

## 2. 인공지능, 머신러닝과 Python 소개

# CONTENTS

01

## AI - ML - DL

02-1 인공지능이란?

02-2 머신러닝(Machine Learning)이란?

02-3 딥러닝(Deep Learning)이란?

02

## Python 기초

인공지능 소개

# 01 AI - ML - DL

## 인공지능-머신러닝-딥러닝

## AI - ML - DL

## Artificial Intelligence

## 인공지능

사고나 학습 등 인간이 가진  
지적 능력을 컴퓨터를 통해  
구현하는 기술



## Machine Learning

## 머신러닝

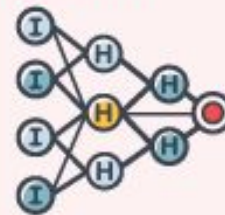
컴퓨터가 스스로 학습하여  
인공지능의 성능을  
향상 시키는 기술 방법



## Deep Learning

## 딥러닝

인간의 뉴런과 비슷한  
인공신경망 방식으로  
정보를 처리



## 02-1 인공지능이란?

### 인공지능에는 어떤것들이 있나?

- 자연어처리 (NLP - Natural language processing)
- \*머신러닝 (Machine Learning)
- 로봇틱 (Robotics) - 스마트 자동차(무인 자동차), 스마트 팩토리, 스마트 홈...
- 비전 (Vision) - 컴퓨터 비전(Image recognition)
- Speech - speech to text, text to speech, translation



## 02-1 인공지능이란?



## 02-1 인공지능이란?

### AI관련 회사 외부투자

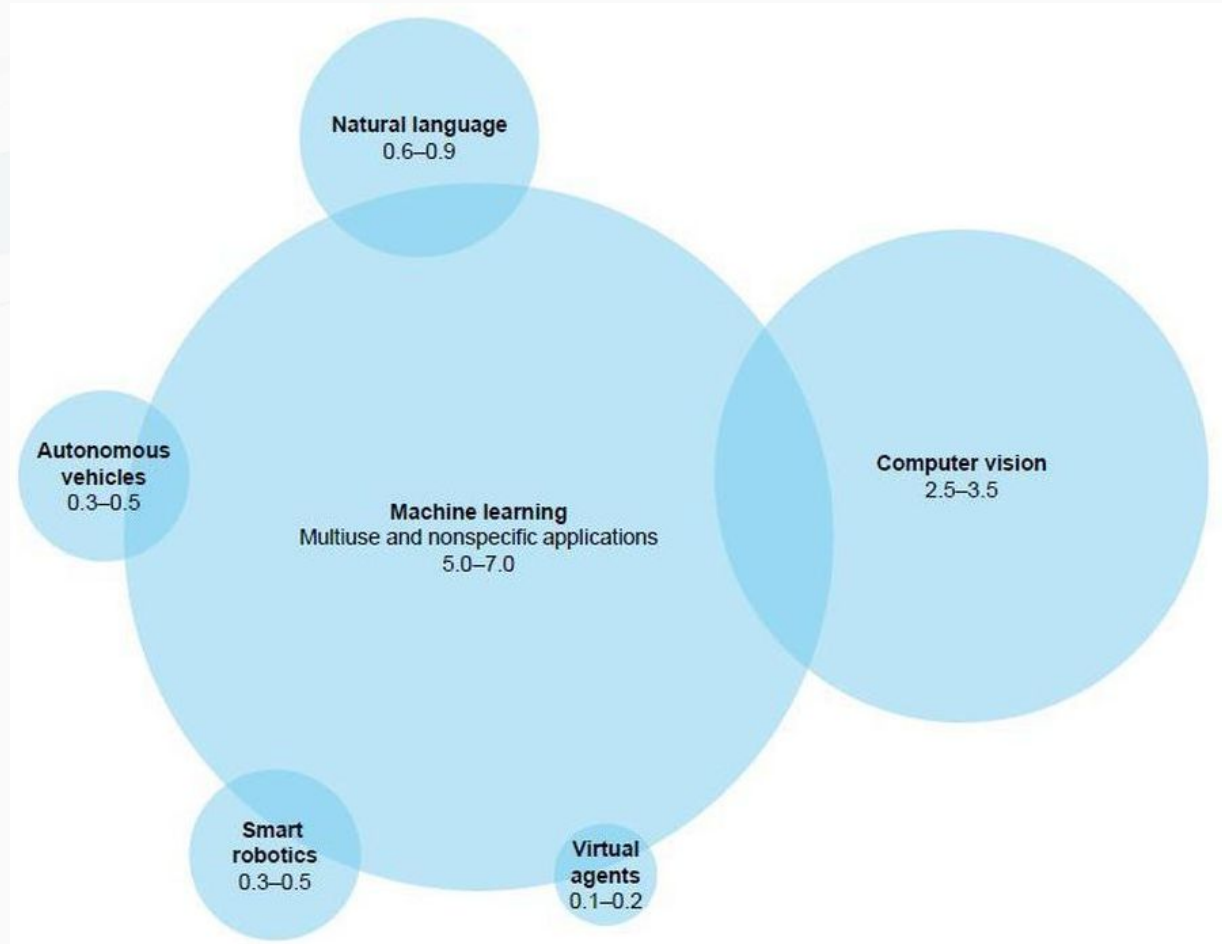
Forbes.com

2016년 기준

단위: \$billion

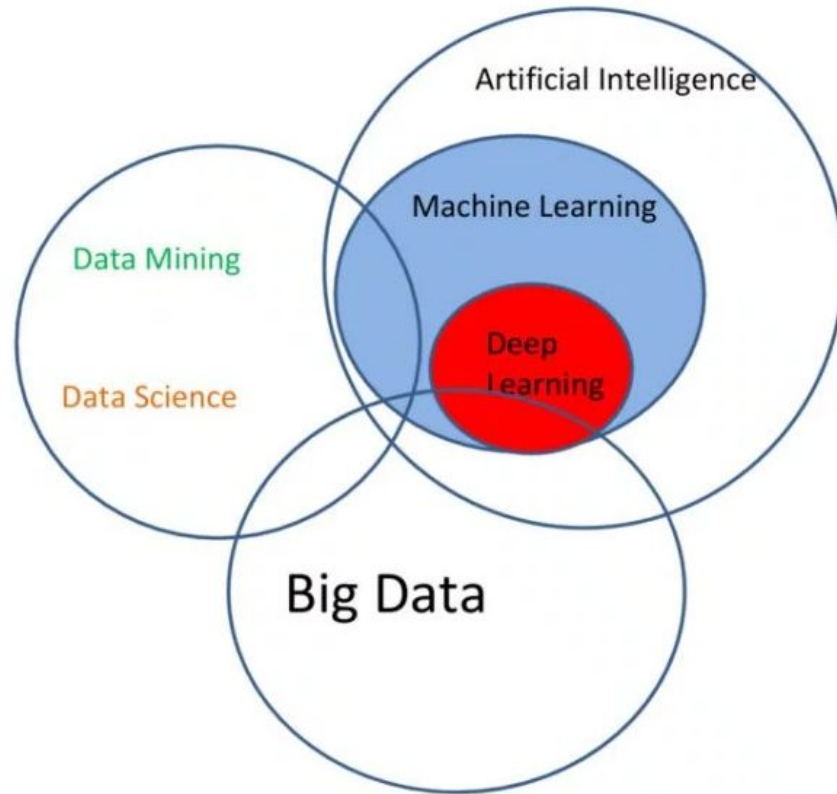
(1조1300억원)

출처: Capital IQ; Pitchbook;  
Dealogic; McKinsey Global  
Institute analysis



# Data Mining, Data Science, Big Data

- 데이터 마이닝 - 통계적 규칙이나 패턴
- 데이터 과학 - 통계학, 수학, 프로그래밍, 데이터





## 02-2 머신러닝(Machine Learning) 이란?

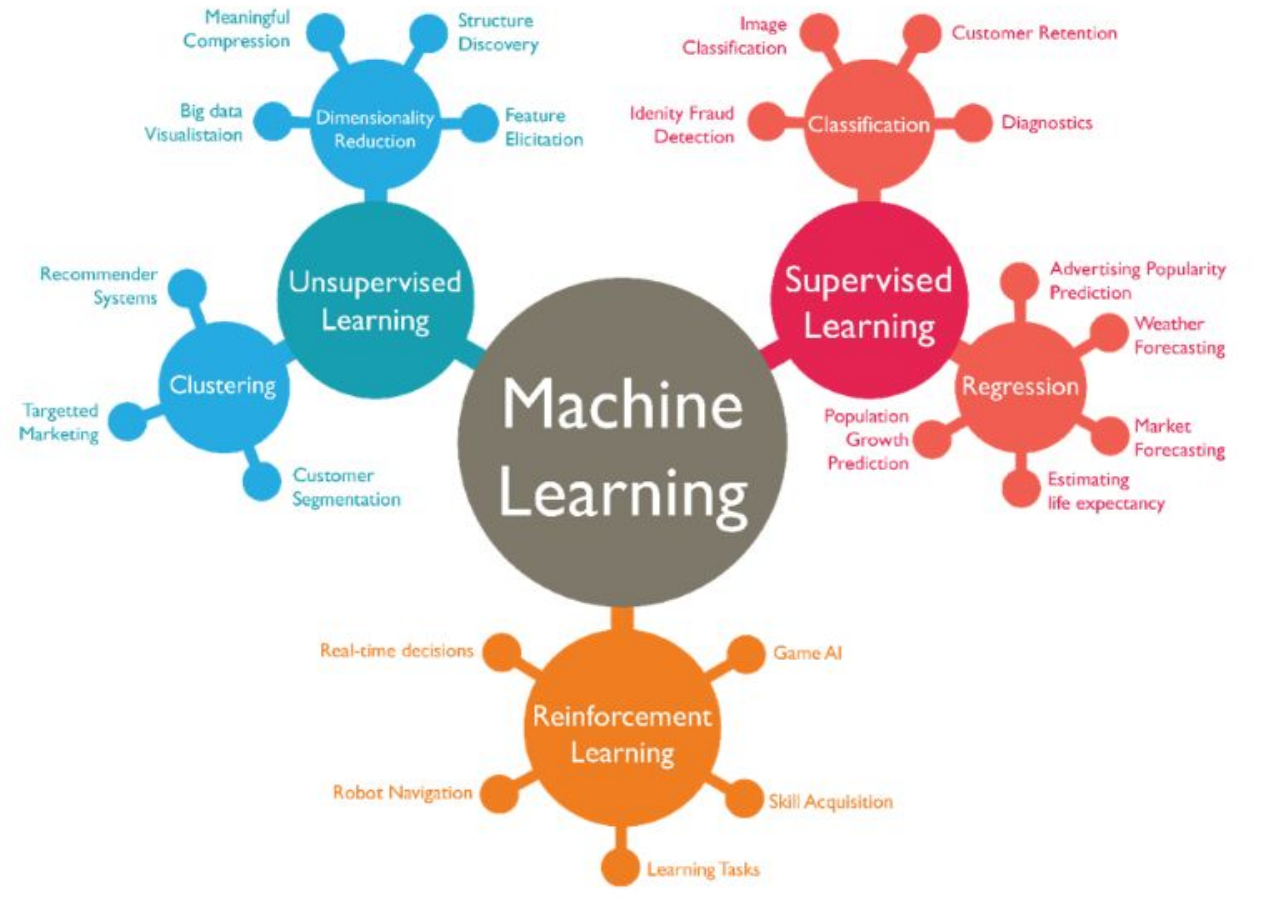
## 지도 학습

- 회귀분석
- 분류

비지도 학습  
(자율학습)

- 군집화
- 차원 축소

## 강화 훈련



## 02-2 머신러닝(Machine Learning) 이란?

### 지도학습 (Supervised Learning)

지도 학습(Supervised Learning)은 데이터에 대한 레이블(Label)-명시적인 정답-이 주어진 상태에서 컴퓨터를 학습시키는 방법이다.

즉, [데이터(data), 레이블(label)] 형태로 학습을 진행하는 방법이다.

대표적으로 **분류(Classification)**와 **회귀(Regression)**가 있다.

## 02-2 머신러닝(Machine Learning) 이란?

### 분류 (Classification)

사과 사진과 오렌지 사진

(데이터) 오렌지색 40% 와 연두색 60% => (레이블) 사과

(데이터) 오렌지색 90% 와 연두색 10% => (레이블) 오렌지

바나나 사진은?



## 02-2 머신러닝 (Machine Learning) 이란?

### 회귀 (Regression)

Small size = \$70,000

Large size = \$160,000

Medium size = \$120,000

위치, 학군, 방#, 층...



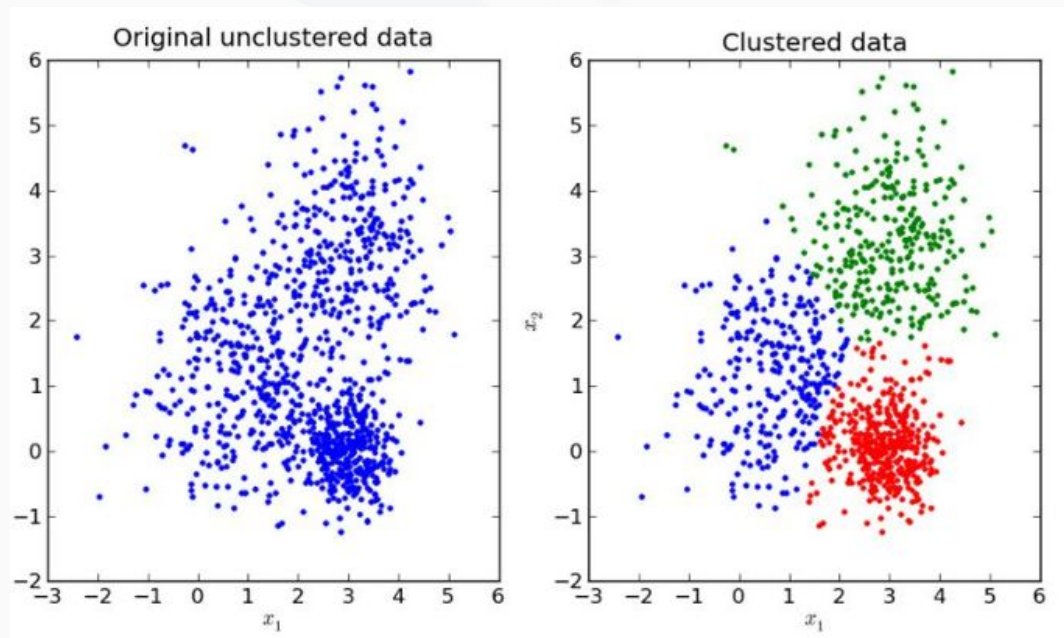
## 02-2 머신러닝(Machine Learning) 이란?

### 비지도학습 (Unsupervised Learning)

**비**지도 학습(Supervised Learning)은 데이터에 대한 레이블(Label)-명시적인 정답-이 **안** 주어진 상태에서 컴퓨터를 학습시키는 방법이다.

비지도 학습은 데이터의 숨겨진 (Hidden) 특징 (Feature)이나 구조를 발견하는데 사용된다.

군집화(Clustering)가 가장 큰 예이다.

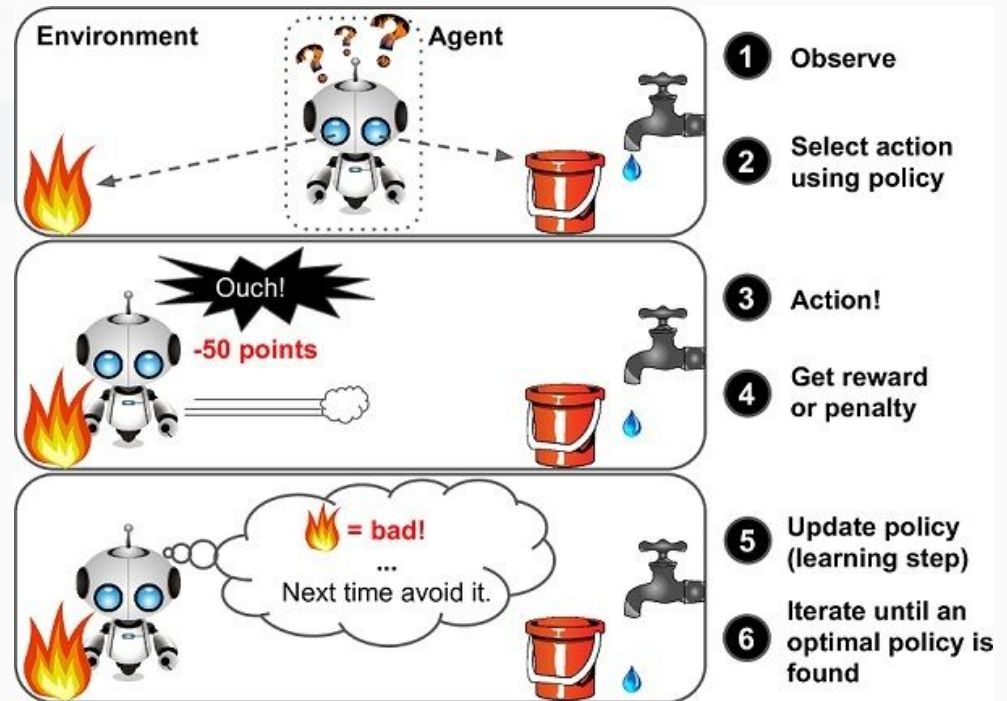




## 02-2 머신러닝(Machine Learning) 이란?

### 강화학습 (Reinforcement Learning)

행동심리학에서 영감을 받았으며,  
어떤 환경 안에서 정의된  
에이전트가 현재의 상태를 인식하여,  
선택 가능한 행동들 중 보상을  
최대화하는 행동 혹은 행동 순서를  
선택하는 방법이다.



## 02-2 머신러닝(Machine Learning) 이란?

### 강화학습 (Reinforcement Learning)

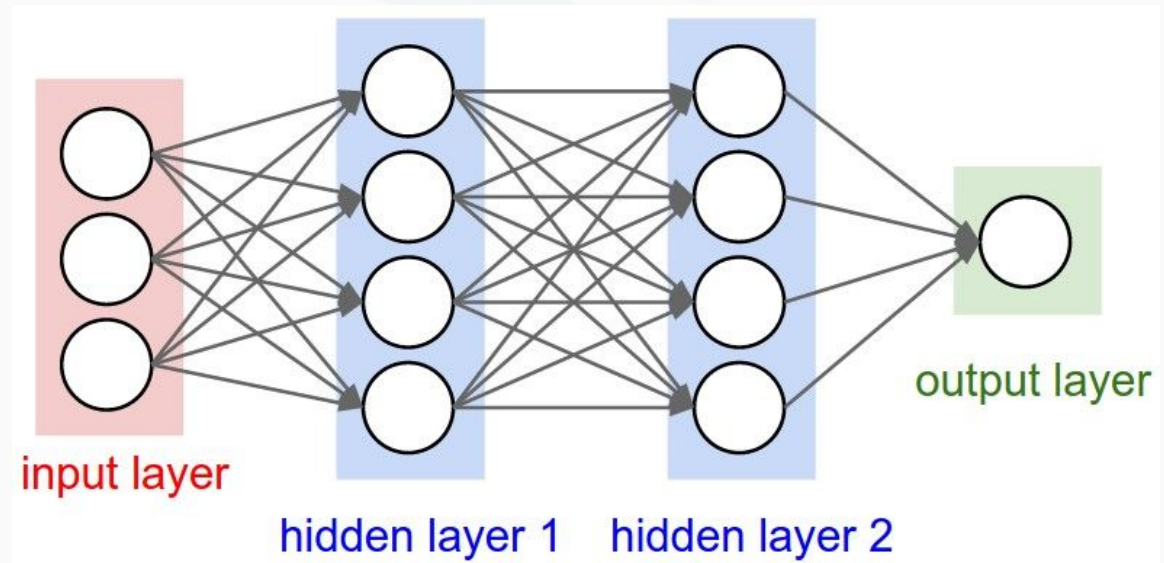
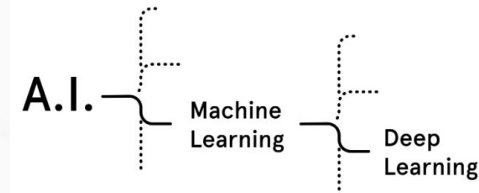


## 02-3 딥러닝(Deep Learning) 이란?

### 딥러닝 (Deep Learning)

다층 신경망의 계산  
(Computation of multi-layer  
neural network)

인공신경망  
(Artificial Neural Network;  
ANN)

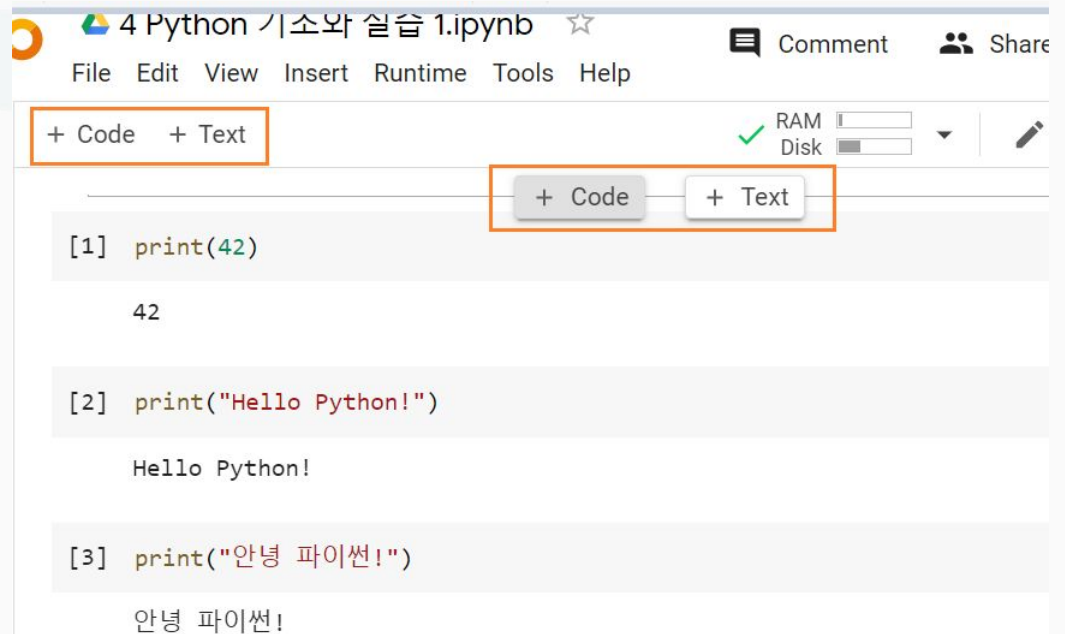




# 02 파이썬(Python) 기초

## 실습 준비

- 중간지점에 마우스를 가지고 가면  
CODE와 TEXT의 옵션이 나타남
- 새로운 코드박스를 만들기
- Ctrl+Enter의 단축키로 실행 가능





## 출력하기(print)

- 숫자출력
- 문자출력

```
[1] print(42)
```

42

```
[2] print("Hello Python!")
```

Hello Python!

```
[3] print("안녕 파이썬!")
```

안녕 파이썬!

## 출력하기

- 작은따옴표('')
- 큰따옴표("")
- print() 함수로 출력할 문장(문자열)은 '' 또는 "" 로 감쌈
- 쉼표/кома(,) - 문자열을 나열할 경우 공백(기본값)이 자동으로 추가
- 연산자 더하기(+) - 문자열을 공백없이 연결

\*쉼표(,) 사용할때 파라미터(매개변수)가 둘(혹이 둘 이상)이 됨



```
print('Hello Python!')

print("Nice to meet you.")

print('Hello "Python"')

print("Hello 'Python'")

print('Hello', 'Python!')

print('Hello' + 'Python!')
```

```
Hello Python!
Nice to meet you.
Hello "Python"
Hello 'Python'
Hello Python!
HelloPython!
```

## 문제풀기

### 출력하기

- 파이썬은 출력전에 연산을 할 수 있을까요? 다음 출력값은 무엇일까요?



```
print(10 - 1)
print(8 - 5)
print(3 + 4)
```

## 주석달기(Comment)

코드는 컴퓨터에게 전달되는 내용이지만, 코드의 일부는 작성자나 다른 사람을 위한 설명으로 채워집니다. 이부분을 주석(comment)이라고 하고 앞에 #을 붙입니다. 연습과정에서는 코드를 연습한 날짜라던가 주의할 부분을 간단하게 기록하는 것부터 시작하면 됩니다.

주석은 중요한 기능중에 하나입니다.

- 내가 작성한 코드를 남들이 보았을때 코드가 긴경우나 복잡한 경우 주석으로 설명
- 보는사람이 이해하기 쉽게
- 주석은 컴퓨터가 읽지 않음

또한, 코딩을 하다보면 오래된 소스인 경우 기억이 잘 나지 않는경우가 있기 때문에, 주석을 설정합니다.

소스에 주석을 다는 습관을 길들이면 정말 유용합니다. 하지만 너무 복잡하게 쓰면 오히려 더 혼란이 올 수도 있습니다.

주석은 **한줄주석**과 **여러줄 주석**으로 나뉩니다.

## 주석달기(Comment) - 한줄주석

한줄주석 에는 샵(#)을 사용

1. 공백에 공간에 사용가능
2. 코드 뒷부분에 사용가능



```
# a는 apple를 의미함  
print("a를 출력")
```

```
print("a를 출력") # a는 apple를 의미함
```

```
a를 출력  
a를 출력
```



## 주석달기(Comment) - 여러줄 주석

- 큰따옴표 `"""` 3개 연달아
- 작은따옴표 `'''` 3개를 연달아

시작과 끝부분에 사용



#큰 따옴표 `"""` 주석방법

```
print("퇴근하고 싶습니다 Start")
"""
오늘은 날씨가 참 좋아요.
시계를 보니 아직 퇴근시간은
아직멀었어요. 일하기가 싫네요.
"""
print("퇴근하고 싶습니다 End ")
```

#작은 따옴표 `'''` 주석방법

```
print("오늘은 연차 사용날 Start")
'''
오늘은 날씨가 참 좋아요.
시계를 보니 아직 퇴근시간은
아직멀었어요. 일하기가 싫네요.
'''
print("벌써 하루가 저물었네요 End ")
```

```
퇴근하고 싶습니다 Start
퇴근하고 싶습니다 End
오늘은 연차 사용날 Start
벌써 하루가 저물었네요 End
```

## 주석달기(Comment) - 주석 에러

‘들여쓰기’ -

주석은 소스와 동일하게 들여쓰기를  
해야합니다.



```
print("오늘은 연차 사용날 Start")
```

```
'''
```

```
    오늘은 날씨가 참 좋아요.  
    시계를 보니 아직 퇴근시간은  
    아직 멀었어요. 일하기가 싫네요.
```

```
'''
```

```
print("벌써 하루가 저물었네요 End ")
```

```
File "<ipython-input-3-f6c964ceb6ec>", line 2
```

```
'''
```

```
^
```

```
IndentationError: unexpected indent
```

SEARCH STACK OVERFLOW

## 변수 (Variable)

지금까지 파이썬에서 숫자와 연산자를 사용하여 직접 계산을 해보았습니다. 그런데 계산 결과를 바로 출력할 수는 있었지만 결과를 계속 가지고 있을 수는 없었습니다.

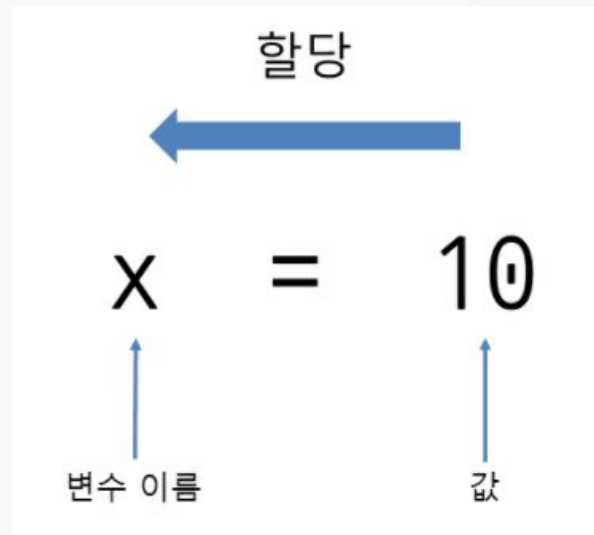
이때는 변수를 사용해서 결과를 저장해야 합니다.

이번 유닛에서는 변수(variable)를 만들어 결과를 저장하는 방법을 알아보겠습니다.

## 변수 (Variable)

### 변수 만들기

파이썬에서는 다음 그림과 같은 형식으로 코드를 입력하여 변수를 만듭니다.



## 변수 (Variable)

### 변수 만들기

변수 이름은 원하는 대로 지으면 되지만 다음과 같은 규칙을 지켜야 합니다.

- 영문 문자와 숫자를 사용할 수 있습니다.
- 대소문자를 구분합니다.
- 문자부터 시작해야 하며 숫자부터 시작하면 안 됩니다.
- \_(밑줄 문자)로 시작할 수 있습니다.
- 특수 문자(+, -, \*, /, \$, @, &, % 등)는 사용할 수 없습니다.
- 파이썬의 키워드(if, for, while, and, or 등)는 사용할 수 없습니다.



## 변수 (Variable)

변수 만들기

\*변수의 경우 `print()` 함수 없이 출력 가능



```
x = 10
x
```

10

The image shows a code execution window. The top part contains the code `x = 10` followed by `x` on a new line. The bottom part shows the output `10`.

## 변수 (Variable)

변수 만들기

\*문자열도 가능



```
y = 'Hello world!'  
y
```

```
'Hello world!'
```

## 변수 (Variable)

변수 여러 개를 한 번에 만들기

\*심표(.)를 사용해서 여러 개의 변수만들기가  
한 줄에 가능

```
[15] x, y, z = 10, 20, 30  
x
```

10

```
[16] y
```

20

```
▶ z
```

30

## 변수 (Variable)

“Variables are the dynamic heart of programming. Variables make code more than a static set of instructions. They allow logic to occur, enabling developers to measure time, analyze data, and customize the program to the user.”

“변수는 프로그래밍의 역동적인 핵심입니다. 변수는 코드를 정적 명령어 세트 이상으로 만듭니다. 이를 통해 로직이 발생하여 개발자가 시간을 측정하고 데이터를 분석하고 사용자에게 맞게 프로그램을 사용자 정의 할 수 있습니다.” (google translate)

## 변수 (Variable)

1. Be Clear (명확하게)
  - N\_FLT
  - dRange vs. dateRange
  - tmp, x, i (for loop에서는 쓰임)
2. Be Concise (간결하게)
  - ThisPersonsFavoriteColor
3. Be Consistent (일관성있게)
  - PascalCase, camelCase, snake\_case, lowercase, or hyphen-case
  - Hungarian Notation(헝가리안 노테이션) - 예: iAge, sName

## 정수와 실수 (int & float)

### 정수(int)

- 영어로 *integer*, 줄여서 파이썬에서는 `int`라고 표현
- 정수끼리 더하거나 곱하거나 빼면 정수
- 정수끼리 나누면 실수가 나올 수 있으나, 나눗셈의 몫만을 구하려면 `//`연산자를 이용

```
a = 5//3
```

- 실수를 정수로 바꾸려면 `int`를 이용

`a=int(5.4)`라고 하면 `a`는 5를 값으로 가지게 된다.

```
[2] int_div = 5 / 3  
int_div
```

```
1.6666666666666667
```

```
[4] int_div_int = 5 // 3  
int_div_int
```

```
1
```



## 정수와 실수 (int & float)

### 실수

- 부동소수점(floating point)이라는 표현법을 이용해 소수점을 표시할 수 있는 숫자
- 어느정도의 계산 정확도는 가지지만, 계산에 있어서 완벽한 정확성은 가지지 않는다.

```
0.1+0.1+0.1 == 0.3
```

- 정수를 실수로 바꾸려면 float를 사용

a=float(5)라고 하면 a는 5.0을 값으로 가지게 된다.

## 문자열 (string)

1. 텍스트 두개를 더하면 문자열이 이어붙여짐
2. 텍스트는 더하기만 가능하고, 빼기(-) 등 다른 계산은 불가능



```
text = '2015' + '1991'  
text
```

```
'20151991'
```



```
text = '2015' - '1991'  
text
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-8-57cadd6c2ad1> in <module>()  
----> 1 text = '2015' - '1991'  
      2 text
```

```
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

SEARCH STACK OVERFLOW

## type() 함수

데이터타입을 확인 하기 위해서는 **type()** 을 사용하여 확인



```
print(type(123))  
print(type(12.3))  
print(type('123'))
```

```
<class 'int'>  
<class 'float'>  
<class 'str'>
```

## 문제풀기

**type() 함수**

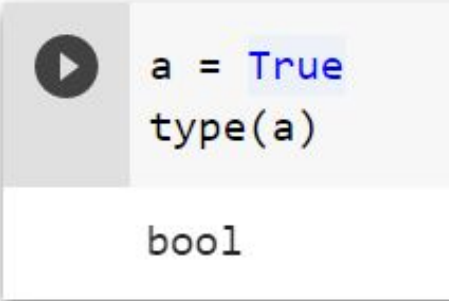
각 값을 변수에 넣고 (혹은 바로), **type**를 맞춰봅시다.

1. 5
2. 'c'
3. 'Ireland'
4. 7.0
5. 'Colin'

## 불(bool)

불리언(boolean)의 줄임말로 불(bool) 자료형이란 참(True)과 거짓(False)을 나타내는 자료형. 불 자료형은 다음 2가지 값만을 가질 수 있음:

- True - 참
- False - 거짓



```
a = True
type(a)

bool
```

※ True나 False는 파이썬의 예약어로 `true`, `false`와 같이 사용하지 말고 첫 문자를 항상 대문자로 사용

## 불(bool)

불 자료형은 조건문의 반환 값으로도 사용. 조건문에 대해서는 if문에서 자세히 배우겠지만 잠시 살펴보고 넘어가자.



`1 == 1` 은 "1과 1이 같은가?"를 묻는 조건문. 이런 조건문은 결과로 **True** 또는 **False**에 해당되는 불 자료형을 돌려줌. 1과 1은 같으므로 **True**를 돌려줌.



## 불(bool)

2는 1보다 크기 때문에  $2 > 1$  조건문은 **True**를 돌려준다.



2는 1보다 작지 않기 때문에  $2 < 1$  조건문은 **False**를 돌려준다.



## 조건문 - if

특정 조건에 따라 다른 동작을 할 수 있도록 해 주는 구문



```
people = 12  
apple = 9  
if people > apple:  
    print('사람이 너무 많아! 몇 명은 못먹겠네')
```

사람이 너무 많아! 몇 명은 못먹겠네

구조

1. if 예약어 : 조건문의 시작을 알림
2. 조건: `people > apple`와 같이 참/거짓을 판단할 수 있는 조건
3. : 조건이 끝났다는걸 표현한하는 명령
4. 실행하고자 하는 코드. 코드는 **탭(tab)**키를 이용해서 들여서 쓴다.
  - 예. `print('사람이 너무 많아! 몇 명은 못먹겠네')`

## 조건문 - if

조건식에 대소비교가 쓰인 경우



```
if 3<5:  
    print("조건식이 True이므로 실행됩니다.")  
if 3>5:  
    print("조건식이 False이므로 실행되지 않습니다.")
```

위의 예처럼 3은 5보다 작으므로 3<5가 참인 if문만 실행됩니다.

## 조건문 - if

조건식에 True/False가 쓰인 경우



```
if True:
    print("조건식이 True이므로 실행됩니다.")
if False:
    print("조건식이 False이므로 실행되지 않습니다.")
```

조건식이 True이므로 실행됩니다.

조건문이 참인 True의 if문만 실행됩니다.

## 조건문 - 들여쓰기

‘들여쓰기’는 파이썬 코더한테 실수(혹은 에러)가 잘 일어나는 부분입니다.



# 들여쓰기가 잘못된 예제

```
a, b = 4, 6
```

```
if a>b:
```

```
print("a가 b보다 크다")
```

File "<ipython-input-9-13097351e3c6>", line 4

```
print("a가 b보다 크다")
```

^

IndentationError: expected an indented block

SEARCH STACK OVERFLOW

## 조건식 - **boolean(bool)**연산

- 예. a는 20대이다.

```
20 <= a and a < 30
```

- 예. a는 18세 미만 또는 60세 이상이다.

```
a < 18 or 60 <= a
```

## 블럭 (block)

1. 함께 실행 되는 하나의 코드 덩어리
2. 들여쓰기로 블럭을 구분한다.
3. 들여쓰기가 어긋나면 오류가 발생한다.
4. 블럭 안에 다른 블럭이 들어갈 수 있다.
5. 내부의 블럭은 외부의 블럭에 종속적
6. 파이썬 코드 전체를 하나의 블럭으로 볼 수 있다.



```
a = 10
b = 7
if a > 5:
    print(a)
    if b < 8:
        print('b < 8')
    if b > 7:
        print(a)
        print('b > 7')
```



## 조건문 - if else

else

- if의 조건이 맞지 않는 경우 항상 실행
- 반드시 if뒤에 나와야 한다.



```
mine = "가위"  
yours = "바위"  
DRAW = "비김"  
if mine == yours:  
    result = DRAW  
else:  
    result = '이기거나 지거나'  
result
```

## 조건문 - elif

elif - (else if)의 줄임

- else 와 if의 결합으로 조건이 맞지 않는 경우 다른 경우를 검사



```
mine = 1
SCISSOR = 1
ROCK = 2
if mine == SCISSOR:
    result = '가위'    # 조건이 참일 때 실행
elif mine == ROCK:
    result = '바위'    # 다른 조건이 참일 때 실행
else:
    result = '나머지'  # 조건이 거짓일 때 실행
```



# 01 기초와 실습 3

01-1 리스트(list)

## List 사용

- 여러개의 값을 담을 수 있는 변수
- 값 읽어오기
  - 리스트를 사용할 때는 0번째가 첫번째
  - 첫번째 값 `list1[0]`
  - 두번째 값 `list1[1]`
  - 뒤에서 첫번째 값 `list1[-1]`
  - 뒤에서 두번째 값 `list1[-2]`
  - 리스트에 들어있는 값 보다 큰 값을 읽어오려고 하면 에러
  - 예. 위의 `list1`에서 `list1[5]` 또는 `list1[-6]`은 에러
- 값 쓰기
  - 변수와 같이 `list1[0]=10`이라고 하면 `list`의 첫번째 값이 10으로 변경



```
list1 = [1, 2, 3, 4, 5]
```

## List 사용

list의 길이는 len() 함수를 사용



```
numbers= [1, 2, 3, 4, 5]  
print(len(numbers))
```

5

## List 수정

리스트에 새로운 값을 추가하는 방법

- `list1=[1,2,3]` 이라고 할 때
- `append`를 사용
  - `list1.append(4)`
  - `append`를 이용하면 리스트에 새로운 값이 추가된다.
- 뒤에 새로운 리스트를 더하기
  - `list2 = list1 + [4]`
  - `list1`은 그대로 두고, 새로운 리스트를 만들어 낸다.

## List 수정

리스트에 값이 들어있는지 확인하는 방법

- in 연산을 이용



```
#12라는 값이 리스트에 들어있는지 확인하는 코드  
n=12  
if n in list1:  
    print('{}가 리스트에 있다.'.format(n))
```



## List 2차원

- 한 줄로 늘어선 1차원 리스트, 평면 구조의 2차원 리스트
- 2차원 리스트는 다음과 같이 가로×세로 형태로, 행(row)과 열(column) 모두 0부터 시작



The diagram illustrates a 2D list structure. A horizontal blue arrow at the top is labeled '가로 크기' (Horizontal Size). A vertical blue arrow on the left is labeled '세로 크기' (Vertical Size). Below these arrows is a 3x4 grid representing the list. The columns are labeled '열 0', '열 1', '열 2', and '열 3' from left to right. The rows are labeled '행 0', '행 1', and '행 2' from top to bottom.

	열 0	열 1	열 2	열 3
행 0				
행 1				
행 2				

## List 2차원

2차원 리스트는 리스트 안에 리스트를 넣어서 만들 수 있으며 안쪽의 각 리스트는 ,(콤마)로 구분

- 리스트 = [[값, 값], [값, 값], [값, 값]]



```
a = [[10, 20], [30, 40], [50, 60]]
```

```
a
```

```
[[10, 20], [30, 40], [50, 60]]
```

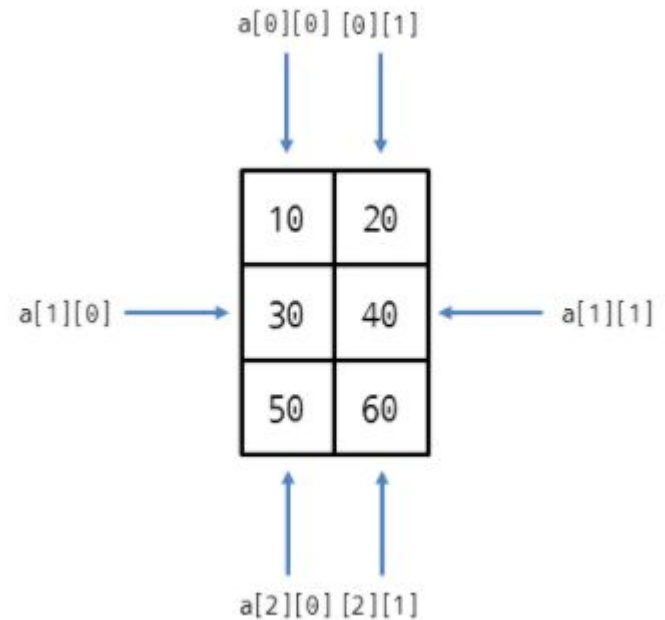
## List 2차원

2차원 리스트의 요소에 접근하거나 값을 할당할 때는 리스트 뒤에 `[]`(대괄호)를 두 번 사용하며 `[]` 안에 세로(row) 인덱스와 가로(column) 인덱스를 지정

- 리스트[세로인덱스][가로인덱스]
- 리스트[세로인덱스][가로인덱스] = 값

2차원 리스트도 인덱스는 0부터 시작 따라서 리스트의 가로 첫 번째, 세로 첫 번째 요소는 `a[0][0]`

```
▶ a = [[10, 20],
        [30, 40],
        [50, 60]]
a
[[10, 20], [30, 40], [50, 60]]
```



## List 2차원

```
▶ a = [[10, 20],  
        [30, 40],  
        [50, 60]  
        ]  
print('a[0][0] = ', a[0][0])  
print('a[1][1] = ', a[1][1])  
a[2][1] = 1000  
print('a = ', a)
```

```
a[0][0] = 10  
a[1][1] = 40  
a = [[10, 20], [30, 40], [50, 1000]]
```

## List - 인덱스 슬라이싱 [:]

Monday이 오프셋(인덱스)은 0, Tuesday는 1, Wednesday는 2, Thursday는 3, Friday는 4.  
따라서 week[2:5] 란? week 2인 Wednesday 부터, 5 - 1 = 4 즉, 4는 Friday 이므로  
['Wednesday', 'Thursday', 'Friday']가 출력 됨.



```
week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']  
week[0:2]
```

```
['Monday', 'Tuesday']
```

```
week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

## List - 인덱스 슬라이싱 [:]

숫자생략이 가능

**[start:]** start오프셋(인덱스)부터 끝까지

**[:end]** 처음부터 end-1 오프셋(인덱스)까지

- `week[3:]`
- `week[:2]`
- `week[::2]` ? 이건 값이 이상하네요?

=> 다음 페이지에서 설명

```
[5] week[3:]
```

```
['Thursday', 'Friday']
```

```
[6] week[:2]
```

```
['Monday', 'Tuesday']
```

```
[8] week[::2]
```

```
['Monday', 'Wednesday', 'Friday']
```

## 같이하기

### List - 인덱스 슬라이싱 [:]

step

**[start : end : step]** step만큼 문자를 건너뛰면서 추출

예)



```
rainbow=['빨강','주황','노랑','초록','파랑','남색','보라']  
rainbow[0:8:2]
```

## for in list

### for in 반복문

- 코드를 필요한만큼 반복해서 실행



```
for pattern in patterns:  
    print (pattern)
```

- 리스트 `patterns`의 값을 하나씩 꺼내 `pattern`으로 전달
- 리스트의 길이만큼 `print (pattern)` 실행
- `:`(콜론)은 꼭 사용해야 하며, 들여쓰기도 중요



## for in range

range() 함수

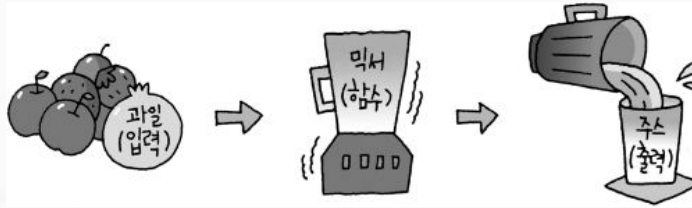
- 필요한 만큼의 숫자를 만들어내는 유용한 기능



```
for i in range(5):  
    print(i)
```

```
0  
1  
2  
3  
4
```

## 함수란?



함수를 설명하기 전에 믹서를 생각해 보자. 우리는 믹서에 과일을 넣는다. 그리고 믹서를 사용해서 과일을 갈아 과일 주스를 만든다.

우리가 믹서에 넣는 과일은 "입력"이 되고 과일 주스는 "출력(결과값)"이 된다.

그렇다면 믹서는 무엇인가? (믹서는 과일을 입력받아 주스를 출력하는 함수와 같다.)

우리가 배우려는 함수가 바로 믹서와 비슷하다. 입력값을 가지고 어떤 일을 수행한 다음에 그 결과물을 내어놓는 것, 이것이 바로 함수가 하는 일이다. 우리는 어려서부터 함수에 대해 공부했지만 함수에 관해 깊이 생각해 본 적은 별로 없다. 예를 들어  $y = 2x + 3$ 도 함수이다.

하지만 이를 수학 시간에 배운 직선 그래프로만 알고 있지  $x$ 에 어떤 값을 넣었을 때 어떤 변화에 의해서  $y$  값이 나오는지 그 과정에 대해서는 별로 관심을 두지 않았을 것이다.

이제 우리는 함수에 대해 조금 더 생각해 보는 시간을 가져야 한다. 프로그래밍에서 함수는 정말 중요하기 때문이다. 자, 이제 파이썬 함수의 세계로 깊이 들어가 보자.

## 함수는 왜 사용하는가?

- 똑같은 내용을 반복해서 작성할때
- "반복적으로 사용되는 가치 있는 부분"을 한 문치로 묶음
- 어떤 입력값을 주었을 때 어떤 결과값을 돌려준다
- 프로그램을 함수화하면 프로그램 흐름을 일목요연하게 볼 수 있음.
- 프로그램 흐름도 잘 파악할 수 있고 오류가 어디에서 나는지도 알기 쉬움
- 함수를 잘 사용하고 함수를 적절하게 만들 줄 아는 사람이 능력 있는 프로그래머

## 파이썬 함수의 구조

- **def**는 함수를 만들 때 사용하는 예약어
- 함수 이름은 자유
- 함수 이름 뒤 괄호 안의 **parameter**(매개변수)는 이 함수에 입력으로 전달되는 값을 받는 변수

```
def 함수명(매개변수):  
    <수행할 문장1>  
    <수행할 문장2>  
    ...
```



```
def add(a, b):  
    return a + b
```

- "이 함수의 이름(함수명)은 **add**이고 입력으로 2개의 값을 받으며 결과값은 2개의 입력값을 더한 값이다."
- 여기에서 **return**은 함수의 결과값을 돌려주는 명령어이다.

## 함수란?

1. 함수는 코드의 덩어리에 이름을 붙인 것이다.
2. 새 함수를 정의할 수 있다.
3. `print()`는 미리 만들어진 함수이다.
4. 함수를 한번 만들고 나면, 그 안은 잊어버려도 좋다.

```
[35] def hello(): # 함수의 정의  
      print('안녕, 함수!')
```

```
▶ print('첫줄 실행')  
   hello() # 함수의 호출  
   print('끝줄 실행')
```

```
첫줄 실행  
안녕, 함수!  
끝줄 실행
```

## 매개변수 (parameter)

- 매개변수 - 함수를 정의할 때 사용하는 이름
- 실행인자 - 함수를 실행할 때 넘기는 변수, 값
- 매개변수와 실행 인자
  - 매개변수와 실행 인자의 개수는 동일해야 한다.
  - 여러 개일 경우 쉼표로 구분

## 매개변수

```
▶ def print_round(number):    # 함수의 정의
    rounded = round(number)
    print(rounded)

print_round(4.6)              # 함수의 호출
print_round(2.2)
```

```
5
2
```

- 매개변수 - 함수를 정의할 때 사용하는 이름
- 실행 인자 - 함수를 실행할 때 넘기는 변수, 값
- 매개변수와 실행 인자
  - 매개변수와 실행 인자의 개수는 동일해야 한다.
  - 여러 개일 경우 쉼표로 구분

## 함수의 값 (return)

- return을 이용해 값을 돌려줌
- 여러 값 반환 - return 뒤에 여러 값을 쉼표로 구분해서 값을 보내고, 받을때도 쉼표로 구분하여 받는다.



```
def add_10(value):  
    result = value + 10  
    return result
```

```
n = add_10(5)  
print(n)
```

15



## 클래스란?

- 변수와 함수를 모아놓은 것
- 객체(Object) - ex) **person1**
- 인스턴스(instance) - ex) **person1 = Person()**

### 조금 쉽게 이해해 보기

머린시절 뽀기를 해 본적이 있다면 아래 그림과 비슷한 모양의 뽀기 틀을 본적이 있을 것이다. 뽀기 아저씨가 뽀기를 불에 달군 후 평평한 바닥에 '탁'하고 소리나게 떨어뜨려 납작하고 동그랗게 만든후에 아래와 비슷한 틀로 모양을 찍어준다. 찍어준 모양대로 모양을 만들어 오면 아저씨는 뽀기 한개를 더 해 준다.

이곳에서 설명할 클래스라는 것이 마치 위 뽀기의 틀(별모양, 하트모양)과 비슷하다. 별 모양의 틀(클래스)로 찍으면 별모양의 뽀기(인스턴스)가 생성되고 하트 모양의 틀( 클래스)로 찍으면 하트모양의 뽀기( 인스턴스)가 나오는 것이다.

클래스란 똑같은 무엇인가를 계속해서 만들어 낼 수 있는 설계도면 같은 것이고(뽀기 틀), 인스턴스란 클래스에 의해서 만들어진 피조물(별 또는 하트가 찍혀진 뽀기)을 뜻하는 것이다.

## 클래스 속성 변경

```
▶ class Person:  
    name = "철수"
```

```
[9] person1 = Person()  
    person1.name
```

```
'철수'
```

```
▶ person1.name = "Pamela Harris"  
    person1.name
```

```
'Pamela Harris'
```

## 클래스 만들기

### 함수 만들기

```
[34] class Person:
```

```
    def print_name(self):  
        print("print_name() 안에 Mr. 바틀즈")
```



```
bartles.print_name()
```

```
print_name() 안에 Mr. 바틀즈
```

### self란?

- class의 인스턴스(instance)를 나타내는 변수
- self는 class내 들의 첫번째 인자로 전달

## 클래스 만들기

self를 사용해 함수로 내장된 변수  
사용하기

- class안에 있는 변수를 함수에서 사용할 때 self를 사용

```
[36] class Student:  
      name = "김철수"  
      def info(self):  
          print("제 이름은 " + self.name + "입니다.")
```

```
[37] inst = Student()  
      type(inst)
```

```
__main__.Student
```

```
[38] inst.name
```

```
'김철수'
```

```
▶ inst.info()
```

```
제 이름은 김철수입니다.
```



## Reference

<https://www.datasciencecentral.com/profiles/blogs/artificial-intelligence-vs-machine-learning-vs-deep-learning>

[https://en.wikipedia.org/wiki/Arthur\\_Samuel](https://en.wikipedia.org/wiki/Arthur_Samuel)

<https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/#2f803ca115e7>

<http://tobetong.com/?p=9393&ckattempt=1>

[https://www.alibaba.com/product-detail/ANPR-no-stop-hand-free-automatic\\_60612355438.html](https://www.alibaba.com/product-detail/ANPR-no-stop-hand-free-automatic_60612355438.html)

[http://news.chosun.com/site/data/html\\_dir/2011/08/10/2011081000126.html](http://news.chosun.com/site/data/html_dir/2011/08/10/2011081000126.html)

비정형데이터분석\_9회차\_인공지능의 개념

<https://dataconomy.com/2015/01/whats-the-difference-between-supervised-and-unsupervised-learning>



## Reference

<https://withcoding.com/64>

<https://Insideout.tistory.com/entry/Python-%ED%8C%8C%EC%9D%B4%EC%84%A0-%EC%A3%BC%EC%84%9D-%EC%82%AC%EC%9A%A9%EB%B2%95%ED%95%9C%EC%A4%84%EC%97%AC%EB%9F%AC%EC%A4%84%EB%8B%A8%EC%B6%95%ED%82%A4>

<https://www.datacamp.com/>

<https://dojang.io/mod/page/view.php?id=2176>

<https://www.freshconsulting.com/development-principle-1-choose-appropriate-variable-names/>

<https://wikidocs.net/18507>

<https://programmers.co.kr/learn/courses/2>

[https://zetawiki.com/wiki/%ED%8C%8C%EC%9D%B4%EC%8D%AC\\_%EC%9E%90%EB%A3%8C%ED%98%95\\_%ED%99%95%EC%9D%B8\\_type\(\)](https://zetawiki.com/wiki/%ED%8C%8C%EC%9D%B4%EC%8D%AC_%EC%9E%90%EB%A3%8C%ED%98%95_%ED%99%95%EC%9D%B8_type())

<https://wikidocs.net/17>

<https://withcoding.com/65>



## Reference

<https://dojang.io/mod/page/view.php?id=2291>

<https://vision-ai.tistory.com/entry/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%EB%A6%AC%EC%8A%A4%ED%8A%B8-%EC%BD%9C%EB%A1%A0-%EC%8A%AC%EB%9D%BC%EC%9D%B4%EC%8B%B1-List-Slicing>

<https://velog.io/@kpl5672/python-2%EC%B0%A8%EC%9B%90%EB%A6%AC%EC%8A%A4%ED%8A%B8>

<https://wikidocs.net/24>

<http://hleecaster.com/python-class/>

<https://sjs0270.tistory.com/140>

<https://blog.hexabrain.net/284>