

# **Big Data Analytics Programming**

## **Recap**

**Jungwon Seo, 2020-Fall**

# 배울 내용

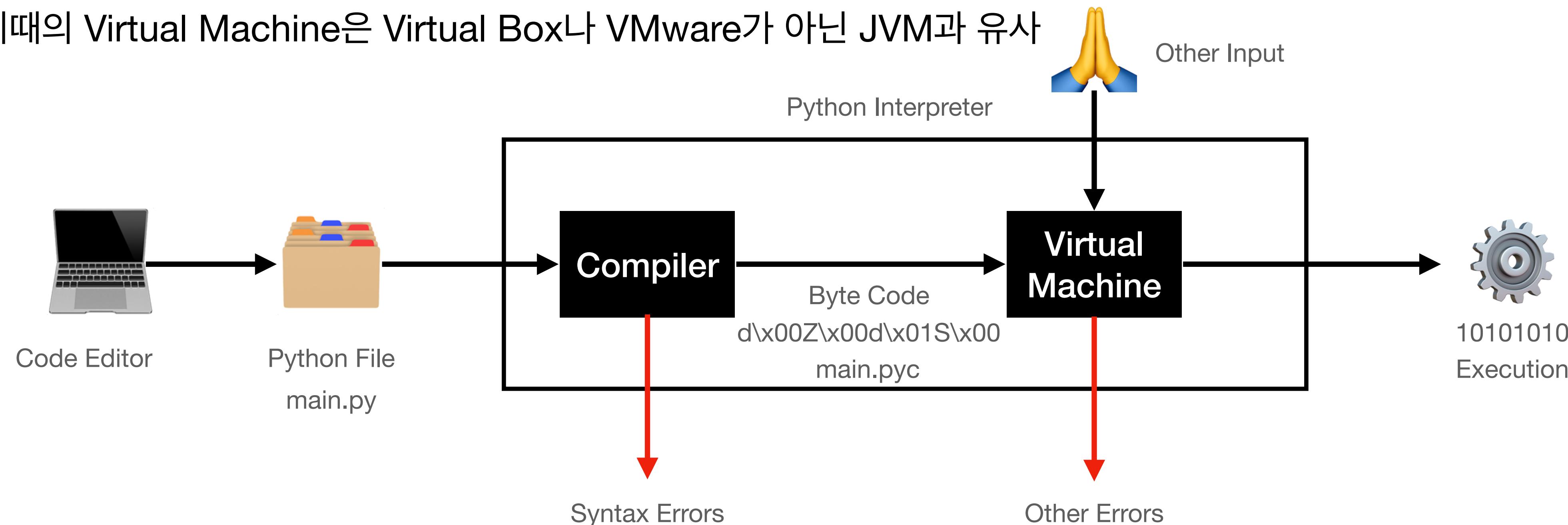
## Week-02. Python Basic

- Python 동작 방식
- 변수와 데이터 타입
- 조건문: if-statement
- 그룹형 데이터 타입
- 반복문: loop-statement

# How Python Works?

코드를 작성했다.. 그 다음은?

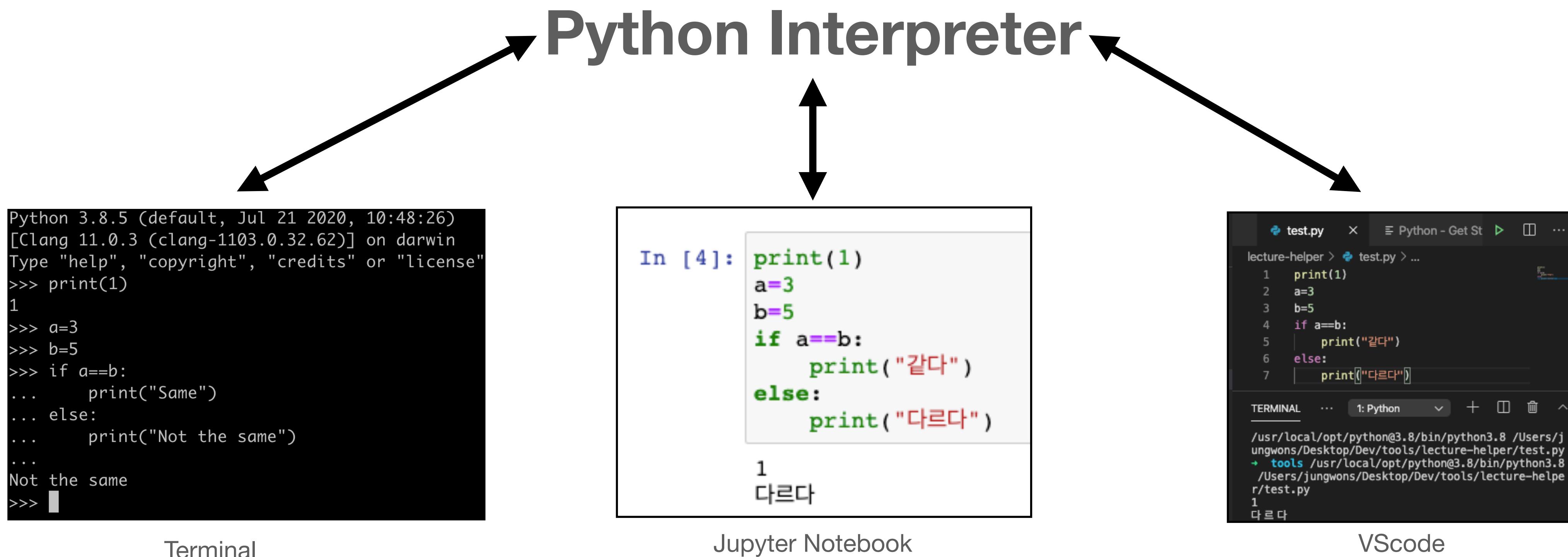
- 작성된 코드는, 파이썬 인터프리터를 통해서 Byte Code로 컴파일 된후 Virtual Machine에 의해 실행이 된다.
  - Virtual Machine에 의해 Python은 OS independent 하게 사용 가능하다.
  - 이때의 Virtual Machine은 Virtual Box나 VMware가 아닌 JVM과 유사



# Python 코딩 환경

Editor는 결국 사람을 위한 것일 뿐..

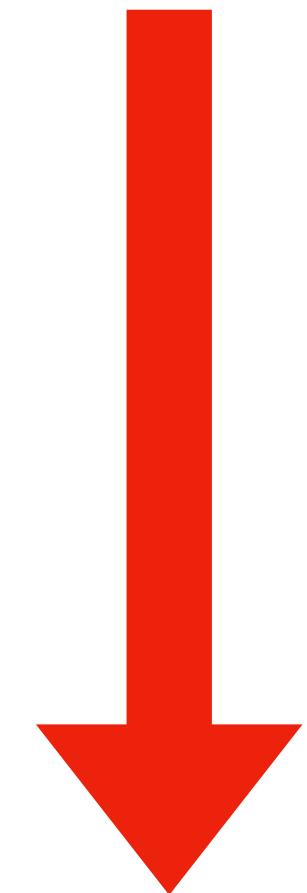
- 어떤 환경에서 작업을 하든, 파이썬 코드는 인터프리터를 통해 동작을 한다.



# 코드 실행 과정

## 머리부터 끝까지

- 기본적으로, 파이썬 코드는 위에서 아래 방향으로 코드가 실행된다.
  - \* 내부적으로는 다른 언어와 마찬가지로 Stack 구조를 가져간다.
- 항상 명심해야 될 것은, 위쪽에서 정의(define) 되지 않은 내용을 아래쪽에서 다룰 수 없다.
  - 오른쪽의 코드는 실행 가능한가?
- “Rules... without them we live with the animals”  
- John Wick 中



```
print("Hello")
a=5
c = a+b
print(c)
```

# Python Errors - Part 1

## 에러의 종류

- **Syntax Error**

- 파이썬 코드가 Byte code로 변환되는 과정중에 감지됨
- 다르게 말해, “문법에 맞지 않게 썼다.” 의 의미

```
a 5
```

```
File "<ipython-input-6-1a22f7fb4338>", line 1
      a 5
      ^
SyntaxError: invalid syntax
```

- **Name Error**

- 해당 변수가 발견되지 않았을 경우
- “처음 보는 변수인데?” 의미

```
print(k)
```

```
-----  
NameError                                         Traceback (most recent call last)  
<ipython-input-7-eb2fa875d160> in <module>  
----> 1 print(k)  
  
NameError: name 'k' is not defined
```

# Python Errors - Part 1

## 에러의 종류

- Type Error

- 어떠한 연산을 할 때, 연산이 되어지는 변수들의 변수형(Type)이 호환되지 않았을 경우
- `1+"가"=?`

```
1+"a"  
-----  
TypeError                                         Traceback (most recent call last)  
<ipython-input-8-4ddca89a012c> in <module>  
      1 1+"a"  
  
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

- Indentation Error

- 파이썬의 경우에는 변수/수행의 범위를 tab을 기준으로 나눔
- 들여쓰기를 잘못 사용

```
if 5==5:  
    print(1)  
  
File "<ipython-input-9-269a27d2924a>", line 2  
    print(1)  
          ^  
  
IndentationError: expected an indented block
```

# 언어를 배울 때 가장 먼저 배우는 것은?

ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ

ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ

ㅏ ㅑ ㅓ ㅕ ㅗ ㅘ ㅜ ㅛ ㅡ ㅣ

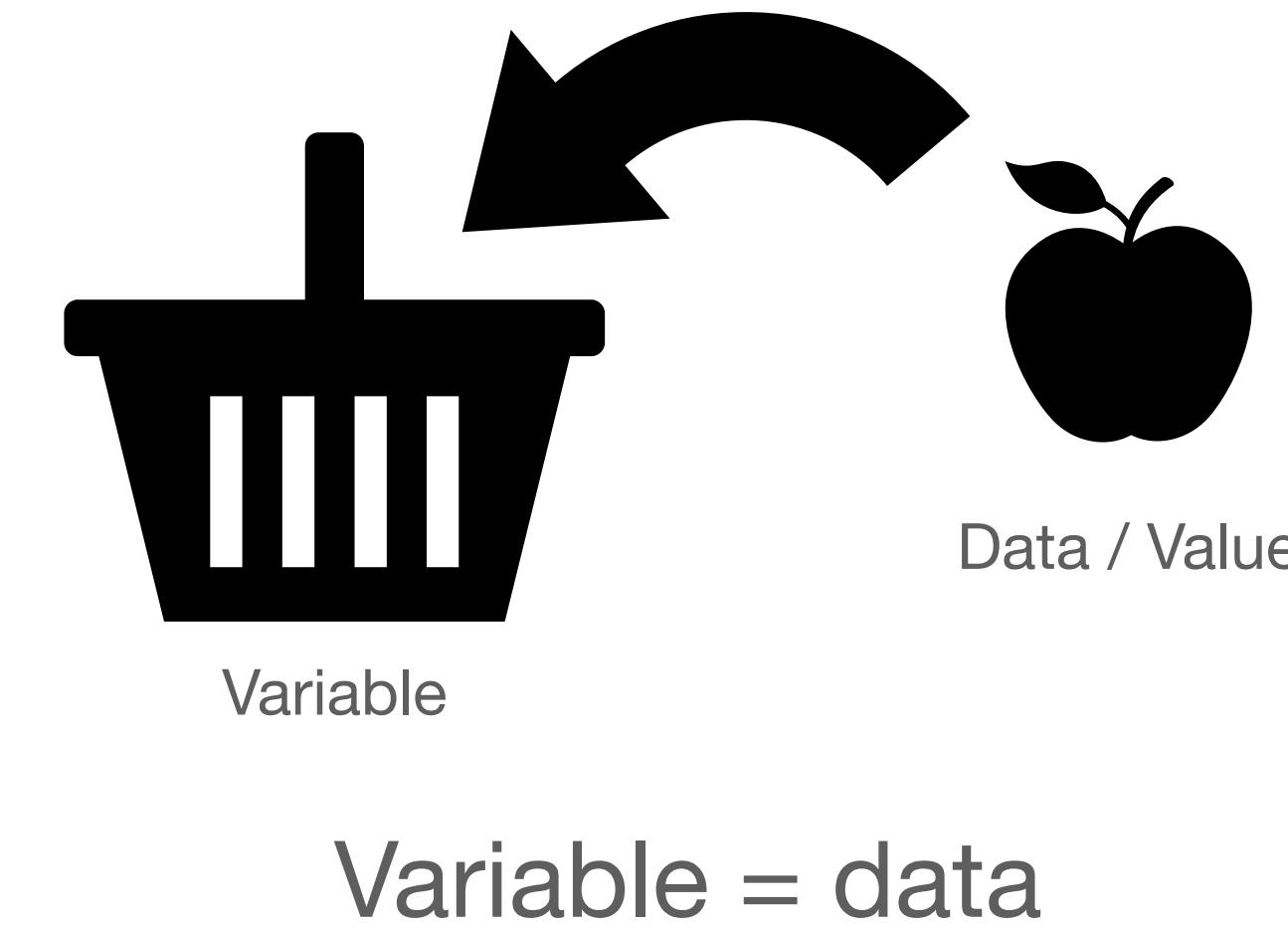
ㅐ ㅒ ㅔ ㅖ

ㅕ ㅖ ㅕ ㅕ

# Python 변수

## Variable과 Data Type

- 변수 (variable): 메모리에 값을 저장(=)하기 위한 명명(Named)된 위치
  - Equal sign(=)은 같다의 의미가 X
  - <변수명> = <값>
  - a = 1
  - my\_name = "Jungwon"
- 동일 한 변수명에 여러번 값을 대입할시,
  - 최종 대입 값으로 대체된다.
- 변수명 작성시 주의할점
  - 숫자로 시작 X, 띄어쓰기 X
  - 한글도 가능, 하지만 아무도 안씀

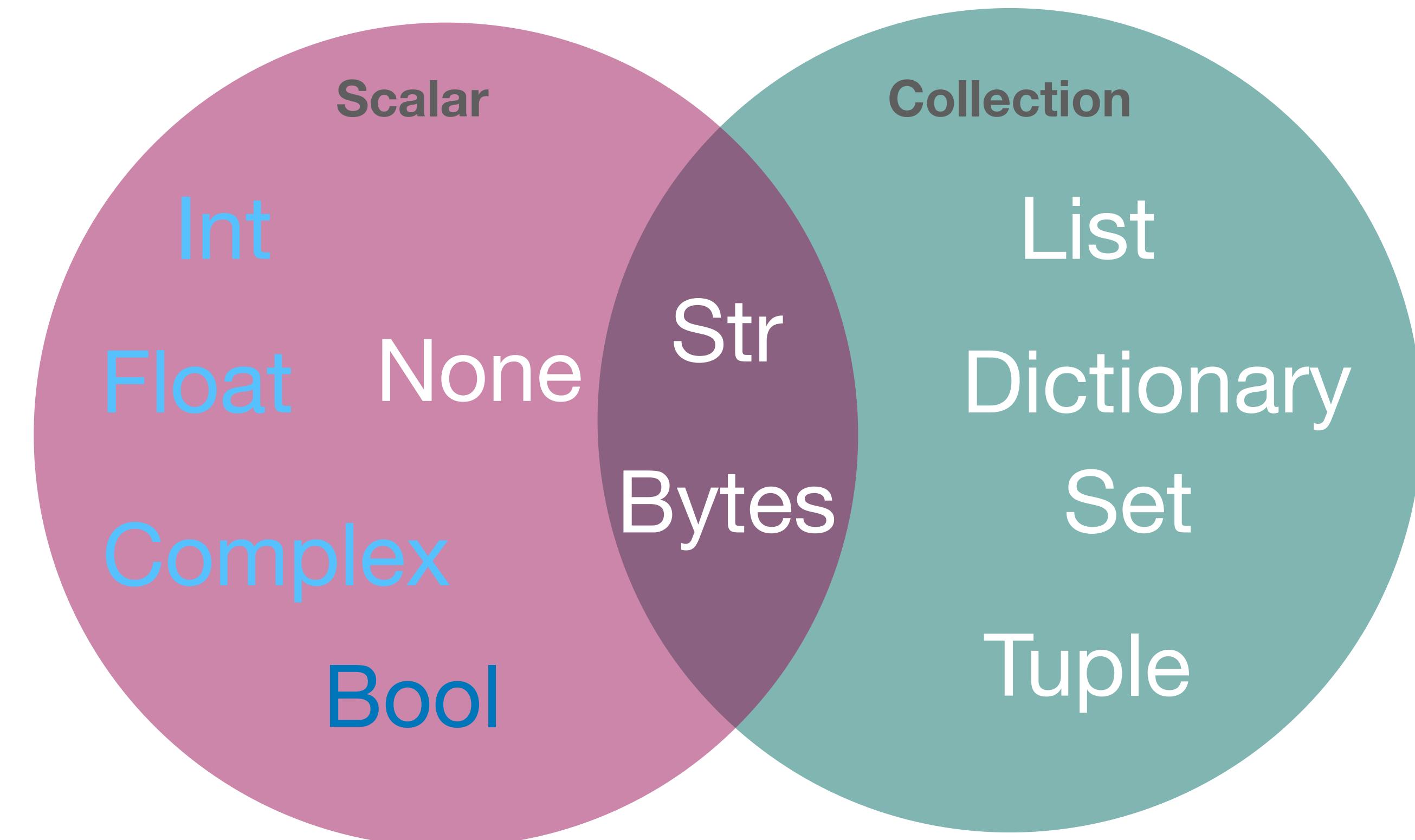


a = 10  
a = "Hello"  
a = True  
print(a)

# Python Built-in Data Type

## Scalar and Collection

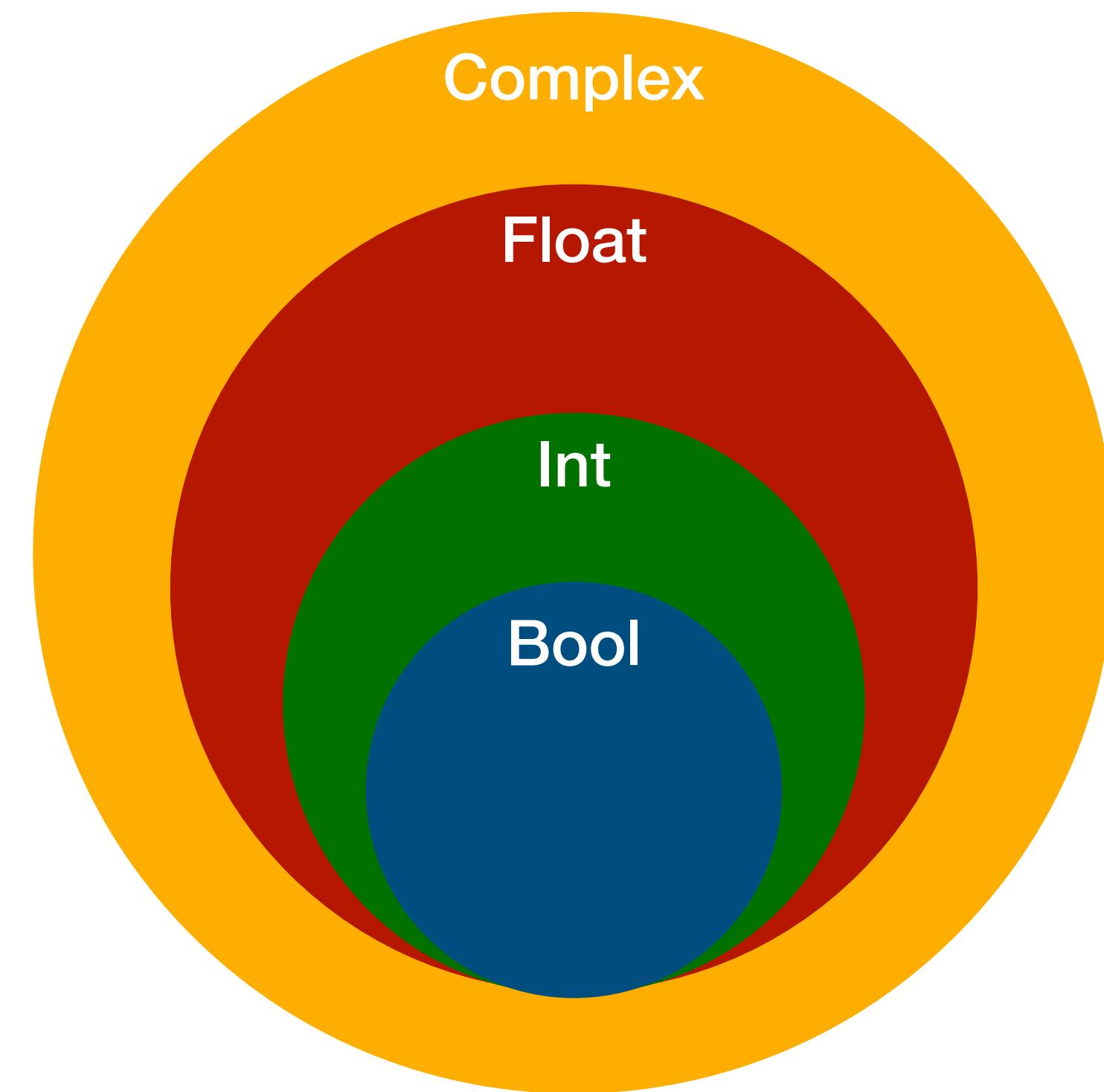
- Scalar: 단일 값을 갖음
- Collection: 값"들"을 갖음



# Python Data Type - Scalar

## 숫자형 Data Type (Numerical)

- 세부 종류: int, float, complex , (bool)
- 사칙연산의 경우 일반 수학적 연산과 동일하게 동작
  - $3 + 3 = 6$ ,  $3 - 3 = 0$ ,  $3 * 3 = 9$ ,  $3 / 3 = 1.0$
  - 추가적인 연산자들
    - %: 모듈러 연산자로 나머지 값(Remainder)을 출력 =>  $5 \% 3 = 2$
    - //: 내림(floor)가 포함된 나누기 연산 = 몫(quotient)을 출력 =>  $5 // 3 = 1$
    - \*\*: 지수(exponential) 연산 =>  $5 ** 2 = 25$
    - &, | : 비트연산 => 이진수로 변환뒤 연산
- 부울형 (boolean) : True or False
  - Bool 변수라고 불리우며, 우리가 흔히 말하는 1과 0
  - 사칙 연산과 논리 연산 제공, 단 사칙연산 적용시 int나 float으로 자동형변환
- 숫자형 변수간의 연산시 형변환
  - 연산변수간의 타입의 다를 경우 또는 연산결과의 타입이 다를 경우 상위 변수형으로 형변환



# Python Data Type - Scalar

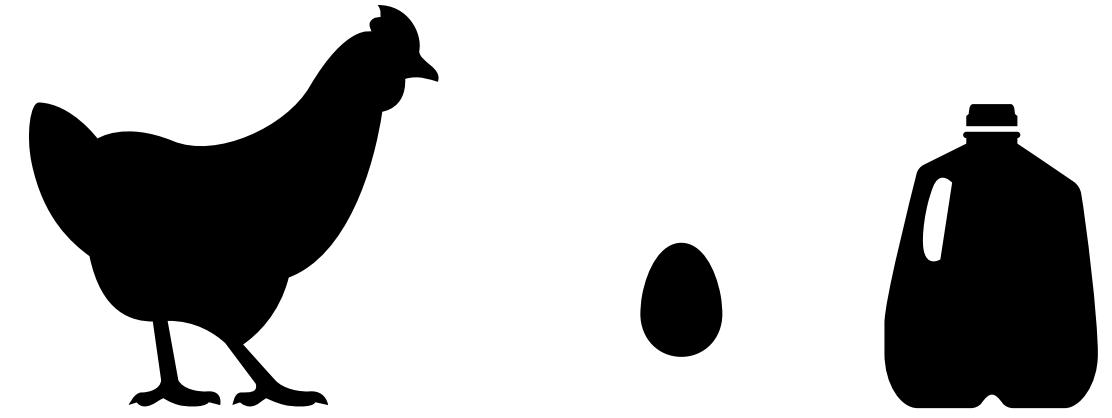
## 문자형 Data Type (String)

- 큰 따옴표(") 또는 작은 따옴표(')로 값을 감싸줘서 표현
  - 1: int, "1":string
  - "abcde", 'abcde'
- 문자형: 더하기(Addition) 연산만 지원
  - 이때의 더하기 연산은 이어쓰기(concatenate)의 효과
  - "hello" + "world" = "helloworld"
- Byte string

# 논리적으로 생각한다는 것은?

부모님께서 프로그래머인 나에게 심부름을 시키셨다.  
"슈퍼가서 우유 하나 사와. 아, 계란 있으면 6개사와"  
그래서 우유를 6개 사갔다.  
"왜 우유를 6개나 샀어?"  
"계란이 있길래.."

- From 인터넷 어딘가..

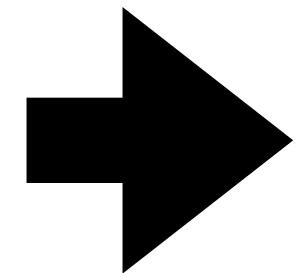


# 조건문(conditional statement)

What if... anything else?

- 사람은 항상 특정한 조건/상황에 기반해서 판단
- 세상에서 일어나는 모든 상황을 Logically 코드로 표현 가능

If I were a boy  
Even just for a day  
I'd roll outta bed in the morning  
and throw on what I wanted and go



```
if user.gender == "boy" and user.duration >= "1":  
    user.wake_up(from="bed", when="morning")  
    user.wear(style="whatever")  
    user.go_out()
```

- Beyonce, If I Were A Boy 中

# 조건문(conditional statement)

## If문 사용법

- 조건문 작성법
  - if 라는 예약어로 시작
  - If 뒤에는 True or False로 나올 수 있는 조건
  - 해당 하는 조건의 ending 은 콜론 (:)으로 마무리
  - 그 조건문이 "만족" 됐을 때 동작할 코드는 들여쓰기(indentation) 이후 작성
  - If 안에서만 동작해야할 코드는 들여쓰기를 유지

The diagram illustrates the structure of an if statement. It starts with the word 'if' in blue, followed by a red box containing '<condition>' and a colon ':'. Below this, three red boxes each contain '<do something>'. An upward-pointing arrow from the text 'Tab!!' to the first red box indicates that the code block below it should be indented.

```
if <condition> :  
    <do something>  
    <do something>  
    <do something>
```

↑  
Tab!!

# 조건문(conditional statement)

True, False?

- 기본적인 Boolean 타입은 그대로 유지 : True is True, False is False
- Boolean 타입 외에도 조건으로 사용가능

Team True!! 😬

True

All numbers except 0

All collections except  
collection **with no element**

All functions

Team False!! 🤖

False

0

0.0

None

[]

()

""

{ }

# 조건문(conditional statement)

## 비교에 의한 조건

- 값들간의 비교를 통해서 조건을 만들 수 있다.
  - A가 B보다 크다 : **A > B**
    - A=5, B=3 일때의 결과는?
    - A=3, B=5 일때의 결과는?
  - A가 B보다 작다 : **A < B**
  - A가 B보다 크거나 같다: **A >=B**
  - A가 B보다 작거나 같다: **A <=B**
  - A와 B가 같다: **A == B, A is B**
  - A와 B가 다르다: **A != B, A is not B**
- 결과는 결국 True or False로 반환

A=5

B=3

print(A>B)

print(A<B)

print(A>=B)

print(A<=B)

print(A==B)

print(A is B)

print(A is not B)

# 영어를 배웠을 때 복수형을 배웠듯이..

s나 es를 붙인다.

y를 i로 바꾸고 es를 붙인다.

f나 fe를 v로 바꾸고 es를 붙인다

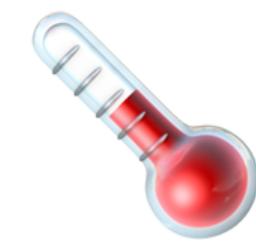
.....

# Python Data Type - Collection

## 묶음형 Data Type

- 단일 값만 저장 하는 것이 아니라, 값을 저장
  - 복수 값 저장의 나쁜 예: val1 = 1, val2 = 2, val3 = 3 .....
  - 복수 값 저장의 좋은 예: my\_list = [1, 2, 3]
- 목적에 따라서 다양한 묶음 형 데이터 타입을 사용
  - List: [1,2,3,4,5,6]
  - Dictionary: {"name": "Jungwon", "score": 100}
  - Tuple: (1, "b")
  - Set : {1,2,3}
- 값들의 묶음이기 때문에, string은 list와 유사한 특징을 띤
  - my\_list = ['a', 'b', 'c']
  - my\_str = 'abc'

**변수(variable)** = **상태(state)**



**함수(function)** = **행동(behavior)**



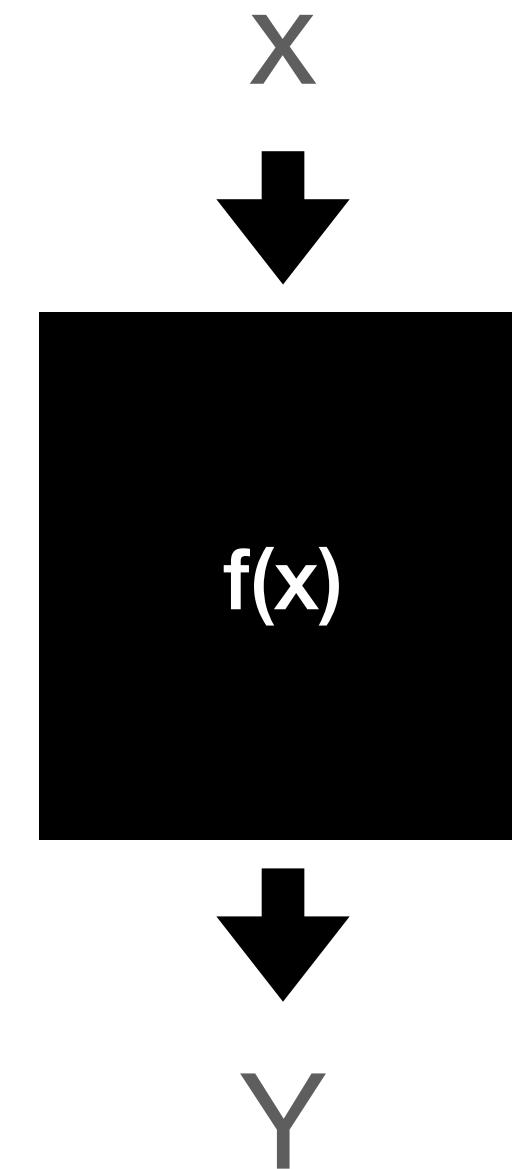
**클래스(class)** = **객체(object)**



# Python Function

$y=f(x)$

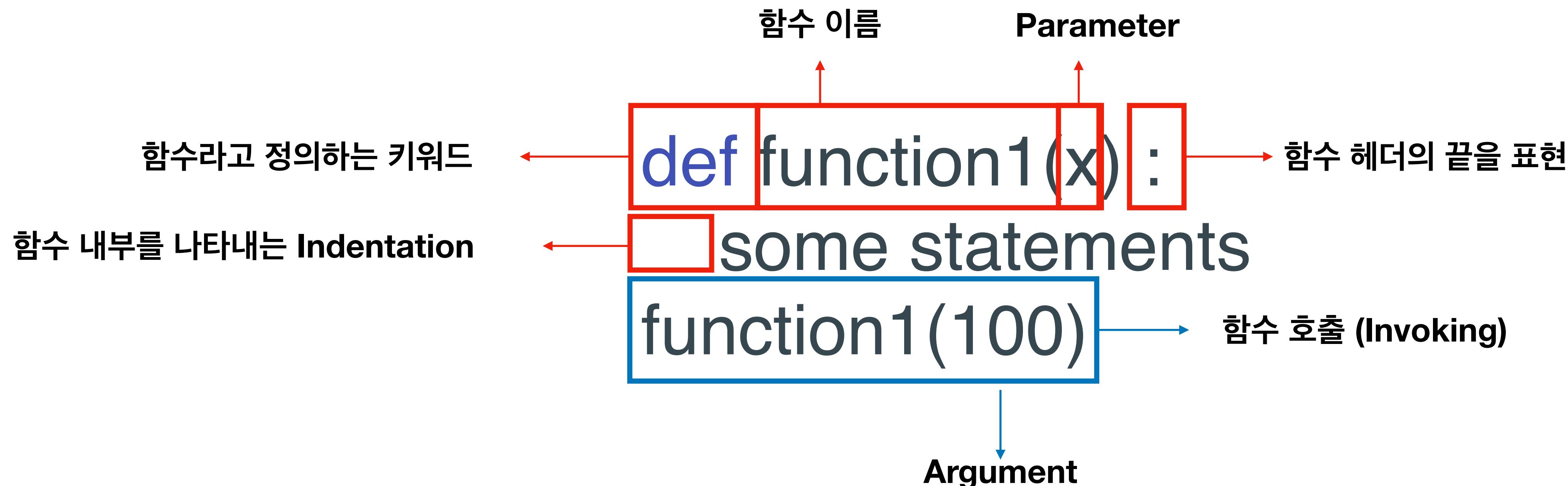
- 기본적으로 수학의 function과 비슷한 개념
  - $y=f(x)$  : f라는 함수에 x라는 인자를 넣으면 y라는 값이 반환!
  - $y=2x$
- 명령/연산들을 한곳에 묶어두기 위해
  - 중복된 코드 사용 방지
  - 코드의 가독성 향상
- Python에서 함수는 keyword **def**로 시작



```
def count_char(text):  
    count = 0  
    for char in text:  
        count += 1  
    return count  
  
cnt = count_char("Hello World")  
print(cnt)
```

# Python Function

## 함수 작성법 / 호출 법



# Python Class

## Object-Oriented Programming

- 컴퓨터 프로그램을 명령어의 목록으로 보는 것이 아닌, 여러개의 독립된 단위(객체)들의 모임으로 파악하고자 함
- OOP concepts
  - 상속 (Inheritance) : 다른 클래스(부모)의 Method 및 변수들을 상속 받을 수 있다.
  - 캡슐화 (Encapsulation) : 객체의 속성과 행위를 하나로 묶고, 그 중 일부를 외부에 감출 수 있다.
  - 추상화 (Abstraction) : 복잡한 자료, 모듈, 시스템등으로 부터 핵심적인 개념 및 기능을 간추려 나타낼 수 있다.
  - 다형성 (Polymorphism) : 한 요소에 여러 개념을 넣을 수 있다
- **Everything is an object in Python!**
  - int, float, string, dict, list, function 등 모두 사실은 Object

# Python Class

## 객체의 청사진(Blueprint)

```
class Player:  
    def __init__(self, name, team, nat, salary):  
        self.name = name  
        self.team = team  
        self.nat = nat  
        self.salary = salary  
    def update_team(self, team):  
        self.team = team  
    def update_salary(self, salary):  
        self.salary = salary
```

```
player1 = Player("손흥민", "토트넘", "대한민국", 1000)  
player1.update_team("레알 마드리드")  
player1.update_salary(2000)
```

# Python Class

## OOP Terms

Class

```
class Mammal:  
    def __init__(self):  
        pass
```

초기화 함수 / 생성자

(Initialization function/  
Constructor)

class Human(Mammal): 부모 클래스(Parent Class)

```
def __init__(self,date_of_birth, name, nationality):
```

```
    self.date_of_birth=date_of_birth
```

```
    self.name = name
```

```
    self.nationality = nationality
```

```
def think(self):
```

```
    pass
```

```
def move(self):
```

```
    pass
```

Variable

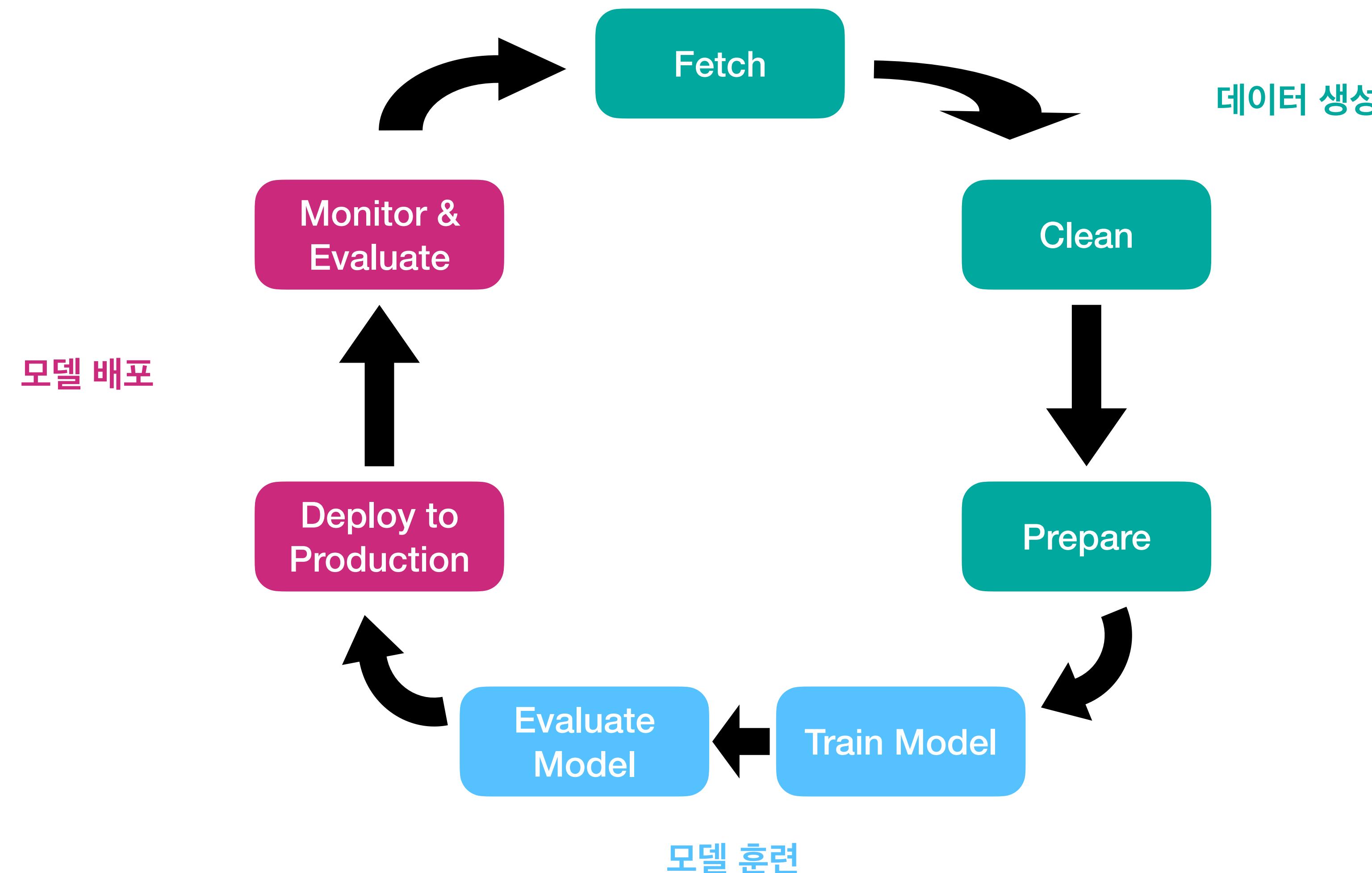
Method  
(function)

인스턴스  
(Instance)

인스턴스화(Instantiation)

```
jungwon=Human("900302", "서중원", "대한민국")
```

# 머신러닝 사이클



# 데이터란?

- 데이터란 데이터 객체의 모음 및 속성(attribute)들
- 속성은 객체의 특성
  - 예 : 사람의 눈 색깔, 온도 등
  - variable, field, feature라고도 불림
- 속성의 모음은 객체를 설명
  - 객체 동의어: 레코드, 포인트, 사례, 샘플, 엔터티 또는 인스턴스

# 속성종류

- Nominal: 명목자료
  - 비교 X, 차이 O
  - 예: 우편번호, 주민번호 뒷자리, 피부색, 성별
- Ordinal: 서열자료
  - 순서가 있는 명목자료
  - 예: 점수 (10점 만점에 몇점), 학년, 키 (크다/보통/작다)
- Interval: 구간자료
  - True Zero가 존재하지 않음
  - 0도가 가장 낮은 온도인가? 1월 1일이 가장 이른 날짜인가?
  - 예: 달력 날짜, 온도 (섭씨/화씨)
- Ratio: 비율자료
  - True Zero가 존재
  - 절대 온도 0도가 가장 낮은 온도, 0M는 길이가 없음, 0초, 0번 등
  - 예: 절대온도, 길이, 시간, 수

# 이산속성 및 연속속성

- 이산 속성
  - 한정적이거나 셀 수 있을 정도로 무한대의 값 집합만 있음
  - 예 : 우편 번호, 횟수, 문서 모음에서 단어의 수
  - 보통 정수 변수로 표시
  - 참고 : 이진 속성은 특수한 경우의 이산 속성
- 연속속성
  - 속성 값으로 실수를 가짐
  - 예 : 온도, 높이 또는 무게.
  - 하지만 유한한 갯수의 숫자로 측정되거나 표현 될 수밖에 없음
  - 일반적으로 소수점을 포함한 수로 표현됨

# 정형 데이터의 중요한 특징

- Dimensionality: 차원
  - Curse of Dimensionality : 차원의 저주
- Sparsity: 희소성
  - 유효한 부분만 카운팅
- Resolution: 해상도
  - 배율에 따라 패턴이 달라짐

# 데이터 품질

- 어떤 종류의 데이터 품질 문제가 있나?
- 데이터 문제를 어떻게 감지 할 수 있나?
- 이 문제들에 대해 무엇을 할 수 있나?
- 데이터 품질 문제의 예 :
  - Noise 와 outliers
  - Missing value
  - Duplicate data

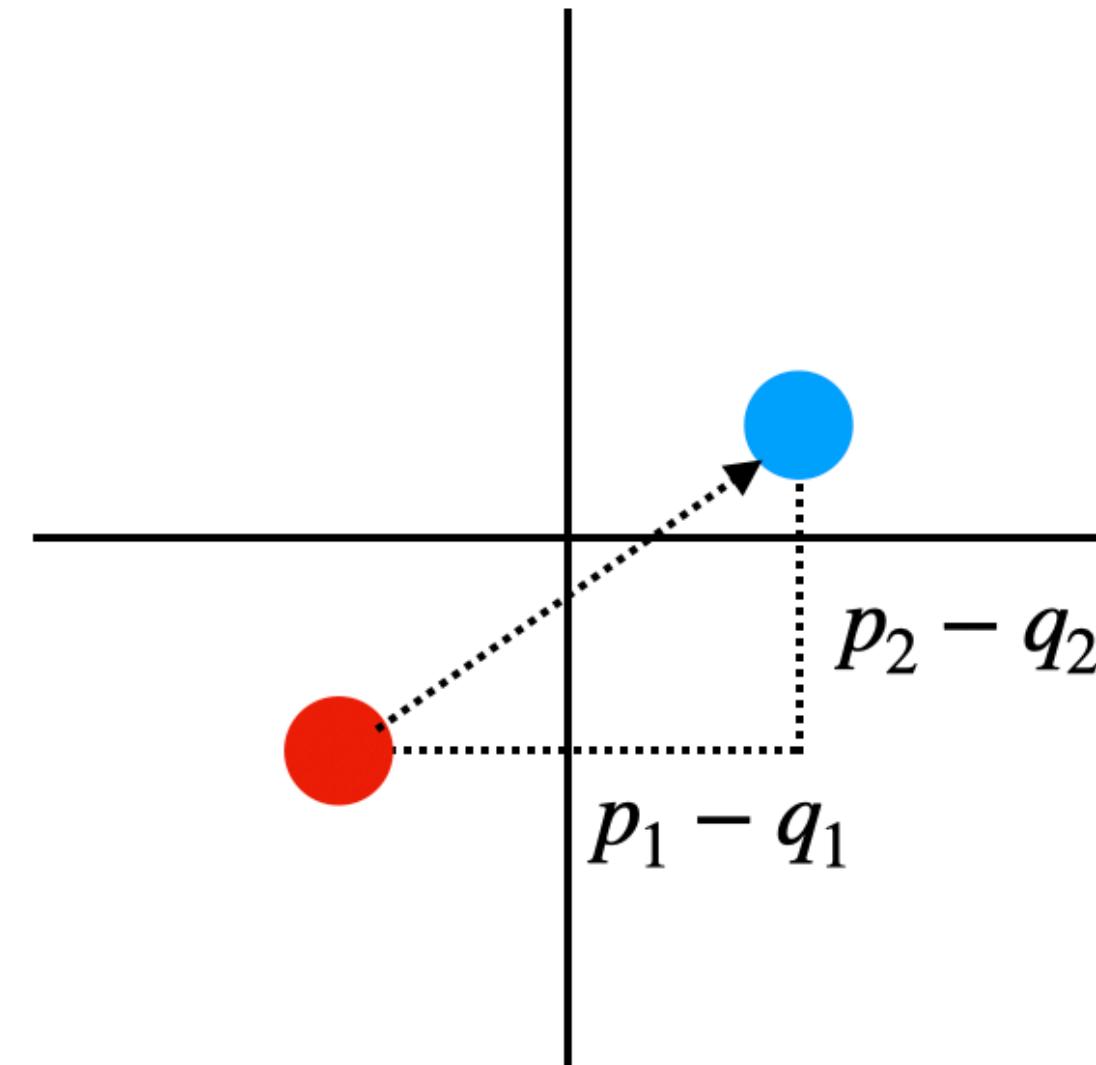
# 결측값

- 결측값이 생기는 이유
  - 정보가 항상 수집되지는 않음
  - 예 : 사람들은 나이와 체중을 숨기려고 함)
  - 모든 경우에 속성을 적용 할 수있는 것은 아님
  - 예 : 연간 소득은 어린이에게는 적용되지 않음)
- 결측값 처리
  - 데이터 객체 제거
  - 결측값 추정
  - 분석 중 결측값 무시
  - 가능한 모든 값으로 대체 (확률에 따라 가중치가 부여됨)

# 유클리디안 거리

- 유클리드 거리
- 여기서 n은 차원 (속성)의 수이고  $p_k$  및  $q_k$ 는 p, q의 k번째 속성(feature)

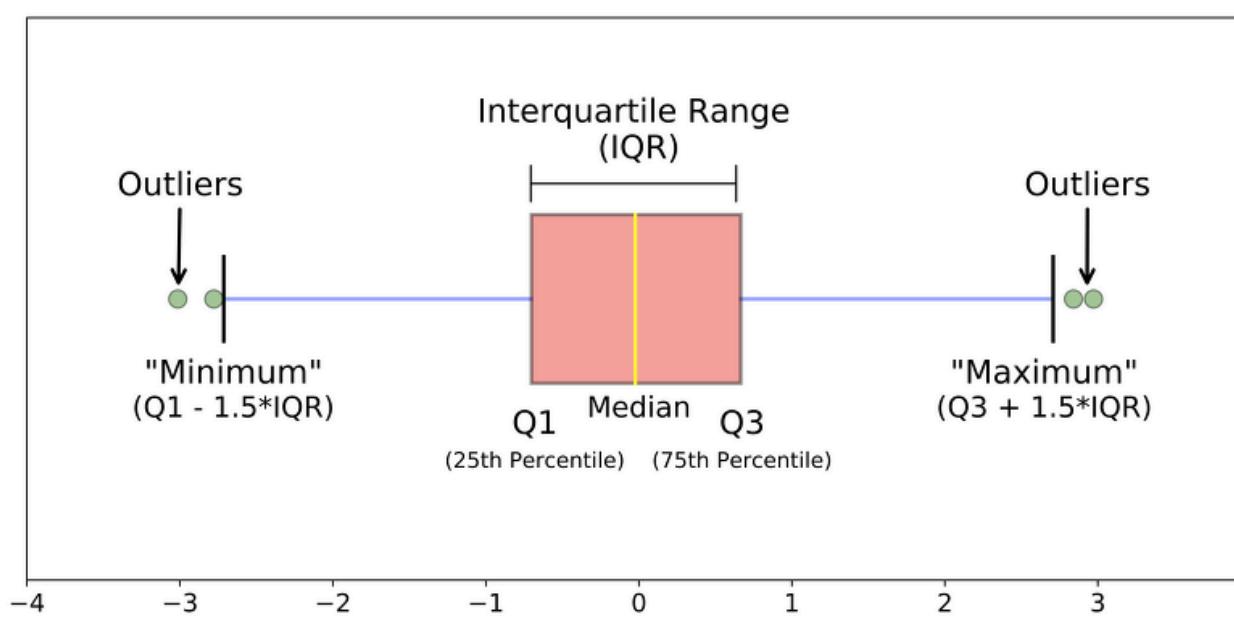
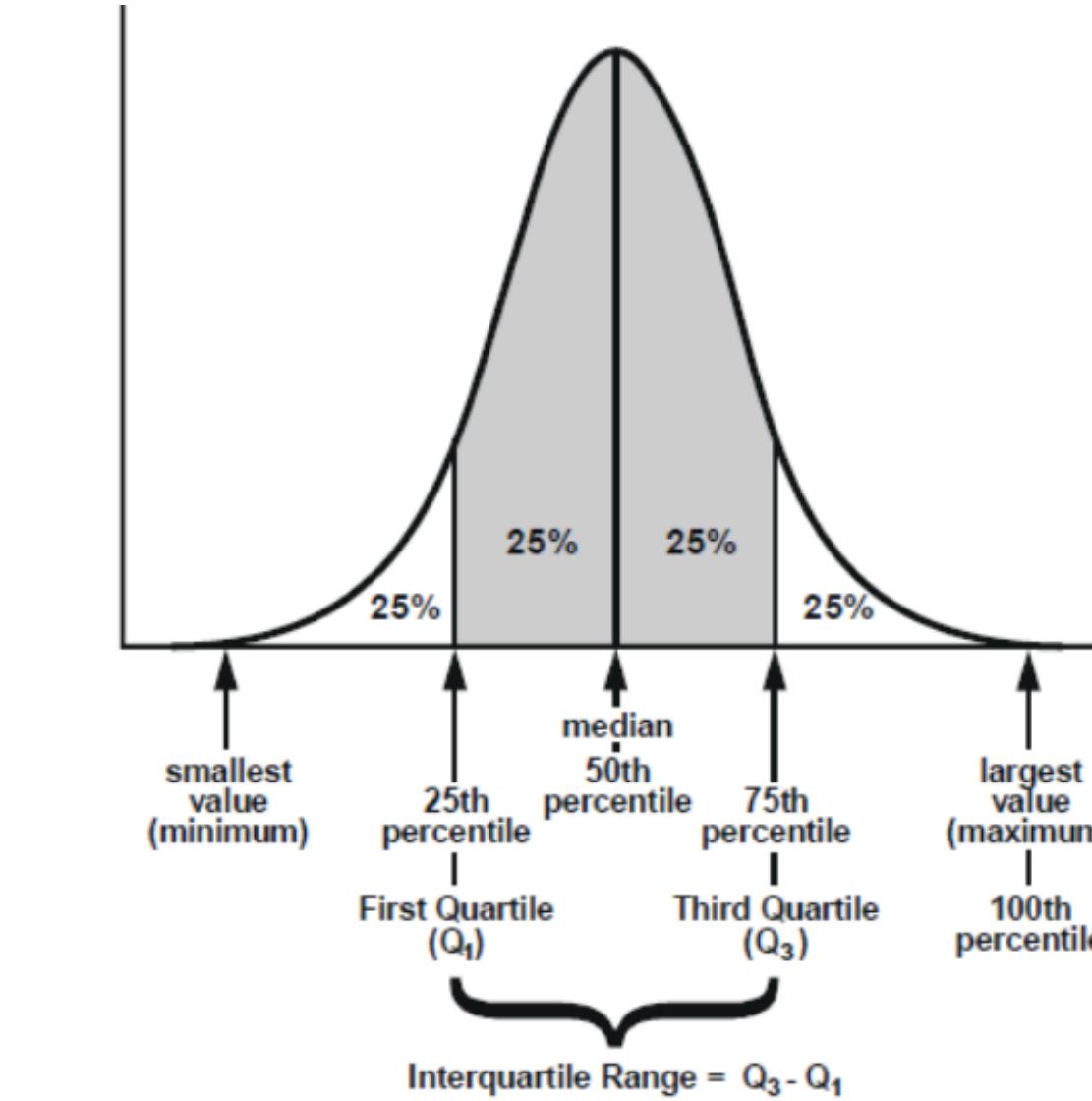
$$dist = \sqrt{\sum_{k=1}^n (p_k - q_k)^2}$$



# 탐색적 데이터 분석

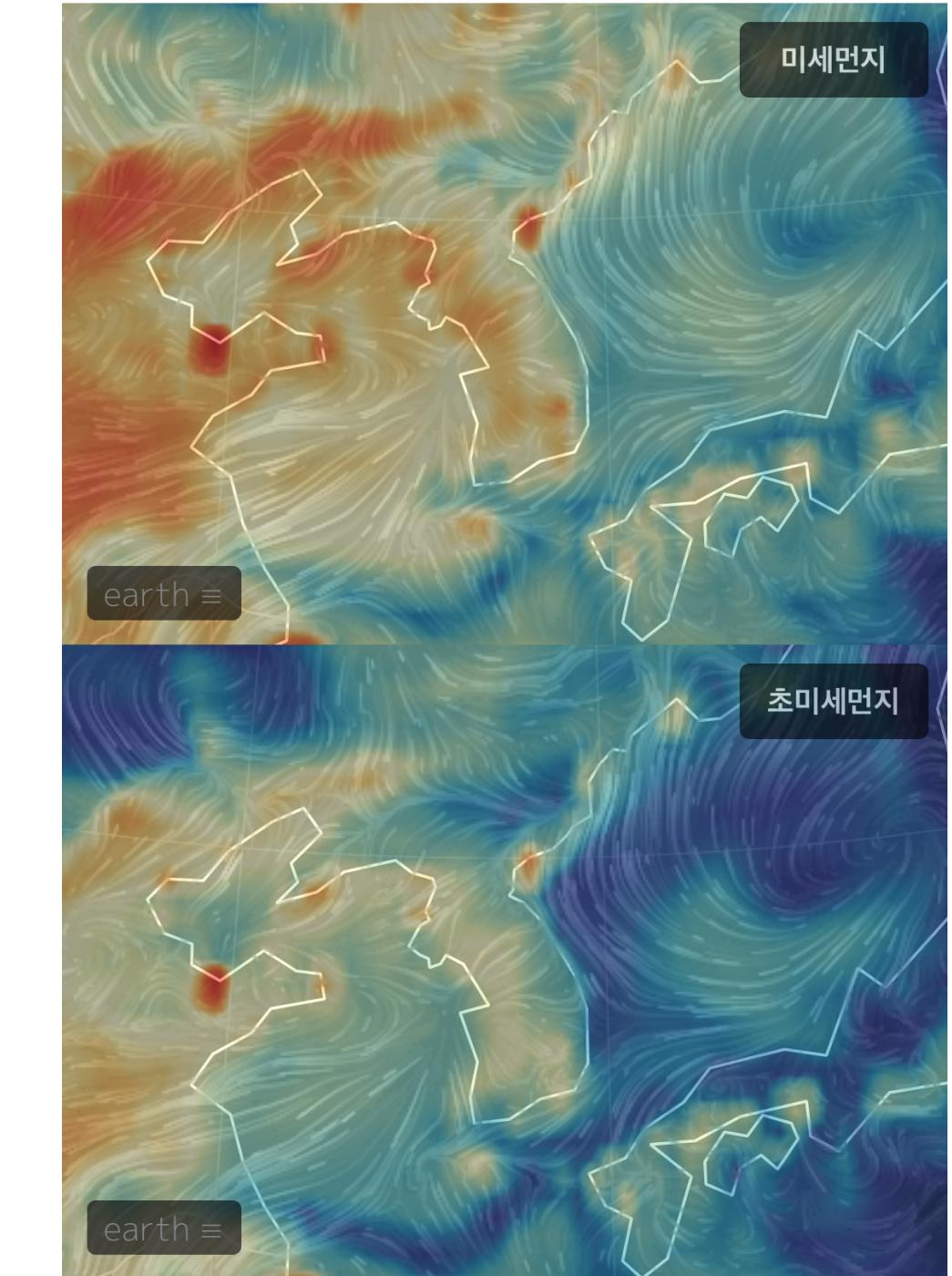
- 데이터의 특성을 더 잘 이해하기 위한 데이터의 예비 탐색
- 데이터 탐색의 주요 동기
  - 전처리 또는 분석에 적합한 도구를 선택하도록 지원
  - 인간의 능력을 활용하여 패턴 인식이 가능하게 하기 위함
    - 인간은 데이터 분석 도구로 캡처되지 않은 패턴을 인식 가능
    - 극단적인 예: 말투 파악
- 탐색적 데이터 분석(EDA)
  - 통계학자 John Tukey에 의해 창시
  - <https://www.itl.nist.gov/div898/handbook/index.htm>

# EDA의 몇가지 예



국가	확진자	사망자	완치	사망 (%)	완치 (%)	발생률*
1 미국 🇺🇸	1,212,900 (+20,960)	69,921 (+1,206)	188,068 (+9,397)	5.8	15.5	3,664
2 스페인 🇪🇸	248,301	25,428	151,633	10.2	61.1	5,311
3 이탈리아 🇮🇹	211,938 (+1,221)	29,079 (+195)	82,879 (+1,225)	13.7	39.1	3,505
4 영국 🇬🇧	190,584 (+3,985)	28,734 (+288)	344	15.1	0.2	2,807
5 독일 🇩🇪	166,152 (+407)	6,993 (+127)	135,100 (+2,400)	4.2	81.3	1,983
6 러시아 🇷🇺	145,268	1,356	18,095	0.9	12.5	995
7 프랑스 🇫🇷	131,863 (+576)	25,201 (+306)	51,371 (+587)	19.1	39.0	2,020
8 터키 🇹🇷	127,659 (+1,614)	3,461 (+64)	68,166 (+5,015)	2.7	53.4	1,514
9 브라질 🇧🇷	108,620 (+6,794)	7,367 (+316)	45,815 (+2,824)	6.8	42.2	511
10 이란 🇮🇷	98,647	6,277	79,379	6.4	80.5	1,174
11 중국 🇨🇳	82,881 (+1)	4,633	77,944	5.6	94.0	58
12 캐나다 🇨🇦	60,772 (+1,298)	3,854 (+172)	26,017 (+1,109)	6.3	42.8	1,610
13 벨기에 🇧🇪	50,267	7,924	12,378	15.8	24.6	4,337
14 페루 🇵🇪	47,372 (+1,444)	1,344 (+58)	14,427 (+877)	2.8	30.5	1,437

널스쿨



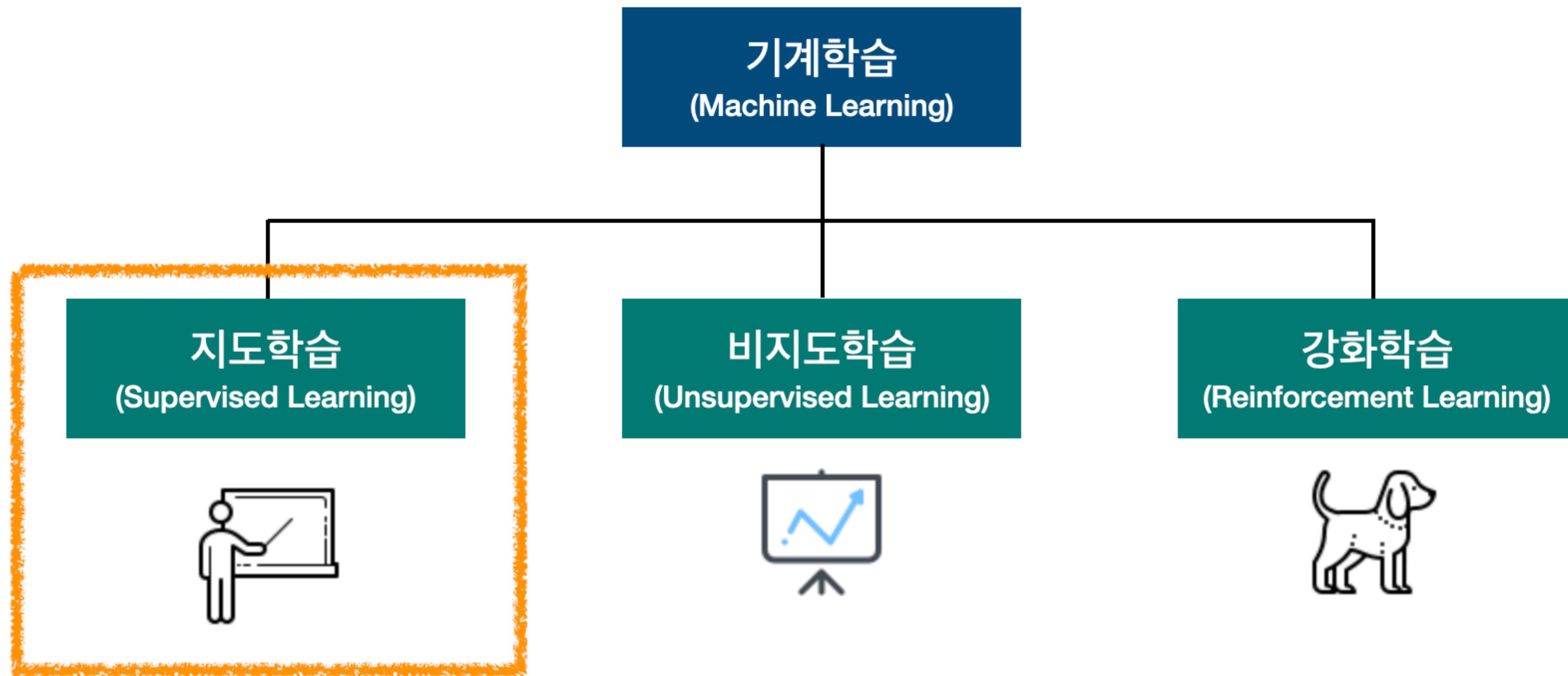
미세미세 지도

안양대연구소

일본기상청

널스쿨

# 머신러닝 지도학습



# 지도학습 - 분류문제

## Supervised Learning - Classification

- Categorical 값을 예측하는 문제

- True or False
- A, B, C, D

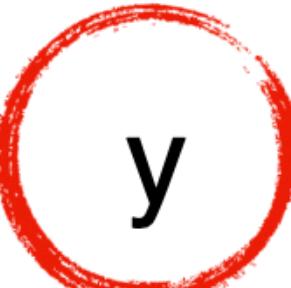
- 훈련 데이터셋 구조

- $X' = [x_1, x_2, x_3, \dots, x_n]$
- $y = x_n$  (클래스)
- $X = [x_1, x_2, x_3, \dots, x_{n-1}]$

- 테스트 데이터셋 구조

- $X' = [x_1, x_2, x_3, \dots, x_{n-1}]$
- $x_n$  즉,  $y$ 를 예측하는 것이 목적

$[x_1, x_2, x_3, \dots, x_{n-1}]$

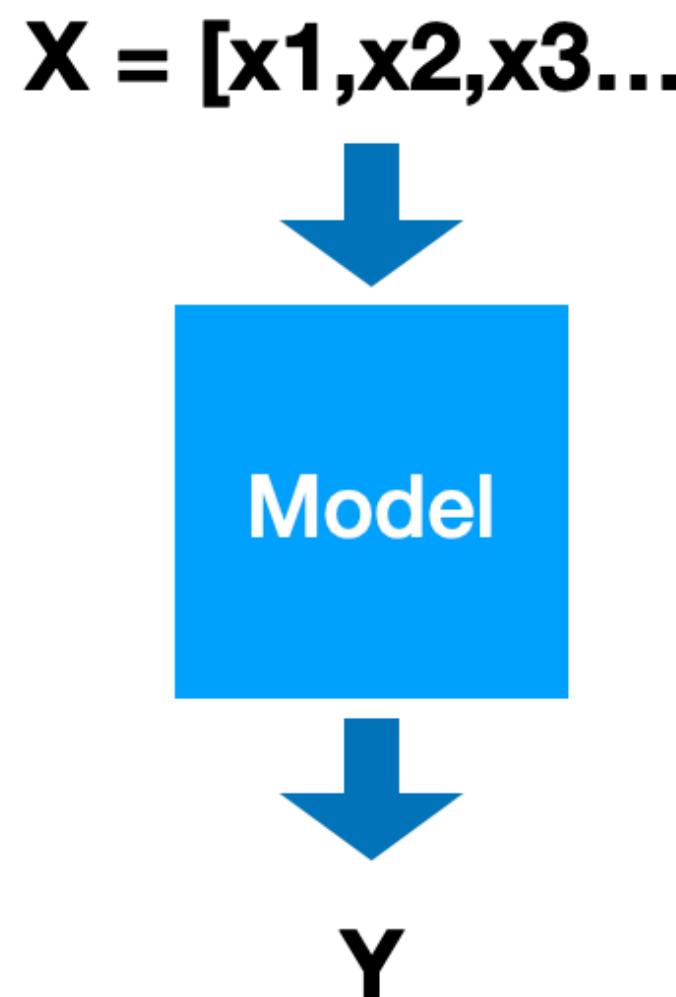


# Classification 기술의 종류

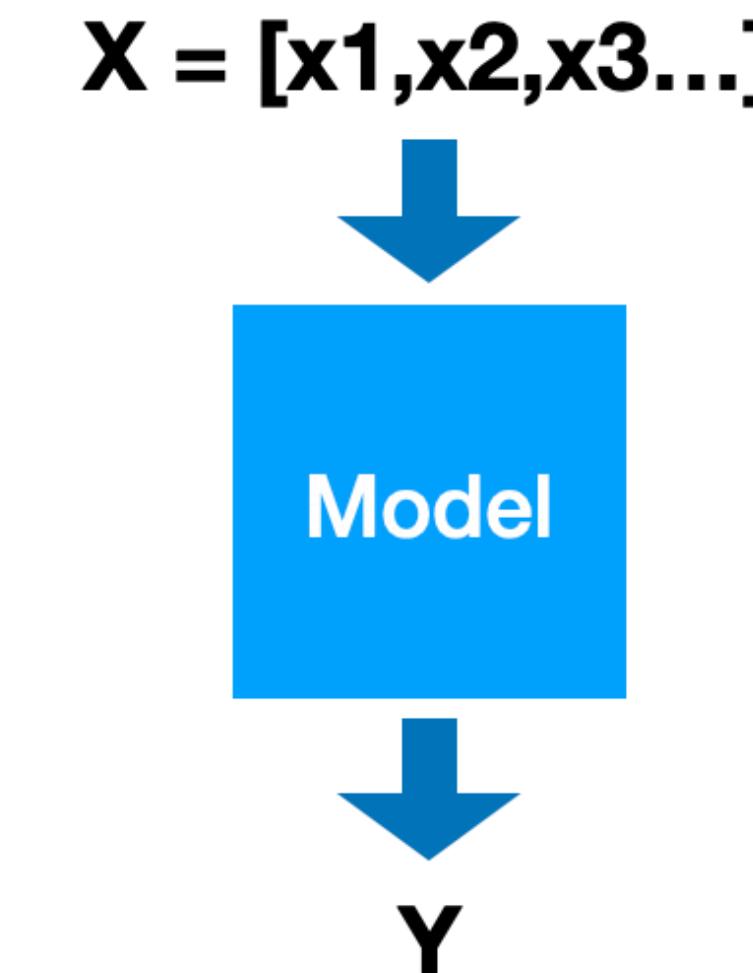
- Statistical Methods
- Rule-based Methods
- Decision Tree based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

# Classification vs Regression

분류와 회귀의 차이

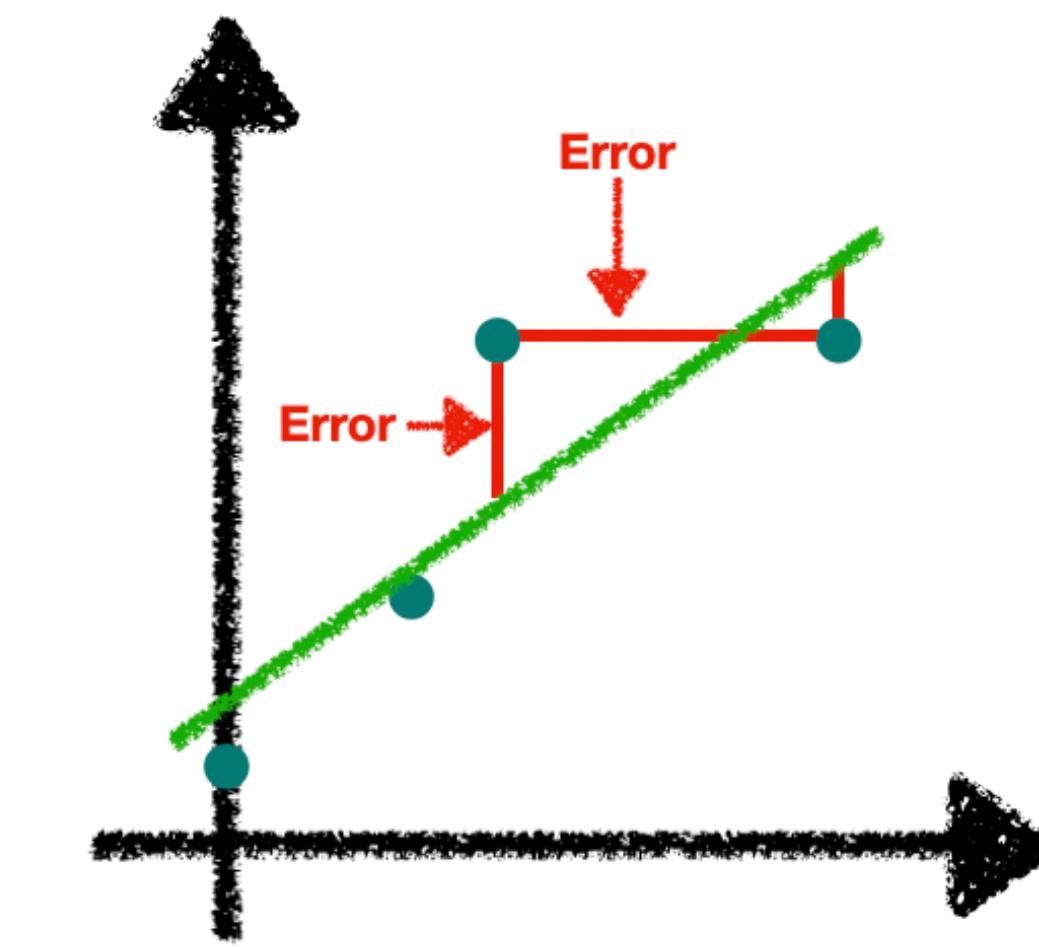
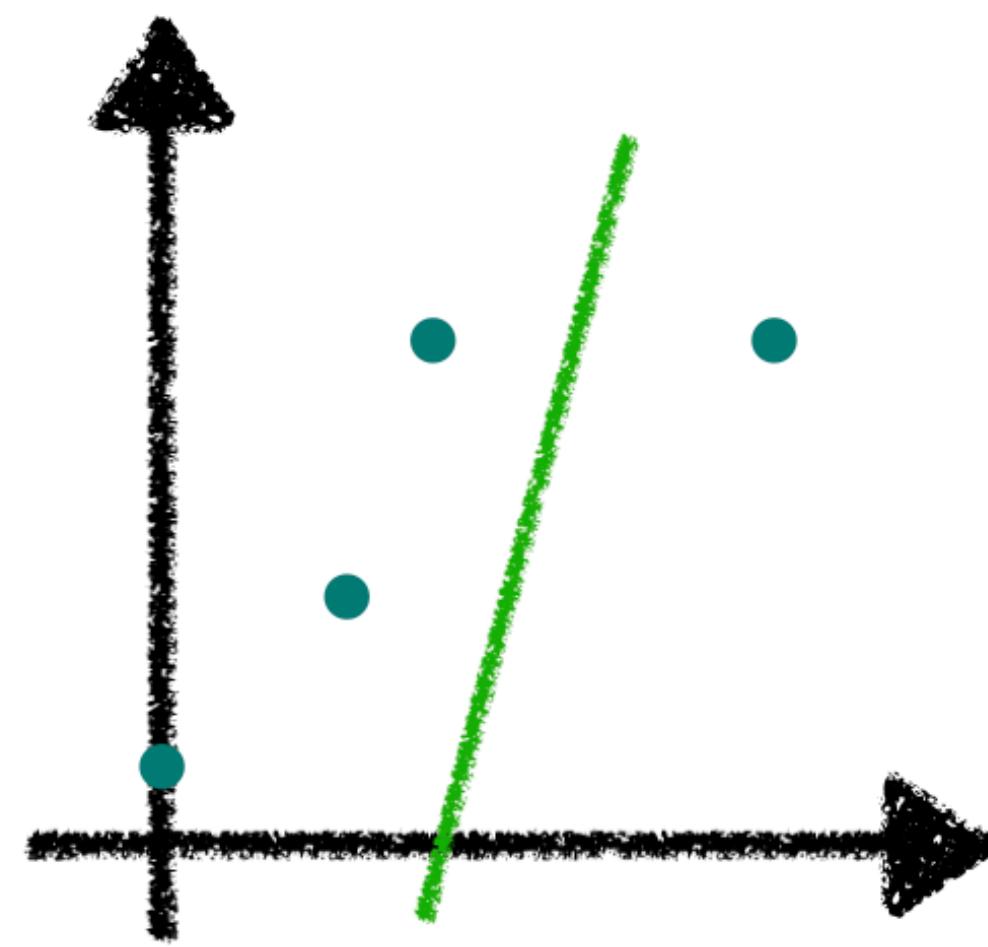
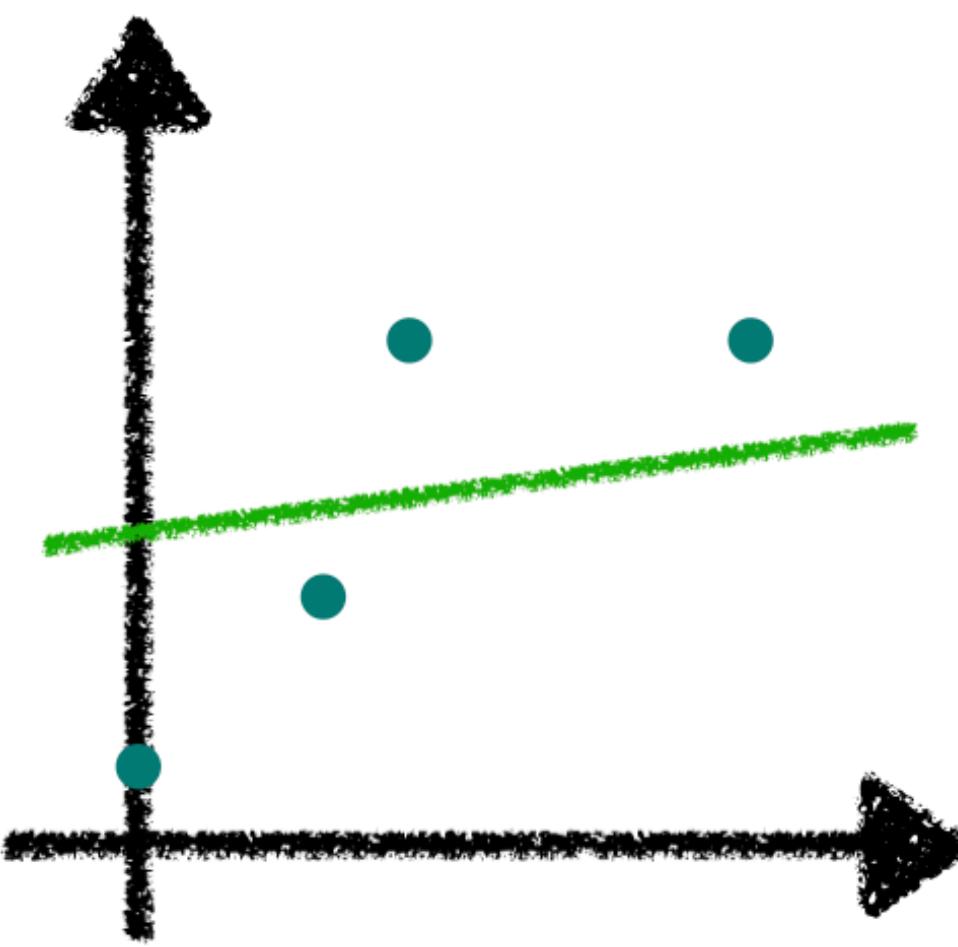


이때  $Y$ 는 범주형  
e.g., (1,0), (True,False), (A, B, C, D)



이때  $Y$ 는 숫자형  
e.g., 133, 0.11, 103030

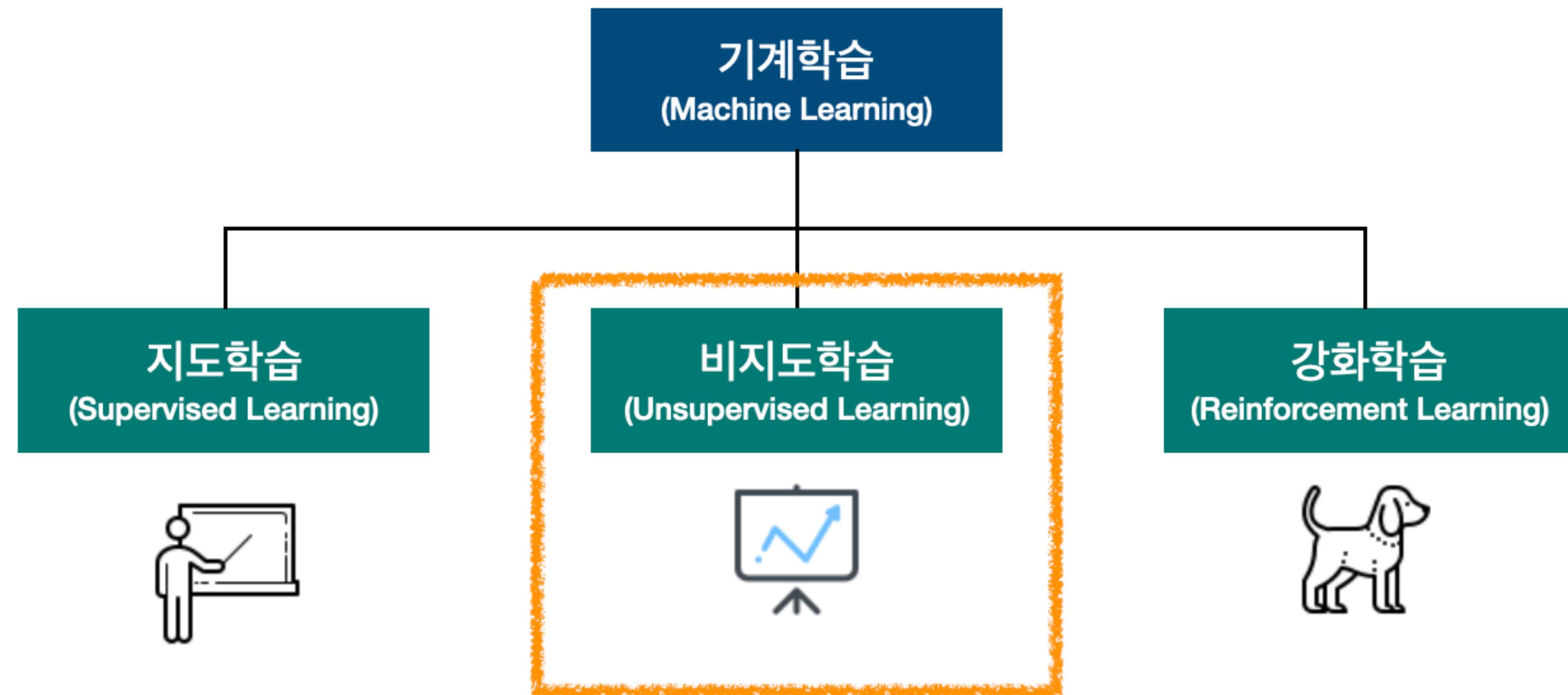
# Linear Regression의 기본 원리



# Regression 모델의 종류

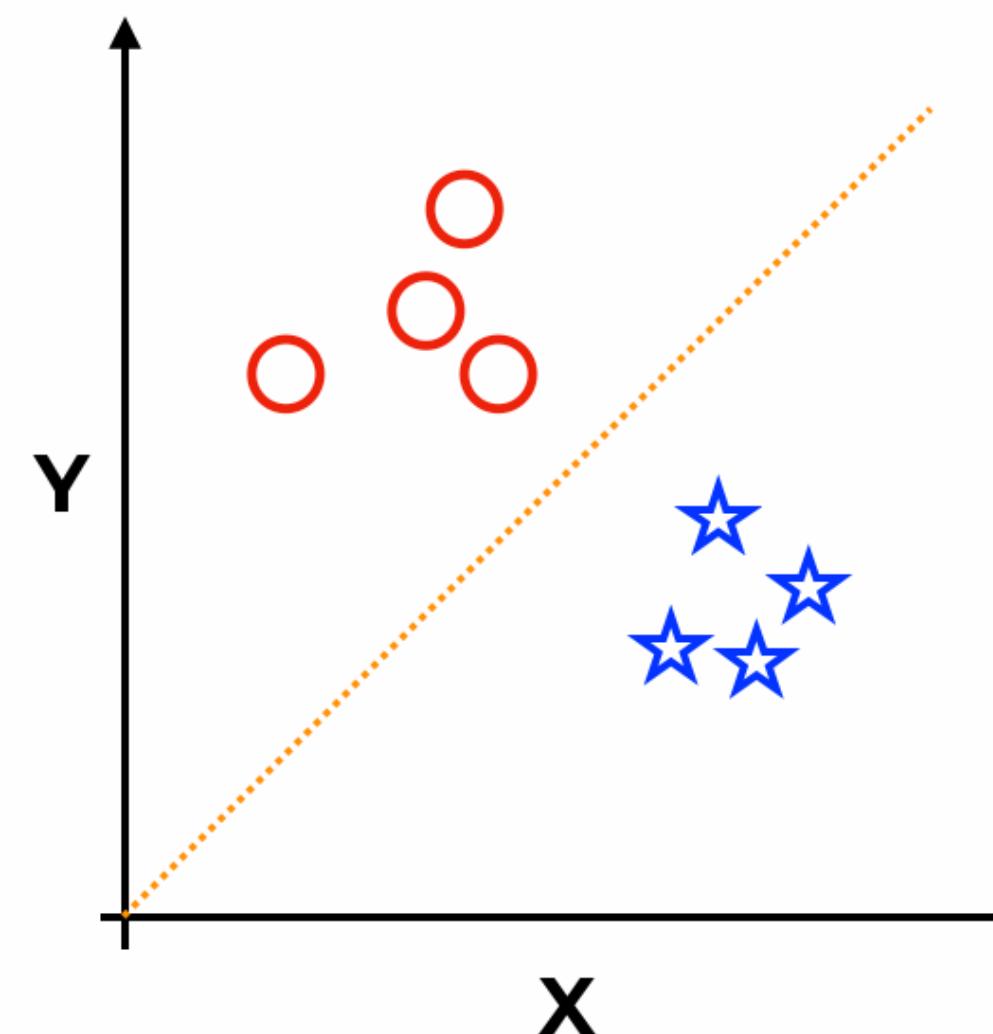
- Linear Regression
- Polynomial Regression
- Support Vector Regression
- Decision Tree Regression
- Random Forest Regression
- Ridge Regression
- Lasso Regression

# 머신러닝 비지도학습

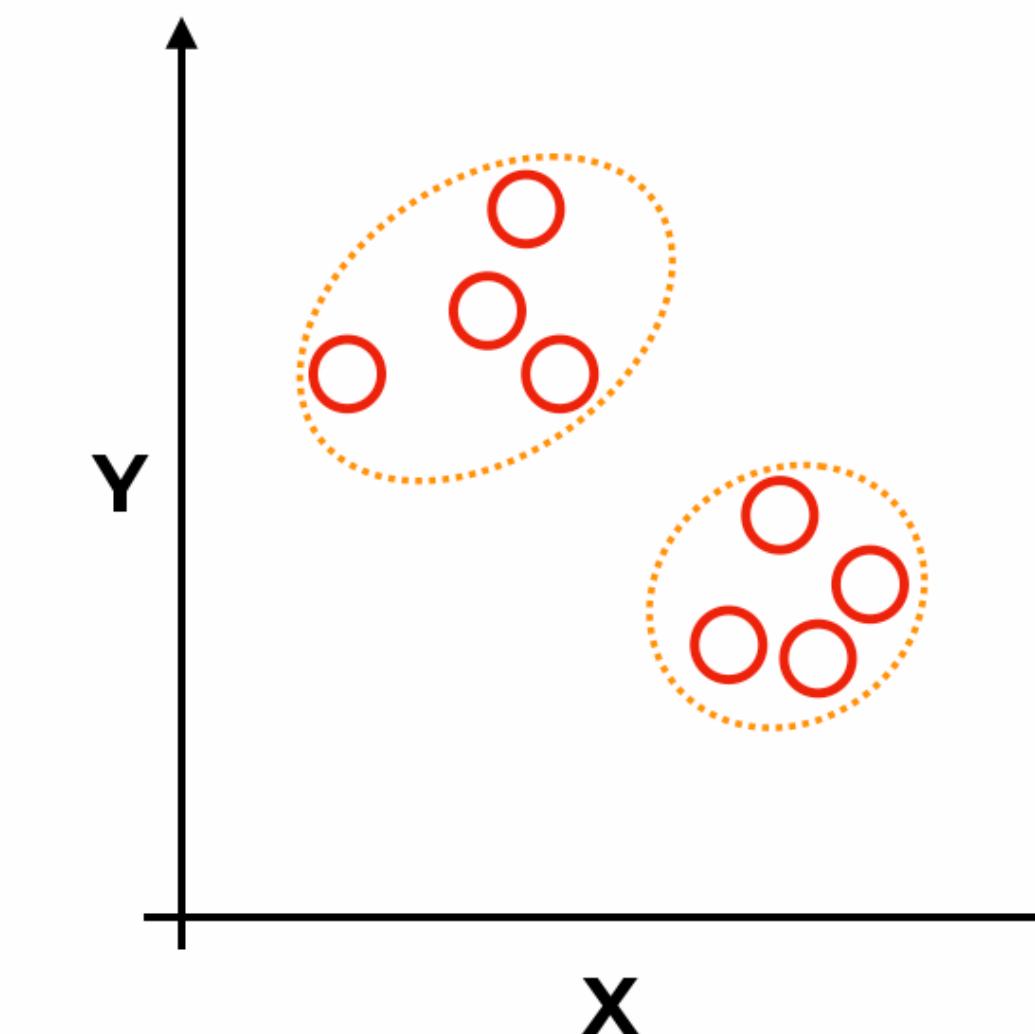


# 레이블이 주어지지 않은 경우

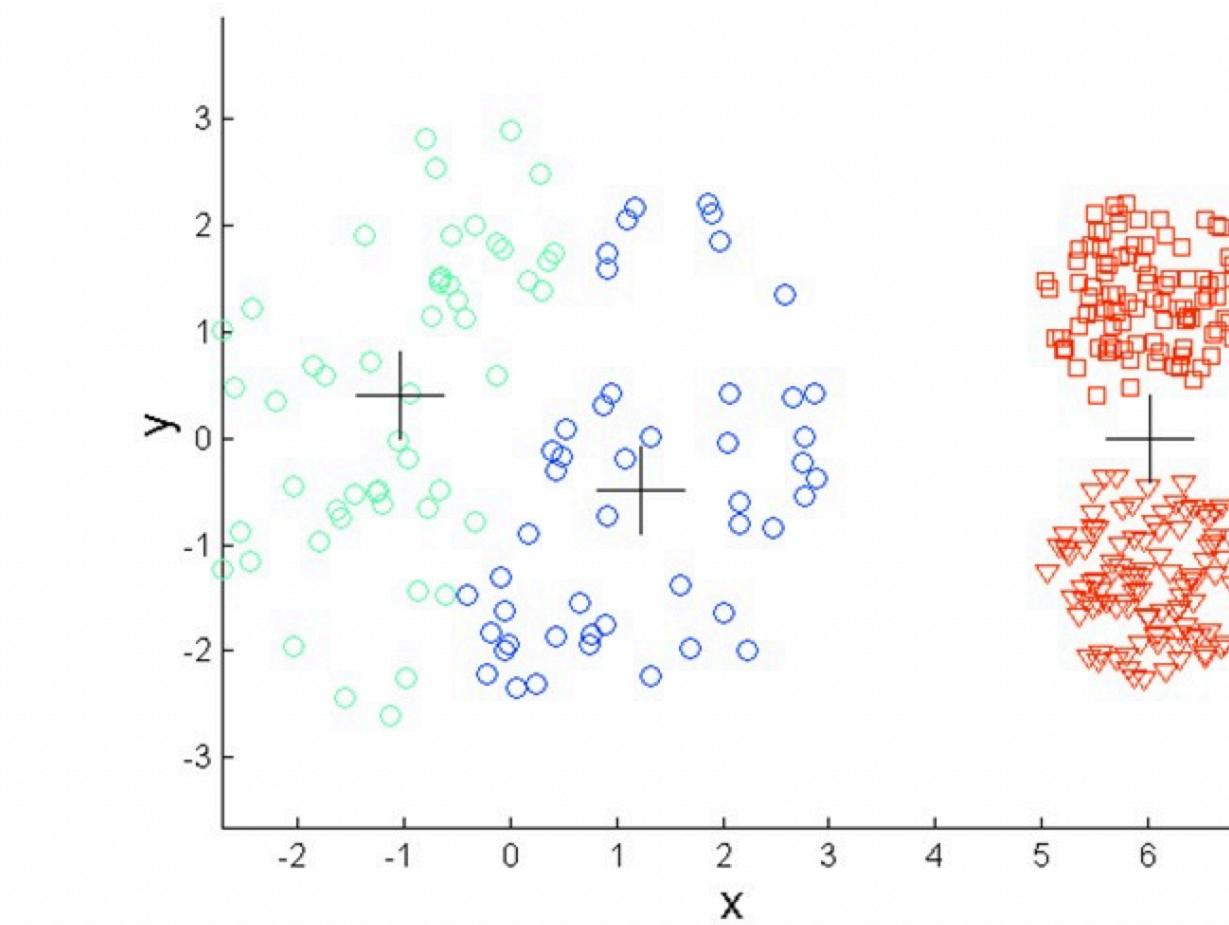
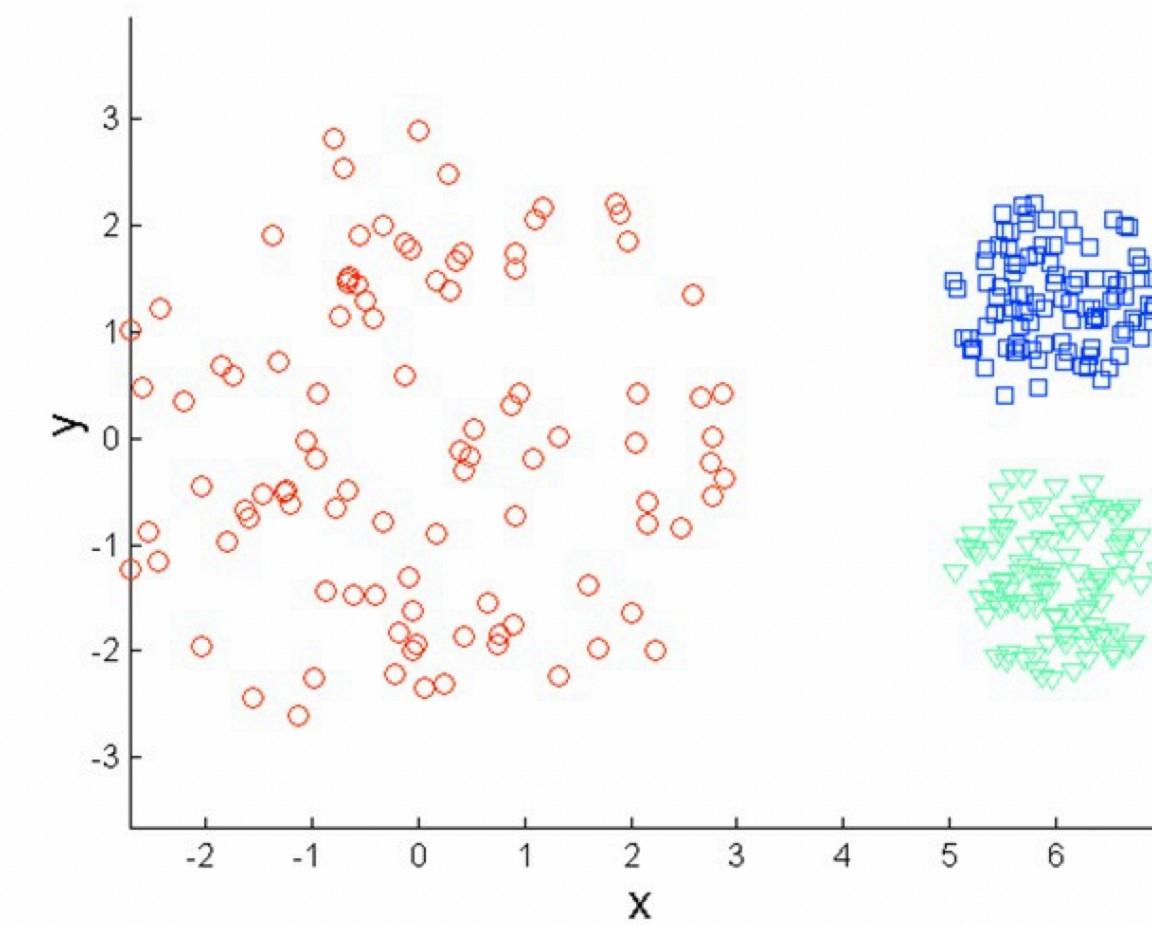
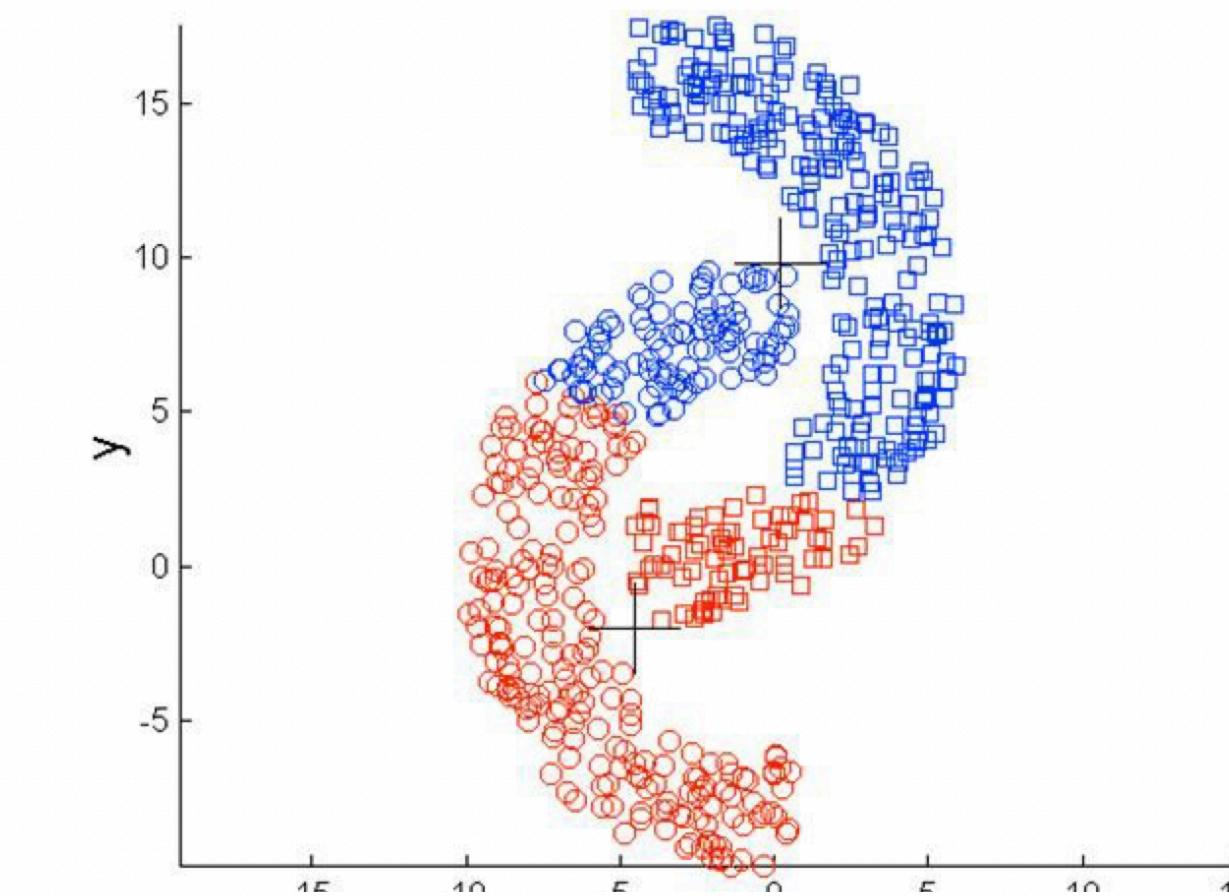
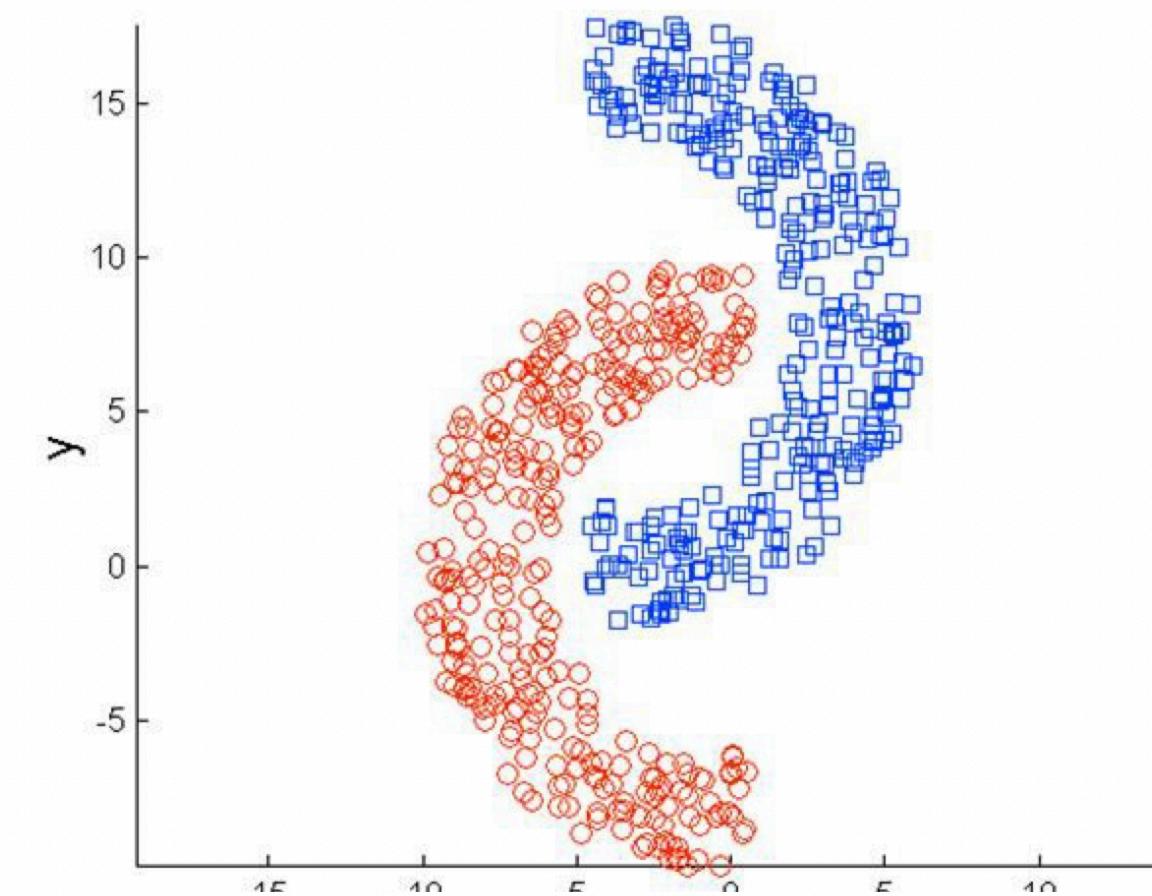
지도학습 (*Supervised Learning*)



비지도학습 (*Unsupervised Learning*)

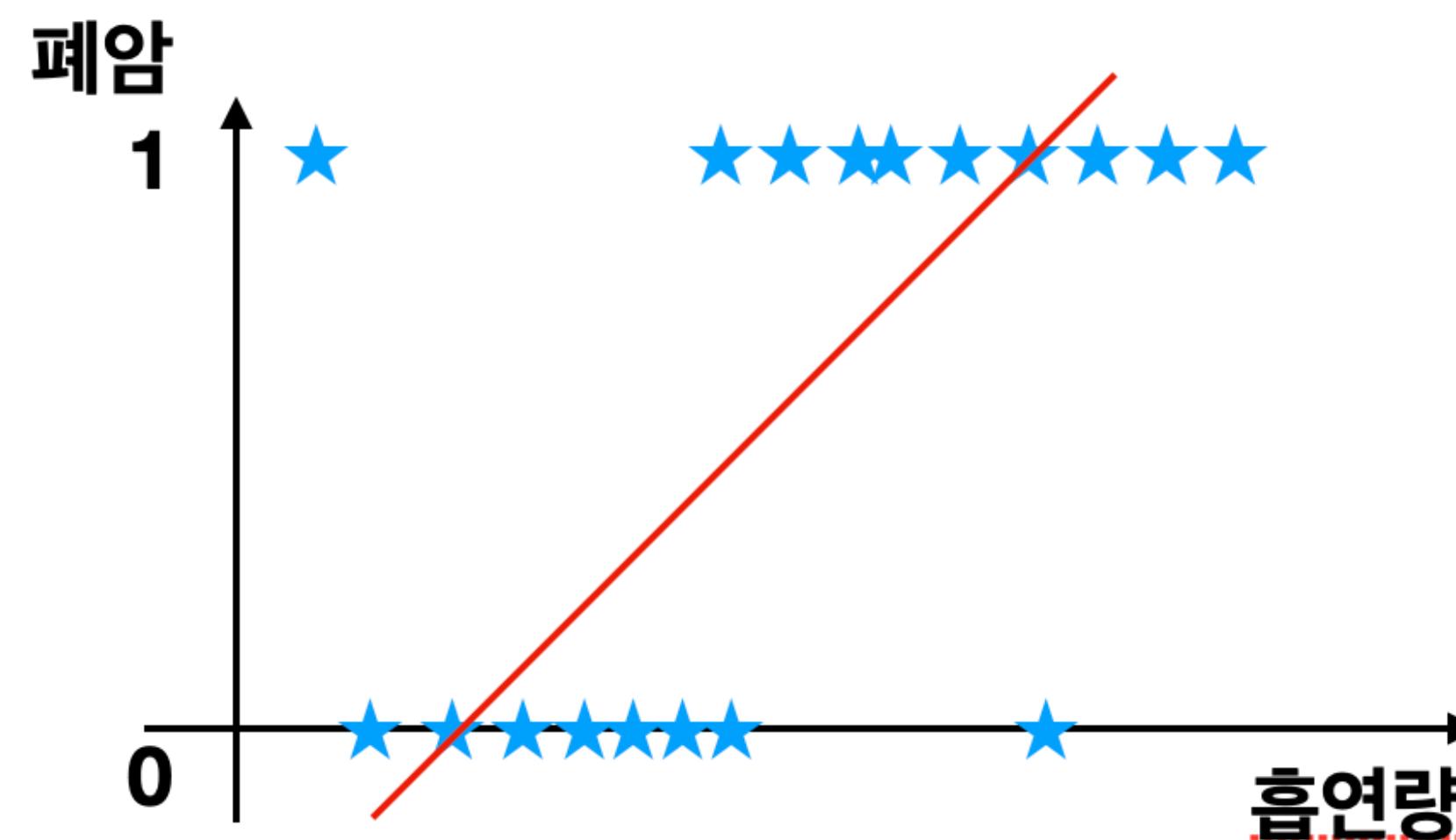


# K평균 군집대 밀도기반 군집

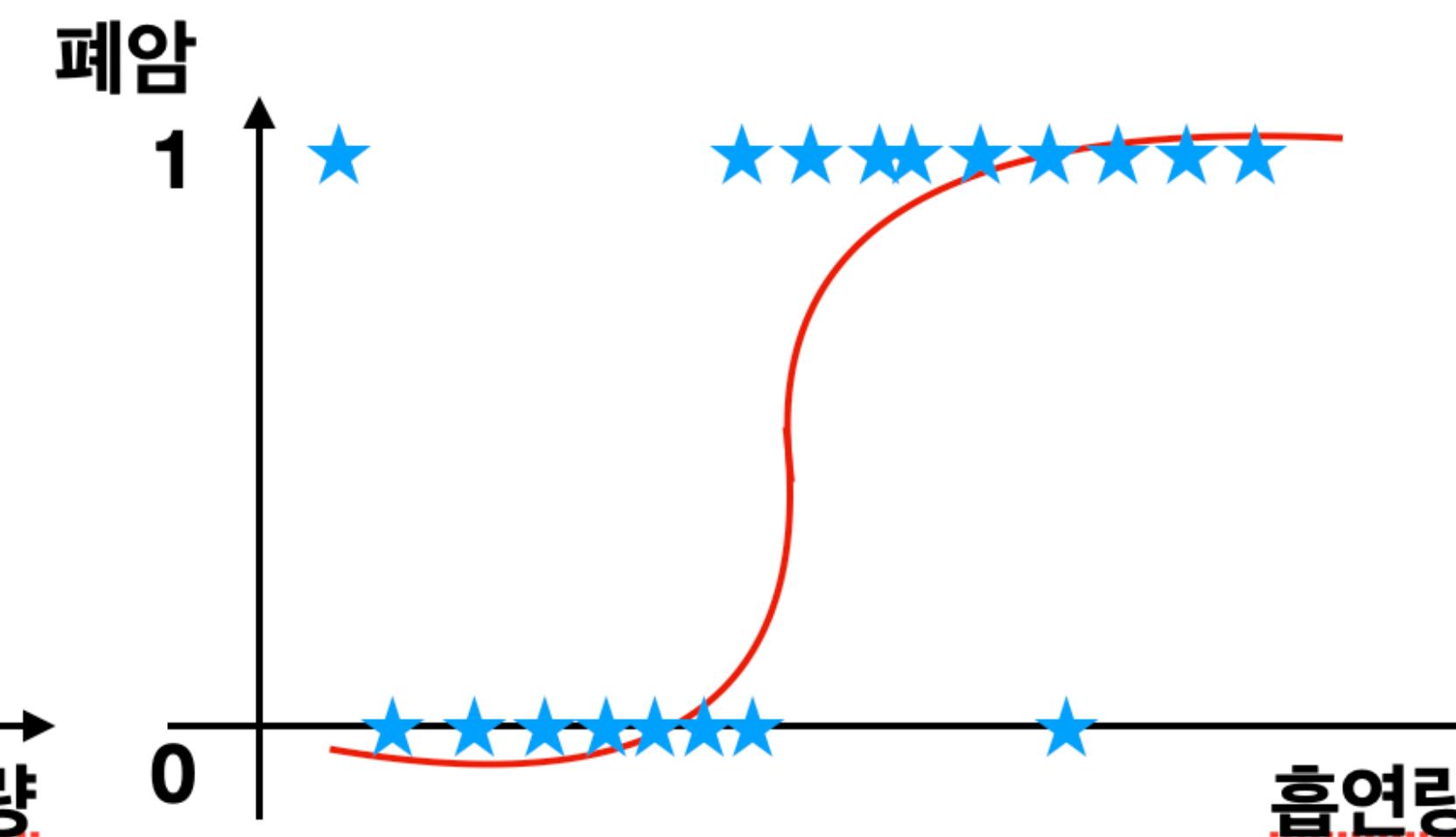


# 뉴럴 네트워크

Linear -> Logistic -> Neural Networks

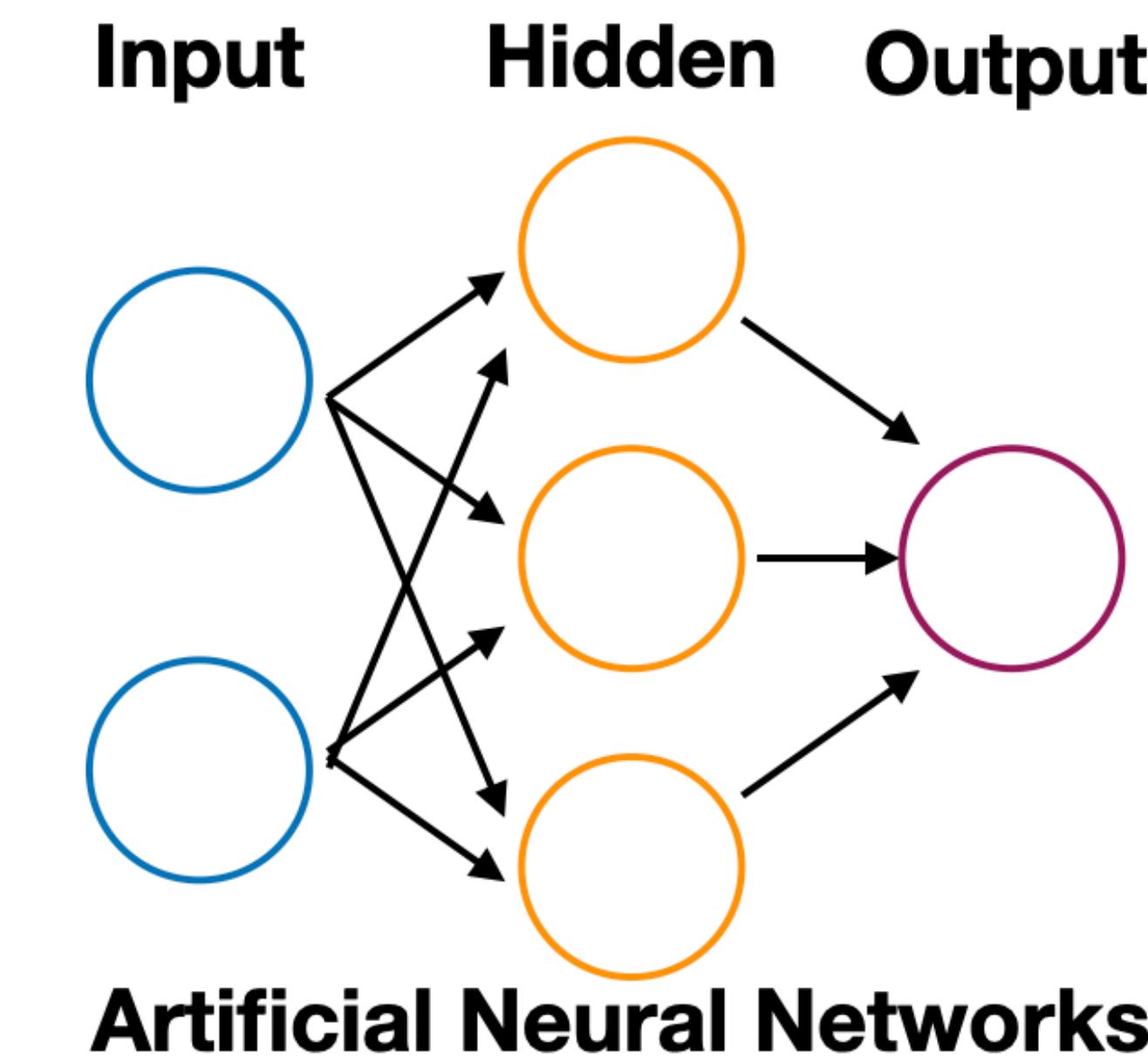
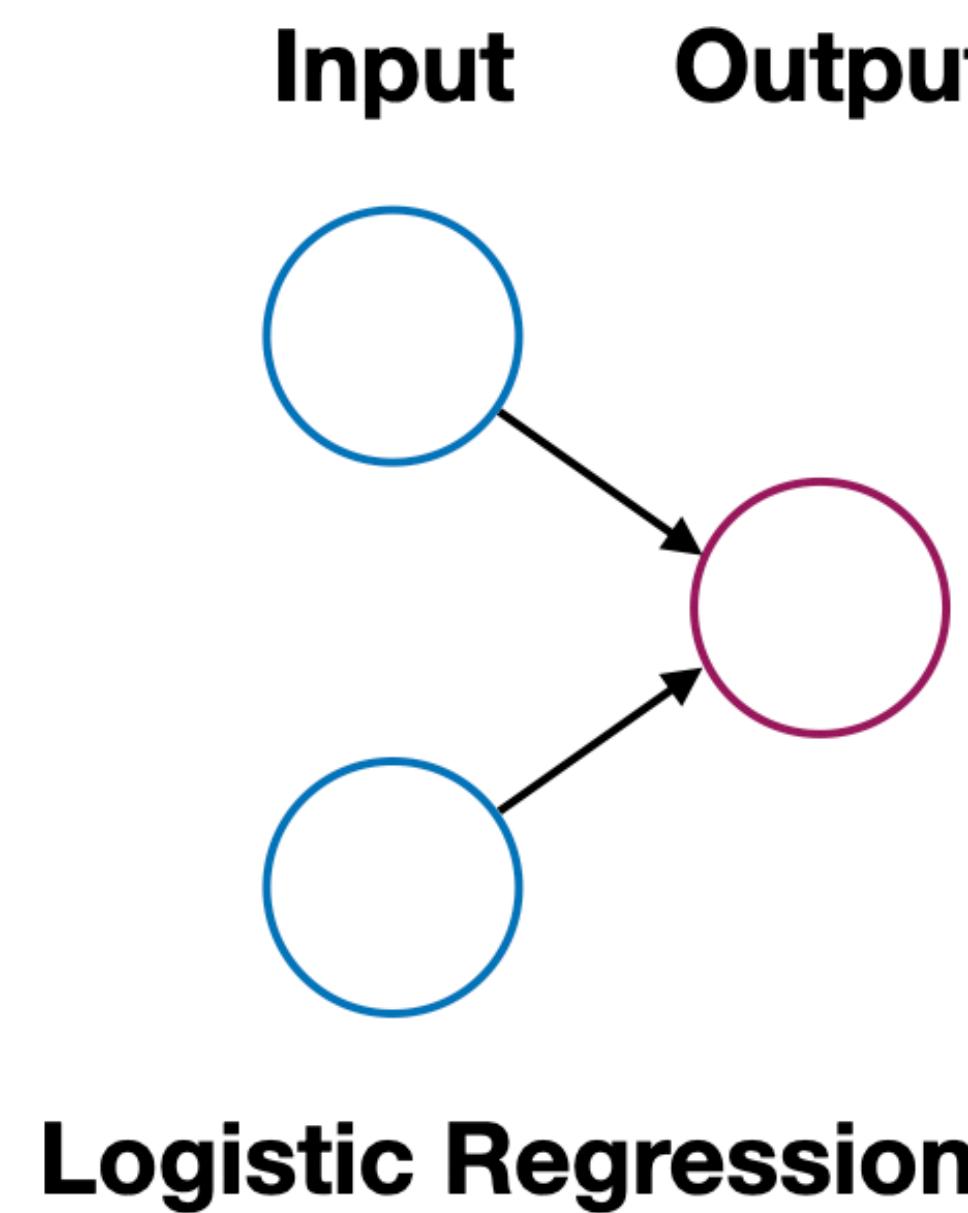


$$y = ax + b$$

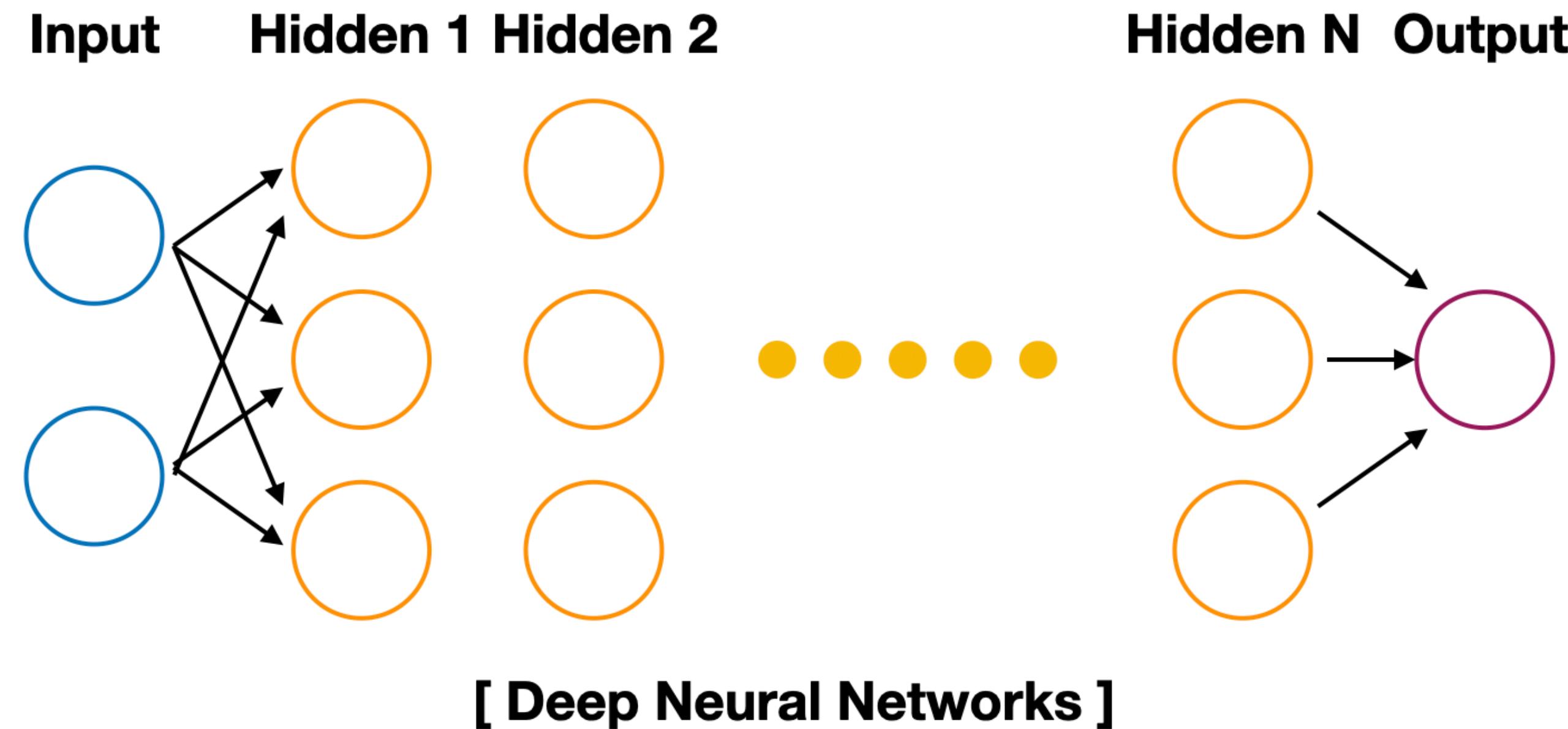


$$y = \frac{1}{1 + e^{-x}}$$

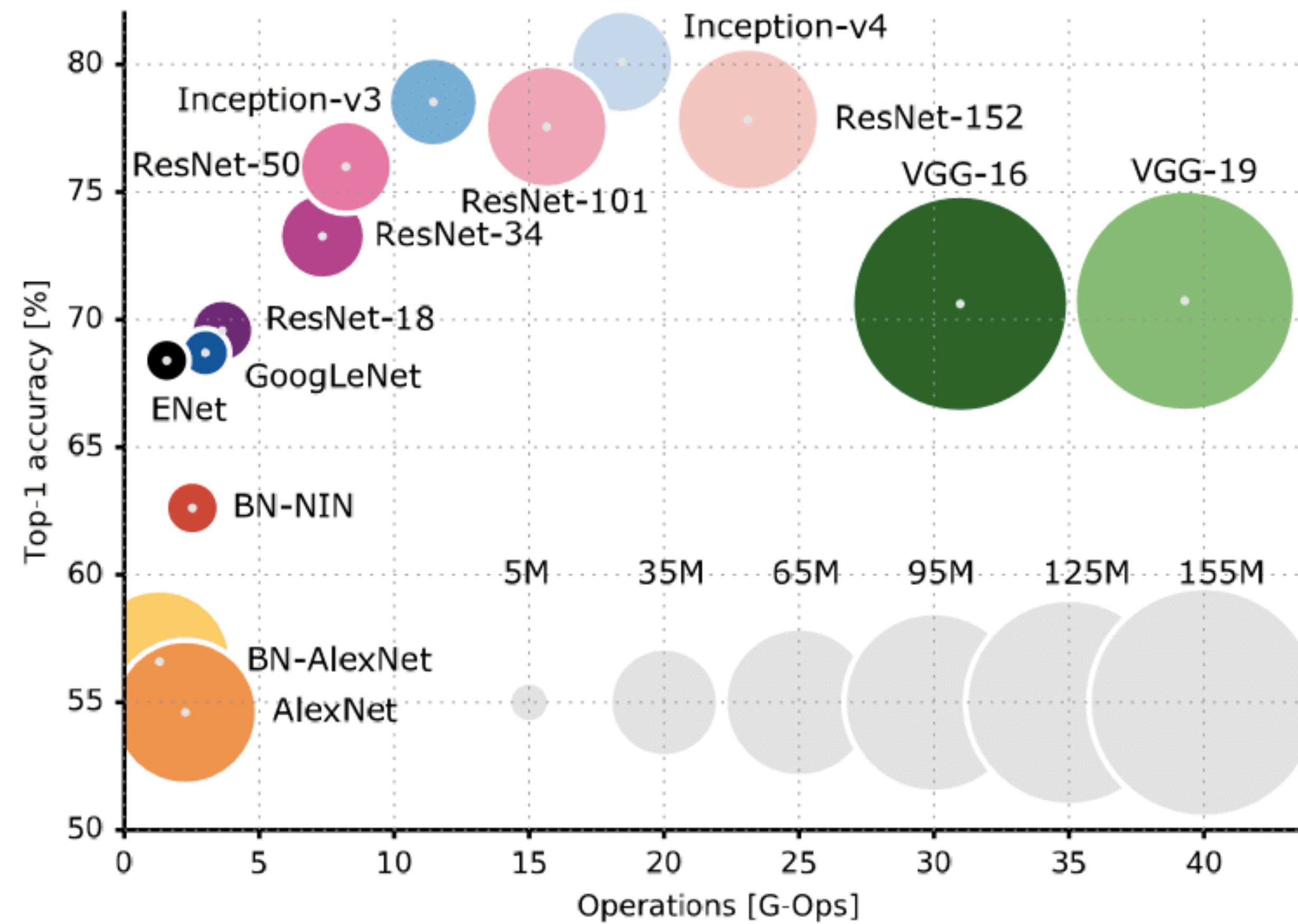
# Logistic Regression의 확장



# Hidden Layer의 증가

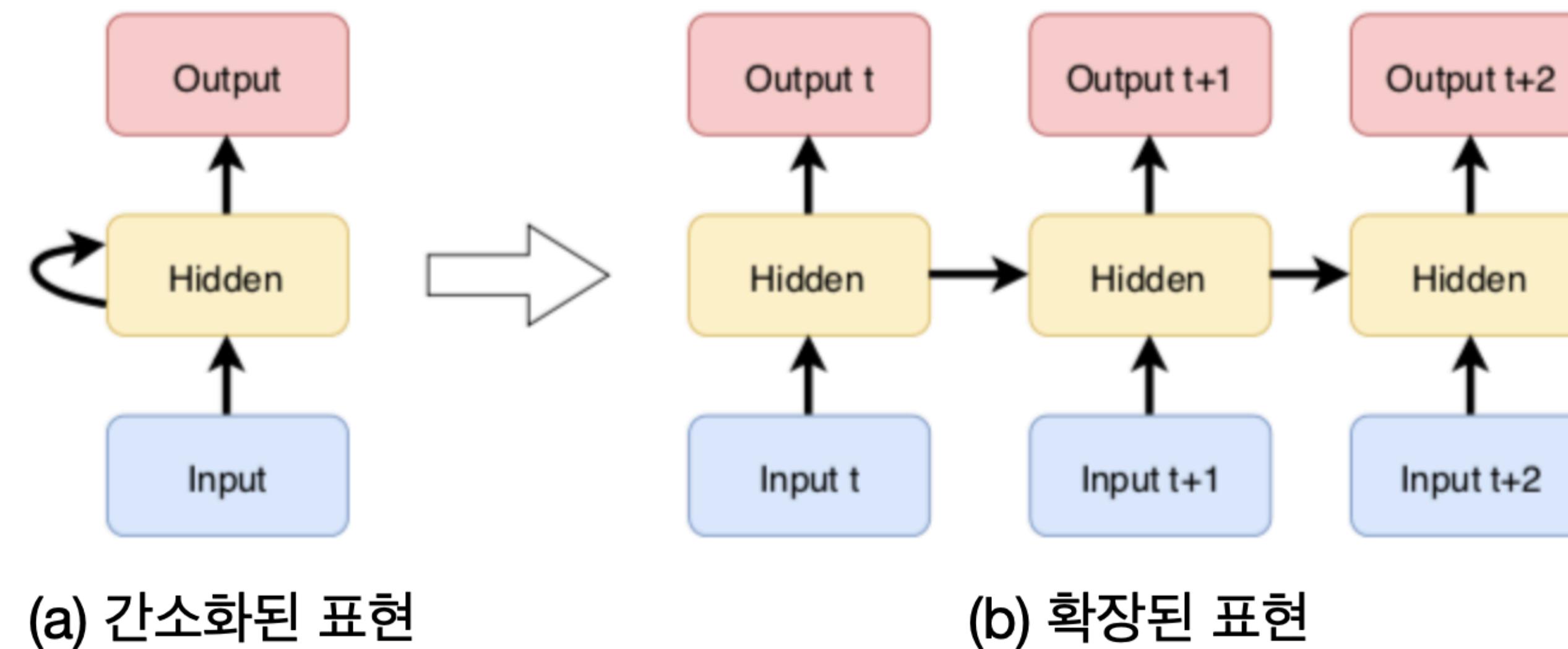


# 다양한 구조의 딥 뉴럴 네트워크들



# 시계열을 고려한 네트워크

## Recurrent Neural Networks



[ RNNs 의 다이어그램 ]

**E.O.D**