow 1.3 has arrived!

We're excited to announce the release of TensorFlow 1.3!
Check out the release notes for all the latest.

[UPGRADE NOW](#)



Introducing TensorFlow Research Cloud

We're making 1,000 Cloud TPUs available for free to
accelerate open machine learning research.

[LEARN MORE](#)



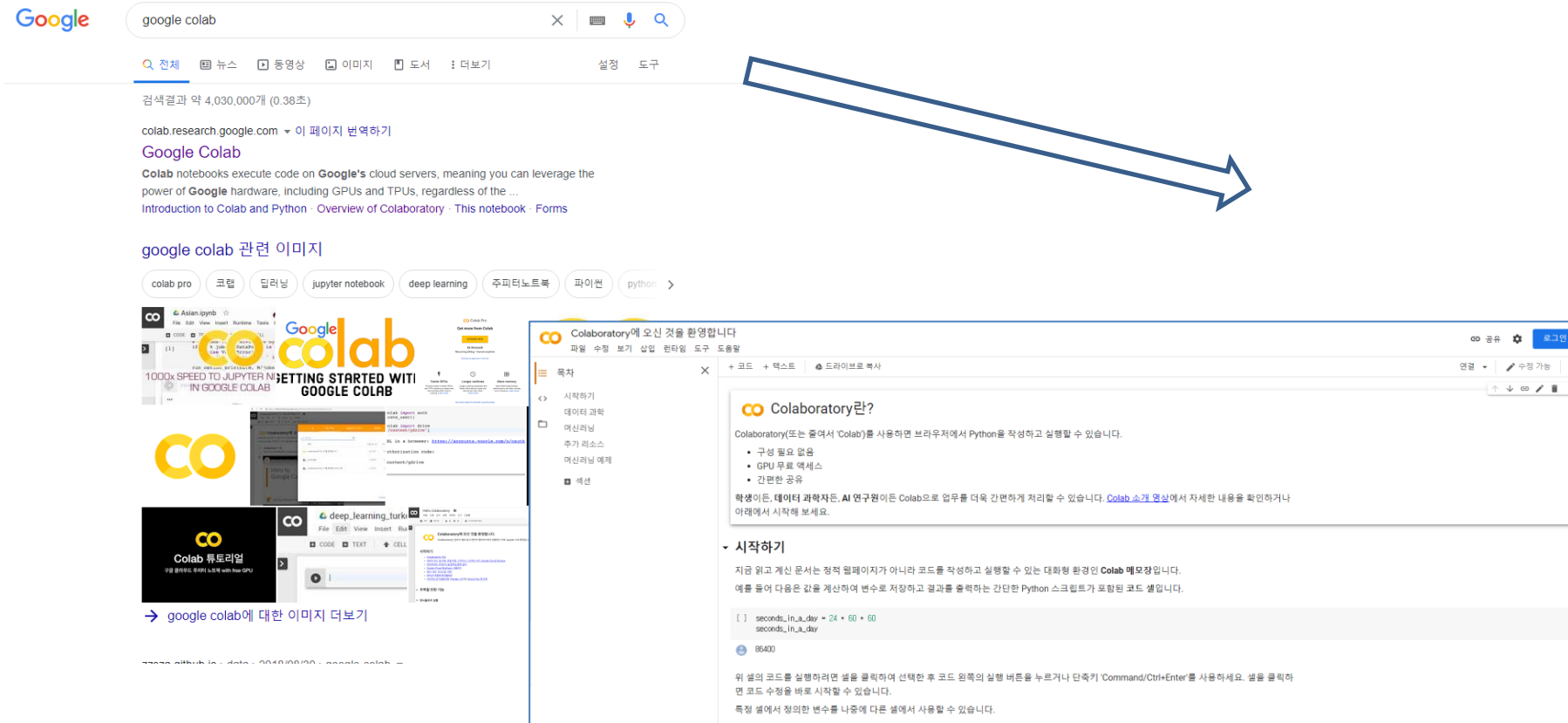
The 2017 TensorFlow Dev Summit

Thousands of people from the TensorFlow community
participated in the first flagship event. Watch the keynote
and talks.

[WATCH VIDEOS](#)

III. 자료구조와 파이썬 문법

- 구글 Colab
 - 클라우드 상에서 GPU를 지원
 - 12시간 세션 시간의 제한
 - 구글 드라이브



III. 자료구조와 파이썬 문법

➤ 데이터 구조?

- 데이터를 일정한 기준에 의해 모아 놓은 것 (collection of data elements, structured in some way)

➤ List

- 여러 같은 type의 값을 하나의 객체로 모음
- 관련 함수: methods (object에 관련된 함수): len, max, min
- object.method(arguments) 방식으로 사용: lst.append(값), lst.count(값), lst.insert(위치, 값), lst.pop(), lst.reverse(), lst.remove(값), lst.sort()

III. 자료구조와 파이썬 문법

➤ List

- 숫치형 값 또는 문자열 등 여러 값의 모임
 - indexing: Sequence 내의 element에 대해 순서 별 index로 접근
 - slicing: 일정 범위 내의 element를 access하는 것
- 값의 추가: 같은 type의 값을 추가

III. 자료구조와 파이썬 문법

➤ Set

- list와 같은 값의 모임
- 값의 순서가 없음, 집합으로 이해
- { }로 생성

III. 자료구조와 파이썬 문법

➤ Tuple

- 다른 형태의 sequence, list와 유사, 0 index에서 시작하는 값
- (, ,)로 표현
- Immutable sequence: 일단 생성 후에는 값을 변경할 수 없음. 문자열과 유사
- sort, append, reverse 등의 method는 사용 불가

III. 자료구조와 파이썬 문법

➤ Dictionary

- 각 값마다 이름이 있는 값의 모음(A "bag" of values, each with its own label)
- 파이썬의 유용한 기능으로, key-value 구조를 지원
- Dictionary는 { }를 이용하여 생성 / 값으로는 key:value 쌍을 , 로 나열
- 빈 dictionary는 {}를 통해 생성

III. 자료구조와 파이썬 문법

➤ 조건문과 반복문

- 조건문: 조건의 참/거짓여부에 따라 문장을 각각 실행, if else, if elif else를 사용
- 반복문: 파라미터 값을 변경하면서 동일한 작업을 여러 번 실행, for 사용
- 조건문-bool을 사용하여 조건 충족 여부 판단
- block은 { }로 표현하지 않고 indentation으로 표현(참고: 한 레벨당 공백 4개)

III. 자료구조와 파이썬 문법

➤ 조건문과 반복문

```
var = 100
```

```
if var < 200:
```

if에 대한 block

```
    print("Expression value is less than 200")
```

```
    [ if var == 150: ] 하위 block
```

```
        print("Which is 150")
```

```
    elif var == 50:
```

```
        print("Which is 50")
```

```
    elif var < 50:
```

```
        print("Expression value is less than 50")
```

```
    else:
```

```
        print("Could not find true expression")
```


III. 자료구조와 파이썬 문법

➤ 조건문과 반복문

- 반복문-while과 for를 사용
- for: set/sequence(또는 iterable object(iterate가능 객체)의 각 값에 대해 Code block 수행

```
x =1
```

```
while x <=100:
```

```
    print(x)
```

```
    x+=1
```

```
words = ['this','is','a','wonder']
```

```
for word in words:
```

```
    print(word)
```

III. 자료구조와 파이썬 문법

➤ 조건문과 반복문의 들여쓰기

- 같은 level의 statement들은 같은 들여쓰기(indentation)값을 갖아야 함

```
if a==1:
```

```
    print(1)  
    print(0)
```

X

```
if a==1:
```

```
    print(1)  
    print(0)
```

O

III. 자료구조와 파이썬 문법

분기(Branch):

일방향적인 프로그램의 흐름이 어느 지점에서 2개 이상으로 나뉘게 됨

if : 분기(branch)를 위한 명령어

사용 방법:

if 조건식:

코드

코드

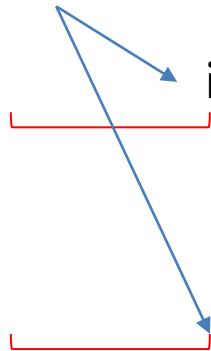
코드

└──────────┘

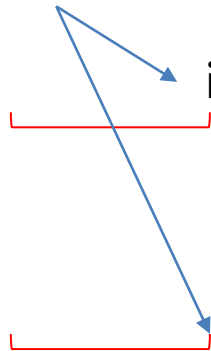
들여쓰기!

조건문의 중첩

if 반짝이는 것 == 동전:

if 반짝이는 것 == 500원짜리 동전:

줍기

if 반짝이는 것 < 500원짜리 동전:

그냥 가던 길 가기

III. 자료구조와 파이썬 문법

else 사용 방법:

if 조건식:

코드1

else:

코드2

True & True는 True
True & False는 False
False & True는 False
False & False는 False

True | True는 True
True | False는 True
False | True는 True
False | False는 False

여러 개의 조건식

if 조건식1 &(또는 |) 조건식2:

코드

if 조건식1 &(또는 |) 조건식2:

코드1

else

코드2

if 국어>90점 & 수학>90점:

피자

Else:

식빵

III. 자료구조와 파이썬 문법

for + if 사용

- ② 1에서 선택된 값이 임시 변수에 임시로 할당
- ① 시퀀스 자료구조에서 값이 하나씩 선택됨

for 임시변수 in 시퀀스자료구조:

- ③ 실행할 코드
임시변수에 조건문을 적용

시퀀스 자료구조의 모든 값이 다 사용될때 까지 이 과정이 반복

for i in sales:

if i > 50:

들여쓰기!

print(i)

또 들여쓰기!

III. 자료구조와 파이썬 문법

while 반복문

while 반복문은 조건식으로만 동작하며 반복할 코드 안에 조건식에 영향을 주는 변화식

사용방법

초기식

while 조건식:

반복할 코드

변화식

```
i = 0                # 초기식
while i < 10:        # while 조건식
    print('펏하 ' )  # 반복할 코드
    i = i + 1        # 변화식
```

For+리스트

```
a = [1,2,3,4]
result = []
for num in a:
    result.append(num*3)
print(result)
```



사용방법:
[표현식 for 항목 in 반복가능객체 if 조건]

```
result = [num * 3 for num in a]
print(result)
```

```
result = [num * 3 for num in a if num % 2 == 0]
print(result)
```


III. 자료구조와 파이썬 문법

함수(Function)

반복적으로 계속 사용되는 일정한 작업을 수행하는 코드들의 모음

함수: 함수이름() 의 형식으로 사용

예: 함수 객체 확인하는 함수
globals()

함수의 입력과 출력

입력값없는 경우: 함수명()

입력값있는 경우: 함수명(입력값, 입력값2, ...)

출력값 있는 경우: 함수 내 return을 통해 반환

결과값 받는 변수 = 함수명()

III. 자료구조와 파이썬 문법

➤ 함수 정의

- def 라는 예약어와 함께 정의
- 함수의 사용은 함수명(입력값)의 형태

```
def Times(a,b):  
    return a*b
```

- 함수 객체 확인: globals()
- return 없으면 None 반환

```
def setValue( newValue):  
    x=newValue  
  
retval = setValue(10)  
print(retval)
```

III. 자료구조와 파이썬 문법

모듈, 패키지, 라이브러리

모듈: 특정 기능을 .py 파일 단위로 작성

패키지: 특정 기능과 관련된 여러 모듈을 묶음. 패키지는 모듈에 네임스페이스(namespace, 이름공간)를 제공

네임스페이스?

패키지1: 모듈1

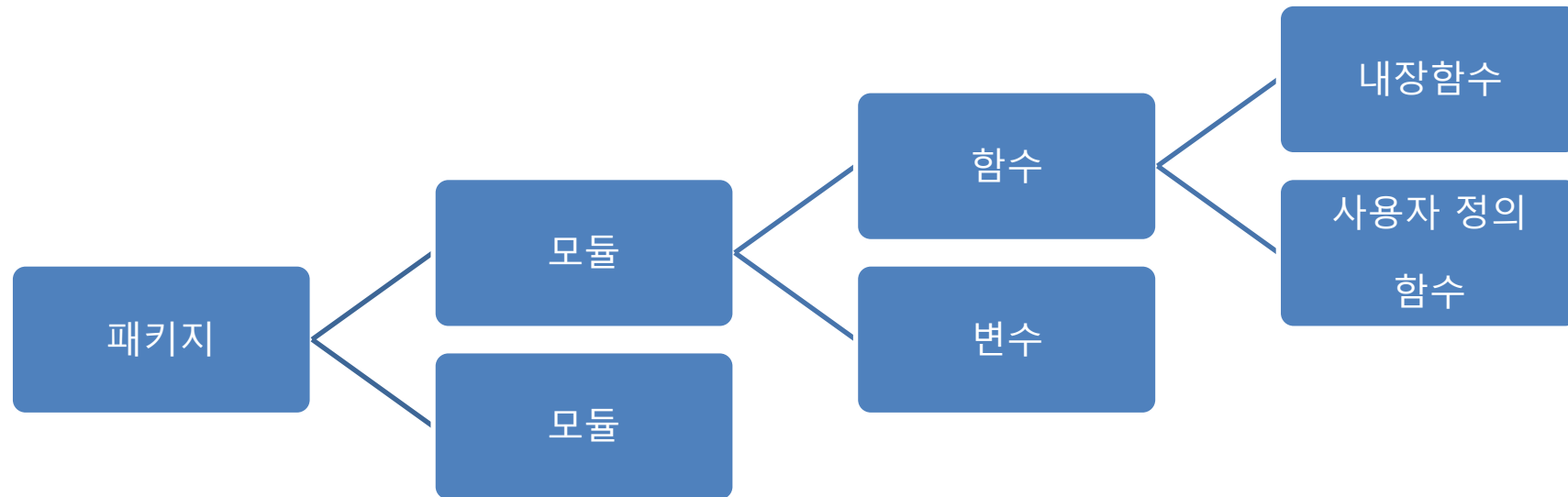
패키지2: 모듈1

패키지1의 모듈1을 쓰고 싶다면!

패키지1.모듈1

III. 자료구조와 파이썬 문법

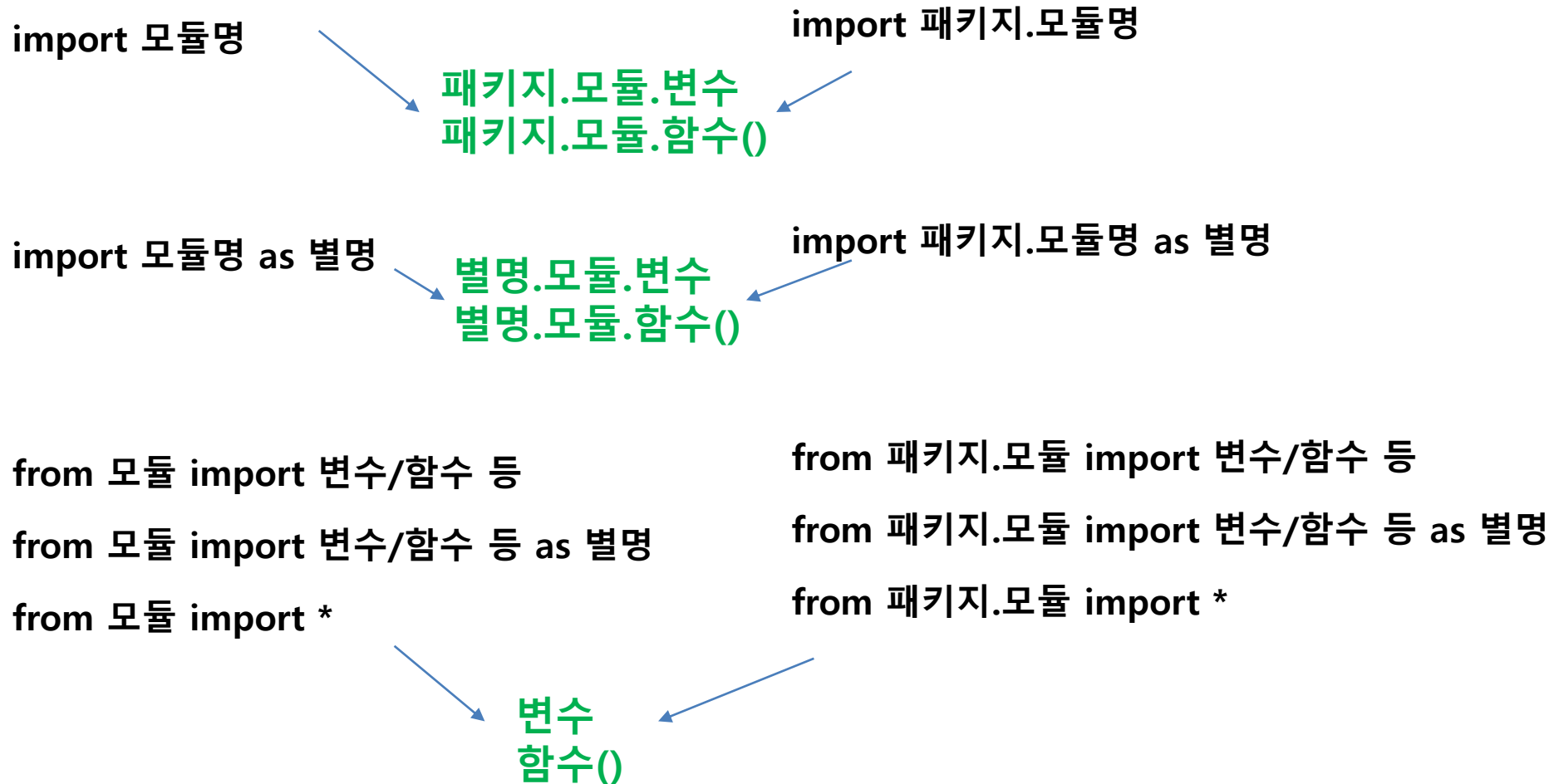
모듈, 패키지, 라이브러리



파이썬 표준 라이브러리: 파이썬에 기본으로 설치된 모듈과 패키지, 내장 함수를 통칭하여 파이썬 표준 라이브러리

III. 자료구조와 파이썬 문법

모듈이나 패키지를 불러오는 3가지 방법



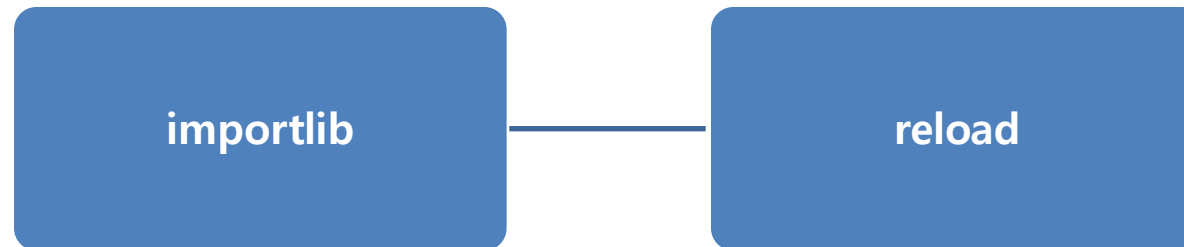
III. 자료구조와 파이썬 문법

모듈 해제와 다시 불러오기

`del 모듈`

`import importlib`

`importlib.reload(모듈)`



모듈 설치하기

pip ?

파이썬 패키지 인덱스(Python Package Index, PyPI)의 패키지 관리 명령어

패키지 설치/삭제/업데이트 지원

쥬피터노트북에서 터미널-> **pip install 패키지 (또는 모듈) 이름**

예: **pip install mlxtend**

문자열(String)

따옴표, 또는 홑따옴표로 둘러싼 문자들의 집합

아무리 수치값이어도 문자열로 표현되면 더 이상 숫자가 아님!

"문자열은 어렵나요?"

'문자열은 사용하면 안되요'

'I can speak English'

'123'

파일 열고 값 출력

내장함수 open 사용

이름=open(파일명, 파일열기모드)

*이름은 Open한 파일의 실제 값을 나타내는 역할을 함(객체)

파일열기모드

- 1) r: 읽기
- 2) w: 쓰기(기존 내용을 무시!)
- 3) a: 추가하기(기존 내용에 추가!)

파일 열고 출력의 3단계

1. 파일 열고 값 출력

```
file1=open( "file1.txt", 'r')
```

```
file1=open( "file1.txt", 'w') #해당 파일명이 없다면 파일이 생성됨
```

```
file1=open( "C:\\file1.txt", 'r')
```

2. 열린 파일에 값 쓰기

```
f.write(값)
```

3. 파일 열고 작업 후에는 파일을 닫기

```
file1.close()
```

파일 입력: 제공된 파일을 읽기

파일을 열은 후,

파일 객체.readline()

또는 파일 객체.readlines()를 이용

또는 파일 객체.read()를 이용

파일 읽기 3단계

1. 파일 열기 : r 모드
2. 파일 내용 읽어서 할당
3. 파일 닫기

with 사용

```
with open( " 파일명 " , " w " ) as file1:  
    file1.write( " 파일 닫기, 거참 번거롭구먼 허허")
```

III. 자료구조와 파이썬 문법

예외 처리: 오류 발생 시 적용

try except 사용

```
try:
    실행할 구문(들)
except:
    실행할 구문
```

```
try:
    실행할 구문(들)
except 발생 오류 코드1:
    실행할 구문(들)
except 발생 오류 코드2:
    실행할 구문(들)
except 발생 오류 코드3:
    실행할 구문(들)
```

오류 그냥 넘어가기: pass

array

▮ 1차원, 2차원, 3차원, ... 가능

▮ +, -, *, / 계산 지원

참고. arange 함수: 일정한 범위내 일련의 정수를 갖는 array 생성

```
import numpy as np
```

```
arr4 = np.arange(10)
```

#0부터 9까지 값을 갖는 1차원 array

```
arr5 = np.arange(3,10)
```

#3부터 9까지 값을 갖는 1차원 array

array의 다차원!

1차원 / 2차원 / 3차원

```
import numpy as np
```

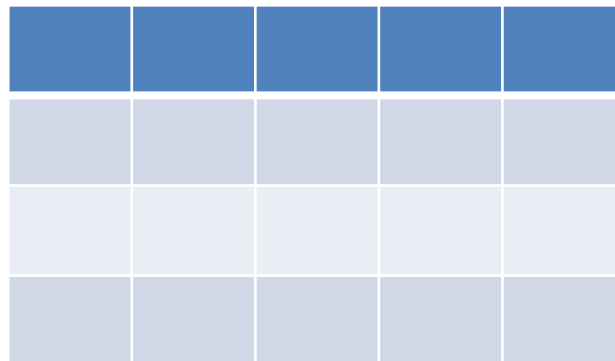
```
arr4 = np.array( [1,2,3] )
```

```
arr5 = np.array( [ [1,2,3], [4,5,6] ] )
```

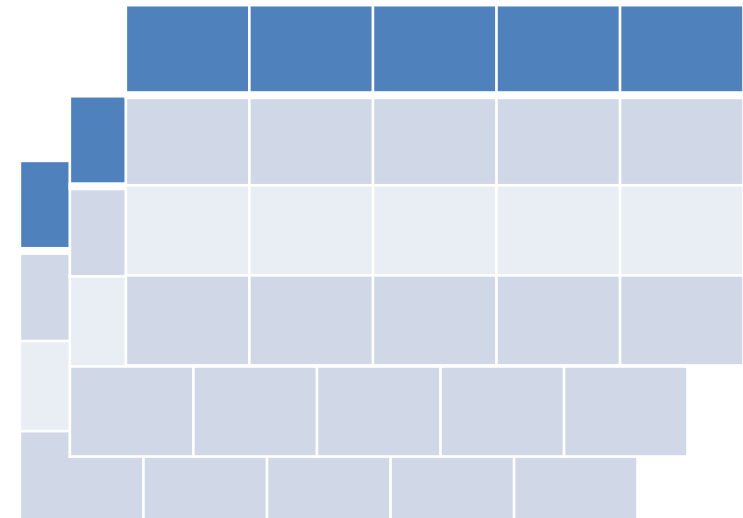
1차원



2차원



3차원



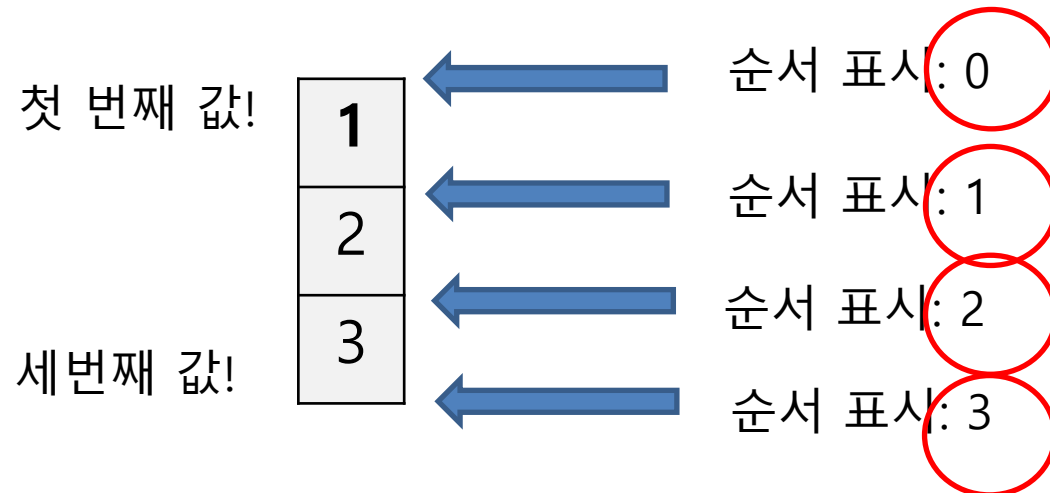
array 값들의 순서!

```
import numpy as np
```

```
arr4 = np.array( [1,2,3] )
```

```
arr4.tolist() #array의 list 변환 함수!
```

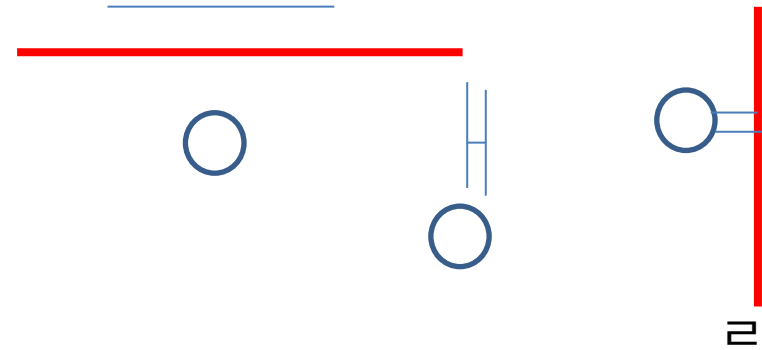
값의 순서가 있다!



array 값들의 순서!

```
import numpy as np
```

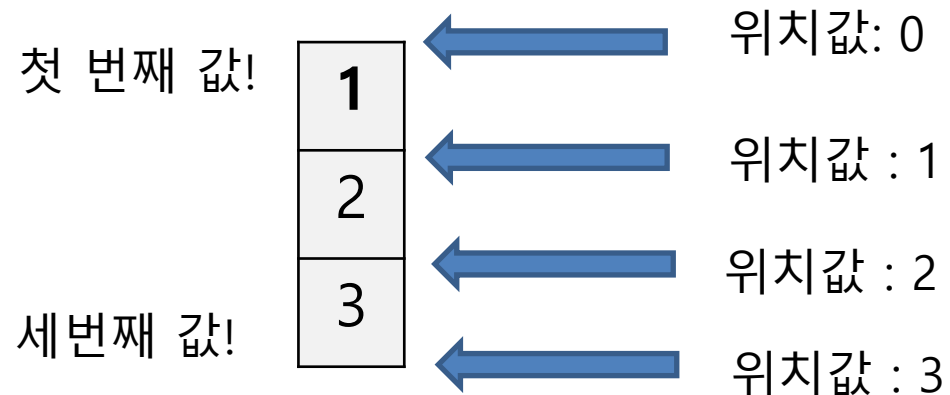
```
arr5 = np.array( [ [1,2,3], [4,5,6] ] )
```



| | | | | |
|---------|----------|----------|----------|------------|
| | 첫 번째 열! | 두 번째 열! | 세 번째 열! | |
| 첫 번째 행! | 1 | 2 | 3 | ← 순서 표시: 0 |
| 두 번째 행! | 4 | 5 | 6 | ← 순서 표시: 1 |
| | | | | ← 순서 표시: 2 |
| | ↑ | ↑ | ↑ | ↑ |
| | 순서 표시: 0 | 순서 표시: 1 | 순서 표시: 2 | 순서 표시: 3 |

1차원 array의 Indexing

```
import numpy as np
arr4 = np.array( [1,2,3] )
```



[]

arr4의 2번째 값! arr4[1]
1번째, 3번째 값! Arr4[[0,2]]

2차원 array의 Indexing

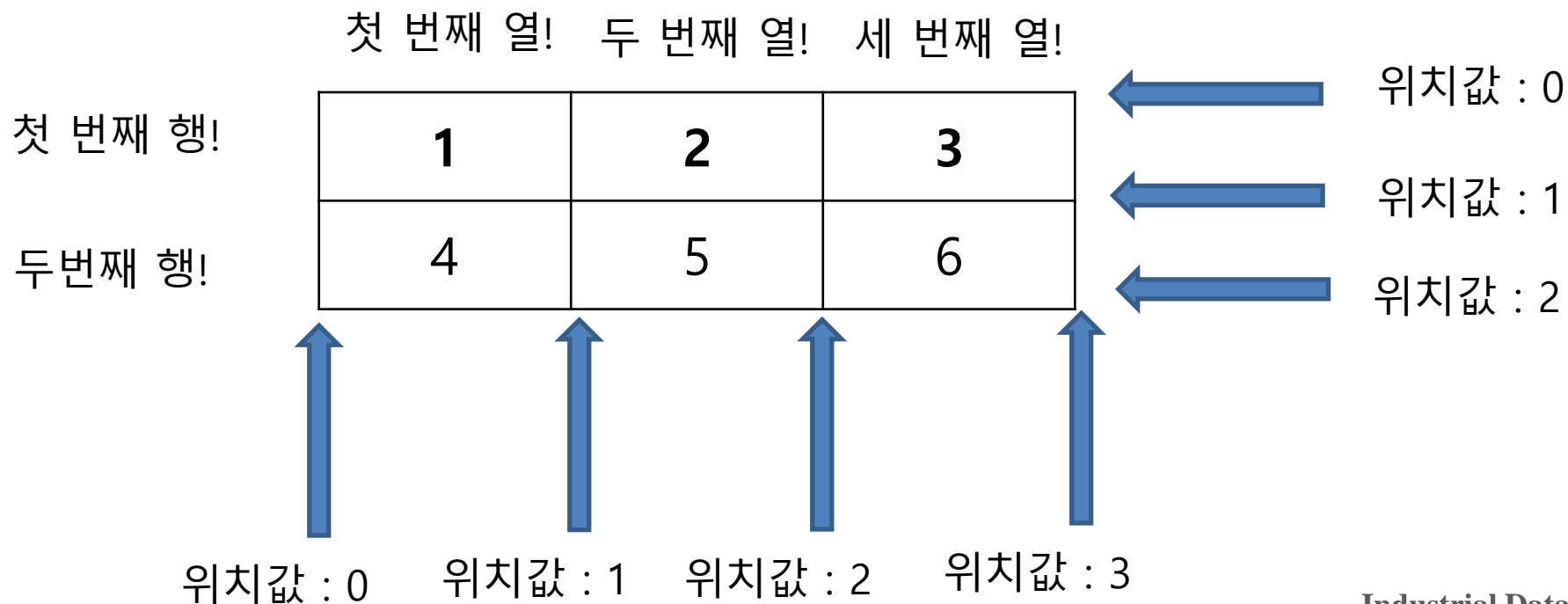
```
import numpy as np
```

```
arr5 = np.array( [ [1,2,3], [4,5,6] ] )
```

```
arr5[ 0 , :] #첫번째 행
```

```
arr5[ : , 1] #두번째 열
```

```
arr5[ 0 , 1] #첫번째 행, 두번째 열
```



다차원 array에서의 Slicing



Slicing에 사용되는 연산자!

[:]

Slicing에 : 의 사용!

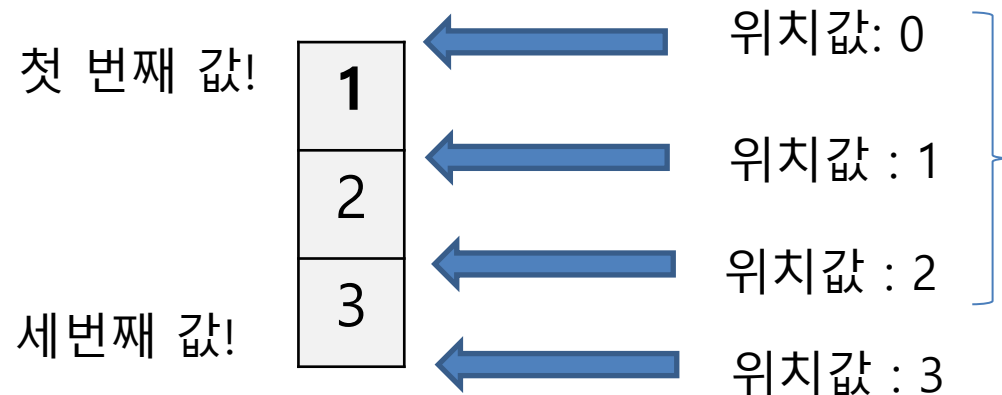
시작되는 위치:끝나는 위치

Slicing 사용 방식!

array이름[시작되는 위치값:끝나는 위치값]

1차원 array의 slicing

```
import numpy as np  
arr4 = np.array( [1,2,3] )
```



```
arr4의 1~2번째 값! arr4[ 0:2 ]  
arr4의 1~2번째 값! arr4[ 0:3 ]
```

2차원 array의 Slicing

```
import numpy as np
```

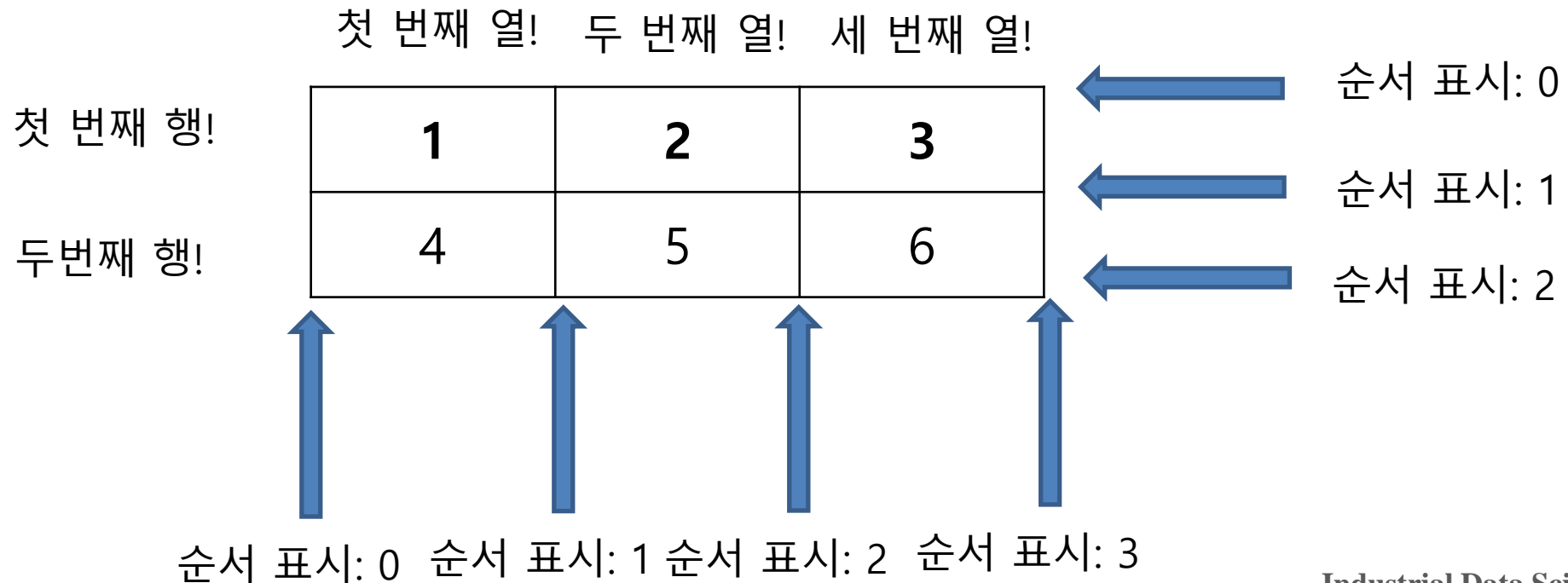
```
arr5 = np.array( [ [1,2,3], [4,5,6] ] )
```

```
arr5[ 0:1 , 1:3]
```

#1행, 2,3열

```
arr5[ 0:2 , 1:2]
```

#1,2행, 2열



array

array에 Scalar를 사칙연산이 가능

`arr4 * 3`

`arr5 + 1`

`arr4 - 3`

`arr5 / 2`

계산 결과를 새로운 array에 할당(assignment)

`arr6 = arr4*3`

array간 연산(1차원)

array 간 덧셈

`arr8+arr9`

array 간 뺄셈

`arr8-arr9`

array 간 곱셈

`arr8*arr9`

array 간 나눗셈

`arr8/arr9`

`arr8.size`

`arr8.shape`

`arr8.ndim`

주요 통계량 함수를 이용한 array 계산

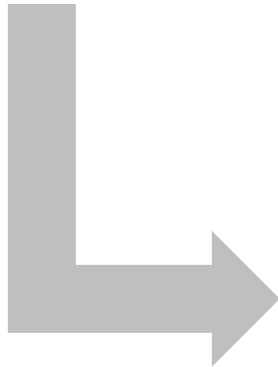
통계량? 주어진 자료의 특성을 나타내는 요약값

```
import numpy as np
```

```
arr12 = np.array( [3,5,2,4] )
```

```
arr13 = np.array( [3.1,5.2,4.5] )
```

```
arr14 = np.array( [-2,3,1,10] )
```



예: 평균, 편차, 최대, 최소값 등

평균: `np.mean(arr12)`

표준편차: `np.std(arr12)`

최대: `np.max(arr13)`

최소: `np.min(arr13)`

주요 통계량 함수를 이용한 array 계산

그 외에도 다양한 수학함수를 지원

sum, prod, abs, sqrt, exp, log 등등...

절대값: `np.abs(arr14)`

제곱근: `np.sqrt(arr12)`

지수: `np.exp(arr13)`

로그: `np.log(arr13)`

지수? Exponential!

로그? Log!

$$e = 2.71828182846$$

pi와 e를 다음처럼 미리 정의
 $pi = 3.14159265359$
 $e = 2.71828182846$

➤ pandas

- 데이터 처리를 위한 최고의 패키지!
- 데이터 처리 기능을 풍부하게 제공
- 빠르고 직관적인 자료 구조!

➤ 특징

- DataFrame 제공
- 엑셀, CSV 등을 포함한 다양한 포맷 지원
- 데이터 정렬과 결측치 처리, 피벗 등 기능!
- 고급 인덱싱과 슬라이싱
- 데이터 합치기와 형태 변경 지원!



IV. Numpy와 Pandas

➤ Pandas의 데이터 구조

➤ 데이터 구조

➤ Series (1D)

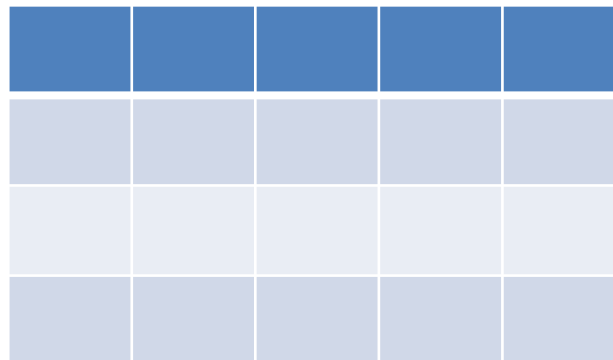
➤ DataFrame (2D)

➤ Panel (3D)

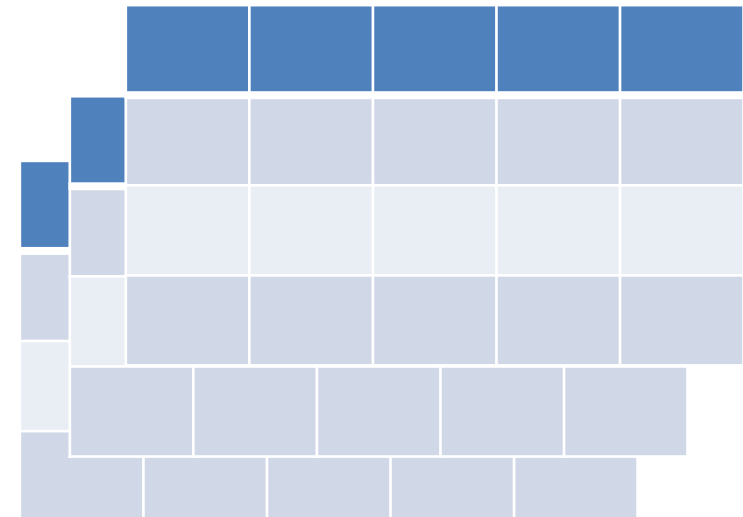
Series



DataFrame



Panel



IV. Numpy와 Pandas

➤ Series

- 1차원
- Subclass of numpy.ndarray
- Index, Values를 갖음
- Index는 원하는 값을 넣을 수 있음-시계열

| index | | values |
|-------|---|--------|
| A | → | 5 |
| B | → | 6 |
| C | → | 12 |
| D | → | -5 |
| E | → | 6.7 |

IV. Numpy와 Pandas

➤ DataFrame

- 행과 열이 있는 2차원
- 열마다 값의 타입이 다를 수 있음!
- 행, 열 모두 인덱스
- 크기가 바뀔 수 있음-행, 열 추가 및 삭제!

The diagram illustrates a DataFrame structure. At the top, a box labeled 'columns' contains the headers '국', '영', '수', and '출석'. Below this, a box labeled 'index' contains the row labels '갑', '을', '병', '정', and '무'. Arrows point from the column headers to the corresponding columns of data, and from the index labels to the corresponding rows of data. The data is organized into a grid of cells.

| | columns | 국 | 영 | 수 | 출석 |
|-------|---------|-----|----|----|-------|
| index | 갑 | 100 | 40 | 55 | True |
| 을 | → | 90 | 70 | 33 | True |
| 병 | → | 50 | 50 | 88 | False |
| 정 | → | 70 | 78 | 99 | False |
| 무 | → | 94 | 44 | 88 | False |

- **Series**
 - pandas에서 제공하는 1차원 자료 구조 , 연산, 인덱싱, 슬라이싱을 포함해 다양한 기능을 지원
- **Series의 인덱싱과 슬라이싱**
 - Series는 Index와 Value를 나누어 활용할 수 있으며, 이때 Index Label이나 Array를 이용한 인덱싱, 슬라이싱을 지원
- **Series 연산**
 - 수치 계산 및 벡터화 계산을 지원하며, Series 간 연산 시 같은 Index 값이 계산

IV. Numpy와 Pandas

➤ Series

- 1차원
- Index, Values를 가짐

| index | values |
|-------|--------|
| A | 1 |
| B | 3 |
| C | 5 |
| D | 7 |
| E | 9 |

IV. Numpy와 Pandas

➤ Series 생성하기

➤ 1차원 시퀀스 형태의 자료구조(예: 리스트 array)로 부터 생성 가능

➤ **Series = Value + Index**

➤ Index Label: Index에 원하는 값을 지정할 수 있음

```
import numpy as np
import pandas as pd

S1 = pd.Series( [1,2,3] )

S2 = pd.Series( np.array([1,2,3]))

S3 = pd.Series( [80,90,75], index=["가","나","다"] )
```

➤ **Series명.values**, 예를 들어 **S1.values**

➤ **Series명.index**, 예를 들어 **S3.index**

IV. Numpy와 Pandas

➤ Series 인덱싱과 슬라이싱

- Index Label을 활용해서 원하는 범위의 값을 선택할 수 있음
 - Index가 지정 안 된 경우: 첫번째값이 0부터 시작하는 정수 index를 갖음
 - Index가 지정된 경우: 별도로 지정한 Index Label을 사용할 수 있으며, 첫번째값이 0부터 시작하는 정수 index도 사용할 수 있음

```
S4 = pd.Series( [1,2,3,4,5] )  
  
S5 = pd.Series( [1,2,3,4,5], index=["a","b","c","d","e"] )  
  
S4[0] # S4의 첫번째 값  
S4["a"] # 오류, S4에는 별도로 지정된 index 없음  
  
S5[0] # S5의 첫번째 값  
S5["a"] # S5의 첫번째 값  
  
S4[ 0:3 ] #S4의 1~4번째 값  
S5[ ["a", "c", "e"] ] #S5의 1,3,5번째 값 출력
```

IV. Numpy와 Pandas

➤ Series 연산

➤ Series는 수치 계산을 지원

➤ $+, -, *, /$

➤ Series 간 연산은 같은 Index 값끼리 계산

```
S4 = pd.Series( [1,2,3,4,5], index=["a","c","d","b","e"] )
```

```
S5 = pd.Series( [1,2,3,4,5], index=["a","b","c","d","e"] )
```

S4+S5

S4

| values | index |
|--------|-------|
| 1 | a |
| 2 | c |
| 3 | d |
| 4 | b |
| 5 | e |

S5

| index | values |
|-------|--------|
| a | 1 |
| b | 2 |
| c | 3 |
| d | 4 |
| e | 5 |

- **DataFrame**
 - pandas에서 제공하는 2차원 자료 구조, 연산, 인덱싱, 슬라이싱을 포함해 다양한 기능을 지원
- **DataFrame의 인덱싱과 슬라이싱**
 - DataFrame은 행은 슬라이싱을, 열은 인덱싱을 지원할 수 있으며, 추가적인 함수를 통해 인덱싱과 슬라이싱을 자유롭게 사용 가능.
- **DataFrame 활용**
 - 수치 계산 및 벡터화 계산을 지원하며, 자료처리를 위한 다양한 함수를 활용할 수 있음

IV. Numpy와 Pandas

➤ DataFrame의 특징!

- 2D table with rows and column labels
- column 별로 서로 다른 dtype을 가질 수 있음
- 행과 열 Index Label을 지원

The diagram illustrates a DataFrame structure with columns and index labels. The columns are labeled 'foo', 'bar', 'baz', and 'qux'. The index labels are 'A', 'B', 'C', 'D', and 'E'. The values for each row are as follows:

| | foo | bar | baz | qux |
|---|-----|-----|-----|-------|
| A | 0 | x | 2.7 | True |
| B | 4 | y | 6 | True |
| C | 8 | z | 10 | False |
| D | -12 | w | NA | False |
| E | 16 | a | 18 | False |

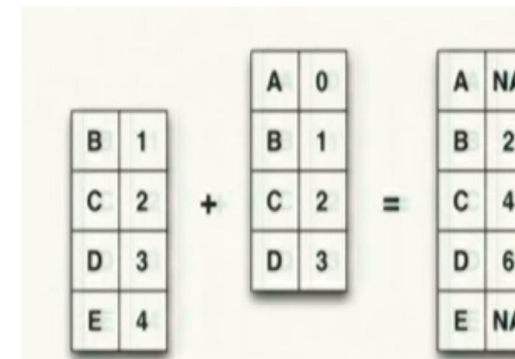
IV. Numpy와 Pandas

➤ DataFrame의 더 많은 특징!

- 데이터의 reshaping 및 pivoting
- Label활용 슬라이싱과 인덱싱, 부분집합 구하기가 쉬움
- Group by 엔진 지원: Data Aggregation/ Data Transformation

➤ Data Alignment

- "Outer join"을 지원: 두 데이터프레임의 행 또는 열 Index를 기준으로 계산!



| | |
|---|---|
| B | 1 |
| C | 2 |
| D | 3 |
| E | 4 |

 +

| | |
|---|---|
| A | 0 |
| B | 1 |
| C | 2 |
| D | 3 |

 =

| | |
|---|----|
| A | NA |
| B | 2 |
| C | 4 |
| D | 6 |
| E | NA |

IV. Numpy와 Pandas

➤ DataFrame 생성하기

- 2차원 형태의 자료구조(예: 리스트의 리스트, 2차원 array)와 Dictionary(Value가 1차원 자료구조인 경우)로 부터 생성 가능
- **DataFrame = Series + Series + Series**
- 행과 열의 Index Label: Index에 원하는 값을 지정할 수 있음!

```
import numpy as np
import pandas as pd

df1 = pd.DataFrame( [[1,2,3], [1,2,3]], columns=['a','b'], index=['r1','r2','r3'] )
df2 = pd.Series( np.array([[1,2,3], [1,2,3]]) , columns=['a','b'],
index=['r1','r2','r3'])
raw_data = {'col0': [1, 2, 3, 4],
            'col1': [10, 20, 30, 40],
            'col2': [100, 200, 300, 400]}
df3 = pd.Series( raw_data )
```

- DF명.values, 예를 들어 df1.values
- DF명.index, 예를 들어 df1.index
- DF명.columns, 예를 들어 df1.columns

➤ DataFrame의 인덱싱과 슬라이싱

- 인덱싱: 열 이름을 지정, 복수인 경우 리스트로 묶어서 지정
- 슬라이싱: 선택을 원하는 행을 슬라이싱 / 단 행 단위에 인덱싱 사용 불가

```
#행
df1[ 0 ] #오류
df1[ 0:1 ]

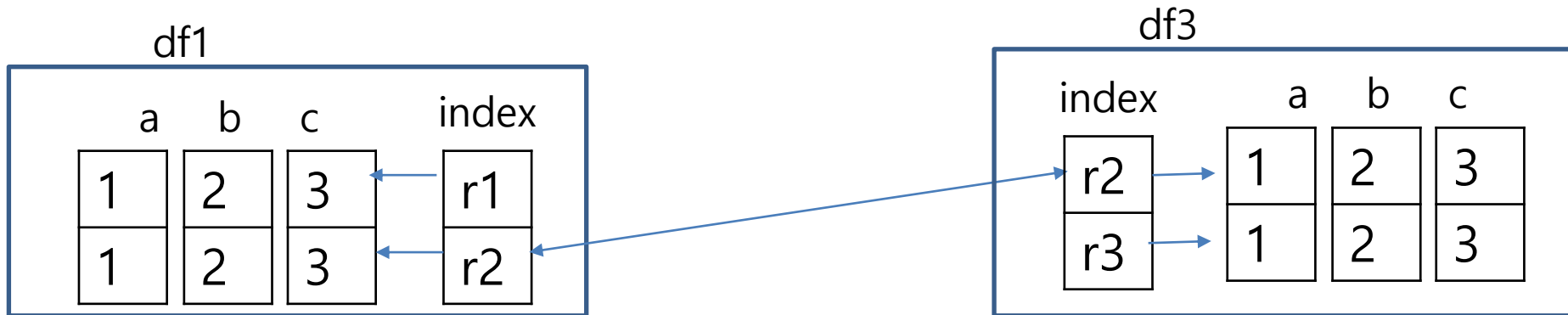
#열
df1['a']
df1[['a','c']]
```


IV. Numpy와 Pandas

➤ DataFrame 활용

- .head(), .describe() 등의 유용한 기능 활용
- 두 데이터프레임의 수치 계산은 같은 인덱스의 해당되는 컬럼끼리 계산!
 - +, -, *, /

```
df3 = pd.DataFrame( [[1,2,3], [1,2,3]], columns=['a','b','c'], index=['r2','r3'] )  
df1+df3
```





Industrial Data Science Lab

Contact:

won.sang.1@gmail.com

<https://sites.google.com/view/idslab>