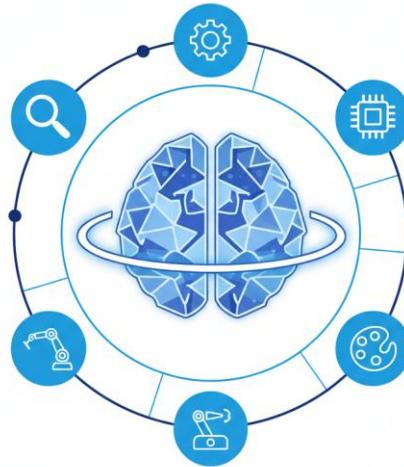


# 생성형 AI 및 파이썬 기초 문법

생성형 AI/ 제어문/ 자료구조



# 인공지능(Artificial Intelligence)

+ 인공지능에 대해 설명해줘

0



## 🤖 인공지능이란?

인공지능은 사람처럼 생각하고, 배우고, 판단하는 능력을 컴퓨터에 구현한 기술이에요.  
즉, 사람이 하던 “지능적인 작업”을 기계가 대신하도록 만드는 거죠.

예를 들면:

- 음성 인식 🎤 (말을 텍스트로 변환)
- 얼굴 인식 😊
- 자동 번역 🌎
- 추천 시스템 📽️ (넷플릭스, 유튜브)
- 자율주행 자동차 🚗



# 인공지능과 머신러닝 그리고 딥러닝

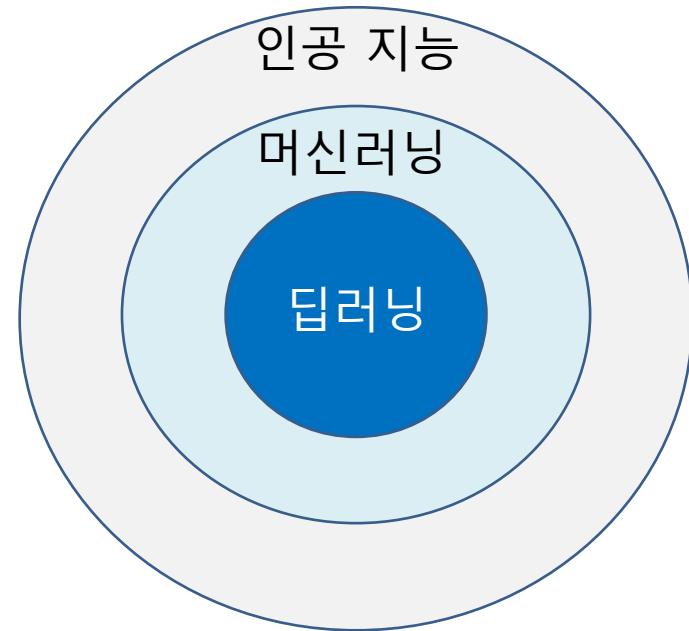
## \* 인공지능의 핵심 기술

### ◆ 머신러닝 (Machine Learning)

- 데이터를 이용해 스스로 학습
- “규칙을 사람이 직접 안 만들어도 됨”

### ◆ 딥러닝 (Deep Learning)

- 머신러닝의 한 종류
- 인간의 뇌를 본뜬 신경망 사용
- 이미지·음성·자연어 처리에 강함



# 머신러닝(Machine Learning)



## 기존 프로그래밍 vs 머신러닝

- 기존 방식  
규칙 + 데이터 = 결과
- 머신러닝  
데이터 + 정답 = 규칙(모델)



# 딥러닝(Deep Learning)

## E 대표적인 딥러닝 모델

- ◆ CNN (합성곱 신경망)
  - 이미지 처리 특화
  - 예: 얼굴 인식, 자율주행
- ◆ RNN / LSTM
  - 순서가 중요한 데이터
  - 예: 번역, 음성 인식
- ◆ Transformer
  - 현재 가장 강력 🔥
  - 챗GPT, 번역기, 요약 AI의 핵심

# 생성형 AI(Generative Artificial Intelligence)

## ● 생성형 AI

기존 데이터를 기반으로 학습하여 세상에 없던 새로운 결과물을 만들어 내는 인공지능의 한 분야.

인간의 뇌 신경망에서 영감을 얻은 인공신경망(Artificial Neural Networks)에 기반합니다. 이 신경망을 통해 데이터의 패턴과 구조를 학습하고, 이를 바탕으로 새로운 텍스트, 이미지, 음성 등을 만들어냅니다.

### ▪ 주요 생성 가능 콘텐츠

- 텍스트(Text): 기사 작성, 이메일 초안, 소설 창작
- 이미지(Image): 사실적인 사진, 예술 작품
- 음악(Audio): 특정 작가 스타일의 연주곡 작곡
- 영상(Video): 간단한 텍스트 설명 기반의 비디오 클립 제작

# LLM(Large Language Model)

- 대규모 언어 모델(LLM)

사람이 쓰는 언어를 이해하고 생성하도록 훈련된 인공지능 모델입니다.

OpenAI의 챗GPT, 구글의 제미나이, MS의 코파일럿이 대표적입니다.

- **원리:** 문장에서 다음에 올 단어를 통계적(확률)으로 예측합니다.  
예를 들어 " 나는 오늘... " 라고 하면 " 점심(45%), 날씨(30%), 밤(20%)" 가 나올 확률이 높다는 것을 방대한 데이터를 통해 학습한 것입니다.
- **특징:** 단순히 말을 흉내 내는 것이 아니라 문맥, 의도, 지식까지 파악해 사람처럼 대화하고 요약하며 코딩까지 수행합니다.
- **환각(할루시네이션-)** 현상 주의  
모든 LLM은 가끔 \*\*"모르는 것을 마치 아는 것처럼 당당하게 거짓말"\*\*을 하는 경우가 있습니다. 따라서 중요한 정보는 반드시 다시 한번 직접 확인(Fact-check)하는 과정이 필요합니다.

# 챗GPT(Chat Generative Pre-trained Transformer)

- 챗GPT(Chat Generative Pre-trained Transformer)
  - **정의:** OpenAI에서 **2022년 11월 30일**에 처음 공개한 인공지능 대규모 언어 모델(LLM) – 회원수(2025년 기준 8억명)
  - **핵심 기술:** 방대한 데이터를 사전 학습하여 문장으로 생성하는 대화형 기술
  - **특징:** 인간이 질문을 던지고 답변을 받는 형태를 사용하며 기존 AI와 달리 대화의 맥락(Context)을 정교하게 이해하여 자연스러운 대화가 가능함.
  - **의미**
    - Chatting – 채팅의 줄임말
    - Generative – 생성하는
    - Pre-trained – 사전 학습된
    - Transformer – 트랜스포머(딥러닝 아키텍쳐)

# 챗GPT(ChatGPT)

- 챗GPT 사용해 보기

- 홈페이지 접속 및 시작

- 공식 주소: chatgpt.com에 접속합니다.
    - 가입/로그인 버튼: 화면 우측 상단의 [Sign up](회원가입) 또는 [Log in](로그인) 버튼을 클릭합니다.

- 계정 생성 (회원가입)

- 간편 연동: 구글(Google), 마이크로소프트(Microsoft), 애플(Apple) 계정 중 하나를 선택하면 복잡한 절차 없이 가입됩니다.
    - 직접 이메일 가입: 이메일 주소를 입력하고 사용할 비밀번호를 설정 합니다.

# 챗GPT(ChatGPT)

## ● 새 채팅 사용하기



# 챗GPT(ChatGPT)

## ● 새 채팅 사용하기

구분	예방 방법	상세 내용
수도계량기	보온 처리	계량기함 내부에 헌 옷·수건·스티로폼 넣기, 틈새는 비닐로 밀봉
외부 배관	배관 보온	베란다·옥상·외부 노출 배관에 보온재 또는 헌 옷 감싸기
혹한기 대비	물 조금 틀어두기	영하 10°C 이하 시 연필심 굵기로 물 흐르게 유지
장기 외출	수도 잠금	계량기 잠그고 배관 내 물 완전히 빼기
보일러	동파방지 설정	외출 모드 또는 최소 온도 유지, 전원 완전 차단 금지

+ 이 내용을 표로 정리해 주세요

0 ↑

# 챗GPT(ChatGPT)

## ● 프롬프트 팁

### 1. 페르소나(Persona) 부여하기

AI에게 특정한 역할을 정해주는 것만으로도 답변의 수준이 달라집니다.

Bad: "마케팅 문구 써줘."

Good: "너는 10년 차 전문 카피라이터야. MZ세대를 타겟으로 한 신제품 운동화 광고 문구 5개를 작성해줘."

### 2. 구체적인 상황과 맥락(Context) 제공하기

AI는 정보를 많이 출수록 더 정확한 추론을 합니다.

필수 요소: 대상(Target), 목적(Goal), 톤앤매너(Tone), 제약 사항(Constraints).

예시: "회사 신입사원 교육용이야. 너무 전문적인 용어는 피하고, 친절하고 격려하는 말투로 작성해줘."



# 챗GPT(ChatGPT)

## ● 프롬프트 팁

### 3. 단계별로 생각하게 하기 (Chain of Thought)

복잡한 문제는 한 번에 답을 내라고 하기보다 과정을 보여달라고 하세요.

팁: 질문 끝에 \*\*"단계별로 차근차근 생각해봐"\*\*라는 문구만 추가해도 논리적 오류 (환각 현상)가 크게 줄어듭니다.

### 4. 출력 형식(Output Format) 지정하기

답변을 얻은 뒤 다시 정리할 필요가 없도록 처음부터 형식을 지정하세요.

예시: "결과는 표(Table) 형태로 정리해주고, 핵심 요약은 세 줄만 적어줘."

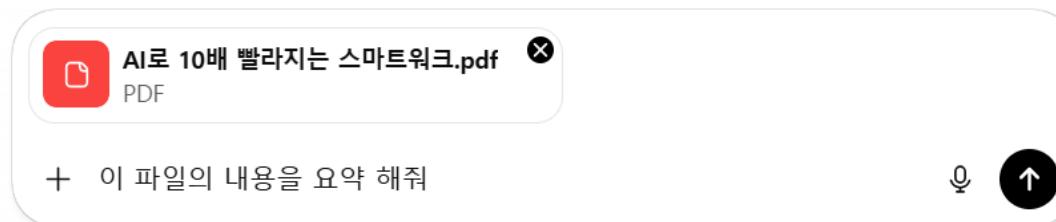
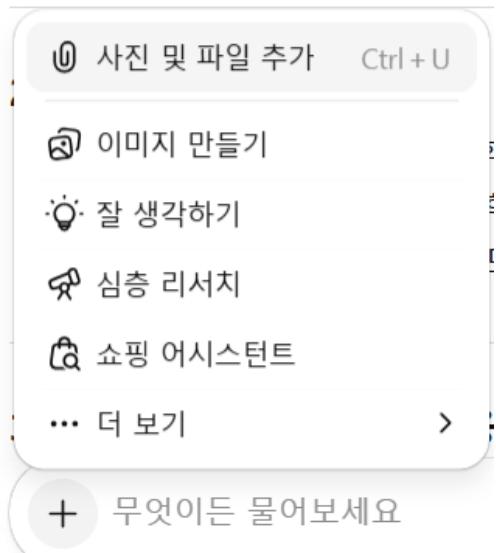
### 5. 예시(Few-shot) 제공하기

내가 원하는 스타일의 샘플을 1~2개 보여주면 그 패턴을 완벽하게 복사합니다.

구조: "질문: [예시 질문] -> 답변: [예시 답변]. 자, 이제 다음 질문에 답해줘: [진짜 질문]"

# 챗GPT(ChatGPT)

## ● 사진 및 파일 추가



# 챗GPT(ChatGPT)

## ● 사진 및 파일 추가

### 1. 사진 추가 후 지브리 스타일과 디즈니 스타일로 만들기



지브리 스타일



디즈니 스타일

### 2. 카카오톡 프로필 사진 바꾸기



# 제미나이(Gemini)

## ● 제미나이(Gemini) 사용해 보기

제미나이에 대해 설명해줘

+ ☰ 도구

Pro ▾



안녕하세요! 저는 구글(Google)에서 개발한 인공지능, \*\*제미나이(Gemini)\*\*입니다.

저는 여러분의 생각 파트너(Thought Partner)로서, 궁금한 점을 해결해 드리고 창의적인 작업을 돋기 위해 만들어졌습니다. 저에 대해 조금 더 자세히 설명해 드릴게요.

### 1. 저는 '멀티모달(Multimodal)' AI입니다.

이것이 저의 가장 큰 특징 중 하나입니다. 단순히 텍스트만 이해하는 것이 아니라, 다음과 같은 다양한 형태의 정보를 받아들이고 처리할 수 있습니다.

- **텍스트 & 코드:** 글을 쓰고, 번역하고, 복잡한 프로그래밍 코드를 짜거나 수정할 수 있습니다.
- **이미지 & 비디오:** 사진이나 영상을 보고 그 내용을 이해하거나, 새로운 이미지와 비디오를 생성할 수 있습니다.
- **오디오:** 음성을 이해하고 대화를 나눌 수 있습니다.

# 제미나이(Gemini)

## ● 제미나이(Gemini) 사용해 보기

로그인 방법

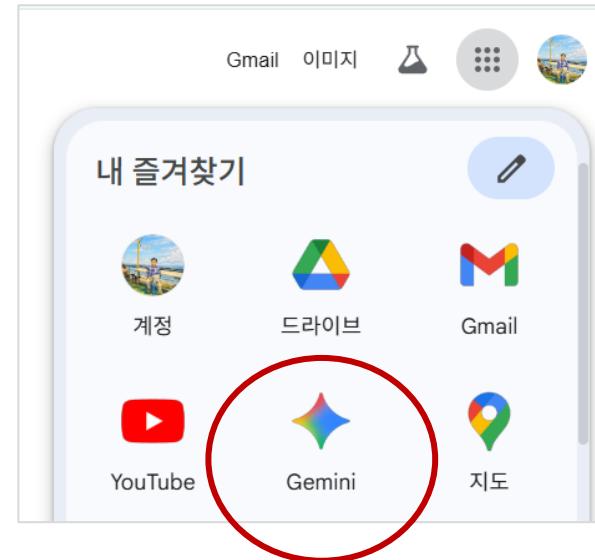
이미 구글 계정이 있는 경우

화면 우측 상단 [로그인] 클릭

구글 계정 선택

비밀번호 입력

로그인 완료 → 바로 Gemini 사용 가능



# 제미나이(Gemini)

## ● 구글 워크 스페이스와 연동

### ❄️ 겨울철 동파 방지 요령 요약

구분	주요 조치 방법	상세 내용 및 팁
수도 계량기	보온재 채우기	내부를 헌 옷, 수건, 에어캡으로 채우고 틈새 밀봉
수도꼭지	물 가늘게 흘리기	영하 10도 이하일 때 온수 쪽으로 가늘게 흘리기
노출 배관	단열 조치	외부 배관을 보온재로 감싸고 젖지 않게 관리
보일러	외출 모드 유지	전원을 끄지 말고 외출 모드 또는 낮은 온도(10~15도) 설정
마당 수도	물 빼기	부동전의 앞 꼭지는 열고 땅속 밸브만 잠가 배관 비우기

feld Sheets로 내보내기



이 내용을 표로 정리해 주세요

+ 도구

빠른 모드 ▾



# 제미나이(Gemini)

## ● 구글 스프레드 시트 사용

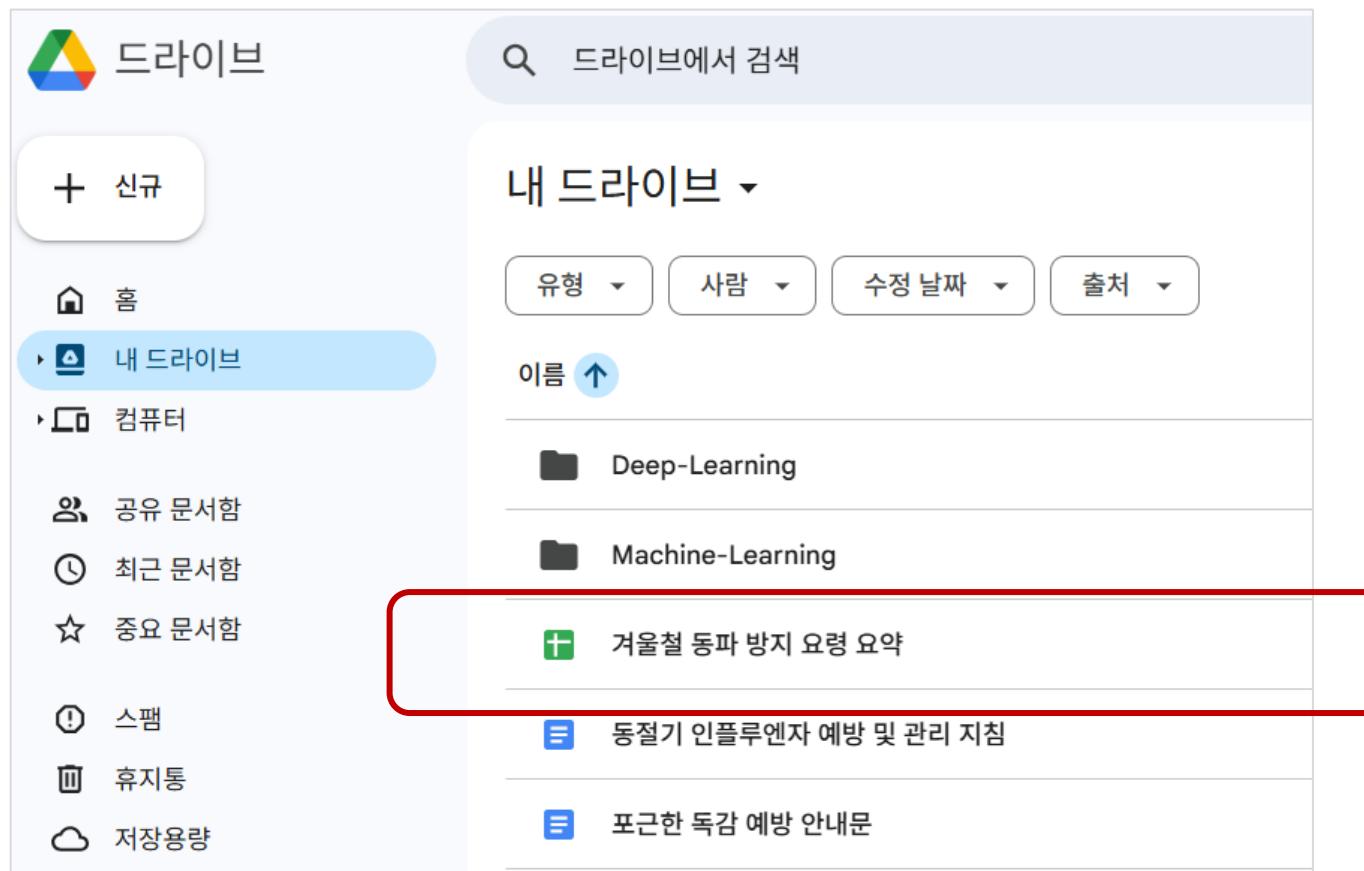
The screenshot shows a Google Sheets interface with the following details:

- Title:** 겨울철 동파 방지 요령 요약
- Toolbar:** 파일, 수정, 보기, 삽입, 서식, 데이터, 도구, 확장 프로그램, 도움말
- Search/Filter:** Q, ↺, ↻, ⌂, 100%, ⌄, ⌅, ⌆, ⌇, ⌈, ⌉, 기본값 ..., -
- Cell Address:** C11
- Table Data:**

	A	B	C	D	E	F
1	구분	주요 조치 방법	상세 내용 및 팁			
2	수도 계량기	보온재 채우기	내부를 헌 옷, 수건, 에어캡으로 채우고 틈새 밀봉			
3	수도꼭지	물 가늘게 흘리기	영하 10도 이하일 때 온수 쪽으로 가늘게 흘리기			
4	노출 배관	단열 조치	외부 배관을 보온재로 감싸고 젖지 않게 관리			
5	보일러	외출 모드 유지	전원을 끄지 말고 외출 모드 또는 낮은 온도(10~15도) 설정			
6	마당 수도	물 빼기	부동전의 앞 꼭지는 열고 땅속 밸브만 잠가 배관 비우기			

# 제미나이(Gemini)

## ● 구글 드라이브에 연동



# 제미나이(Gemini)

- 이미지 생성하기 – 나노 바나나

도구 > 나노바나나



유영하는 우주 비행사의 이미지를 만들어 줘



이미지



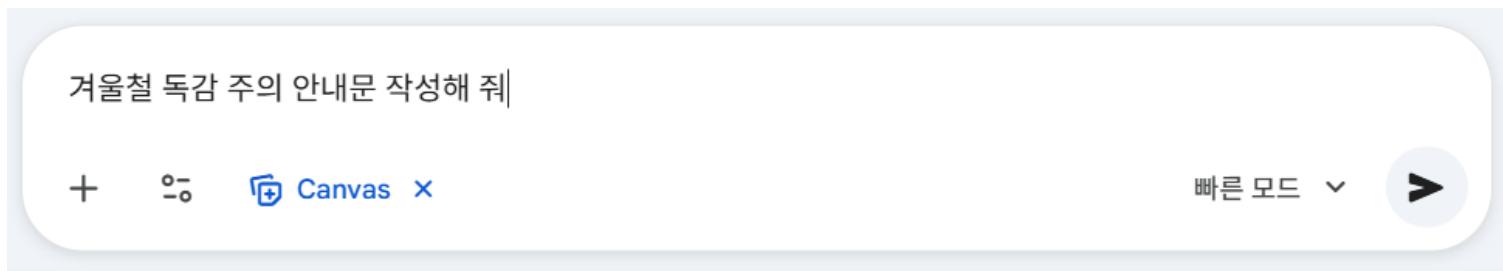
빠른 모드



# 제미나이(Gemini)

## ● 캔버스 – canvas

아이디어를 문서, 인포그래픽, 웹페이지 등으로 구현하는 공간입니다.



↑ → 길이 변경 – 짧게, 길게

↙ → 어조 변경 – 편하게, 정중하게

☰ → 수정제안



# 제미나이(Gemini)

## ● 캔버스 – canvas

- 공유 – Docs로 내보내기

The screenshot shows a Google Docs document with the following details:

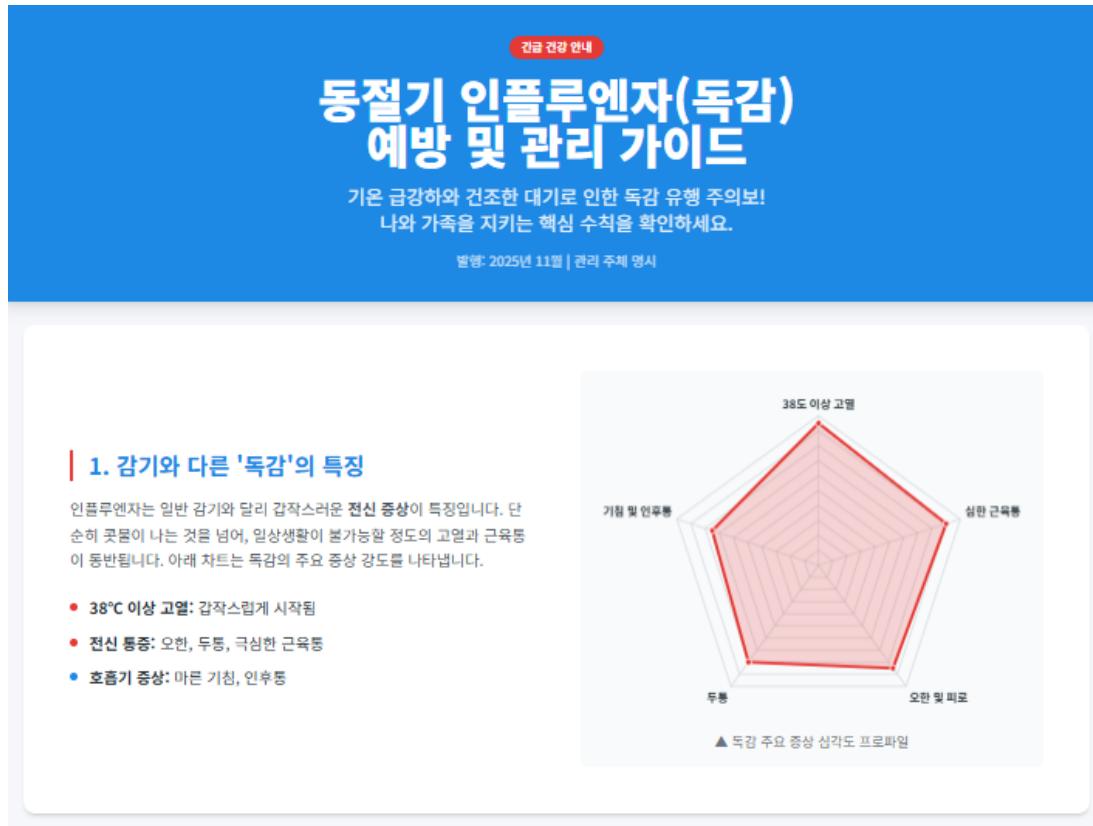
- Title:** 동절기 인플루엔자(독감) 예방 및 관리 수칙 안내
- Content:** 최근 동절기 기온 하강 및 대기 건조로 인해 인플루엔자 바이러스의 확산 위험이 증대되고 있습니다. 인플루엔자는 전염성이 매우 높으며, 고위험군의 경우 심각한 합병증을 유발할 수 있으므로 아래의 예방 및 관리 수칙을 엄격히 준수하여 주시기 바랍니다.
- Section 1:** 1. 주요 증상 및 임상적 특징
- Text:** 인플루엔자는 일반적인 상기도 감염(감기)과 달리 급격한 전신 증상을 동반하는 특징이 있습니다.
- List:**
  - 전신 증상: 38°C 이상의 고열, 오한, 두통 및 극심한 근육통(통증)과 무기력증
  - 호흡기 증상: 인후통, 지속적인 기침 및 비충혈(콧물) 등

- 파일 – 다운로드 – Microsoft Word(.docx), PDF 문서(.pdf)

# 제미나이(Gemini)

## ● 캔버스 – canvas

- 만들기 - 인포그래픽



# AI 시대의 필수 코딩 언어

## ● 파이썬(Python)

+ 파이썬에 대해 알려 줘



### 파이썬이 뭐야?

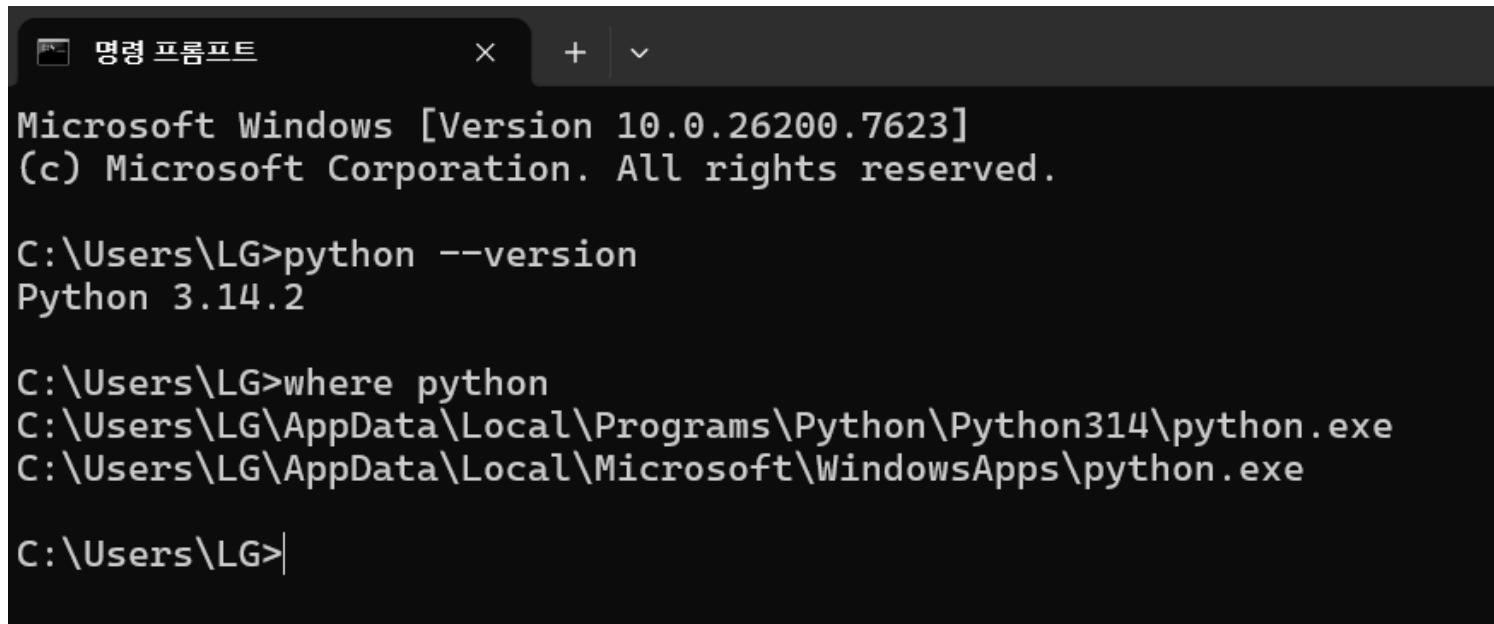
- 1991년 구도 반 로섬(Guido van Rossum)이 만든 언어
- 문법이 사람 말처럼 직관적
- 인터프리터 언어 → 코드 작성 후 바로 실행 가능

### 파이썬으로 뭘 할 수 있나?

- 데이터 분석 & 시각화
- 인공지능 / 머신러닝
- 웹 서버 & API 개발
- 자동화 스크립트
- 간단한 게임
- 파일 처리, 크롤링

# AI 시대의 필수 코딩 언어

## ● 파이썬 설치 유무 확인

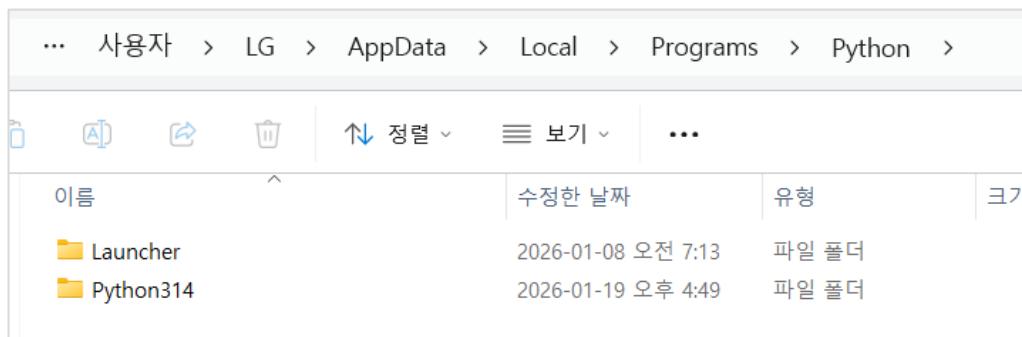


```
명령 프롬프트
Microsoft Windows [Version 10.0.26200.7623]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LG>python --version
Python 3.14.2

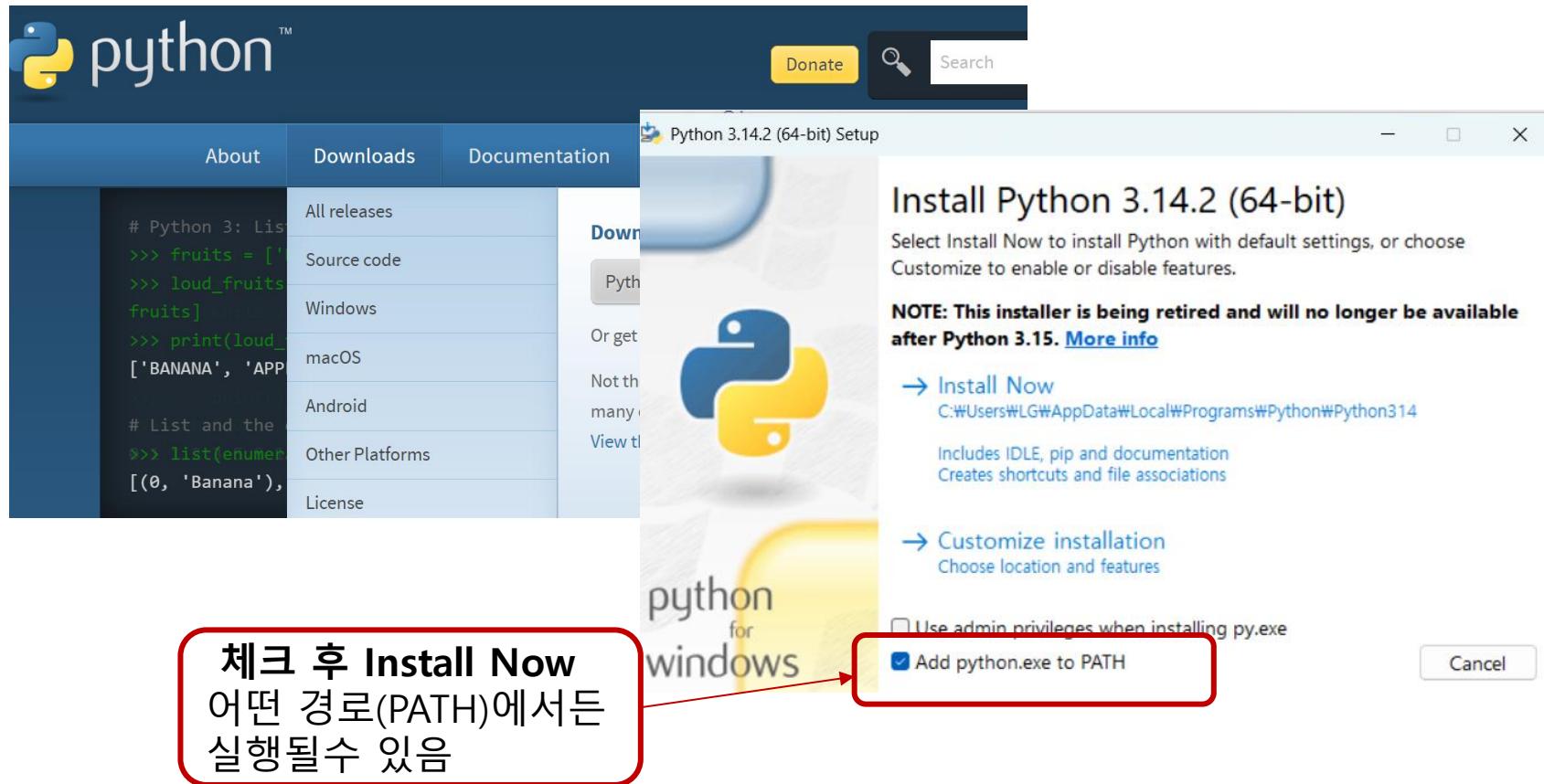
C:\Users\LG>where python
C:\Users\LG\AppData\Local\Programs\Python\Python314\python.exe
C:\Users\LG\AppData\Local\Microsoft\WindowsApps\python.exe

C:\Users\LG>
```



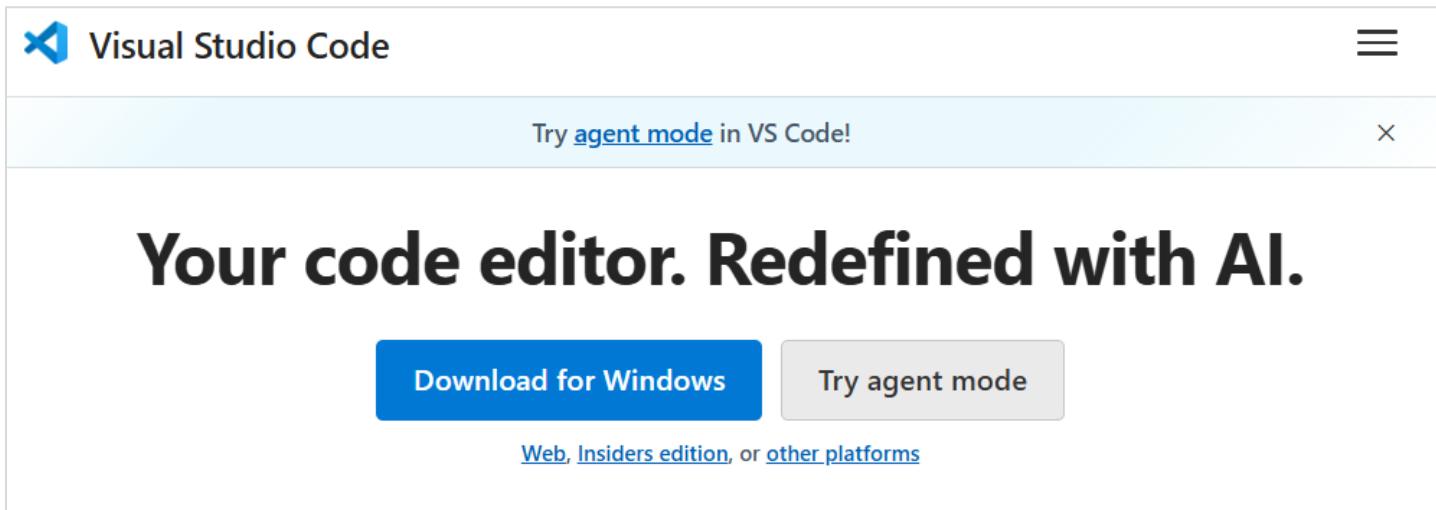
# AI 시대의 필수 코딩 언어

- 파이썬 - [www.python.org](http://www.python.org) > Downloads



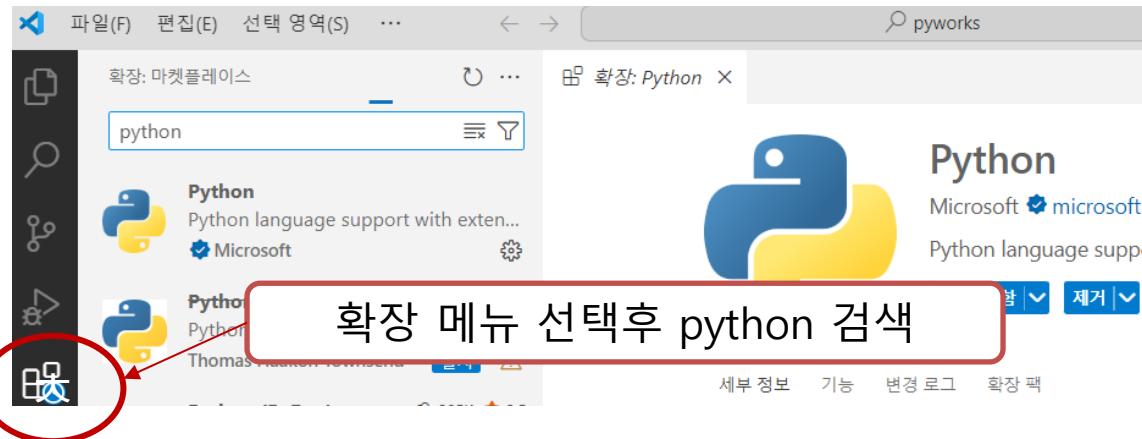
# AI 시대의 필수 코딩 IDE

## ◆ VS code 설치

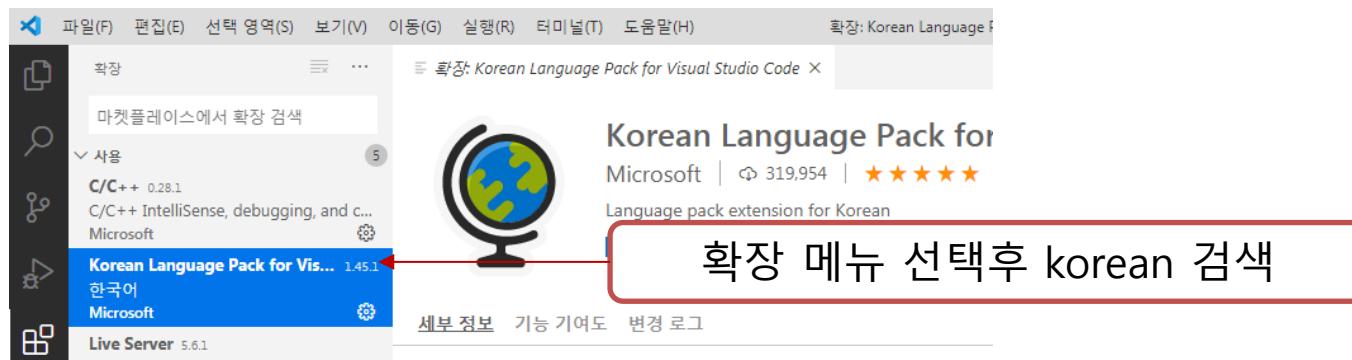


# AI 시대의 필수 코딩 IDE

## ◆ Python 언어 지원 확장 팩 설치하기

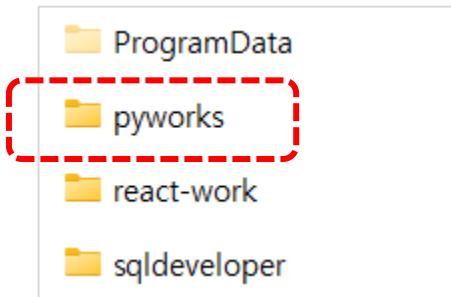


## ◆ 한국어 팩 설치

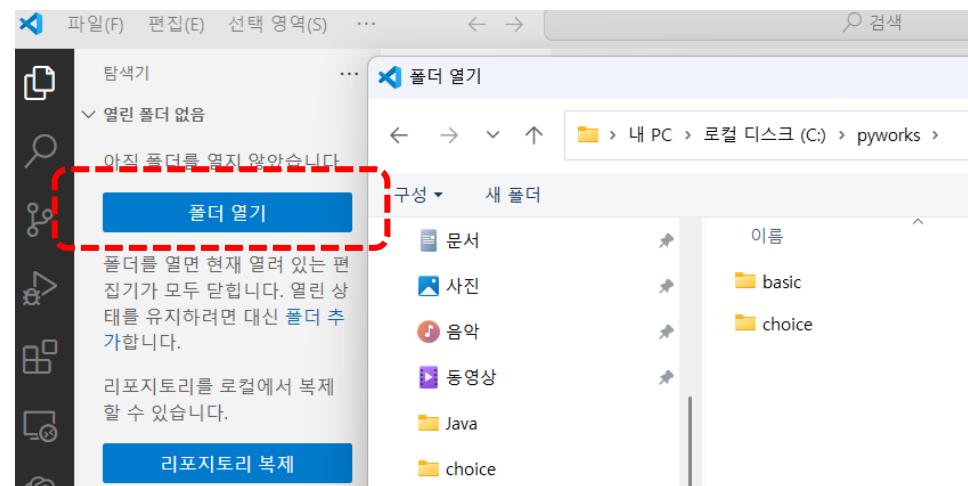


# AI 시대의 필수 코딩 IDE

## ◆ VS code – 작업 폴더 설정



① 작업영역 폴더 만들기



② 작업영역 폴더 설정

# AI 시대의 필수 코딩 IDE

## ◆ VS code – 관리 도구

The image shows the VS Code interface. On the left, the sidebar has a gear icon highlighted with a red circle, and a callout box labeled "관리 > 설정" (Manage > Settings). The main area shows the "General" tab of the settings editor. A red circle highlights the "afterDelay" input field under "Files: Auto Save", with a callout box labeled "자동 저장" (Auto Save). Another red circle highlights the "17" input field under "Editor: Font Size", with a callout box labeled "글꼴 크기" (Font Size).

사용자 작업 영역

일반적으로 사용되는 ...

- > 텍스트 편집기
- > 워크벤치
- > 창
- > 기능
- > 애플리케이션
- > 보안
- > 확장

일반적으로 사용되는 설정

Files: Auto Save

저장되지 않은 변경 사항이 있는 편집기의 자동 저장을 제어합니다.

afterDelay

자동 저장

Editor: Font Size

글꼴 크기(픽셀)를 제어합니다.

17

글꼴 크기

Editor: Font Family

글꼴 패밀리를 제어합니다.

Consolas, 'Courier New', monospace

관리 > 설정

# 출력 – print()

## ▪ 출력 – print() 함수

```
"""
주석(Comment) : 코드 내용을 설명하기 위한 메모
- 한 줄 주석: # 을 붙임
- 여러줄 주석: 쌍따옴표(or 흘따옴표) 3개로 열고 닫기
"""

# 출력 : print() 함수 사용
# 문자 - 흘따옴표 또는 쌍따옴표로 감싼다.
print('A')
print("coffee")
print("아이스 아메리카노")
#print(이름)

# 숫자 : 정수, 실수
print(17)
print(3.3)

# 산술 계산
print(1 + 2) #3
print(50 - (5 * 6)) #20
print((50 - (5 * 6))/4) #5.0
```

# 변수[Variable]

## ■ 변수란?

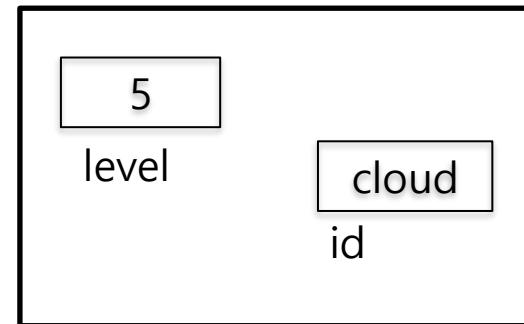
- 프로그램에서 사용되는 자료를 저장하기 위한 메모리 공간(영역)
- 프로그램 실행 중에 값 변경 가능, variable 이라 함

## ■ 변수의 선언 및 초기화

- 변수 선언은 어떤 타입의 데이터를 저장할 것인지 그리고 변수 이름은 무엇인지를 결정한다. (자료형)은 생략함.

(자료형) 변수이름 = 초기값;

예) level = 5 , id = "cloud"



# 변수(Variable)

## ● 변수 사용 예제

```
# 상수 - 변하지 않는 수
print(4 + 5)
print(4 * 5)

# 변수 선언과 초기화
n1 = 4 # 정수형 변수 n1에 4를 저장
n2 = 5

print(n1 + n2)
print(n1 - n2)
print(n1 * n2)
print(n1 / n2)

level = 4
print("레벨 :", level)

userid = "cloud"
print("ID :", userid)
```

# 변수[Variable]

## ● 변수명(이름) 작성시 주의

- 변수 이름은 영문, 한글, 특수문자, 숫자의 결합으로 만든다.
- 변수 이름은 숫자로 시작할 수 없고, 공백이 있으면 안됨.
- 변수 이름은 대, 소문자를 구분함
- 예약어는 사용할 수 없음.(if, for, while 등)

```
# 변수명(이름) -> 스네이크 표기법
name_of_fruit = "사과"
print("그 과일의 이름은", name_of_fruit, "입니다.")

rate_of_birth = 0.81
print("대한민국의 2025년 출산율은", rate_of_birth, "입니다")

# 변수명 오류
#2n = 10 # 숫자로 시작 불가
#user id = "sky123" # 공백 불가
#for = 3 # 예약어 사용 불가
```

# 자료형(Data Type)

## ■ 자료형이란?

- 사용할 데이터의 종류에 따라 메모리 공간을 적절하게 설정해 주는 것
- 파이썬에서는 표기하지 않음 (예, num=10, name="홍길동")

분류	자료형	설명	예
정수	int	소수점이 없는 수	-2, -1, 0, 1, 2
실수	float	소수점(.)이 있는 수, 부동소수점수라고도 불린다.	-3.5, 0.0, 1.25
문자열	str	알파벳과 다른 문자로 이루어진 문장	"a", 'hello', "비"
논리형	bool	참과 거짓을 표현	True, False(2가지 값만 있음)

# 자료형(Data Type)

- 자료형(Data Type) 예제

```
# 자료형 - type() 함수
# 숫자형
n1 = 4
rate_of_birth = 0.81

print(type(n1)) #int
print(type(rate_of_birth)) #float

# 논리형
b1 = False

print(type(b1)) #bool
print(4 > 5) #False
print(5 == 5) #True

# 문자형
name_of_fruit = "사과"

print(type(name_of_fruit)) #str
```

# 연산자(Operator)

## ▪ 대입 연산자

- 오른쪽의 값을 왼쪽의 변수에 대입
- '=' 연산자를 사용.

age = 20

user\_id = "abc123"

- 연산의 순서

오른쪽 > 왼쪽

산술 연산 > 대입 연산

total = num1 + num2



```
# '=' 은 대입 연산자
user_id = "smile"
password = "k1234"

print("user_id =", user_id)
print("password =", password)

print("user_id = " + user_id)
print("password = " + password)
```

# 연산자(Operator)

## ▪ 대입 연산자

### ▪ 구조 분해 할당

```
# 구조 분해 할당  
a, b = 10, 20  
print("a:", a)  
print("b:", b)
```

```
# a, b 변수의 값을 서로 교환  
a, b = b, a  
print("a:", a)  
print("b:", b)
```

```
a: 10  
b: 20  
a: 20  
b: 10
```

# 연산자(Operator)

## ■ 산술 연산자

연산자	연산 작업	설명
+	$n1 + n2$	더하기
-	$n1 - n2$	빼기
*	$n1 * n2$	곱하기
/	$n1 / n2$	나누기
//	$n1 // n2$	몫
%	$n1 \% n2$	나머지
**	$n1 ** n2$	거듭제곱

# 연산자(Operator)

## ■ 산술 연산자 예제

```
n1 = 10
n2 = 4

print("n1 + n2 =", n1 + n2)
print("n1 - n2 =", n1 - n2)
print("n1 * n2 =", n1 * n2)
print("n1 / n2 =", n1 / n2)      # 나누기
print("n1 // n2 =", n1 // n2)    # 몫
print("n1 % n2 =", n1 % n2)      # 나머지
print("n1 ** n2 =", n1 ** n2)    # 거듭제곱
```

# 연산자(Operator)

## ■ 비교 연산자

연산자	의미	예	결과
<	보다 작다	$9 < 10$	True
>	보다 크다	$9 > 10$	False
$\leq$	작거나 같다	$9 \leq 10$	True
$\geq$	크거나 같다	$9 \geq 10$	False
$=$	같다	$9 == 10$	False
$\neq$	같지 않다	$9 \neq 10$	True
is	같다(객체)	a is b	False
is not	같지 않다(객체)	a is not b	True

# 연산자(Operator)

## ■ 비교 연산자

```
x = 10  
y = -10  
  
print(x > 0)  
print(y > 0)  
print()  
  
print(x > y)  
print(x < y)  
print()  
  
print(x == 10)  
print(x == y)  
print(x != y)  
print(x is y)  
print(x is not y)  
print()
```

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

# 연산자(Operator)

## ■ 논리 연산자

연산자	규칙
x and y	x, y 가 모두 참이면 참, 나머지는 거짓
x or y	x, y 중 둘 중 하나가 참이면 참
not x	x가 참이면 거짓, 거짓이면 참

```
# and - 2개의 조건이 모두 참일때 참이다.  
# or - 2개의 조건중 1개만 참이어도 참이다.  
print(x > 0 and y > 0) #False  
print(x > 0 or y > 0) #True  
print(not (y > 0)) #True
```

# 입력 – input()

## ➤ input() – 입력 함수

```
"""
print("문자 입력: ")
ch = input()
print(ch)
```

```
"""
ch = input("문자 입력: ")
print(ch)
```

문자 입력:  
happy  
happy

문자 입력: apple  
apple

# 입력 – input()

## ➤ input() – 입력 함수

```
# 정수에 1 더하기  
num = input("정수 입력: ")  
num = int(num) #문자를 정수로 변환  
print(type(num))  
  
print(num + 1)  
  
# 몸무게의 2배 계산  
weight = input("몸무게 입력: ")  
weight = float(weight) #문자를 실수로 변환  
print(type(weight))
```

# 입력 – input()

## ➤ input() – 입력 함수

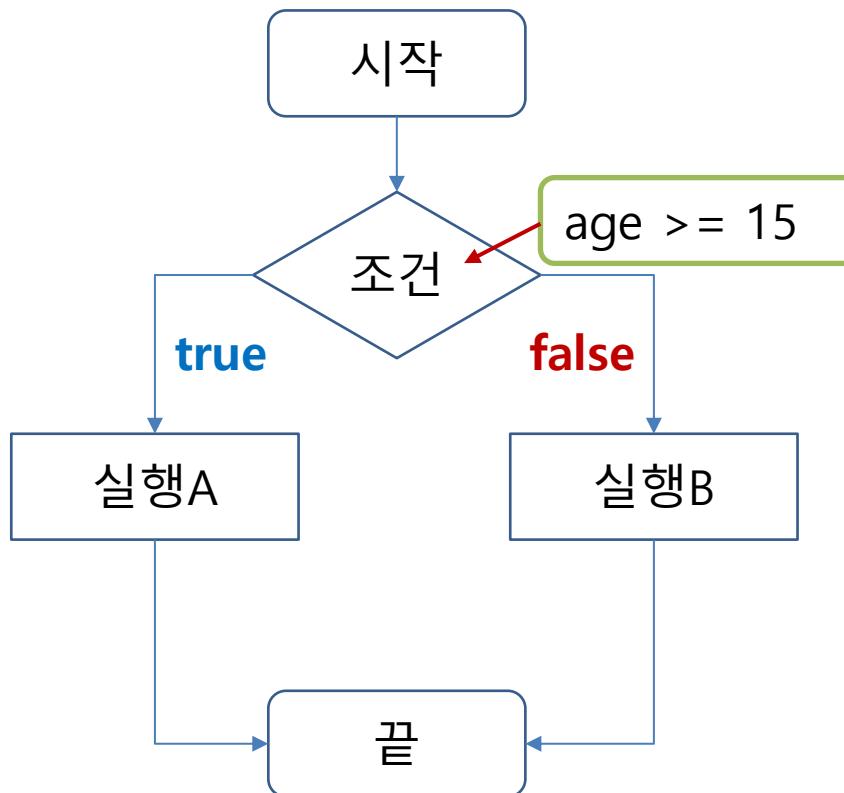
```
print("이름을 입력해 주세요:")
name = input()
print(name + '님 반갑습니다.')

age = input('나이를 입력해 주세요: ')
age = int(age)
print("당신의 나이는 " + str(age) + "세 이군요!")
```

# 조건문(Choice Structure)

## 조건문

- 특정한 조건에 의해서 프로그램 진행이 분기되는 구문



# 조건문(Choice Structure)

## ▪ if 문

**if** 조건식:

수행문

4칸 들여쓰기  
인덴트(indent)

# 조건식이 참이면 수행문 실행

## ▪ if ~ else 구문

• **if** 조건식:

•      수행문 1

• **else** :

•      수행문 2

# 조건식이 참이면 수행문1 실행,  
아니면 수행문2 실행

# 조건문(Choice Structure)

- if 문

```
age = 17
if age < 20:
    print("미성년자입니다.")

print(f"나이는 {age}세입니다.")
```

- if ~ else 구문

```
age = 17
if age < 20:
    print("미성년자입니다.")
else:
    print("성인입니다.")

print(f"나이는 {age}세입니다.")
```

# 조건문(Choice Structure)

- **if ~ elif ~ else** 구문

- **if** 조건1:
  - 수행문1
- **elif** 조건2:
  - 수행문2
- **else :**
  - 실행문 3

# 조건1이 참이면 수행문1 실행, 조건2가 참이면 수행  
문2 실행, 조건1,2가 모두 거짓이면 수행문3 실행

# 조건문(Choice Structure)

- 미니 프로그램 - 신호등

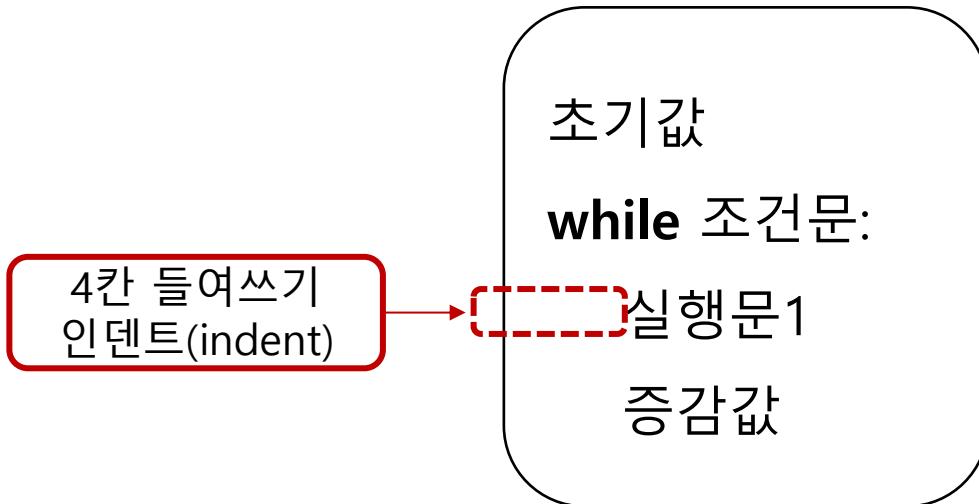


```
color = input("신호등 색상 입력(빨강/노랑/초록): ")

if color == "빨강":
    print("정지")
elif color == "노랑":
    print("주의")
elif color == "초록":
    print("진행")
else:
    print("잘못된 입력입니다")
```

# 반복문 – while문

- while문



# 반복문 – while문

## ● while문 예제

```
a = 1  
print(a)  
  
a += 1  
print(a)  
  
a += 1  
print(a)
```

```
# 1부터 5까지 출력하는 while 반복문  
print("1부터 5까지 출력")  
n = 1  
while n <= 5:  
    print(n)  
    n += 1  
  
print("5부터 1까지 출력")  
n = 5  
while n >= 1:  
    print(n)  
    n -= 1
```

# 반복문 – while문

## ● while문 예제

```
# 1부터 10까지의 합을 구하는 프로그램  
i = 1  
total = 0  
while i <= 10:  
    total += i  
    print("i=", i, ", total=", total)  
    i += 1  
  
print(f"1부터 10까지의 합은 {total}입니다.")
```

```
i= 1 , total= 1  
i= 2 , total= 3  
i= 3 , total= 6  
i= 4 , total= 10  
i= 5 , total= 15  
i= 6 , total= 21  
i= 7 , total= 28  
i= 8 , total= 36  
i= 9 , total= 45  
i= 10 , total= 55  
1부터 10까지의 합은_55입니다.
```

# 반복문 – while문

## ● 반복 조건문

반복문에서 break 문을 만나면 더 이상 반복을 수행하지 않고,  
반복문을 빠져 나옴

```
while True:
```

```
    수행문
```

```
    if 조건 :
```

```
        break
```

# 반복문 – while문

## ● 반복 조건문

```
# 1부터 5까지 출력하는 프로그램 (break 사용)
n = 1
while True:
    if n > 5:
        break
    print(n)
    n += 1
print("반복문 종료")
```

# 반복문 – while문

## ● 챗봇 프로그램

```
from datetime import datetime

while True:
    user_input = input("챗봇에게 질문하세요 (종료하려면 'exit' 입력): ")
    if user_input == "exit":
        print("챗봇을 종료합니다.")
        break
    elif "안녕" in user_input:
        print("챗봇: 안녕하세요! 무엇을 도와드릴까요?")
    elif "이름" in user_input:
        print("챗봇: 제 이름은 챗봇입니다.")
    elif "시간" in user_input:
        now = datetime.now()
        print(f"챗봇: 현재 시간은 {now.hour}시 {now.minute}분입니다.")
    else:
        print("챗봇: 죄송합니다, 잘 모르겠어요.")
```

# 반복문 – for문

## ▪ for문

순서열의 각 원소를 처음부터 순회하면서 반복변수에 담아 낸다.

순서열은 **리스트**, **튜플**, **문자열** 등을 사용

for - in range() 와 for - in 문을 사용함.

**for 반복변수 in range(시작값, 종료값, 증감값):**

**실행문**

- for 반복변수 in 순서열:
- 실행문

# 반복문 – for문

- range() 함수 사용하기

range(^|작값, 종료값, 증감값):

- ✓ 시작값을 생략하면 **0**부터 시작하고, 종료값은 (**종료값-1**) 이다.

```
print(range(5)) # range(0, 5)
print(list(range(0, 5))) # [0, 1, 2, 3, 4]
print(list(range(1, 5, 1))) # [1, 2, 3, 4]
```

```
# 1부터 10까지 출력
for n in range(1, 11):
    print(n)
```

# 반복문 – for문

## ▪ range() 함수 사용하기

```
# 1부터 10까지의 합 구하기
total = 0
for n in range(1, 11):
    total += n
print(f"1부터 10까지의 합: {total}")

# 1부터 10까지의 짝수 출력
for n in range(2, 11, 2):
    print(n, end=" ")
print()

# 1부터 10까지의 홀수 출력
for n in range(1, 11):
    if n % 2 == 1:
        print(n, end=" ")
```

# 반복문 – for문

- 단을 입력받아 구구단 출력하기

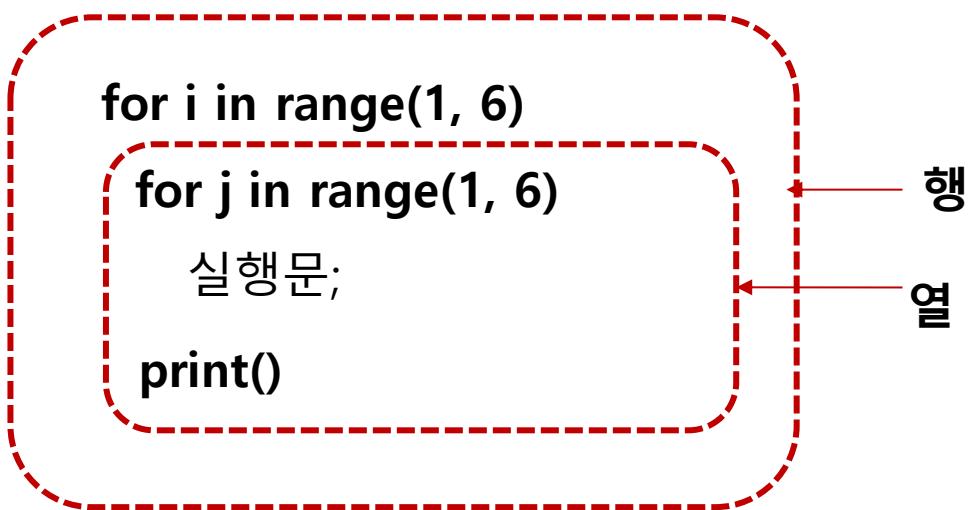
```
단을 입력하세요: 7  
7 x 1 = 7  
7 x 2 = 14  
7 x 3 = 21  
7 x 4 = 28  
7 x 5 = 35  
7 x 6 = 42  
7 x 7 = 49  
7 x 8 = 56  
7 x 9 = 63
```

```
dan = int(input("단을 입력하세요: "))  
  
for n in range(1, 10):  
    result = dan * n  
    print(f"{dan} x {n} = {result}")
```

# 반복문 – for문

## ❖ 이중 for문

- 행, 열 구현하기



5행 5열

	열1	열2	열3	열4	열5
행1					
행2					
행3					
행4					
행5					

# 반복문 – for문

## ❖ 중첩 for문

```
가가가가  
가가가가  
가가가가
```

```
for i in range(3): # 바깥쪽 for문
    for j in range(4): # 안쪽 for문
        print("가", end="")
    print() # 줄바꿈
```

# 반복문 – for문

## ❖ 구구단 전체 출력 프로그램

```
# 구구단 전체 출력하기
for dan in range(2, 10): # 2단부터 9단까지
    for n in range(1, 10): # 각 단의 1부터 9까지
        result = dan * n
        print(f"{dan} x {n} = {result}")
    print() # 단이 바뀔 때마다 줄바꿈
```

2 x 1 = 2	4 x 1 = 4	6 x 1 = 6
2 x 2 = 4	4 x 2 = 8	6 x 2 = 12
2 x 3 = 6	4 x 3 = 12	6 x 3 = 18
2 x 4 = 8	4 x 4 = 16	6 x 4 = 24
2 x 5 = 10	4 x 5 = 20	6 x 5 = 30
2 x 6 = 12	4 x 6 = 24	6 x 6 = 36
2 x 7 = 14	4 x 7 = 28	6 x 7 = 42
2 x 8 = 16	4 x 8 = 32	6 x 8 = 48
2 x 9 = 18	4 x 9 = 36	6 x 9 = 54

3 x 1 = 3	5 x 1 = 5	7 x 1 = 7
3 x 2 = 6	5 x 2 = 10	7 x 2 = 14
3 x 3 = 9	5 x 3 = 15	7 x 3 = 21
3 x 4 = 12	5 x 4 = 20	7 x 4 = 28
3 x 5 = 15	5 x 5 = 25	7 x 5 = 35
3 x 6 = 18	5 x 6 = 30	7 x 6 = 42
3 x 7 = 21	5 x 7 = 35	7 x 7 = 49
3 x 8 = 24	5 x 8 = 40	7 x 8 = 56
3 x 9 = 27	5 x 9 = 45	7 x 9 = 63

# 리스트(List)

## ▪ 챗GPT 프롬프트 입력

+ 파이썬 리스트 자료구조에 대해 설명해줘

0



### 1. 리스트란?

리스트는 여러 개의 값을 순서대로 저장하는 자료구조입니다.

대괄호 [] 를 사용해서 만들어요.

python

코드 복사

```
numbers = [1, 2, 3, 4]
fruits = ["apple", "banana", "cherry"]
mixed = [1, "hello", 3.14, True]
```

👉 서로 다른 자료형을 섞어서 저장할 수 있다는 게 특징이에요.



# 리스트(List)

## ▪ 정수형 리스트

```
# 리스트 생성  
my_list = [1, 2, 3, 4, 5]  
  
# 리스트 출력  
print("리스트:", my_list)  
print(type(my_list))  
  
# 리스트의 길이 출력  
print("리스트의 길이:", len(my_list))  
  
# 리스트의 특정 요소 접근  
print("리스트의 첫 번째 요소:", my_list[0])  
print("리스트의 세 번째 요소:", my_list[2])  
print("리스트의 마지막 요소:", my_list[-1])  
  
# 리스트의 슬라이싱(':' 연산자 사용-부분 리스트 추출)  
print("리스트의 첫 세 요소:", my_list[0:3])  
print("리스트의 마지막 두 요소:", my_list[-2:])  
print("리스트의 모든 요소:", my_list[:])
```

# 리스트(List)

## ▪ 정수형 리스트

```
# 리스트 요소 변경  
my_list[1] = 20  
print("요소 변경 후 리스트:", my_list)  
  
# 리스트에서 요소 제거 - remove() 메서드 사용  
del my_list[2]  
print("요소 제거 후 리스트:", my_list)  
  
# 리스트에 요소 추가 - append() 메서드 사용  
# my_list.append(6)  
# print("요소 추가 후 리스트:", my_list)
```

# 리스트(List)

- **for** 변수 **in** 리스트:

```
my_list = [10, 20, 30, 40, 50]
print(20 in my_list) # True
print(25 in my_list) # False

print("for문을 사용한 리스트 출력:")

for item in my_list:
    print(item, end=' ')
print()

# 40보다 큰 요소만 출력
print("40보다 큰 요소 출력:")

for item in my_list:
    if item > 40:
        print(item, end=' ')
print()
```

# 리스트(List)

- **for** 변수 **in** 리스트:

```
my_list = [10, 20, 30, 40, 50]
print(20 in my_list) # True
print(25 in my_list) # False

print("for문을 사용한 리스트 출력:")

for item in my_list:
    print(item, end=' ')
print()

# 40보다 큰 요소만 출력
print("40보다 큰 요소 출력:")

for item in my_list:
    if item > 40:
        print(item, end=' ')
print()
```

# 리스트(List)

## ▪ if 변수 in [list]

리스트 내부에 값이 있으면 True, 없으면 False

```
# 음식 분류하기 - 한식, 일식, 중식
foods = ["비빔밥", "짜장면", "초밥", "김치찌게"]

for food in foods:
    if food in ["짜장면", "짬뽕"]:
        print(f'{food}는(은) 중식입니다.')
    elif food in ["초밥", "우동"]:
        print(f'{food}는(은) 일식입니다.')
    else:
        print(f'{food}는(은) 한식입니다.)
```

비빔밥는(은) 한식입니다.  
짜장면는(은) 중식입니다.  
초밥는(은) 일식입니다.  
김치찌게는(은) 한식입니다.

# 리스트(List)

## ■ 리스트의 주요 메서드(함수)

함수	기능	사용 예
append()	요소 추가	a = [1, 2, 3] a.append(4) <b>a = [1, 2, 3, 4]</b>
insert()	특정 위치에 추가	a = [2, 4, 5] a.insert(1,3) #1번 위치에 3 삽입 <b>a = [2, 3, 4, 5]</b>
pop()	요소 삭제	a = [1, 2, 3, 4, 5] a.pop() # 마지막 위치의 요소 제거 <b>a = [1, 2, 3, 4]</b> a.pop(1) # 1 번 위치의 2 제거 <b>a = [1, 3, 4]</b>
remove()	특정 요소 삭제	s = ['모닝', 'BMW', 'BENZ', '스포티지'] s.remove('BMW') # 요소 직접 삭제 <b>s = ['모닝', 'BENZ', '스포티지']</b>

# 리스트(List)

## ■ 리스트의 주요 메서드(함수)

함수	기능	사용 예
sort()	정렬	<code>a = [1, 4, 2, 3] a.sort() [1, 2, 3, 4]</code>
reverse()	뒤집기	<code>lower = ['b', 'c', 'a'] lower.reverse() ['a', 'b', 'c']</code>
extend(리스트)	리스트의 끝에 리스트 추가	<code>li = ['a', 'b'], li.extend(['c','d']) ['a', 'b', 'c', 'd'],</code>
copy()	리스트 복사	<code>n = [1, 2, 3] m = n.copy()</code>

# 리스트(List)

## ■ 리스트의 주요 메서드(함수)

```
# 요소 추가
fruits = ['사과', '바나나', '귤']
fruits.append('포도') # 맨 끝에 요소 추가
print("append 후:", fruits)

# 요소 삽입
fruits.insert(1, '딸기') # 인덱스 1에 요소 삽입
print("insert 후:", fruits)

# 요소 제거
fruits.remove('바나나') # 값으로 요소 제거
print("remove 후:", fruits)

# 맨 마지막 요소 제거
last_fruit = fruits.pop() # 맨 마지막 요소 제거 및 반환
print("pop 후:", fruits)
```

# 리스트(List)

## ■ 리스트의 주요 메서드(함수)

```
# 요소 정렬
numbers = [5, 2, 9, 1, 5, 6]
numbers.sort() # 오름차순 정렬
print("sort 후:", numbers)

# 요소 뒤집기
numbers.reverse() # 리스트 뒤집기
print("reverse 후:", numbers)

# 리스트 복사
copied_numbers = numbers.copy() # 리스트 복사
print("copy 후:", copied_numbers)

# 리스트 확장
numbers.extend([10, 11, 12]) # 리스트 확장
print("extend 후:", numbers)
```

# 리스트(List)

- 문자열은 특별한 차원 리스트이다.

문자열(시작번호:끝번호)

\* 끝번호는 (끝번호 -1)과 같다

```
s = "Hello, World!"  
print("문자열:", s)  
print("문자열의 길이:", len(s))  
  
# 문자열의 특정 문자 접근  
print("첫 번째 문자:", s[0])  
print("일곱 번째 문자:", s[6])  
print("마지막 문자:", s[-1])  
  
# 문자열 슬라이싱  
print("처음 다섯 문자:", s[0:5])  
print("쉼표부터 끝까지:", s[5:])  
print("모든 문자:", s[:])
```

```
문자열: Hello, World!  
문자열의 길이: 13  
첫 번째 문자: H  
일곱 번째 문자:  
마지막 문자: !  
처음 다섯 문자: Hello  
쉼표부터 끝까지: , World!  
모든 문자: Hello, World!
```

# 리스트(List)

## ▪ 문자열 함수(메서드) 정리

메서드	설명
<b>split()</b>	s = 'banana, grape, kiwi' s = fruit.split(',') [구분기호로 나누고 리스트로 만듬] s ['banana', 'grape', 'kiwi']
<b>replace()</b>	s = 'Hello, World' s = s.replace('World', 'Korea') [문자를 변경함] 'Hello, Korea'
<b>find()</b>	s = "Hello" s.find('H') 0 s.find('k') -1 [문자열이 존재하는 위치 반환. 없으면 -1반환]
<b>strip()</b>	s = " Hi, lee" s.strip() Hi, lee

# 리스트(List)

- 문자열 함수(메서드)

```
# 대소문자 변환
print("소문자 변환:", s.lower()) # hello, world!
print("대문자 변환:", s.upper()) # HELLO, WORLD!

# 문자 개수 세기
print("문자열에서 'o'의 개수:", s.count('o')) # 2

# 특정 문자 위치 찾기
print("문자열에서 'World'의 위치:", s.find('World')) # 7

# 문자열 교체
print("문자열 교체:", s.replace('World', 'Python')) # Hello, Python!
```

# 리스트(List)

- 문자열 함수(메서드)

```
# split과 join
csv = "apple,banana,cherry"
fruits = csv.split(',')
print("분리된 파일 리스트:", fruits) # ['apple', 'banana', 'cherry']
# apple,banana,cherry
print("파일 리스트를 다시 문자열로:", ", ".join(fruits))

# strip
msg = "    Good Luck!    "
print("양쪽 공백 제거:", msg.strip()) # "Good Luck!"
print("왼쪽 공백 제거:", msg.lstrip()) # "Good Luck!    "
print("오른쪽 공백 제거:", msg.rstrip()) # "    Good Luck!"
```

# 리스트(List)

## ▪ 2차원 리스트의 선언 및 생성

- 리스트 내부에 리스트를 가진 자료 구조이다.
- 행과 열의 표(테이블) 형태를 이루고 있다.

리스트 이름 = [ 요소1, 요소2, [요소1, 요소2, 요소3]]

	열1	열2
행1	a[0][0]	a[0][1]
행2	a[1][0]	a[1][1]
행3	a[2][0]	a[2][1]

# 리스트(List)

## ▪ 2차원 리스트 생성 및 출력

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

# 2차원 리스트의 크기
print("2차원 리스트의 크기:")
print("행의 수:", len(matrix)) # 3
print("첫 번째 행의 열의 수:", len(matrix[0])) # 3
print("두 번째 행의 열의 수:", len(matrix[1])) # 3

print("2차원 리스트의 요소 접근:")
print("첫 번째 행, 두 번째 열:", matrix[0][1]) # 2
print("세 번째 행, 첫 번째 열:", matrix[2][0]) # 7
print("두 번째 행 전체:", matrix[1]) # [4, 5, 6]

print("2차원 리스트 (행렬):")
for row in matrix:
    print(row)
```

# 리스트(List)

```
# 요소 추가
matrix.append([10, 11, 12]) # 새로운 행 추가
print("새로운 행 추가 후:")
for row in matrix:
    print(row)

# 요소 수정
matrix[0][0] = 99 # 첫 번째 행, 첫 번째 열 수정
print("첫 번째 행, 첫 번째 열 수정 후:")
for row in matrix:
    print(row)

# 요소 삭제
del matrix[1] # 두 번째 행 삭제
print("두 번째 행 삭제 후:")
for row in matrix:
    print(row)
```

# 딕셔너리(Dictionary)

## ◆ 딕셔너리

리스트 처럼 여러 개의 값을 저장할 수 있고, 키(key)와 값(value)으로 대응시켜 저장하는 자료구조이다.  
중괄호{ }를 사용한다.

딕셔너리 이름 = { 키:값, 키:값....}

{ 'name': '한국민', 'age': 28 }

dictionary

키	키	키
값	값	값

# 딕셔너리(Dictionary)

## ◆ 딕셔너리 주요 메서드

함수	사용 예
<code>d[key] = value</code>	<code>d = {'Tomas':13, 'Jane':9}</code> <code>d['Mike'] = 10 # 요소 추가</code> <code>{'Tomas':13, 'Jane':9, 'Mike':10 }</code>
<code>del d[key]</code>	<code>del d['Jane'] #요소 삭제</code> <code>{'Tomas':13, 'Mike':10 }</code>
<code>d.pop(key)</code>	<code>d.pop('Mike')</code> 10 <code>{'Tomas':13}</code>
<code>clear()</code>	<code>d.clear() # d={} 빈 딕셔너리</code>
<code>d.keys()</code>	<code>d.keys() # 모든 키 가져오기</code> <code>d_keys(['Tomas', 'Mike'])</code>
<code>d.values()</code>	<code>d.Values() # 모든 값 가져오기</code> <code>d_values([13, 10])</code>
<code>d.get(key)</code>	<code>d.get('Jane') #9</code>

# 딕셔너리(Dictionary)

## ◆ 딕셔너리 생성 및 관리

```
student = {  
    "name": "한강",  
    "age": 21,  
    "major": "컴퓨터 공학"  
}  
  
print("딕셔너리 내용:", student)  
print(type(student)) # <class 'dict'>  
  
print("딕셔너리의 크기:", len(student)) # 3  
  
print("딕셔너리의 요소 접근:")  
print("이름:", student["name"]) # 한강  
print("나이:", student["age"]) # 21  
print("전공:", student["major"]) # 컴퓨터 공학  
  
# 딕셔너리 요소 추가  
student["university"] = "한강대학교"  
print("요소 추가 후 딕셔너리:", student)
```

# 딕셔너리(Dictionary)

```
# 딕셔너리 요소 수정  
student["age"] = 22  
print("요소 수정 후 딕셔너리:", student)  
  
# 딕셔너리 요소 삭제  
# del student["major"]  
student.pop("major") # pop 메서드를 사용하여 요소 삭제  
print("요소 삭제 후 딕셔너리:", student)  
  
# 키 목록과 값 목록  
keys = student.keys()  
values = student.values()  
print("키 목록:", list(keys))  
print("값 목록:", list(values))  
  
print("딕셔너리 순회:")  
for key, value in student.items():  
    print(f"{key}: {value}")
```

# 딕셔너리(Dictionary)

## ● 용어 사전 만들기

### ♠ 컴퓨터 용어 사전 ♠

검색할 용어를 입력하세요(종료: q or Q): 이진수

컴퓨터가 사용하는 0과 1로 이루어진 수

검색할 용어를 입력하세요(종료: q or Q): 버그

프로그램이 적절하게 동작하는데 실패하거나 오류가 발생하는 코드 조각

검색할 용어를 입력하세요(종료: q or Q): 함수

정의된 단어가 없습니다.

검색할 용어를 입력하세요(종료: q or Q): q

프로그램 종료!

1. Dictionary 자료구조에 컴퓨터 용어와 정의를 저장한다.
2. 용어를 계속 반복해서 검색 할 수 있다.
3. 검색한 용어가 없으면 정의된 단어가 없음을 알려준다.
4. 검색을 종료하려면 'q' 또는 'Q'를 입력한다.

# 딕셔너리(Dictionary)

## ● 용어 사전 만들기

```
print("♠ 컴퓨터 용어 사전 ♠")  
  
dic = {  
    "CPU": "Central Processing Unit - 컴퓨터의 중앙 처리 장치",  
    "RAM": "Random Access Memory - 임의 접근 메모리",  
    "이진수": "0과 1로 이루어진 수 체계",  
    "알고리즘": "문제를 해결하기 위한 절차나 방법"  
}
```

# 딕셔너리(Dictionary)

## ● 용어 사전 만들기

```
while True:  
    word = input("검색할 용어를 입력하세요 (종료: q or Q): ")  
    if word == 'q' or word == 'Q':  
        print("용어 사전 프로그램을 종료합니다.")  
        break  
    definition = dic.get(word) # 용어에 대한 정의 검색  
    if definition:  
        print(f"{word}: {definition}")  
    else:  
        print(f"'{word}' 용어는 사전에 없습니다.")
```

# 튜플(tuple)

- 튜플(tuple)

- 튜플의 요소를 변경(추가, 수정, 삭제)할 수 없다.
- 요소 추가는 초기화나 튜플간 합치기를 하면 가능함
- 리스트처럼 동일한 방식으로 인덱싱과 슬라이싱 가능함
- 소괄호( )를 사용한다.

튜플 이름 = (요소1, 요소2....)

```
t1 = ()  
t2 = (1, )  
t3 = (1, 2, 3)  
t4 = ('a', 'b', 'c')
```

# 튜플(tuple)

## ● 튜플 자료형

```
# 튜플 생성
t = (1, 2, 3)
print(t) # (1, 2, 3)
print(type(t)) # <class 'tuple'>

# 특정 요소 접근
print("첫 번째 요소:", t[0]) # 1
print("두 번째 요소:", t[1]) # 2
print("마지막 요소:", t[-1]) # 3

# 튜플 길이
print("튜플의 길이:", len(t)) # 3

# 슬라이싱
print("첫 두 요소:", t[0:2]) # (1, 2)
print("두 번째 요소부터 끝까지:", t[1:]) # (2, 3)
```

# 튜플(tuple)

## ● 튜플 자료형

```
# 튜플은 변경 불가능(immutable)
# t[0] = 10 # 오류 발생
# 튜플 삭제 불가능
# del t[1] # 오류 발생

# 단일 요소 튜플 생성
t2 = (5)
print("단일 요소 튜플 아님:", t2) # 5

t2 = (5,)
print("단일 요소 튜플:", t2) # (5,)

# 튜플 연결
t3 = t + t2
print("튜플 연결:", t3) # (1, 2, 3, 5)
```

튜플의 요소는 수정 및  
삭제 할 수 없다.

# 튜플(tuple)

## ● 튜플 사용 예제

```
# 점수 입력
scores = []
scores.append((80,))
scores.append((70,))
scores.append((90,))
print(scores)

# 과목과 점수 입력
subjects = []
subjects.append(("국어", 90))
subjects.append(("수학", 80))
print(subjects) # [('국어', 90), ('수학', 80)]

# 요소 조회
print(subjects[0]) # ('국어', 90)
print(subjects[1]) # ('수학', 80)

# 이차원 검색
print(subjects[0][1]) # 90
print(subjects[1][1]) # 80
```