

# SKKU 헬스케어 EDA Intro



# 1. [Motivation] 의료 AI/헬스케어 사례



# [Motivation]

## 코딩, 왜 강력한 무기인가?



- 데이터를 다루는 직군이든, 아니든, 우리는 매일 데이터의 홍수 속에서 살아간다. 의료 환경으로 좁혀 보면, 환자의 전자의무기록(EMR), 수많은 임상 연구 논문, 엑셀에 정리된 실험 데이터 등.
- 이 데이터 안에 숨겨진 패턴을 발견하고, 반복적인 작업을 자동화하는 능력은 단순한 '기술'을 넘어 연구와 진료의 질을 높이는 '핵심 역량'이 된다.
- 파이썬은 이 모든 것을 가능하게 하는 가장 강력하고 접근하기 쉬운 도구이다.

# 의료 분야의 데이터 과제



Sooraj Sushama 박사, 박사, MBA

NYSIF의 수석 보험수리사 겸 보험수리 분석 책임자 | 페퍼다인 대학교  
의 분석 교수 | 블록체인 저자



2022년 9월 21일

의료 산업에서 데이터가 점점 더 중요해지고 있다는 것은 의심의 여지가 없습니다. 방대한 양의 데이터를 수집, 저장, 분석할 수 있게 되면서 환자 개개인에게 맞춤형 치료를 제공하는 "정밀 의학"의 새로운 시대가 열리고 있습니다. 하지만 개인 재정을 관리해 본 사람이라면 누구나 알다시피, 데이터 관리는 쉽지 않습니다. 오늘날 의료 데이터가 직면한 가장 큰 과제들을 소개합니다.

## 1. 생성되는 데이터의 양이 엄청납니다.

전 세계는 매일 2.5조 바이트(약 250경 바이트)의 데이터를 생성하며, 그중 상당 부분은 의료 산업에서 발생합니다. 의료 서비스 제공업체는 환자 의료 기록부터 보험 청구까지 매일 엄청난 양의 데이터를 생성합니다. 한 추정에 따르면, 평균적인 병원은 매년 약 1페타바이트(100만 기가바이트)의 데이터를 생성합니다. 0과 1이 엄청나게 많은 양이죠!

## 2. 데이터에 접근하거나 효과적으로 사용하는 것이 항상 쉬운 것은 아닙니다.

데이터가 존재한다고 해서 사용하기 쉬운 것은 아닙니다. 많은 경우 데이터는 여러 부서, 컴퓨터 시스템, 심지어 여러 조직에 분산되어 있습니다. 심지어 단일 병원 내에서도 개인정보 보호 문제로 인해 여러 의사와 임상이가 환자 정보를 공유하는 것은 어려울 수 있습니다. 결과적으로, 이러한 모든 원시 데이터를 실행 가능한 통찰력으로 변환하는 것은 상당히 어려울 수 있습니다.

# 역대 최고치: 의료 데이터 침해 위협의 급증에 맞서다



스콧 스페란자 헬스락 2025년 5월 30일



우리 사회의 초석인 의료 산업은 방대한 양의 민감한 환자 데이터를 보호하는 데 있어 전례 없는 어려움에 직면해 있습니다. 한때 우려스러웠던 것이 이제는 위기로 변했습니다. 의료 데이터 유출은 단발적인 사건이 아니라 끊임없는 흐름이며, 수백만 명의 사람들에게 영향을 미치고 의료 생태계에 막대한 부담을 주고 있습니다.

IT 및 사이버 보안 전문가에게 이 상황의 심각성, 근본 원인, 그리고 현재 방어 시스템의 단점을 이해하는 것은 더욱 안전한 미래를 구축하기 위한 첫 번째 중요한 단계입니다.

통계는 냉혹한 현실을 보여줍니다. Bluesight에서 발표한 **2025년 데이터 침해 지표(Breach Barometer)**에 따르면, 의료 분야는 2024년에 데이터 침해가 급증하여 3억 건이 넘는 환자 기록이 유출되었는데, 이는 2023년 대비 26% 증가한 수치입니다. 이 수치에는 미국 인구 두 명 중 한 명꼴로 영향을 미친, 역사상 가장 광범위한 의료 데이터 침해가 포함됩니다.<sup>1</sup> <sup>통계</sup>에 따르면 의료 기록은 기존 금융 정보보다 50배 더 가치가 높기 때문에, 사이버 범죄자에게는 위협 대비 보상이 상당합니다.

2025년 초에도 이러한 추세는 지속되고 있습니다. HIPAA 저널은 **2025년 2월 데이터 침해 보고서**에서 500명 이상 관련 대규모 의료 데이터 침해 사건 46건을 보고했으며, 이로 인해 120만 명이 피해를 입었다고 밝혔습니다.<sup>2</sup> 이는 전월 대비 감소세를 보였지만, 2024년 한 해 동안 발생한 침해 건수가 많다는 점은 상승 추세에서 일시적인 하락세일 가능성을 시사합니다.

이 전염병의 원인은 다면적입니다. 보고된 데이터에 따르면, 가장 중요한 원인은 해킹 및 기타 IT 사고로, 2025년 2월 보고된 침해 사고의 74%를 차지했으며, 110만 명 이상(전체 피해 사례의 89%)의 개인 건강 정보(PHI)가 유출되었습니다.

# 사례 1: 수십 개의 엑셀 파일, 5분 만에 하나로 합치고 정리하기

- Before (현실):

여러 명의 연구원이 각자 수집한 환자 데이터를 별도의 엑셀 파일로 관리한다. (A연구원\_혈액검사.xlsx, B연구원\_인적사항.xlsx, C연구원\_과거력.xlsx ...)

데이터 취합을 위해 담당자가 모든 파일을 열어 특정 열을 복사하고, 새로운 엑셀 시트에 붙여넣는 작업을 반복한다.

환자 ID를 기준으로 정렬하고, 오타나 빈칸이 없는지 일일이 눈으로 확인한다.

**결과:** 수 시간이 소요되고, 복사/붙여넣기 과정에서 실수가 발생할 확률이 높다.

- After (Python 활용):

폴더 안에 있는 모든 엑셀 파일을 자동으로 인식하는 파이썬 스크립트를 작성한다.

각 파일에서 필요한 열(환자 ID, 나이, 특정 검사 수치 등)만 추출하고, 환자 ID를 기준으로 모든 데이터를 하나의 표로 자동 병합한다.

병합된 데이터에서 빈칸이 몇 개인지, 특정 수치가 정상 범위를 벗어나는 경우는 몇 건인지 요약 리포트를 생성한다.

**결과:** 단 몇 줄의 코드로 5분 안에 모든 작업이 완료된다. 사람은 분석과 해석이라는 더 중요한 일에 집중할 수 있다.

# 사례 2: 최신 논문, 키워드만 입력하면 자동으로 목록 만들기

- Before (현실):

새로운 연구 주제('Metformin and aging')에 대한 최신 동향을 파악하기 위해 매일 PubMed에 접속한다.

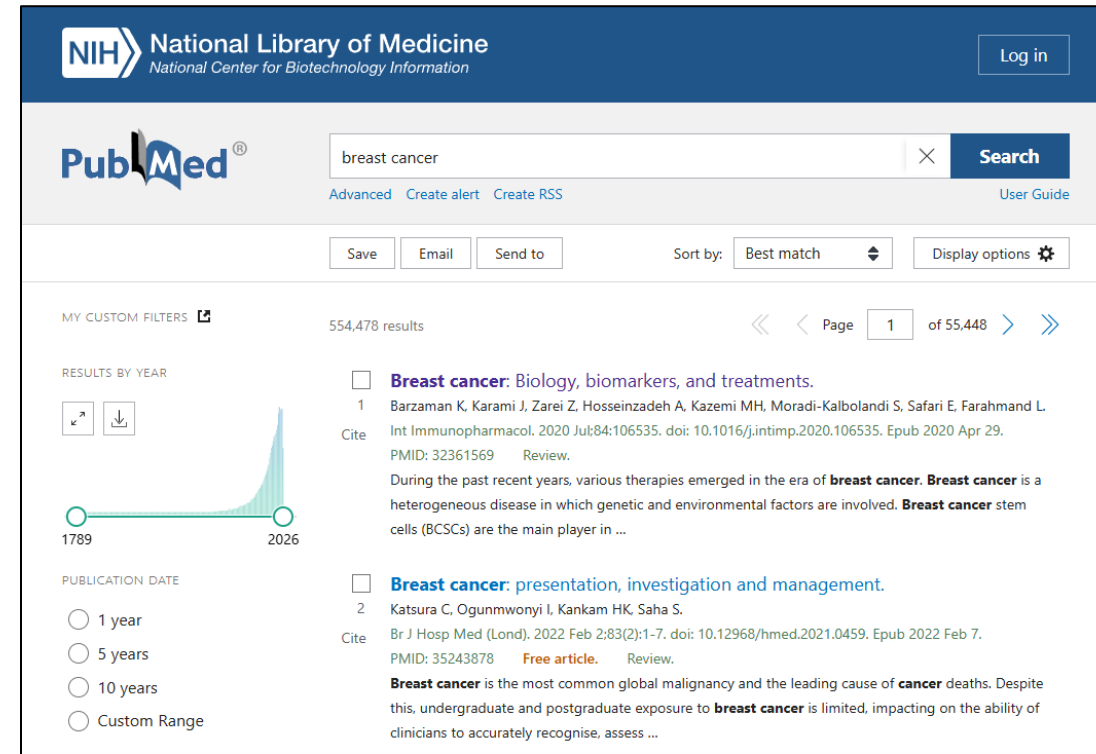
키워드로 검색하고, 결과 목록을 스크롤하며 제목과 초록을 읽고, 유의미해 보이는 논문을 북마크하거나 엑셀에 수동으로 정리한다.

결과: 매일 30분 이상 소요되는 반복적인 작업이며, 중요한 논문을 놓칠 수 있다.

- After (Python 활용):

PubMed에 특정 키워드로 검색을 요청하고, 결과 페이지의 HTML을 가져오는 스크립트를 작성한다. 가져온 HTML에서 논문 제목, 저자, 저널, 초록 부분만 정확히 추출(파싱)한다. 추출된 정보를 날짜순으로 정리하여 CSV 파일로 자동 저장하거나, 이메일로 발송하도록 설정한다.

결과: 스크립트 실행 한 번으로 1분 안에 원하는 정보가 정리된 파일이 생성된다. '자동 비서'가 생긴 것과 같다.



# 사례 3: 복잡한 임상 점수 계산, 환자 ID만 넣으면 자동으로

- Before (현실):

심방세동 환자의 뇌졸중 위험도를 평가하기 위해 CHA<sub>2</sub>DS<sub>2</sub>-VASc 점수를 계산한다. (C: 울혈성 심부전(1점), H: 고혈압(1점), A<sub>2</sub>: 75세 이상(2점) ...)

환자 차트를 보며 각 항목에 해당하는지 일일이 확인하고, 점수를 더하여 총점을 계산한다.

**결과:** 환자가 많을수록 시간이 오래 걸리고, 항목을 누락하거나 점수를 잘못 더하는 실수가 발생할 수 있다.

- After (Python 활용):

환자의 나이, 성별, 과거력(고혈압, 당뇨 등)을 입력받아 CHA<sub>2</sub>DS<sub>2</sub>-VASc 점수를 계산하는 함수를 정의한다.

엑셀 파일에 정리된 수십 명의 환자 리스트를 불러와, 각 환자에게 이 함수를 적용하여 점수를 자동으로 계산하고 결과를 새로운 열에 추가한다.

**결과:** 수백 명의 환자 데이터도 몇 초 만에 정확하게 계산이 완료된다.

## 2. 내 손으로 만드는 개발 환경



## 2. 내 손으로 만드는 개발 환경

- 코딩을 처음 시작할 때 가장 큰 장벽은 바로 이 '개발 환경 설정'이다.

[비유로 이해하기: 우리의 '디지털 연구실']

- 파이썬 (Python): 연구의 기반이 되는 '핵심 과학 원리'이다. 데이터를 분석하고, 기계를 학습시키는 모든 법칙과 언어가 여기에 담겨 있다.
- VSCode (Visual Studio Code): 우리가 연구를 수행할 '연구실 건물'이다. 코드를 작성하고, 실험(실행)하고, 결과를 확인하는 모든 활동이 이루어지는 공간이다.
- 가상 환경 (Virtual Environment): 연구실 안의 '독립된 클린룸(Clean Room)'입니다. A라는 실험에는 시약 1.0 버전이 필요하고, B라는 실험에는 시약 2.0 버전이 필요할 때, 두 시약이 섞이면 실험을 망치게 된다. 가상 환경은 각 프로젝트(실험)마다 필요한 도구(시약)들을 완벽히 분리하여, 충돌 없이 안정적으로 연구를 진행할 수 있게 해주는 필수 공간이다.

# 1단계: 파이썬 설치하기

- 가장 먼저 컴퓨터가 파이썬이라는 언어를 이해할 수 있도록 설치해야 합니다.

- 파이썬 공식 홈페이지 접속

웹 브라우저를 열고 [python.org](https://www.python.org) 로 이동합니다.

메뉴에서 [Downloads] 위에 마우스를 올리면 [Download for Windows] 버튼과 함께 최신 버전(예: Python 3.11.x)이 보입니다. 이 버튼을 클릭하여 설치 파일을 다운로드합니다.

<https://www.python.org/downloads/windows/> 이동

<https://www.python.org/ftp/python/3.12.7/python-3.12.7-amd64.exe> ← 다운 및 설치

- 설치 파일 실행 및 가장 중요한 설정

다운로드한 .exe 파일을 실행합니다.

설치 화면이 나타나면, 반드시 화면 하단의 Add Python.exe to PATH 체크박스를 클릭하여 체크해야 합니다.

- 왜 중요한가요? 이 설정을 해야만 컴퓨터의 어느 위치에서든 "파이썬!" 하고 불렀을 때 컴퓨터가 파이썬을 찾아낼 수 있습니다. 이 설정을 놓치면 많은 문제가 발생합니다.

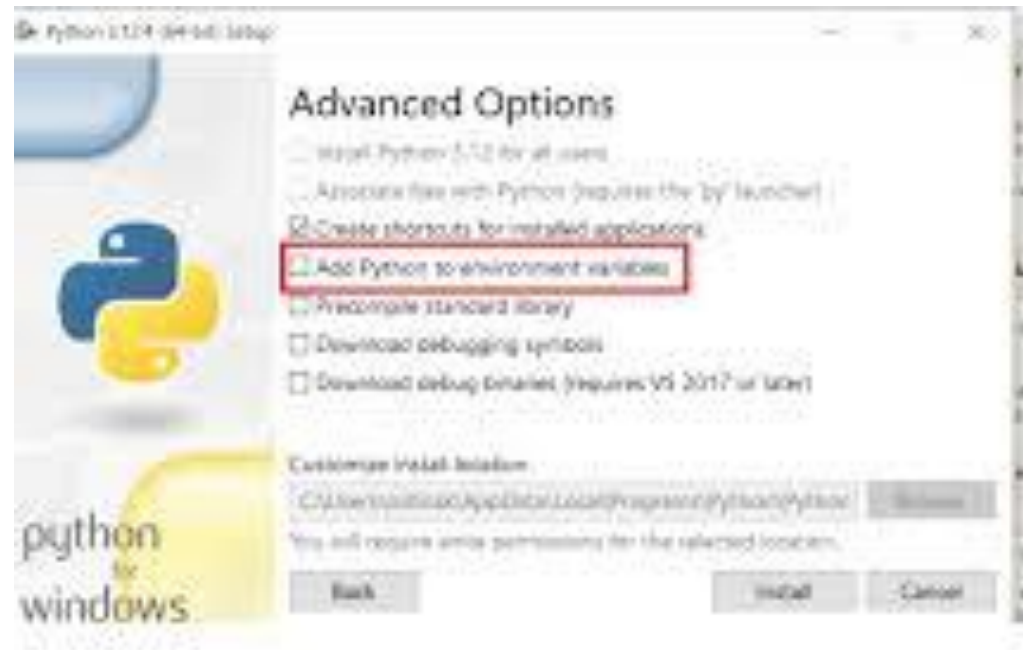
# 1단계: 파이썬 설치하기

- 설치 파일 실행 및 가장 중요한 설정

다운로드한 .exe 파일을 실행합니다.

설치 화면이 나타나면, 반드시 화면 하단의 Add Python.exe to PATH 체크박스를 클릭하여 체크해야 합니다.

- 왜 중요한가요? 이 설정을 해야만 컴퓨터의 어느 위치에서든 "파이썬!" 하고 불렀을 때 컴퓨터가 파이썬을 찾아낼 수 있습니다. 이 설정을 놓치면 많은 문제가 발생합니다.



# 1단계: 파이썬 설치하기

- 설치 진행

Install Now를 클릭하여 설치를 진행합니다. 설치가 완료될 때까지 잠시 기다립니다.

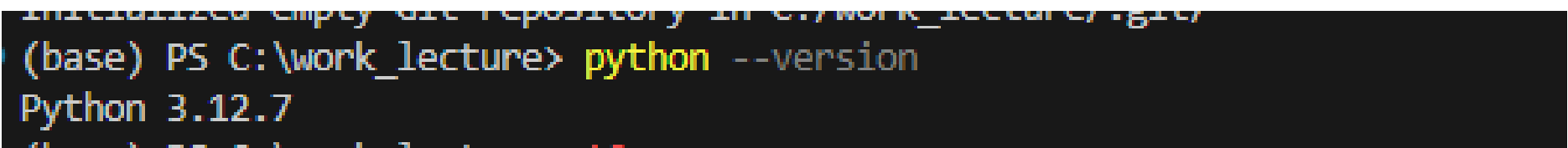
"Setup was successful" 메시지가 나오면 성공적으로 설치된 것입니다.

- 설치 확인

Windows 시작 버튼을 누르고 cmd를 입력하여 '명령 프롬프트'를 실행합니다.

까만 화면에 다음 명령어를 입력하고 Enter 키를 누릅니다.

```
python --version
```



```
Initialized empty Git repository in C:/work_lecture/.git/
(base) PS C:\work_lecture> python --version
Python 3.12.7
```

# For mac user..

Homebrew 사용 (개발자들에게 가장 추천)

- 가장 깔끔하고 관리가 쉬운 표준적인 방법

1. 터미널(Terminal) 앱을 엽니다 (Command + Space 후 'terminal' 검색).

2. Homebrew 설치 (이미 설치되어 있다면 패스):

아래 명령어를 복사해 터미널에 붙여넣고 엔터를 칩니다. (설치 중 비밀번호 입력 필요)

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

3. Python 설치:

```
brew install python
```

4. 설치 확인:

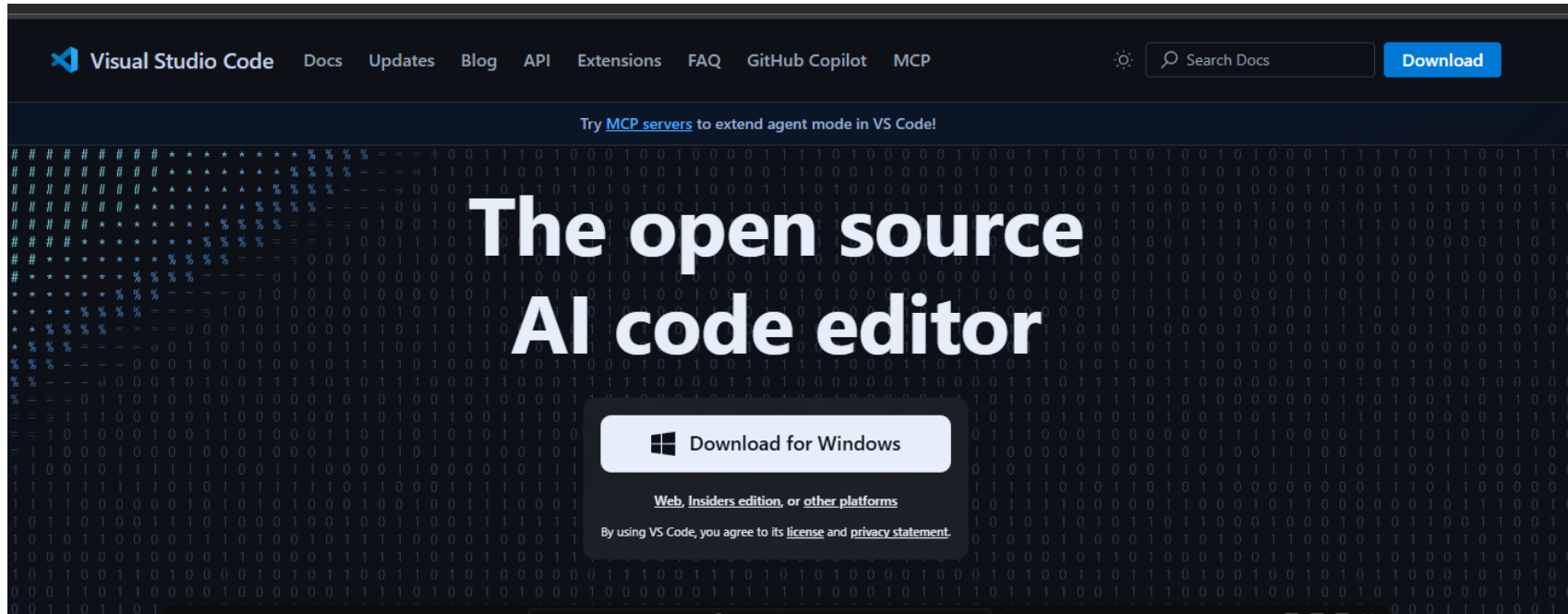
```
python3 --version
```

## 2단계: VSCode 설치하기 (나만의 '연구실 건물' 짓기)

- 이제 코드를 작성하고 관리할 멋진 공간을 만들 차례입니다.
- VSCode 공식 홈페이지 접속

웹 브라우저에서 [code.visualstudio.com](https://code.visualstudio.com) 으로 이동합니다.

파란색 [Download for Windows] 버튼을 클릭하여 설치 파일을 다운로드합니다.



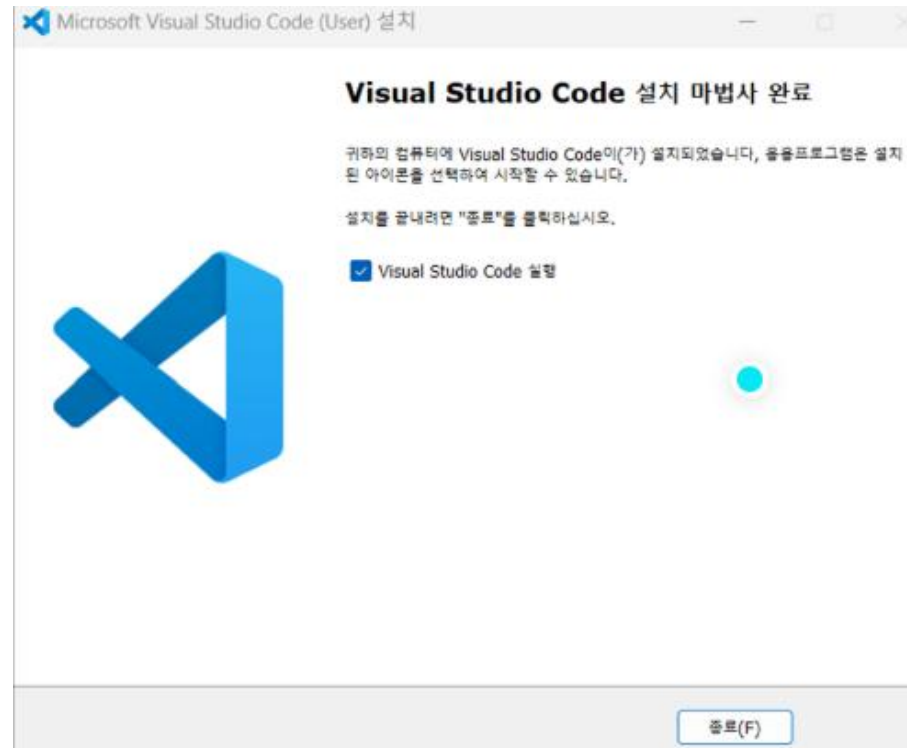
## 2단계: VSCode 설치하기 (나만의 '연구실 건물' 짓기)

- 설치 진행

다운로드한 .exe 파일을 실행합니다.

'동의합니다'를 선택하고, 계속 '다음' 버튼을 눌러 설치를 진행합니다. 기본 설정을 그대로 두어도 괜찮습니다.

설치가 완료되면 VSCode가 자동으로 실행됩니다.



## 2단계: VSCode 설치하기 (나만의 '연구실 건물' 짓기)

- 필수 확장 프로그램(연구 도구) 설치

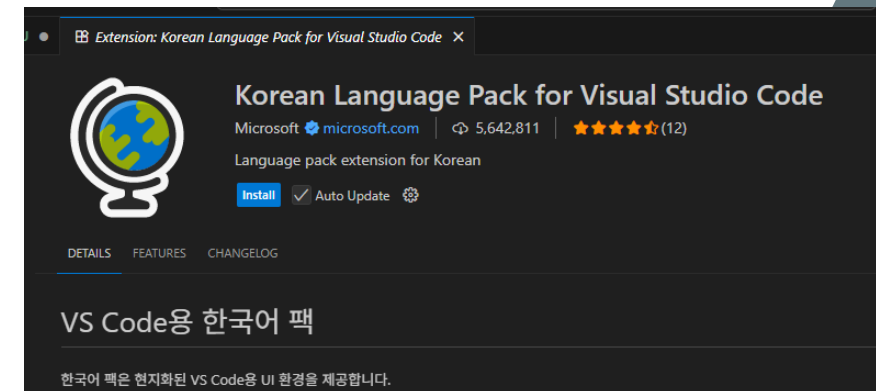
VSCode를 더 강력하게 만들어주는 '추가 도구'를 설치해야 합니다.

VSCode 화면 왼쪽의 아이콘 메뉴에서 레고 블록 모양의 [Extensions] 아이콘을 클릭합니다.

검색창에 Python을 검색합니다.

가장 위에 나오는, Microsoft가 제공하는 Python 확장 프로그램을 찾아 [Install] 버튼을 클릭합니다.

(추가) 같은 방식으로 Korean Language Pack for Visual Studio Code 를 검색하여 설치하면 메뉴가 한글로 바뀝니다.



# For mac user..

1.공식 홈페이지 접속: <https://code.visualstudio.com/> 으로 이동합니다.

2.다운로드: 메인 화면의 파란색 'Download Mac Universal' 버튼을 클릭합니다. (Universal 버전은 Intel 맥과 Apple Silicon 맥 모두 호환됩니다.)

## 3.설치 (중요):

다운로드 폴더에 VSCode-darwin-universal.zip 파일이 받아지면 더블 클릭하여 압축을 풉니다.

Visual Studio Code.app이라는 파란색 아이콘이 나옵니다.

이 아이콘을 드래그해서 Finder의 '응용 프로그램(Applications)' 폴더로 옮겨주세요. (다운로드 폴더에서 바로 실행하면 나중에 권한 문제가 생길 수 있습니다.)

4.실행: 응용 프로그램 폴더에서 VS Code를 실행합니다.

### 터미널에서 code 명령어로 열기

1. 터미널에서 프로젝트 폴더로 이동한 뒤 `code .`이라고 치면 바로 VS Code가 열리게 설정하면 매우 편합니다.
2. VS Code를 켭니다.
3. `Cmd + Shift + P`를 눌러 명령어 팔레트를 엽니다.
4. `shell command`라고 검색합니다.
5. `Shell Command: Install 'code' command in PATH`를 클릭합니다.
6. 이제 터미널에서 어느 폴더에서든 `code .`을 입력하면 해당 폴더가 VS Code로 열립니다.

### 3. 파이썬 가상 환경 설정 (실험별 '클린룸' 만들기)

이제 실제 프로젝트를 진행할 깨끗하고 독립된 공간을 만드는, 가장 전문적인 단계입니다.

- 프로젝트 폴더 만들기

바탕화면이나 문서 폴더 등 원하는 위치에 이 과정을 위한 새 폴더를 만듭니다.

폴더 이름: medical-ai

- VSCode에서 프로젝트 폴더 열기

VSCode를 실행하고, 상단 메뉴에서 파일 > 폴더 열기...를 선택합니다.

방금 만든 medical-ai 폴더를 선택하여 엽니다.

- VSCode 내장 터미널 열기

VSCode 상단 메뉴에서 터미널 > 새 터미널을 클릭하거나, 단축키 Ctrl + J를 누릅니다.

화면 하단에 명령어를 입력할 수 있는 터미널 창이 나타납니다.

### 3. 파이썬 가상 환경 설정 (실험별 '클린룸' 만들기)

- 가상 환경(클린룸) 생성하기

열린 터미널에 다음 명령어를 정확히 입력하고 Enter를 누릅니다.

`python -m venv medi`

```
PS C:\work_lecture> python -m venv medi
PS C:\work_lecture> ls

디렉터리: C:\work_lecture

Mode                LastWriteTime         Length Name
----                -
d-----          2025-07-17 오후 8:03             medi
-a----          2025-07-17 오후 7:12         356 Interactive-1.ipynb
```

해설: "파이썬아(python), 가상 환경 만드는 도구(-m venv)를 사용해서, venv라는 이름의 가상 환경 폴더를 만들어줘."  
명령을 실행하면 왼쪽 폴더 목록에 venv라는 폴더가 새로 생긴 것을 볼 수 있습니다. 이곳이 바로 우리의 클린룸입니다.

- 가상 환경(클린룸) 활성화하기

만들기만 한 클린룸에 직접 '들어가는' 과정입니다. 이 과정을 거쳐야만 독립된 환경에서 작업을 시작할 수 있습니다.

터미널에 다음 명령어를 입력하고 Enter를 누릅니다. (.Wv까지 입력하고 Tab 키를 누르면 자동 완성 기능을 활용할 수 있습니다.)


### 3. 파이썬 가상 환경 설정 (실험별 '클린룸' 만들기)

- 가상 환경(클린룸) 활성화하기

만들기만 한 클린룸에 직접 '들어가는' 과정입니다. 이 과정을 거쳐야만 독립된 환경에서 작업을 시작할 수 있습니다.

터미널에 다음 명령어를 입력하고 Enter를 누릅니다. (.Wm까지 입력하고 Tab 키를 누르면 자동 완성 기능을 활용할 수 있습니다.)

.WmediWScriptsWActivate.ps1



```
PS C:\work_lecture> .\medi\Scripts\Activate.ps1
(medi) PS C:\work_lecture>
```

- 성공 확인:** 명령어 줄 맨 앞에 (venv) 라는 초록색 글자가 나타나면, 성공적으로 클린룸에 들어온 것입니다! 이제부터 설치하는 모든 도구는 이 venv 안에만 설치됩니다.
- 만약 오류가 발생한다면?** PowerShell의 실행 정책 때문일 수 있습니다. 터미널에 Set-ExecutionPolicy Unrestricted -Scope Process를 입력하고 Enter를 친 뒤, 다시 활성화 명령어를 시도해 보세요.

- 가상 환경 비활성화하기
- 빠져 나오고 싶을 때는 터미널에 다음 명령어를 입력하면 됩니다.

deactivate

```
(medi) PS C:\work_lecture> deactivate
PS C:\work_lecture> █
```

(medi) 표시가 사라지는 것을 볼 수 있.

- 이제 여러분은 어떤 파이썬 프로젝트든 독립적이고 안정적으로 수행할 수 있는 완벽한 '디지털 연구실'을 갖추게 되었습니다. 다음 단계는 이 클린룸 안에 분석에 필요한 도구들(Pandas, Matplotlib 등)을 설치하고 실제 데이터 분석을 시작하는 것입니다.

# For mac user..

## 1. 터미널 열기 및 프로젝트 폴더 이동

- 먼저 터미널을 열고, 작업을 진행할 폴더를 하나 만들어서 이동합니다.

# 예: 바탕화면에 'ai\_project'라는 폴더를 만들고 이동

```
mkdir ~/Desktop/ai_project
```

```
cd ~/Desktop/ai_project
```

## 2. 가상환경 생성 (Install)

파이썬 3의 venv 모듈을 실행하여 가상환경을 만듭니다. 관례적으로 가상환경 이름은 venv 또는 .venv를 많이 사용합니다.

# python3 -m venv [가상환경이름]

```
python3 -m venv venv
```

# For mac user..

## 3. 가상환경 활성화 (Activate)

- 가상환경을 만들었다고 바로 적용되는 것이 아니라, **\*\*"지금부터 이 가상환경을 쓰겠다"\*\***고 선언(활성화)해 주어야 합니다.

```
# source [가상환경폴더경로]/bin/activate
```

```
source venv/bin/activate
```

## 4. 가상환경 비활성화 (Deactivate)

- 작업을 마치고 원래 시스템 파이썬 환경으로 돌아가려면 다음 명령어를 입력합니다

```
deactivate
```

### VS Code와 연동하기

터미널에서 가상환경을 만들어도, **VS Code가 그 가상환경을 인식하지 못하면** 코드에서 에러가 난다.

**1.VS Code 실행:** 터미널에서 `code .` 을 입력하거나 VS Code를 켜서 해당 프로젝트 폴더(ai\_project)를 연다.

**2.파이썬 파일 열기:** .py 파일을 하나 만들거나 열기

**3.인터프리터 선택:**

- 화면 **오른쪽 하단**에 있는 파이썬 버전 표시(예: 3.10.x 64-bit)를 클릭
- 또는 단축키 `Cmd + Shift + P` -> `Python: Select Interpreter` 검색.

**4.가상환경 선택:**

- 목록에서 방금 만든 **venv ('venv': venv)** 혹은 **(Recommended)** 라고 적힌 항목을 선택
- 이걸 선택해야 VS Code가 터미널에서 설치한 라이브러리들을 인식*

### 3. 파이썬 핵심 문법 익히기 (1)

#### 데이터 담기 (변수, 타입, 자료형, 자료구조)



# 프로그래밍이란.

프로그래밍:

프로그램을 계획(planning), 작업의 순서를 정하거나 실행하는 것

"주어진 문제를 해결하기 위해" 작업의 순서를 계획하고 수행하는 것



목차 숨기기

처음 위치

[프로그래밍 언어](#)

[소프트웨어 개발](#)

[같이 보기](#)

[참조](#)

[외부 링크](#)

## 컴퓨터 프로그래밍

문서 토론

위키백과, 우리 모두의 백과사전.

🔗 [프로그래밍](#)은 여기로 연결됩니다. 다른 뜻에 대해서는 [프로그래밍 \(동음이의\)](#) 문서를 참고하십시오.

**컴퓨터 프로그래밍**(영어: computer programming) 또는 간단히 **프로그래밍**(programming, 문화어: 프로그램 작성) 혹은 **코딩**(coding)은 하나 이상의 관련된 추상 **알고리즘**을 특정한 **프로그래밍 언어**를 이용해 구체적인 **컴퓨터 프로그램**으로 **구현**하는 기술이다.<sup>[1]</sup> 프로그래밍은 **기법**, **과학**, **수학**, **공학**, **심리학**적 속성들을 가지고 있다.

한편 코딩은 '작업의 흐름에 따라 **프로그램 언어**의 명령문을 써서 **프로그램**을 작성하는 일' 또는 '프로그램의 **코드**를 작성하는 일'로 크게 나누어 언급되고 있는데 이는 알고리즘과의 상관관계를 잘 언급하고 있다.

## 프로그래밍 언어 [ 편집 ]

🔍 이 부분의 본문은 [프로그래밍 언어](#)입니다.

특정한 프로그래밍 언어로 쓰인 프로그램은 **기계어**로 번역되어 컴퓨터에 의해 **실행**되며, 어떤 프로그래밍 언어도 기계어로 번역이 가능하다. 어떤 언어에서는 기계어 대신 ***p-부호***로 불리는 바리를 생성하기도 한다. **프로그래머**가 기계 부호로 직접 작성하는 것도 가능하지만, 이는 굉장히 어려운 작업이다. 때문에 낮은 수준에서의 컴퓨터 제어가 필요한 경우 프로그래머들은 기계어 명령어에 대한 일대일 연상 기호 대응인 **어셈블리어**를 사용한다.

서로 다른 프로그래밍 언어는 다른 프로그래밍 유형을 지원하기 때문에, 분야에 따라 적합한 언어가 존재한다. 또한 언어마다 프로그래머가 알고리즘을 구현할 때 그 구체적인 방법과 수준의 차이가 있기 때문에, 사용의 편의성과 성능 사이에서 적절한 타협이 이루어진다. 또한 프로그래밍의 언어 중 하나이며 컴퓨터에게 명령을 전달할 수 있다.

Who?



러브레이스 백작부인 어거스터 에이다 킹(Augusta Ada King, Countess of Lovelace, 1815년 12월 10일~1852년 11월 27일)은 영국의 수학자이자 작가이다.<sup>[1]</sup> 시인 조지 고든 바이런의 딸로, 출생명은 어거스터 에이다 바이런이며, 대중적으로는 에이다 바이런, 혹은 에이다 러브레이스라고도 불린다.<sup>[2]</sup>

과학만능주의가 팽배하던 19세기를 살다간 귀족여성으로서 이학적(理學的)인 관심과 타고난 지능을 바탕으로 초기 컴퓨터과학에 인상적인 발자취를 남겼다. 에이다는 찰스 배비지의 연구에 대한 좋은 이해자이자 협력자였고, 배비지가 고안한 해석기관을 위한 공동작업으로 널리 알려져 있다.

해석기관에서 처리될 목적으로 작성된 **알고리즘**이 최초의 컴퓨터 프로그램으로 인정되었던 바 ‘세계 최초의 **프로그래머**’라는 수식어가 붙는다. 해석기관을 단순한 계산기 또는 수치 처리 장치로만 생각하던 당대의 과학자들과는 달리 훨씬 다양한 목적으로 활용될 수 있는 가능성에 주목하여 현대 컴퓨터의 출현을 예측하였다.

**프로그래밍 언어**에서 사용되는 중요한 개념인 **루프**, **GOTO문**, **IF문**과 같은 **제어문**의 개념을 소개하였다. 그는 **서브루틴**에 관한 개념도 고안하였는데, 이것은 1948년 **모리스 윌키**

에이다 러브레이스  
Ada Lovelace



**세계 최초의 프로그래머..**

# 프로그래밍이란?

- 컴퓨터 프로그래밍

컴퓨터가 주어진 문제를 해결할 수 있도록 사람이 지시해야 하는 일

- 사람이 하는 일

문제 해석

요구사항 분석

계획

처리 과정(명령) 지시 (코딩) 및 디버깅



최근에는,  
AX 도메인으로 전환

- 컴퓨터가 하는 일

주어진 명령을 처리

# 프로그래밍 언어

- 저수준(Low-level), 고수준(High-level) 언어로 구분  
사람에게 가까운지(고수준) 컴퓨터에게 가까운지(저수준)을 결정
- 저수준 프로그래밍 언어  
사람보다 컴퓨터가 이해하기 쉬움  
하드웨어 종속적  
이진수로 구성된 기계어에 거의 1:1로 사상시킨 어셈블리어언어
- 고수준 프로그래밍 언어  
컴퓨터보다 사람이 이해하기 쉽게 만들어진 언어  
하드웨어에 비종속적  
파이썬(Python), C, C++, 자바(Java) 등

# Python 이란

출처 : <https://ko.wikipedia.org/wiki/%ED%8C%8C%EC%9D%B4%EC%8D%AC>

파이썬은 1980년대 말 고안되어 네덜란드 CWI의 귀도 반 로섬이 1989년 12월 구현하기 시작하였다. 이는 역시 SETL에서 영감을 받은 ABC 언어의 후계로서 예외 처리가 가능하고, 아메바 OS와 연동이 가능하였다. 반 로섬은 파이썬의 주 저자로 계속 중심적 역할을 맡아 파이썬의 방향을 결정하여, 파이썬 공동체로부터 '자선 종신 이사'의 칭호를 부여받았다. 이 같은 예로는 리눅스의 리누스 토발즈 등이 있다.

## 파이썬 2

- 파이썬 2.0은 2000년 10월 16일 배포되었고, 많은 기능이 추가되었다. 그중 전면적인 [쓰레기 수집기](#)(GC, Garbage Collector) 탑재와 [유니코드](#) 지원이 특징적이다. 그러나 가장 중요한 변화는 개발 절차 그 자체로, 더 투명하고 공동체 지원을 받는 형태가 되었다.
- 2020년 1월 1일부로 파이썬 2의 지원이 종료되었다.[\[6\]](#)[\[7\]](#)

## 파이썬 3

- 파이썬3000(혹은 파이썬3k)이라는 코드명을 지닌 파이썬의 3.0버전의 최종판이 긴 테스트를 거쳐 2008년 12월 3일자로 발표되었다. 2.x대 버전의 파이썬과 하위호환성이 없다는 것이 가장 큰 특징이다. 파이썬 3의 주요 기능 다수가 이전 버전과 호환되게 2.6과 2.7 버전에도 반영됐다.
- 파이썬 공식 문서에서는 "파이썬 2.x는 레거시(낡은 기술)이고, 파이썬 3.x가 파이썬의 현재와 미래가 될 것"이라고 요약했는데, 처음 배우는 프로그래머들은 파이썬 3으로 시작하는 것을 권장하고 있다.[\[8\]](#)

# 프로그램 실행 방식

## 실행 방식

### 인터프리터

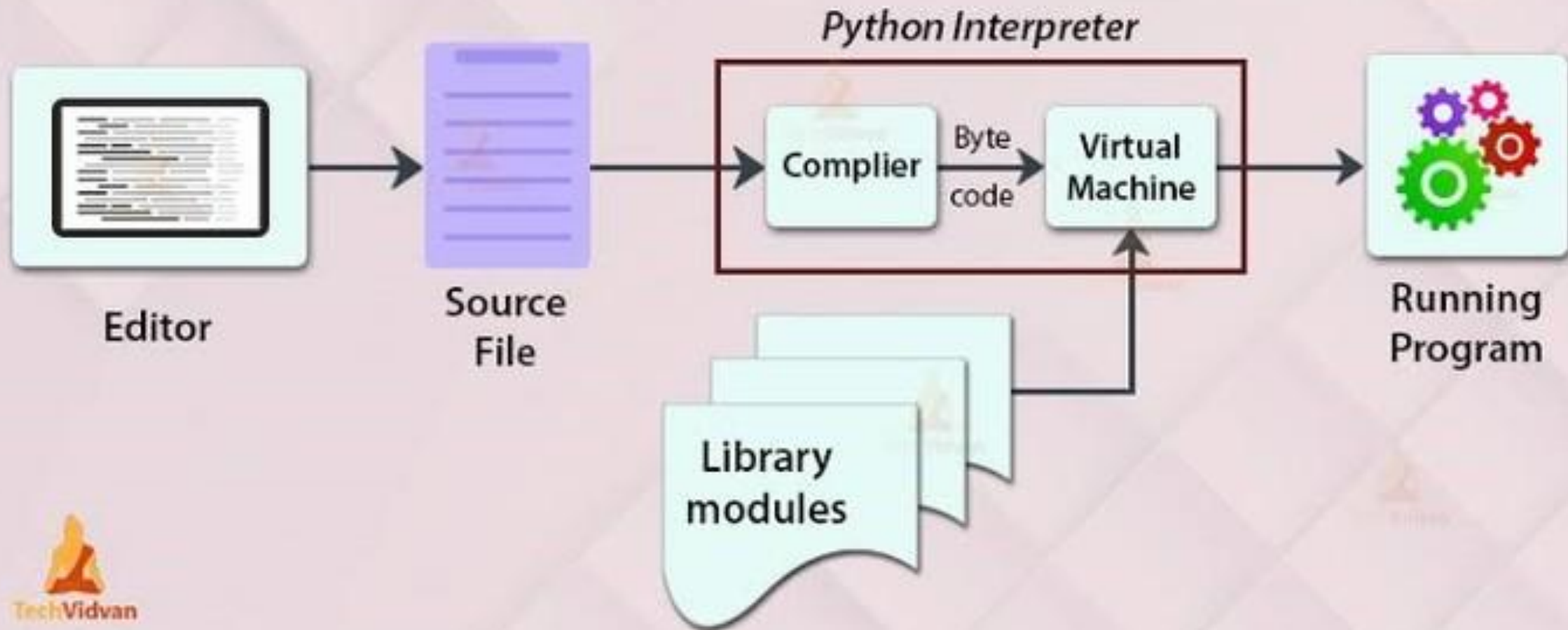
- 한 줄씩 통역해서 전달
- 파이썬

### 컴파일러(Compiler) 또는 번역기(Translator)

- 전체를 번역해서 전달
- C, Java 등

# 파이썬 실행구조

## How Python Interpreter Works?



# 파이썬 프로그램을 실행하는 방법

## 파이썬 셸(shell) 사용

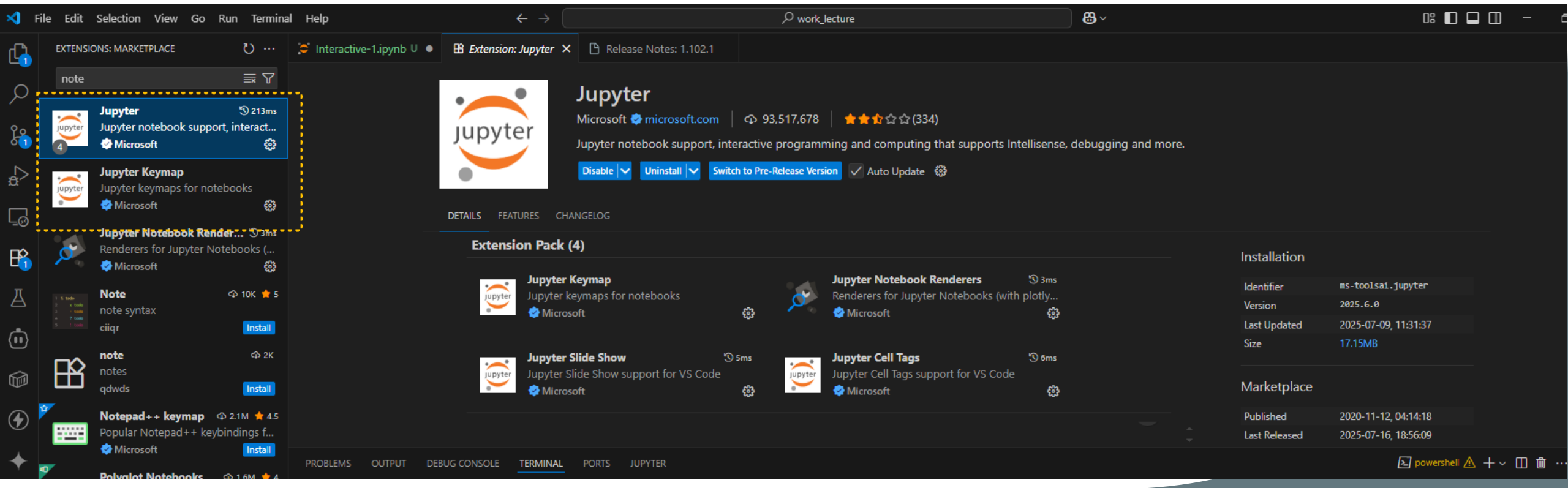
- 한 문장 단위로 파이썬 인터프리터가 처리

## 파이썬 코드 파일을 실행

- 텍스트 에디터를 이용해서 파이썬 코드를 작성해서 파일로 저장
- 파이썬 인터프리터로 하여금 코드 파일을 읽어서 한 문장씩 실행시키도록 함

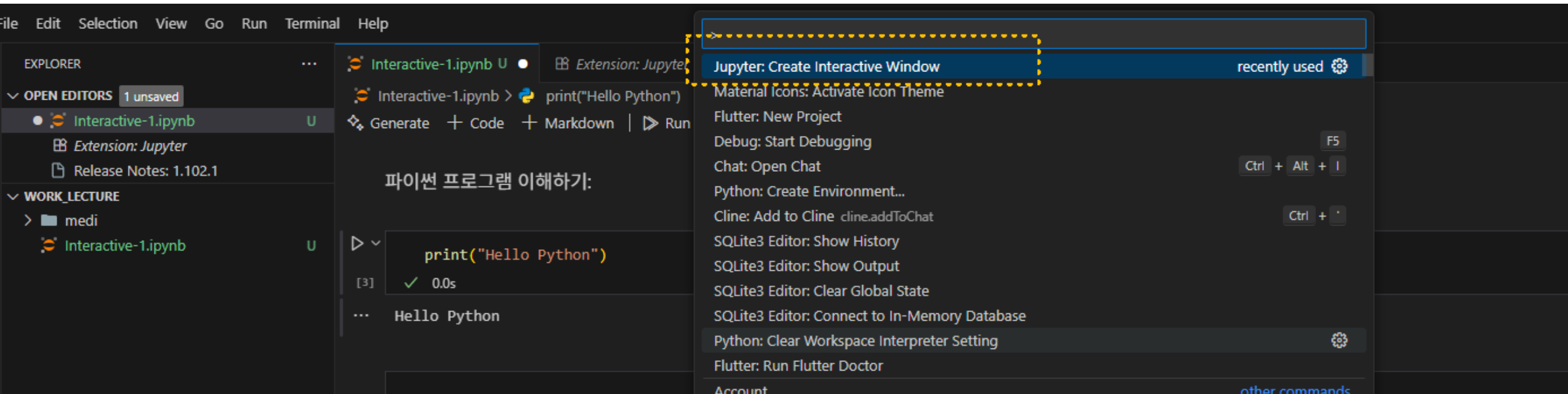
# VSCods 플러그인 추가

- Jupyter notebook 플러그인을 설치한다.
- JupyterLab은 노트북, 코드 및 데이터를 위한 최신 웹 기반 인터랙티브 개발 환경입니다. 유연한 인터페이스를 통해 사용자는 데이터 과학, 과학 컴퓨팅, 계산 저널리즘, 머신러닝 분야의 워크플로를 구성하고 조정할 수 있다.
- 모듈식 디자인은 확장 기능을 통해 기능을 확장하고 강화할 수 있도록 지원한다. (<https://jupyter.org/install>)



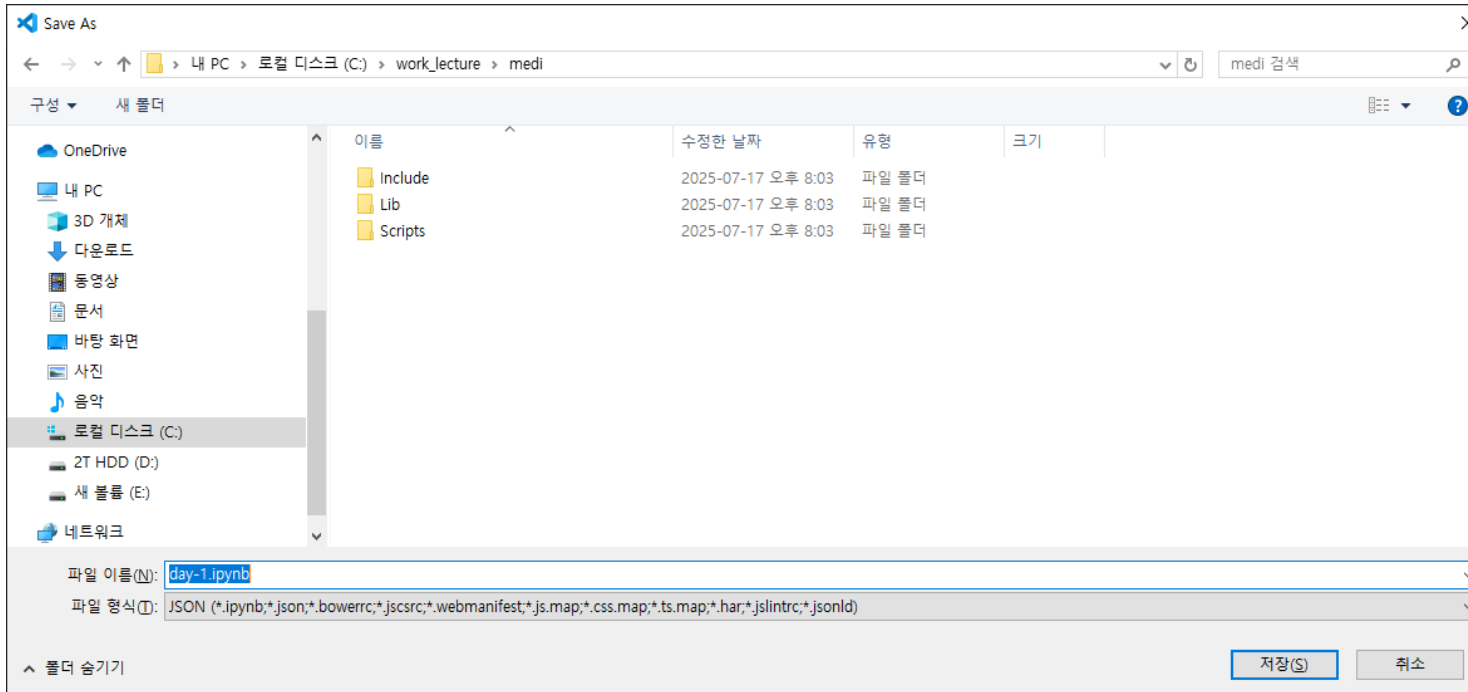
# Notebook 생성

- ctrl+shift + p 클릭
- 쥬피터 윈도우 생성

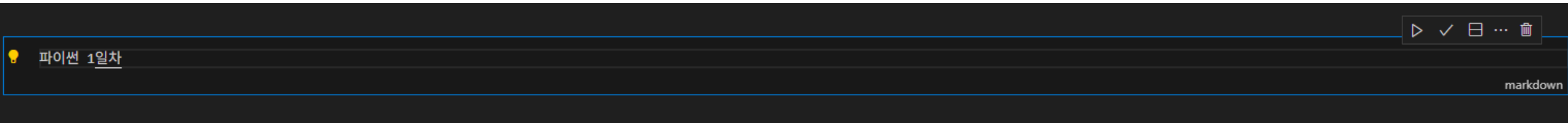


# Notebook 생성

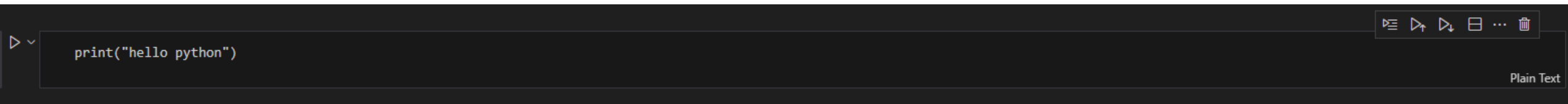
- 가상환경 medi 폴더 안에 day-1.ipynb 로 생성하자.



- Markdown: 주석 역할을 한다.



- Code : 파이썬 코드를 작성한다.

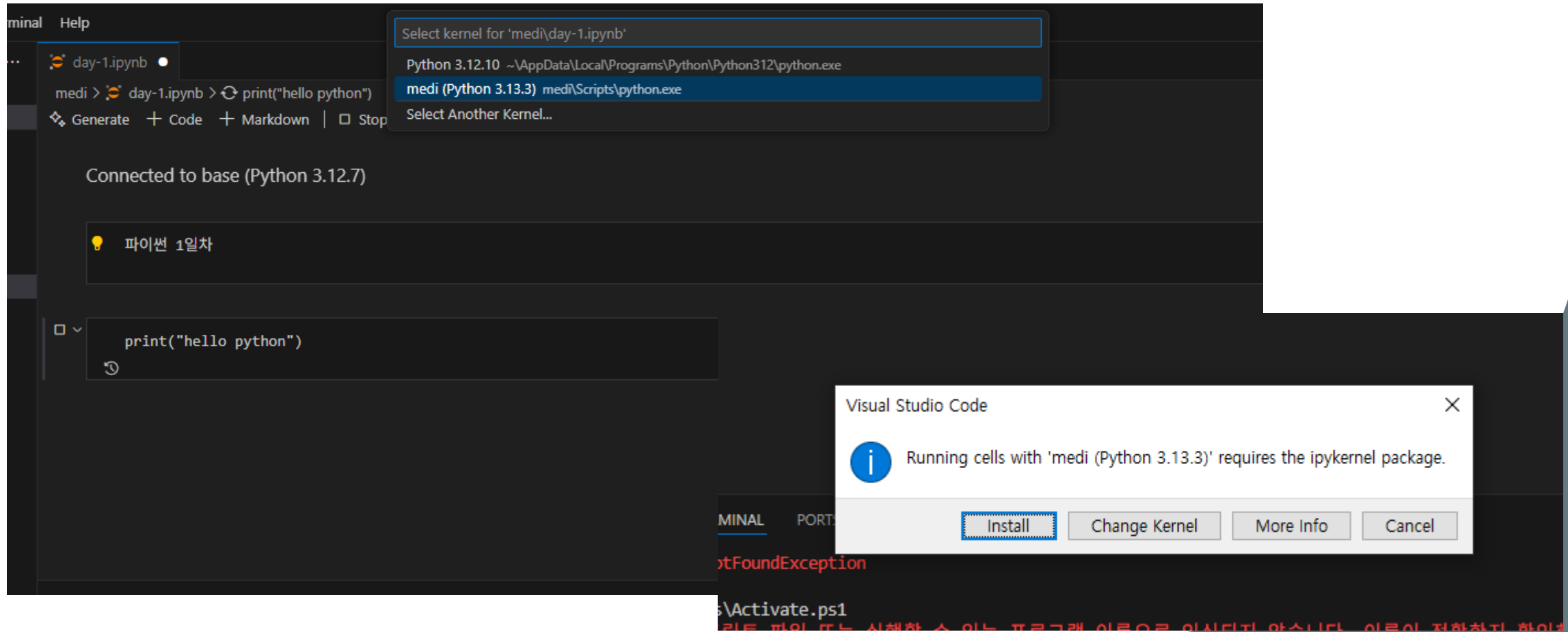


- 실행(ctrl + enter)시:

(파이썬 버전이 선택이 안되 있다면, 자신의 컴퓨터에 설치 파이썬 버전을 선택하면 된다.

- 실행(ctrl + enter)시:

(파이썬 버전이 선택이 안되 있다면, 자신의 컴퓨터에 설치 파이썬 버전을 선택하면 된다.



- 실행 결과:

A screenshot of a Jupyter Notebook cell showing the execution of a Python code snippet. The code is `print("hello python")`. The output area shows a green checkmark, the text `[1]`, and the execution time `0.0s`. Below this, the output of the print statement is displayed as `... hello python`.

```
print("hello python")
```

[1] ✓ 0.0s

... hello python

# 파이썬 활용

- `print("Hello Python")`
- `print()`는 파이썬이 제공하는 명령
- 어떤 작업을 컴퓨터로 하여금 수행할 수 있도록 만들어진 코드의 묶음
  - `print()` 명령은 주어진(전달된) 내용을 화면에 출력한다.
  - 명령에 전달되는 내용을 전달인자(argument) 혹은 인 자라고 함

# 파이썬(Rule)

- 인코딩

문자를 컴퓨터가 어떻게 저장하는지를 나타냄

예: 우리가 'A'라고 보는 것은 실제로 컴퓨터에서는 65라고 저장될 가능성이 높음

- 디코딩

숫자로 저장되어 있는 글자를 우리가 인식할 수 있는 문자로 변환

- 인코딩 방식은 다양하게 있음

- 파이썬 코드

UTF-8 형식으로 저장

특히 코드에 한글이 들어있을 때 조심해야 함

# 파이썬(Rule)

- 문자열

문자들의 조합 또는 나열

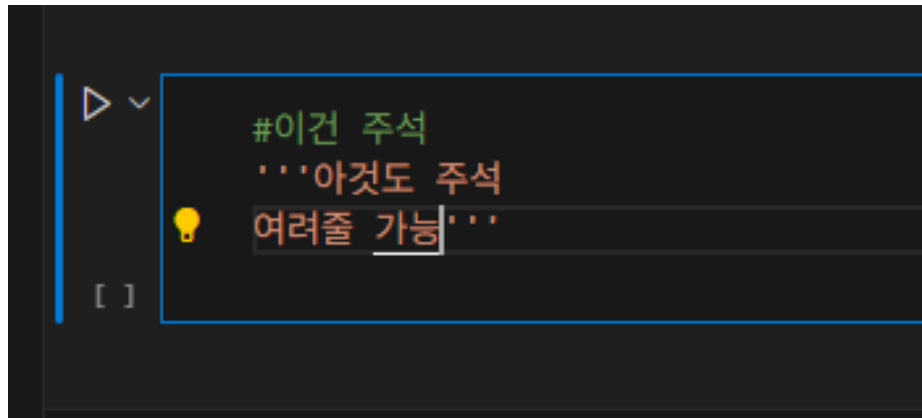
파이썬에서는 "..."와 '...'를 구별 안 함

Print("hi all")

Print('hi all')

# 파이썬(Rule)

- 코드에 코멘트가 필요한 경우, 주석을 사용한다.
- 파이썬은 '#' 또는 ''' 내용 ''' 으로 주석 처리를 할 수 있다.

A screenshot of a code editor with a dark background. It shows three lines of Python code. The first line is a single-line comment starting with '#'. The second line is a multi-line comment starting with '''' and ending with '''. The third line is also a multi-line comment, with the word '가능' underlined. On the left side of the editor, there is a vertical blue bar and a small icon of a lightbulb.

```
#이건 주석  
'''아것도 주석  
여러줄 가능'''
```

# 파이썬(Rule)

예러:

```
print("dddd)
```

[2]   0.0s

... Cell In[2], line 1

```
print("dddd)
```

^

**SyntaxError:** unterminated string literal (detected at line 1)

# 파이썬(Rule)

변수 (Variable): 가장 기본적인 데이터 상자

- 변수는 데이터를 저장하는 가장 기본적인 단위이다. 특정 데이터에 사람이 알아보기 쉬운 '이름표'를 붙여주는 것과 같다. 이 이름표(변수명)를 통해 데이터에 쉽게 접근하고 사용할 수 있다.
- 선언 방법: 변수이름 = 데이터
- 변수 이름은 보통 의미를 알 수 있도록 짓는다. (예: age, patient\_name)

# 파이썬(Rule)

## 자료형 (Data Type): 데이터의 종류

- 변수에 담기는 데이터는 각기 다른 종류(타입)를 가진다. 파이썬은 이 종류를 자동으로 인식한다.
- **정수형 (int)**: 소수점이 없는 숫자. 나이, 환자 수, 약의 개수 등을 나타낸다.  
예: `patient_age = 58`
- **실수형 (float)**: 소수점이 있는 숫자. 체온, 몸무게, BMI 지수, 약물 용량 등 정밀한 수치를 나타낸다.  
예: `body_temperature = 36.5`
- **문자열 (str)**: 글자들의 나열. 환자 이름, 진단명, 주소 등 텍스트 데이터를 나타낸다. 작은따옴표(')나 큰따옴표(")로 감싸야 한다.  
예: `patient_name = "홍길동"`
- **불리언 (bool)**: 참(True) 또는 거짓(False) 두 가지 상태만을 나타낸다. 흡연 여부, 알레르기 유무 등 '예/아니오'로 답할 수 있는 상태를 저장한다.  
예: `is_smoker = True`

# 파이썬(Rule)

자료구조 (Data Structure): 여러 데이터를 담는 그릇

하나가 아닌 여러 개의 데이터를 묶어서 관리해야 할 때 자료구조를 사용한다.

파이썬 자료구조는 데이터를 효율적으로 저장하고 관리하는 데 필수적입니다. 자료구조를 잘 이해하고 문제에 맞게 선택하면 프로그램의 성능을 향상시키고, 복잡한 문제를 더 쉽게 해결할 수 있습니다. 파이썬에서 제공하는 다양한 자료구조 (리스트, 튜플, 딕셔너리, 세트 등)의 특성을 파악하고 활용하는 것이 중요합니다.

# 파이썬(Rule)

## 자료구조의 중요성:

- **효율적인 데이터 관리:**

- 자료구조는 데이터를 저장하고 검색하는 방식을 정의하여 프로그램의 성능을 최적화합니다. 예를 들어, 딕셔너리는 키-값 쌍을 사용하여 빠른 검색을 지원하며, 리스트는 순서대로 데이터를 저장하고 인덱스로 접근할 수 있게 합니다.

- **프로그램 성능 향상:**

- 적절한 자료구조를 선택하면 프로그램 실행 속도를 높이고 메모리 사용량을 줄일 수 있습니다. 예를 들어, 많은 양의 데이터를 처리할 때 해시 테이블 기반의 자료구조를 사용하면 검색 시간을 단축할 수 있습니다.

- **문제 해결 능력 향상:**

- 다양한 자료구조를 활용하면 복잡한 문제를 더 쉽게 해결할 수 있습니다. 예를 들어, 그래프 관련 문제를 풀 때 그래프 자료구조를 사용하면 효율적인 알고리즘을 적용할 수 있습니다.

- **코드 가독성 및 유지보수 용이성:**

- 적절한 자료구조를 사용하면 코드를 더 명확하게 만들고 유지보수를 용이하게 할 수 있습니다. 예를 들어, 데이터를 그룹화하여 관리할 때 클래스를 활용하여 자료구조를 정의하면 코드를 더 구조화할 수 있습니다.

# 파이썬(Rule)

- 리스트 (List []):

특징: 순서가 있는 데이터의 목록이며, 생성 후에도 내용을 수정, 추가, 삭제할 수 있다. (mutable)

용도: 여러 환자의 나이 목록, 한 환자의 시간대별 혈압 기록처럼 순서가 중요하고 내용이 바뀔 수 있는 데이터를 다룰 때 가장 널리 사용된다.

인덱싱(Indexing): `my_list[0]` (첫 번째 항목), `my_list[-1]` (마지막 항목) 처럼 순서(번호)로 특정 항목에 접근한다. (번호는 0부터 시작)

슬라이싱(Slicing): `my_list[1:4]` 처럼 특정 범위를 잘라내어 새로운 리스트를 만든다.

# 파이썬 핵심(Rule)

- 리스트 (List []):

[기초] 리스트 생성 및 길이 확인

- 어느 병동의 환자 5명의 나이를 담고 있는 patient\_ages 리스트를 만드시오. 나이는 각각 34, 52, 28, 65, 41 이다. 그 후, 이 리스트에 총 몇 명의 환자가 있는지 출력하시오.

```
▶ patient_ages = [34, 52, 28, 65, 41]
  number_of_patients = len(patient_ages)

  print(f"현재 병동의 환자 수: {number_of_patients}명")
[3] ✓ 0.0s
... 현재 병동의 환자 수: 5명
```

# 파이썬 핵심(Rule)

- 딕셔너리 (Dictionary {}):

특징: 'Key-Value' 쌍으로 데이터를 저장한다. 순서보다는 각 데이터가 무엇을 의미하는지 나타내는 'Key(이름표)'가 중요하다.  
내용 수정, 추가, 삭제가 가능하다. (mutable)

용도: 한 환자의 차트처럼 '이름': '홍길동', '나이': 58 과 같이 구조화된 정보를 관리하는 데 최적이다.

접근: `my_dict['Key']` 처럼 Key를 통해 Value에 접근한다.

# 파이썬 핵심 (Rule)

- 딕셔너리 (Dictionary {}):

## [기초] 약물 정보 딕셔너리 만들기

- 'Aspirin'이라는 약에 대한 정보를 담은 medication\_info 딕셔너리를 만드시오. 정보는 '용도(purpose)': '해열진통제', '성분(ingredient)': '아세틸살리실산' 이다. 그 후, '용도'를 Key로 사용하여 값을 출력하시오.

```
medication_info = {
    'purpose': '해열진통제',
    'ingredient': '아세틸살리실산'
}

purpose_of_med = medication_info['purpose']
print(f"아스피린의 용도: {purpose_of_med}")
```

[4] ✓ 0.0s

... 아스피린의 용도: 해열진통제

# 파이썬 핵심 (Rule)

- 튜플 (Tuple ( )):

특징: 리스트와 유사하게 순서가 있는 목록이지만, 한 번 만들면 내용을 절대 수정할 수 없다. (immutable)

용도: 환자의 주민등록번호나 병록번호, 유전자 염기 서열처럼 프로그램 실행 중에 절대 바뀌어서는 안 되는 중요한 데이터를 안전하게 보관할 때 사용한다.

# 파이썬 핵심 (Rule)

- 튜플 (Tuple ()):

## [기초] 검체 정보 저장하기

- 검체의 고유 ID('S001')와 채취 시간('09:30')을 sample\_info 라는 튜플로 만드시오. 그 후, 이 튜플의 두 번째 항목(채취 시간)을 출력하시오.

```
▶ sample_info = ('S001', '09:30')
  collection_time = sample_info[1]

  print(f"검체 채취 시간: {collection_time}")
[5] ✓ 0.0s
... 검체 채취 시간: 09:30
```

# 파이썬 핵심 (Rule)

- 셋 (Set {}):

특징: 중복된 데이터를 허용하지 않는 데이터의 묶음이다. 순서의 개념이 없다.

용도: 특정 목록에서 중복된 항목을 모두 제거하고, 유일한 값들만 추려내고 싶을 때 매우 유용하다. (예: 한 병동에서 처방된 모든 약의 '종류'를 알아낼 때)

# 파이썬 핵심 (Rule)

- 셋 (Set {}):

## [기초] 알레르기 유발 항원 종류 확인하기

- 어떤 환자가 여러 검사에서 양성 반응을 보인 알레르기 항원 목록 allergens = ['집먼지진드기', '고양이털', '계란흰자', '집먼지진드기']가 있다. set을 사용하여 중복을 제거하고, 환자가 총 몇 종류의 항원에 알레르기가 있는지 출력하시오.

```
▶ allergens = ['집먼지진드기', '고양이털', '계란흰자', '집먼지진드기']
unique_allergens = set(allergens)
num_of_allergens = len(unique_allergens)

print(f"환자의 알레르기 항원 종류 수: {num_of_allergens}종")
print(f"알레르기 항원 목록: {unique_allergens}")
```

[6] ✓ 0.0s

... 환자의 알레르기 항원 종류 수: 3종  
알레르기 항원 목록: {'고양이털', '계란흰자', '집먼지진드기'}

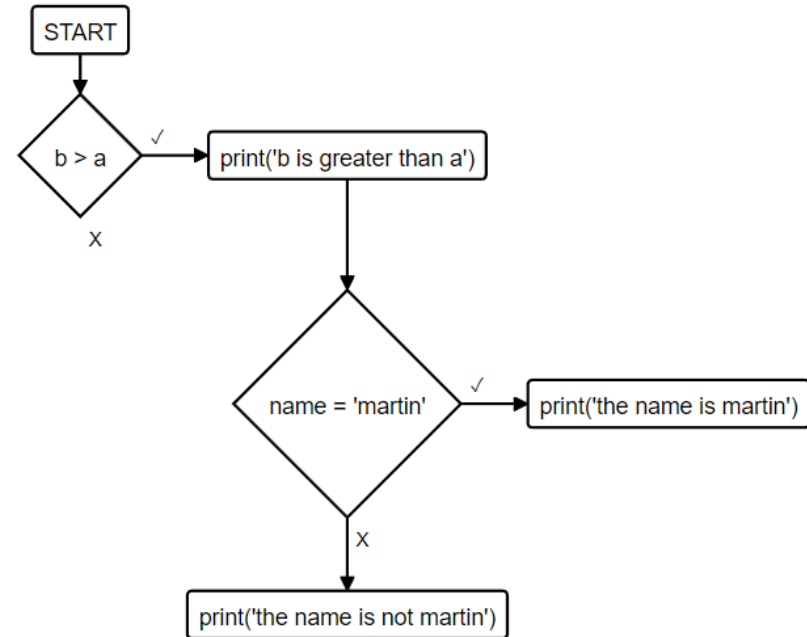
## 4. 파이썬 핵심 문법 익히기 (2) - 작업 시키기 (조건,반복,함수,클래스)



# 파이썬 핵심(Rule)

## [개념 정리]

- 데이터를 다양한 그릇에 담았다면, 이제 그 데이터를 활용하여 컴퓨터에게 의미 있는 '작업'을 시켜야 한다. 이를 위해 프로그램의 논리적 흐름을 제어하는 문법들을 배운다.



# 파이썬 핵심(Rule)

## 조건문 (Conditional Statements): 논리적 분기 만들기

- 조건문은 특정 조건의 참(True) 또는 거짓(False) 여부에 따라 프로그램의 실행 경로를 나누는, 의사결정의 핵심이다. 이는 마치 진료 가이드라인처럼 "만약(if) A 조건이면 B를 실행하고, 그게 아니면(else) C를 실행하라"는 논리적 분기점을 만드는 것과 같다.
- **if**: 가장 기본적인 조건문. if 조건: 뒤의 코드 블록은 조건이 True일 때만 실행된다.
- **elif**: else if의 줄임말. 이전 if나 elif 조건이 False일 때, 새로운 조건을 검사한다. 여러 조건을 순차적으로 검사할 때 사용한다.
- **else**: 위의 모든 if, elif 조건이 False일 때 실행된다. 선택 사항이며, 항상 맨 마지막에 위치한다.

**중요:** if, elif, else 뒤에는 반드시 콜론(:)을 찍어야 하며, 그에 속한 코드 블록은 반드시, 들여쓰기(indentation)를 해야 한다.

# 파이썬 핵심(Rule)

- [기초] 체온에 따른 상태 분류
- 환자의 체온 body\_temp 변수가 주어졌을 때, 37.5도 이상이면 "주의 필요", 그 외에는 "정상"을 출력하는 코드를 작성하시오.

조건 : body\_temp = 37.8

```
▶ body_temp = 37.8

if body_temp >= 37.5:
    print("주의 필요")
else:
    print("정상")

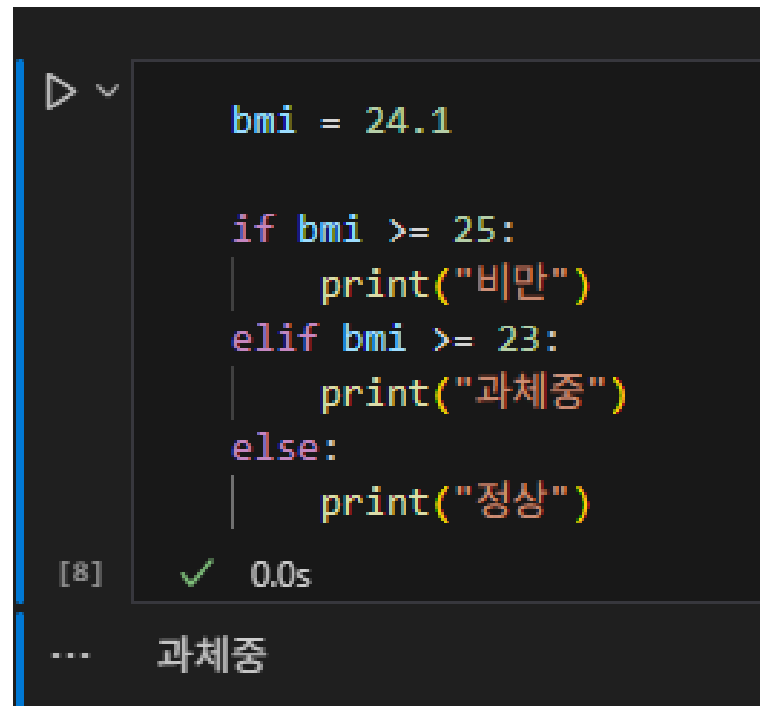
[7] ✓ 0.0s

... 주의 필요
```

# 파이썬 핵심(Rule)

- [기초] 비만도 단계별 분류
- BMI 지수 bmi 변수가 주어졌을 때, 25 이상이면 "비만", 23 이상 25 미만이면 "과체중", 그 외에는 "정상"을 출력하는 코드를 if, elif, else를 모두 사용하여 작성하시오.

bmi = 24.1

A screenshot of a Python code editor showing a script that checks a BMI value. The code defines a variable 'bmi' as 24.1 and uses an if-elif-else structure to print '비만' (Obese), '과체중' (Overweight), or '정상' (Normal). The execution output at the bottom shows '과체중' (Overweight) as the result.

```
bmi = 24.1

if bmi >= 25:
    print("비만")
elif bmi >= 23:
    print("과체중")
else:
    print("정상")
```

[8] ✓ 0.0s

... 과체중

# 파이썬 핵심(Rule)

## 반복문 (Loops): 자동화의 시작

반복문은 특정 작업을 여러 번 자동으로 수행하게 한다. 수백 명의 환자 데이터를 일일이 처리하는 대신, 반복문 하나로 모든 작업을 완료할 수 있다.

- **for 반복문:**

**특징:** 리스트, 튜플, 문자열 등 '반복 가능한(iterable)' 객체의 항목을 처음부터 끝까지 하나씩 순회하며 작업을 수행한다. 반복 횟수가 정해져 있을 때 주로 사용한다.

**활용:**

- `for item in my_list::` 리스트의 항목을 하나씩 꺼내 item 변수에 담는다.
- `for key, value in my_dict.items():` 딕셔너리의 키와 값을 동시에 꺼낸다.
- `for i in range(5):` 0부터 4까지, 총 5번 반복한다.

- **while 반복문:**

**특징:** 특정 조건이 True인 동안 계속해서 작업을 반복한다. 반복 횟수가 정해져 있지 않고, 특정 조건을 만족할 때까지 계속해야 할 때 사용한다.

**주의:** while문의 조건이 영원히 False가 되지 않으면 프로그램이 멈추지 않는 '무한 루프'에 빠질 수 있으므로, 루프 내에서 조건을 변경하는 코드가 반드시 필요하다.

# 파이썬 핵심(Rule)

## [기초] 환자 이름 목록 출력

- 입원 환자 이름이 담긴 patient\_names 리스트가 있다. for문을 사용하여 각 환자의 이름을 한 줄에 한 명씩 출력하시오.

```
patient_names = ['김민준', '이서연', '박도윤', '최아린']
```

```
▶ patient_names = ['김민준', '이서연', '박도윤', '최아린']

print("--- 입원 환자 명단 ---")
for name in patient_names:
    print(name)

[9] ✓ 0.0s

... --- 입원 환자 명단 ---
    김민준
    이서연
    박도윤
    최아린
```

# 파이썬 핵심(Rule)

- [기초] 특정 조건의 환자만 찾기
- 여러 환자의 나이가 담긴 `age_list = [28, 67, 45, 71, 34, 82]`가 있다. `for`문과 `if`문을 함께 사용하여, 나이가 65세 이상인 '고령' 환자의 나이만 출력하시오.

```
▶ ~  
    age_list = [28, 67, 45, 71, 34, 82]  
  
    print("--- 고령 환자 목록 ---")  
    for age in age_list:  
        if age >= 65:  
            print(f"{age}세")  
[10] ✓ 0.0s  
... --- 고령 환자 목록 ---  
    67세  
    71세  
    82세
```

# 파이썬 핵심(Rule)

- [기초] 카운트다운
- count 변수를 5로 시작하여, while문을 사용해 1씩 감소시키면서 숫자를 출력하고, 마지막에 "발사!"를 출력하는 카운트다운 프로그램을 작성하시오.

```
count = 5

while count > 0:
    print(count)
    count -= 1 # count = count - 1 과 같은 의미

print("발사!")
```

[11] ✓ 0.0s

... 5  
4  
3  
2  
1  
발사!

# 파이썬 핵심(Rule)

- [기초] 약물 투여 시뮬레이션
- 환자에게 매 시간 10mg의 약물을 투여하고, 환자의 신체는 시간당 8mg의 약물을 대사(제거)한다고 가정하자. 체내 약물 총량이 목표치인 50mg 이상이 될 때까지 몇 시간이 걸리는지 while문을 사용하여 계산하고, 최종 약물 총량과 걸린 시간을 출력하시오.

```
total_dose = 0 # 초기 체내 약물 총량
hours = 0      # 경과 시간
target_dose = 50 # 목표 총량

while total_dose < target_dose:
    hours += 1
    total_dose += 10 # 10mg 투여
    total_dose -= 8  # 8mg 대사
    print(f"{hours}시간 후, 체내 총량: {total_dose}mg")

print(f"\n목표치 도달! 총 {hours}시간 소요.")
print(f"최종 체내 약물 총량: {total_dose}mg")

# 이 코드의 결과는 25시간이 걸리고 최종 총량은 50mg가 됩니다.
# 24시간 후 -> 48mg. 아직 목표 미달.
# 25시간 후 -> 48 + 10 - 8 = 50mg. 목표 도달 후 루프 종료.
```

✓ 0.0s

1시간 후, 체내 총량: 2mg  
2시간 후, 체내 총량: 4mg  
3시간 후, 체내 총량: 6mg  
4시간 후, 체내 총량: 8mg  
5시간 후, 체내 총량: 10mg  
6시간 후, 체내 총량: 12mg  
7시간 후, 체내 총량: 14mg  
8시간 후, 체내 총량: 16mg  
9시간 후, 체내 총량: 18mg  
10시간 후, 체내 총량: 20mg  
11시간 후, 체내 총량: 22mg  
12시간 후, 체내 총량: 24mg

13시간 후, 체내 총량: 26mg  
14시간 후, 체내 총량: 28mg  
15시간 후, 체내 총량: 30mg  
16시간 후, 체내 총량: 32mg  
17시간 후, 체내 총량: 34mg  
18시간 후, 체내 총량: 36mg  
19시간 후, 체내 총량: 38mg  
20시간 후, 체내 총량: 40mg  
21시간 후, 체내 총량: 42mg  
22시간 후, 체내 총량: 44mg  
23시간 후, 체내 총량: 46mg  
24시간 후, 체내 총량: 48mg  
25시간 후, 체내 총량: 50mg

목표치 도달! 총 25시간 소요.  
최종 체내 약물 총량: 50mg

# 파이썬 핵심(Rule)

## 함수 (Function): 코드 레시피 만들기

- 함수는 특정 기능을 수행하는 코드 묶임에 이름을 붙여 '재사용'할 수 있게 만든 것이다. 'BMI 계산'처럼 자주 사용하는 기능을 함수로 만들어 두면, 필요할 때마다 함수 이름만 호출하여 간편하게 사용할 수 있다. 코드가 간결해지고 유지보수가 쉬워진다.
- 정의: `def 함수이름(매개변수1, 매개변수2):`
- 매개변수 (Parameter): 함수에 전달하는 '입력 값'. 함수를 호출할 때 이 값을 넘겨준다.
- 반환값 (Return): 함수가 모든 작업을 마친 후 되돌려주는 '결과물'. `return` 키워드를 사용한다. `return`을 만나면 함수는 즉시 종료된다.

# 파이썬 핵심(Rule)

## [기초] 간단한 인사말 함수

- 환자의 이름을 입력받아 "안녕하세요, [이름]님. 진료실로 들어오세요." 라고 출력하는 greet\_patient() 함수를 작성하고, '이하은'이라는 이름으로 함수를 호출하시오.

```
def greet_patient(name):  
    """환자의 이름을 받아 인사말을 출력하는 함수"""  
    print(f"안녕하세요, {name}님. 진료실로 들어오세요.")  
  
    # 함수 호출  
    greet_patient('이하은')  
[13]  ✓  0.0s  
...  안녕하세요, 이하은님. 진료실로 들어오세요.
```

# 파이썬 핵심(Rule)

## [기초] 섭씨-화씨 변환 함수

섭씨온도(Celsius)를 입력받아 화씨온도(Fahrenheit)로 변환하여 반환(return)하는 celsius\_to\_fahrenheit() 함수를 작성하시오. 변환 공식은 (섭씨온도 \* 9/5) + 32 이다. 섭씨 37도를 화씨로 변환하여 그 결과를 출력하시오.

```
def celsius_to_fahrenheit(celsius):  
    """섭씨온도를 화씨온도로 변환하여 반환하는 함수"""  
    fahrenheit = (celsius * 9/5) + 32  
    return fahrenheit  
  
# 함수 호출 및 반환값 저장  
f_temp = celsius_to_fahrenheit(37)  
print(f"섭씨 37도는 화씨 {f_temp:.2f}도 입니다.") # 소수점 둘째 자리까지 출력
```

[14] ✓ 0.0s

... 섭씨 37도는 화씨 98.60도 입니다.

# 파이썬 핵심(Rule)

클래스 (Class): 나만의 데이터 타입을 설계하기

- 클래스는 파이썬 객체 지향 프로그래밍의 핵심이다. 이는 관련된 데이터(변수)와 기능(함수)을 하나로 묶어 새로운 '데이터 타입'을 설계하는 '설계도'와 같다.
- 개념: '환자'라는 클래스(설계도)를 만든다고 생각해보자.
  - 속성 (Attributes): 환자가 가지는 데이터. (예: 이름, 나이, 병록번호)
  - 메서드 (Methods): 해당 환자가 할 수 있는 행동. (예: 자기소개하기, 증상설명하기)
- 객체 (Object) / 인스턴스 (Instance): 클래스라는 설계도를 바탕으로 실제로 만들어낸 '실체'이다. '환자' 설계도로 '김민준 환자', '이서연 환자' 등 실제 환자 객체를 만드는 것과 같다.
- `__init__(self, ...)`: 클래스로 객체를 만들 때, 가장 먼저 호출되는 특별한 메서드. '초기화 메서드' 또는 '생성자'라고 부른다. 객체가 가져야 할 초기 속성들을 여기서 설정한다. `self`는 만들어지는 객체 자기 자신을 의미한다.

# 파이썬 핵심(Rule)

## [기초] 환자(Patient) 클래스 정의하기

- 환자를 나타내는 Patient 클래스를 정의하시오. 이 클래스는 생성될 때 환자의 이름(name)과 병록번호(chart\_number)를 속성으로 가져야 한다. 이 설계도를 바탕으로, 이름이 '강지우', 병록번호가 'C00128'인 환자 객체 p1을 생성하고, 객체의 이름과 병록번호를 출력하시오.

```
class Patient:
    # 객체가 생성될 때 호출되는 초기화 메서드
    def __init__(self, name, chart_number):
        self.name = name
        self.chart_number = chart_number

# Patient 클래스로 p1 객체를 생성
p1 = Patient(name='강지우', chart_number='C00128')

# 객체의 속성에 접근하여 출력
print(f"환자 이름: {p1.name}")
print(f"병록번호: {p1.chart_number}")
```

[15] ✓ 0.0s

... 환자 이름: 강지우  
병록번호: C00128

# 파이썬 핵심(Rule)

## [기초] 기능(메서드) 추가하기

- 문제 1에서 만든 Patient 클래스에, 환자의 체온을 업데이트하는 update\_temp() 메서드를 추가하시오. 이 메서드는 새로운 체온을 입력받아 객체의 temperature 속성에 저장해야 한다. p1 객체에 update\_temp(37.6)을 호출하여 체온을 업데이트하고, p1.temperature 속성을 출력하여 확인하시오.

```
class Patient:
    def __init__(self, name, chart_number):
        self.name = name
        self.chart_number = chart_number
        self.temperature = None # 초기 체온은 측정 전이므로 None으로 설정

    # 체온을 업데이트하는 메서드
    def update_temp(self, new_temp):
        self.temperature = new_temp
        print(f"{self.name} 환자의 체온이 {self.temperature}도로 업데이트되었습니다.")

# p1 객체 생성
p1 = Patient(name='강지우', chart_number='C00128')

# 체온 업데이트 메서드 호출
p1.update_temp(37.6)

# 업데이트된 속성 확인
print(f"현재 체온: {p1.temperature}도")
```

[16] ✓ 0.0s

... 강지우 환자의 체온이 37.6도로 업데이트되었습니다.  
현재 체온: 37.6도

## 5. Pandas로 엑셀 작업 자동화하기



# Pandas

개념: Pandas 소개 및 활용 방법

## Pandas란 무엇인가?

- Pandas는 파이썬에서 사용하는 데이터 분석 라이브러리입니다. 의학 연구에서 사용하는 거대한 엑셀 파일이나 환자 데이터 목록을 코드 몇 줄로 자유자재로 다룰 수 있게 해주는, "파이썬의 슈퍼-파워 엑셀"이라고 생각하면 이해하기 쉽다.
- 수작업으로 몇 시간이 걸릴 데이터 정리 및 분석 작업을 단 몇 초 만에 완료할 수 있게 해준다.

Pandas



# Pandas

pandas는 데이터 분석 및 조작을 위한 파이썬 라이브러리이다.

모듈 설치에 아래 명령을 통해 진행한다.

pip install pandas

```
(medi) PS C:\work_lecture> pip install pandas
Collecting pandas
  Downloading pandas-2.3.1-cp313-cp313-win_amd64.whl.metadata (19 kB)
Collecting numpy>=1.26.0 (from pandas)
  Downloading numpy-2.3.1-cp313-cp313-win_amd64.whl.metadata (60 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\work_lecture\medi\lib\site-packages (from pandas) (2.9.0.post0)
Collecting pytz>=2020.1 (from pandas)
  Using cached pytz-2025.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Using cached tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in c:\work_lecture\medi\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Downloading pandas-2.3.1-cp313-cp313-win_amd64.whl (11.0 MB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.0/11.0 MB 15.2 MB/s eta 0:00:00
Downloading numpy-2.3.1-cp313-cp313-win_amd64.whl (12.7 MB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.7/12.7 MB 12.7 MB/s eta 0:00:00
Using cached pytz-2025.2-py2.py3-none-any.whl (509 kB)
Using cached tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: pytz, tzdata, numpy, pandas
```

# Pandas

## 핵심 구성 요소

- **DataFrame**: Pandas의 가장 핵심적인 데이터 구조입니다. 엑셀의 '시트(Sheet)'처럼 행(row)과 열(column)으로 구성된 2차원 표입니다. 각 행은 개별 환자 한 명의 정보를, 각 열은 나이, 성별, 혈압 등 특정 변수를 나타냅니다.
- **Series**: DataFrame에서 하나의 열을 떼어낸 것과 같습니다. 즉, 1차원 데이터 목록입니다. 예를 들어, 모든 환자의 '나이' 데이터만 모아놓은 열이 하나의 Series입니다.

# Pandas

## Pandas로 무엇을 할 수 있는가? (활용 방법)

- **데이터 불러오기 및 저장하기:** CSV, 엑셀 등 다양한 형식의 파일을 손쉽게 불러와 DataFrame으로 만들고, 분석이 끝난 데이터를 다시 파일로 저장할 수 있습니다.
- **데이터 탐색 및 요약:** 데이터가 몇 행 몇 열인지, 각 열은 어떤 데이터 타입인지, 기초적인 통계치(평균, 표준편차, 최댓값, 최솟값 등)는 어떠한지 한눈에 파악할 수 있습니다.
- **데이터 선택 및 필터링 (가장 중요!):**
  - `df['나이']`: '나이' 열만 선택하기
  - `df[['나이', '성별']]`: 여러 열을 동시에 선택하기
  - `df[df['나이'] > 65]`: '나이'가 65보다 큰 환자 데이터만 추출하기
  - `df[(df['나이'] > 65) & (df['성별'] == '남')]`: 여러 조건을 동시에 만족하는 데이터 추출하기
- **데이터 정제 및 가공:**
  - 불필요한 열 삭제하기
  - 비어있는 값(결측치)을 찾아서 채우거나 삭제하기
  - 기존 데이터를 이용해 새로운 의미를 가진 열(예: BMI 지수) 추가하기
- **그룹별 분석:** 데이터를 특정 기준(예: 성별, 흡연 여부)으로 묶어서 각 그룹의 통계치를 비교, 분석할 수 있습니다.

# Pandas

실습: pandas 활용

가상의 데이터 만들기



```
import pandas as pd

# 1. 가상의 환자 데이터로 DataFrame 만들기 (딕셔너리 활용)
# 실제로는 pd.read_csv('파일명.csv') 또는 pd.read_excel('파일명.xlsx')로 파일을 불러온다.
data = {
    '환자ID': ['P001', 'P002', 'P003', 'P004', 'P005'],
    '성별': ['남', '여', '여', '남', '여'],
    '나이': [68, 55, 71, 49, 62],
    '수축기혈압': [152, 138, 160, 125, 141],
    '당뇨여부': [True, False, True, False, True]
}
df = pd.DataFrame(data)

print("--- 원본 데이터 ---")
print(df)
```

[17] ✓ 0.5s

...

```
--- 원본 데이터 ---
   환자ID 성별  나이  수축기혈압  당뇨여부
0  P001  남   68    152    True
1  P002  여   55    138   False
2  P003  여   71    160    True
3  P004  남   49    125   False
4  P005  여   62    141    True
```

# Pandas

데이터 필터링하기 : 65 세 이상 고령, 고위험군 환자

```
# 2. 특정 조건으로 데이터 필터링하기
# [응용문제 1] 65세 이상인 고령 환자만 추출하기
print("\n--- 65세 이상 고령 환자 ---")
senior_patients = df[df['나이'] >= 65]
print(senior_patients)

# [응용문제 2] 당뇨가 있으면서 수축기 혈압이 140 이상인 '고위험군' 환자 찾기
print("\n--- 고위험군 환자 (당뇨 0, SBP >= 140) ---")
high_risk_group = df[(df['당뇨여부'] == True) & (df['수축기혈압'] >= 140)]
print(high_risk_group)
```

[18] ✓ 0.0s

...

```
--- 65세 이상 고령 환자 ---
  환자ID  성별  나이  수축기혈압  당뇨여부
0  P001   남   68     152     True
2  P003   여   71     160     True

--- 고위험군 환자 (당뇨 0, SBP >= 140) ---
  환자ID  성별  나이  수축기혈압  당뇨여부
0  P001   남   68     152     True
2  P003   여   71     160     True
4  P005   여   62     141     True
```

# Pandas

새로운 열 추가하기



```
# 3. 새로운 계산 열 추가하기
# 나이를 기준으로 '연령대' 열 추가하기 (예: 60대, 50대)
df['연령대'] = (df['나이'] // 10) * 10
print("\n--- '연령대' 열 추가 후 ---")
print(df)
```

[20]

✓ 0.0s

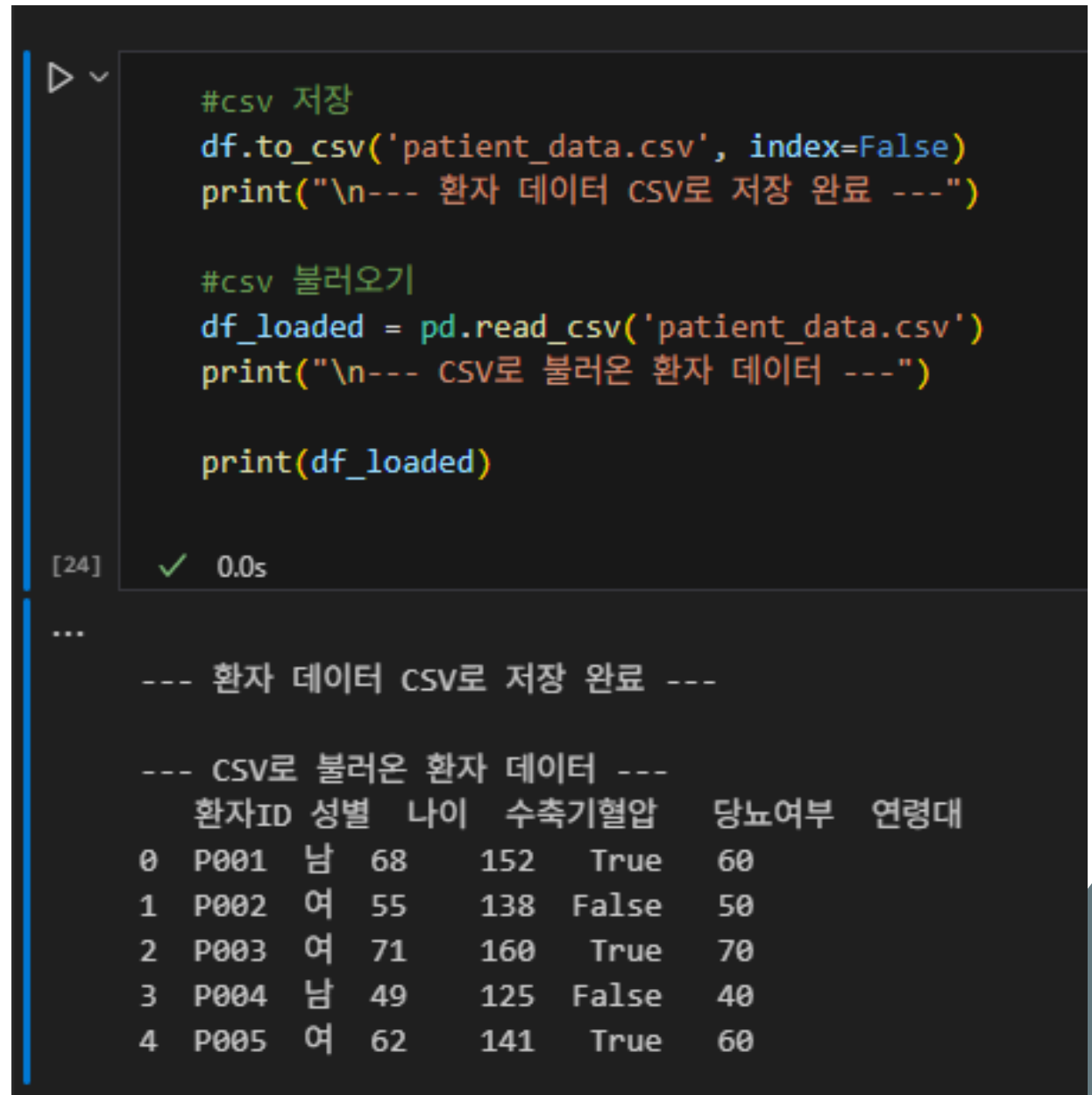
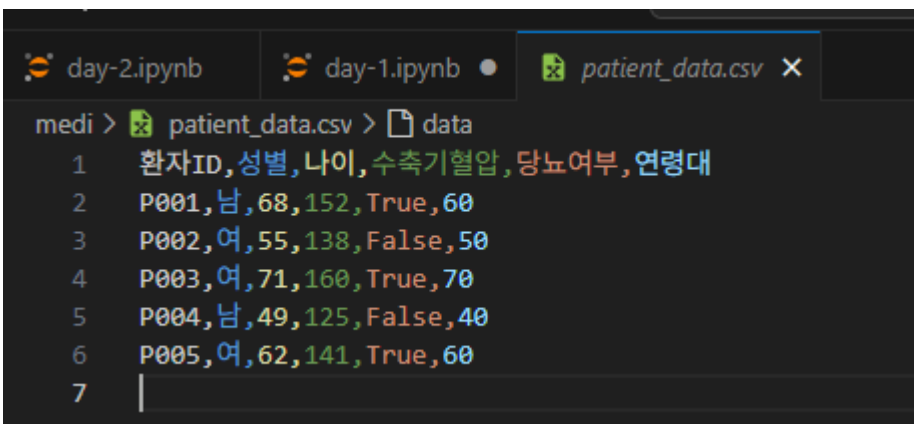
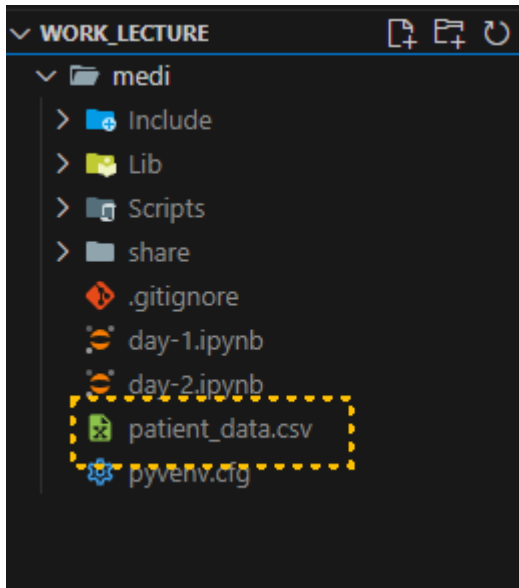
...

--- '연령대' 열 추가 후 ---

	환자ID	성별	나이	수축기혈압	당뇨여부	연령대
0	P001	남	68	152	True	60
1	P002	여	55	138	False	50
2	P003	여	71	160	True	70
3	P004	남	49	125	False	40
4	P005	여	62	141	True	60

# Pandas

엑셀 csv 데이터 저장 / 불러오기



# 헬스케어 데이터 분석 기초 및 비/시계열 데이터 이해



# 1. 헬스케어 데이터의 이해 (Understanding Healthcare Data)

## 1.1. 헬스케어 데이터의 정의와 패러다임의 변화

- 헬스케어 데이터란 개인의 건강 상태, 의료 서비스 이용, 보건 활동과 관련하여 생성되는 모든 종류의 정형 및 비정형 데이터를 총칭한다.
- 과거의 의학이 의사의 경험과 직관에 의존하는 '경험 의학(Empirical Medicine)'이었다면, 현대 의학은 축적된 데이터를 바탕으로 최적의 의사결정을 내리는 '증거 기반 의학(Evidence-Based Medicine)'을 넘어, 개개인의 특성에 맞춘 '정밀 의료(Precision Medicine)'로 나아가고 있다. 이 변화의 핵심 동력이 바로 데이터이다.



# 1. 헬스케어 데이터의 이해 (Understanding Healthcare Data)

## 1.2. 주요 데이터 원천 (Data Sources)

헬스케어 데이터는 생성되는 장소와 주체에 따라 크게 네 가지로 분류된다.

- **임상 데이터 (Clinical Data):** 병원에서 생성되는 데이터이다. 전자의무기록(EMR/EHR)이 대표적이며, 환자의 진단명, 처방 정보, 검사 결과(Lab result), 수술 기록, 입퇴원 기록 등이 포함된다. 의료진의 전문적인 소견이 포함되어 신뢰도가 높으나, 병원마다 기록 방식이 달라 표준화가 어렵다.
- **청구 데이터 (Claims Data):** 병원이 진료비를 건강보험공단이나 심사평가원에 청구하기 위해 생성하는 데이터이다. 전 국민의 의료 이용 내역을 포괄적으로 볼 수 있다는 장점이 있으나, 실제 임상 수치(혈압, 혈당 등)보다는 비용과 진단 코드 위주로 구성되어 있어 임상적 디테일은 부족하다.
- **유전체 데이터 (Genomic Data):** 개인의 DNA 염기서열 정보이다. 질병의 유전적 원인을 규명하고 맞춤형 신약을 개발하는 데 필수적이다. 데이터의 용량이 매우 크고(Big Data), 분석을 위해 고도의 생물정보학적 지식이 요구된다.
- **환자 유래 데이터 (PGHD, Patient Generated Health Data):** 환자가 병원 밖에서 스마트폰, 웨어러블 기기(Apple Watch, Fitbit 등)를 통해 직접 생성하는 데이터이다. 일상생활 속의 활동량, 수면 패턴, 심박수 등을 포함하며, 예방 의학 관점에서 가치가 매우 높다.

# 1. 헬스케어 데이터의 이해 (Understanding Healthcare Data)

## 1.4. 비시계열(Cross-sectional) 데이터와 시계열(Time-series) 데이터

분석의 목적과 데이터 구조에 따라 두 가지로 구분한다.

- **비시계열 데이터 (단면 데이터):** 특정 시점(Snapshot)에서 수집된 데이터이다. 시간의 흐름을 고려하지 않고, '환자 A와 환자 B의 차이'를 비교하는 데 중점을 둔다. 예를 들어, "입원 당시의 혈압이 높은 환자군이 사망률이 높은가?"를 분석할 때 사용한다.
- **시계열 데이터 (종단 데이터):** 한 환자에 대해 시간 순서대로 반복 측정된 데이터이다. '어제의 환자 A와 오늘의 환자 A의 변화'를 추적하는 데 중점을 둔다. 예를 들어, "투약 후 1시간 간격으로 혈당이 어떻게 변하는가?"를 분석할 때 사용한다.

*“먼저 데이터 분석의 기본기를 다지기 위해, 구조가 단순하고 해석이 명확한 비시계열 데이터를 중심으로 학습한다.”*

## 1. Numpy (Numerical Python)

Numpy는 파이썬에서 고성능 수치 계산을 위해 만들어진 라이브러리입니다.

핵심 객체: ndarray (다차원 배열)

주요 특징:

속도: 파이썬 리스트보다 훨씬 빠르고 메모리를 효율적으로 사용합니다.

벡터화 연산: 반복문(for문) 없이 배열 간 연산을 한 번에 처리할 수 있습니다.

용도: 선형 대수, 통계 계산, 푸리에 변환 등 수학적 작업의 기반이 됩니다.

실습 : nb\_1\_1\_env.ipynb

## 2. 데이터 전처리 (Preprocessing)

### 2.1. 전처리의 중요성

- "Garbage In, Garbage Out(GIGO)"은 데이터 분석의 불변의 진리이다.
- 헬스케어 데이터는, 수집되어지는 특수성으로 인해 'Garbage(노이즈, 결측, 오류)'가 필연적으로, 많이 포함되어 있다.
- 이 노이즈를 무엇으로 필터링할수 있을까?
- 바로, 전처리과정을 적용함으로, 데이터의 상당수의 불필요한 노이즈를 감소 시킬수 있다.
- 전처리는 전체 분석 시간의 80% 이상을 차지할 만큼 중요한 단계이며, 임상적 지식(Domain Knowledge)이 가장 많이 요구되는 단계이다.

실습 : nb\_1\_2\_cohort.ipynb



## 2. 데이터 전처리 (Preprocessing)

### 2.2. 결측치 (Missing Value) 처리

의료 데이터에서 결측치는 단순한 공백이 아니다.

- 결측의 유형 파악:

MCAR- Missing Completely At Random (완전 무작위 결측): 전산 오류 등으로 우연히 누락된 경우.

MAR- Missing At Random

(무작위 결측 또는 조건부 무작위 결측): 다른 변수와 관련이 있는 경우 (예: 여성이 남성보다 특정 검사 누락이 많음).

MNAR (비무작위 결측): 결측 자체가 정보를 담고 있는 경우. 예컨대 응급 상황이라 체중 잴 시간이 없어 누락된 경우, '체중 결측=위급 환자'라는 의미가 될 수 있다.

## 2. 데이터 전처리 (Preprocessing)

### 2.2.1. MCAR (Missing Completely At Random)

원어: Missing Completely At Random

- 한국어: 완전 무작위 결측

- 의미:

데이터가 누락된 이유가 데이터의 내용이나 다른 변수와 전혀 상관없이, 순전히 우연(Chance)에 의한 경우.  
마치 주사위를 굴려서 결측이 된 것과 같다.

- 헬스케어 예시:

채혈한 혈액 튜브를 실수로 떨어뜨려 깨져서 검사 결과가 없는 경우.  
전산 시스템 오류로 특정 시간대의 데이터가 랜덤하게 날아간 경우.

- 대처:

데이터를 그냥 삭제(Drop)하고 분석해도 결과에 큰 편향(Bias)을 주지 않는다.

## 2. 데이터 전처리 (Preprocessing)

### 2.2.2. MAR (Missing At Random)

원어: Missing At Random

- 한국어: 무작위 결측 (또는 조건부 무작위 결측)

- 의미:

가장 오해하기 쉬운 개념입니다. 결측된 이유가 '결측된 값 그 자체'와는 관련이 없지만, '관측된 다른 변수'와는 관련이 있는 경우이다.

즉, 다른 변수(예: 성별, 나이)를 알면 결측이 발생할 확률을 설명할 수 있다는 뜻이다.

- 헬스케어 예시:

체중 데이터 누락: 여성 환자들이 남성 환자보다 체중 공개를 꺼려서 체중 데이터 결측이 많다고 가정해보자.

- 이때 결측은 '성별(관측된 데이터)'과 관련이 있다.
- 하지만 '체중이 많이 나가서(결측된 값 자체)' 누락된 것은 아니다(가벼운 여성도 공개를 꺼릴 수 있으므로).

- 대처:

함부로 삭제하면 편향이 생깁니다(위 예시에서 삭제 시 남성 데이터만 남게 됨).

다른 변수를 활용한 대체(Imputation) 기법을 사용해야 한다.

## 2. 데이터 전처리 (Preprocessing)

### 2.2.3. MNAR (Missing Not At Random)

원어: Missing Not At Random

- 한국어: 비무작위 결측

- 의미:

데이터가 누락된 이유가 '결측된 그 값 자체' 때문인 경우입니다.

가장 다루기 까다로운 케이스이며, 결측 자체가 중요한 정보가 됩니다.

- 헬스케어 예시:

우울증 설문: 우울증 점수가 매우 높은(심각한) 환자가 의욕 저하로 인해 설문지 작성을 포기한 경우.

- 점수가 비어있는 이유는 '점수가 높기 때문'입니다.

중환자실 혈압: 환자가 위독하여(혈압이 너무 낮거나 불안정하여) 기계가 측정을 실패한 경우.

- 대처:

이 데이터를 삭제하거나 단순히 평균값으로 채우면 심각한 왜곡이 발생합니다. (중증 환자를 분석에서 배제하는 꼴이 됨).

전문적인 통계적 모델링이 필요하거나, 결측 여부 자체를 변수화(Flagging)해서 분석해야 합니다.

## 2. 데이터 전처리 (Preprocessing)

### 요약 비교

용어	원어	의미 (결측의 원인)	헬스케어 예시	처리 난이도
MCAR	Missing Completely At Random	순수한 우연	튜브 파손, 전산 오류	쉬움 (삭제 가능)
MAR	Missing At Random	다른 변수에 의존	여성이 체중 기입을 덜 함	보통 (대체 필요)
MNAR	Missing Not At Random	값 그 자체가 원인	너무 아파서 검사 못 받음	어려움 (신중 접근)

실습 :

`nb_1_3_cohort.ipynb`

`nb_2_1_cohort.ipynb`

`nb_2_2_cohort.ipynb`

`nb_2_3_cohort.ipynb`

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.1. 통계적 추론의 기초

- 항상, 우리가 가진 데이터는 전체 환자(모집단, Population)가 아닌 일부(표본, Sample)에서 출발을 하게 된다.
- 이상적인 데이터를 통해 추정을 할수 있는 환경을 만드는 것은 극히 어렵다.
- 이러한 어려움을 보상하기 위해 우리는 통계 기법을 사용한다.
- 즉, 표본에서 얻은 결과를 통해 모집단의 특성을 추측하는 것, 이것이 '통계적 추론' 이고 지속적으로 연구가 되어야할 주제이다.
- 헬스케어 연구의 대부분은 "이 신약이 효과가 있는가?", "이 위험요인이 질병을 유발하는가?"를 증명하는 과정이며, 이는 가설 검정(Hypothesis Testing)을 통해 이루어진다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.2. 가설 설정과 P-value (상세 해설)

### 3.2.1. 가설 검정의 논리: "무죄 추정의 원칙"

- 통계적 가설 검정은 법정에서의 재판 과정과 매우 유사하다.

**법정:** 피고인(용의자)은 확실한 증거가 나오기 전까지는 '무죄'로 추정된다. 검사는 증거를 제시하여 판사에게 "이 사람이 무죄일 확률은 희박하다"고 설득해야 한다.

**연구:** 새로운 약물(피고인)은 확실한 데이터가 나오기 전까지는 '효과가 없다'고 추정된다. 연구자(검사)는 실험 데이터를 통해 "이 약이 효과가 없을 확률은 희박하다"는 것을 입증해야 한다.

- 이 과정에서 두 가지 대립되는 가설을 세우게 된다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.2.1.1. 귀무가설 ( $H_0$ , Null Hypothesis)

- 정의: "차이가 없다", "효과가 없다", "0(Null)이다"라는 가설이다.
- 성격: 현재의 상태(Status Quo)이자, 연구자가 기각(Reject)하고 싶어 하는 가설이다. 처음에는 이 가설이 맞다고 가정하고 분석을 시작한다.
- 헬스케어 예시:
  - "A 신약과 위약(Placebo) 간의 혈압 강하 효과에는 차이가 없다."
  - "흡연자와 비흡연자의 폐암 발병률은 같다."

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.2.1.2. 대립가설 ( $H_1$ , Alternative Hypothesis)

- 정의: "차이가 있다", "효과가 있다"라는 가설이다.
- 성격: 연구자가 새로운 데이터를 통해 입증(Prove)하고 싶어 하는 나의 주장이다.
- 헬스케어 예시:
  - "A 신약은 위약보다 혈압 강하 효과가 크다." (단측 검정)
  - "흡연자와 비흡연자의 폐암 발병률은 다르다." (양측 검정)

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.2.2. P-value (유의 확률, Probability Value)

### 3.2.2.1. P-value의 정확한 정의

- P-value는 "귀무가설이 참이라고 가정했을 때, 우리가 수집한 데이터와 같은(또는 더 극단적인) 결과가 우연히 관찰될 확률"이다.

\*P-value는 '귀무가설이 참일 확률'이나 '대립가설이 참일 확률'이 아니다. 이것은 '놀라움의 정도(Surprise factor)'를 나타내는 척도이다.

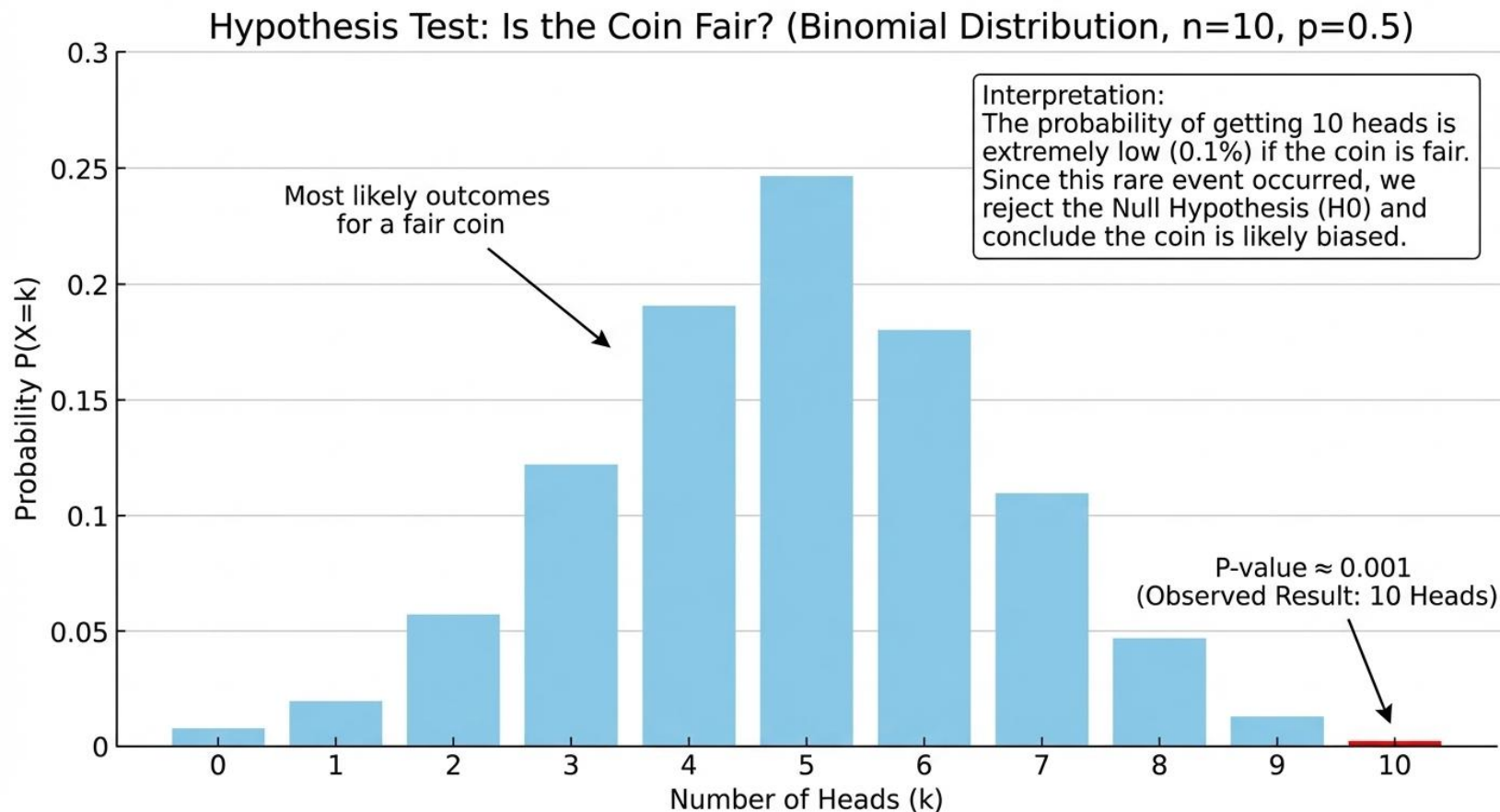
### 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

#### 3.2.2. P-value (유의 확률, Probability Value)

##### 3.2.2.2. 직관적인 해석

연구자가 동전 던지기 실험을 한다고 가정해보자.

- **상황:** 동전을 10번 던졌는데 10번 모두 앞면이 나왔다.
- **귀무가설:** "이 동전은 정상적인 동전이다 (앞/뒤 확률 50%)."
- **해석:** 정상 동전에서 10번 연속 앞면이 나올 확률은  $(1/2)^{10} \approx 0.001(0.1\%)$ 이다.
- **판단:** 0.1%라는 확률은 매우 희박하다. 귀무가설(정상 동전)이 참이라면 이런 일이 일어날 리가 없다. 따라서 "이 동전은 조작되었다(대립가설)"고 판단한다.
- 여기서 0.001이 바로 P-value이다.



### 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

#### 3.2.2.3. P-value의 크기에 따른 의미

- P-value가 크다 (예: 0.3):

"귀무가설 하에서 이런 데이터가 나올 확률이 30%나 된다."

흔히 일어날 수 있는 일이다. → "효과가 있다고 보기 어렵다 (귀무가설 기각 실패)."

- P-value가 작다 (예: 0.001):

"귀무가설 하에서 이런 데이터가 나올 확률은 0.1%밖에 안 된다."

우연이라고 보기엔 너무 이상하다. 뭔가 특별한 이유(효과)가 있다. → "통계적으로 유의미한 차이가 있다 (귀무가설 기각)."

### 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

#### 3.2.3. 의사결정의 기준: 유의수준 ( $\alpha$ , Significance Level)

P-value가 얼마나 작아야 "회박하다"고 할 것인가? 그 기준선이 필요하다.

- **유의수준 0.05 (5%):** 가장 일반적으로 사용되는 기준이다. "우연히 이런 결과가 나올 확률이 5% 미만이라면, 우연이 아니라고 보겠다"는 약속이다.

P-value < 0.05: 귀무가설 기각 → 통계적으로 유의하다 (Significant).

P-value  $\geq$  0.05: 귀무가설 채택(유지) → 통계적으로 유의하지 않다 (Not Significant).

- **엄격한 기준 (0.01 or 0.001):** 생명과 직결된 신약 임상시험 등에서는 더 엄격한 기준인 1%(0.01)를 사용하기도 한다. 오판의 가능성을 더 줄이기 위함이다.

### 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

#### 3.2.4. 통계적 유의성 vs 임상적 유의성 (중요)

헬스케어 데이터 분석가로서 반드시 구분해야 할 개념이다.

통계적 유의성 (Statistical Significance):  $P\text{-value} < 0.05$ 를 만족했는가? (수학적 판단)

임상적 유의성 (Clinical Significance): 그 차이가 실제로 환자의 치료나 삶의 질에 의미가 있는가? (의학적 판단)

#### [사례] 고혈압 신약 개발

- 상황: 10만 명을 대상으로 임상시험을 했다.
- 결과: 위약군은 평균 혈압이 140, 신약군은 139.8이었다. (0.2 mmHg 감소)
- 통계적 판단: 표본 수가 워낙 많아(10만 명), 아주 미세한 차이도 P-value는 0.0001로 나올 수 있다. → 통계적으로 유의하다.
- 임상적 판단: 의사 입장에서 혈압 0.2 떨어뜨리려고 비싼 약을 처방할까? 아니다. → 임상적으로는 유의하지 않다.
- 결론: 데이터 분석가는 P-value에만 집착해서는 안 되며, 효과 크기(Effect Size)와 임상적 맥락을 함께 고려해야 한다.

### 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

#### 3.2. 5. 두 가지 오류 (Type I & II Error)

- 가설 검정은 확률에 기반하므로 언제나 오류의 가능성이 있다.

구분	내용	헬스케어 예시	위험성
제1종 오류 ( $\alpha$ Error)	귀무가설이 참인데 기각함 (위양성, False Positive)	효과 없는 약을 효과가 있다고 판단함.	환자가 불필요한 약을 먹고 부작용을 겪거나 돈을 낭비함.
제2종 오류 ( $\beta$ Error)	대립가설이 참인데 못 밝혀냄 (위음성, False Negative)	효과 있는 약을 효과가 없다고 판단하여 폐기함.	좋은 치료 기회를 놓쳐 환자를 살리지 못함.

- 일반적으로 과학적 검증 과정에서는 제1종 오류(거짓말쟁이가 되는 것)를 통제하는 것을 더 중요하게 생각한다. 그래서 유의수준( $\alpha$ )을 0.05로 고정해두고 분석을 수행한다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3. 주요 검정 기법 소개 (비시계열 중심)

비시계열 데이터 분석에서 가장 빈번하게 수행하는 작업은 '그룹 간의 비교'이다.

"환자군과 정상군의 혈액 수치가 다른가?", "A치료법과 B치료법 중 어느 것이 더 효과적인가?"와 같은,

질문에 답하기 위해, 데이터의 유형(연속형/범주형)과 집단의 수에 따라 적절한 통계적 검정 기법을 선택해야 한다.  
이에 대해 살펴보자

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.1. 검정 기법 선택의 지도 (Decision Map)

통계 검정을 시작하기 전, 반드시 다음 두 가지 질문을 던져야 한다.

- 1. 종속변수(결과값)가 무엇인가? (숫자인가, 카테고리인가?)
- 2. 비교하려는 집단이 몇 개인가? (2개인가, 3개 이상인가?)

비교 대상 (독립변수)	결과 데이터 (종속변수)	추천 검정 기법	Python 함수 (Scipy)
두 집단 (예: 남 vs 여)	연속형 (예: 키, 혈압)	T-test (t-검정)	ttest_ind
세 집단 이상 (예: A/B/C약)	연속형 (예: 키, 혈압)	ANOVA (분산분석)	f_oneway
집단 무관	범주형 (예: 생존/사망)	Chi-square (카이제곱)	chi2_contingency

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.2. 두 집단의 평균 비교: T-test (t-검정)

### - 개념 및 원리:

- 가장 대표적인 모수적(Parametric) 검정 방법이다. 두 집단의 평균(Mean)이 통계적으로 유의미한 차이가 있는지를 평가한다. 단순히 평균값의 차이만 보는 것이 아니라, 데이터가 퍼져 있는 정도인 분산(Variance)을 함께 고려한다.
- *직관적 이해*: 두 집단의 평균 차이가 클수록(Signal이 강함), 그리고 집단 내 분산이 작을수록(Noise가 약함) 두 집단은 '다르다'고 판단할 확률이 높아진다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3. 2. 두 집단의 평균 비교: T-test (t-검정)

- 세부 종류:

- 독립 표본 T-검정 (Independent Samples T-test):

상황: 서로 전혀 다른 두 그룹을 비교할 때 사용한다.

헬스케어 예시: "신약을 투여받은 실험군(Group A)과 위약을 투여받은 대조군(Group B)의 콜레스테롤 수치 비교."

전제조건: 두 그룹은 서로 독립적이어야 하며, 데이터는 정규성을 따라야 한다.

- 대응 표본 T-검정 (Paired Samples T-test):

상황: 동일한 대상의 두 가지 측정값을 비교하거나, 쌍(Pair)을 이룬 대상을 비교할 때 사용한다.

헬스케어 예시: "한 환자의 왼쪽 팔 혈압과 오른쪽 팔 혈압의 차이 비교" 또는 "다이어트 프로그램 참가 전 몸무게와 참가 후 몸무게 비교." (전/후 비교는 시계열적 성격이 있으나, '변화량'이라는 하나의 변수로 취급하여 1일차에 다루기도 한다.)

- 결과 해석:

- T-statistic (t-통계량): 두 집단 간의 차이를 표준오차로 나눈 값이다. 절댓값이 클수록 차이가 크다는 뜻이다.
- P-value < 0.05: 두 집단의 평균은 통계적으로 유의미하게 다르다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.3. 세 집단 이상의 평균 비교: ANOVA (분산분석)

### - 개념 및 원리

- ANalysis Of VAriance의 약자이다. 3개 이상의 그룹(예: 저용량군, 고용량군, 대조군)을 비교할 때 T-test를 세 번 하지 않고 한 번에 분석하는 기법이다.
- 왜 *T-test*를 여러 번 하면 안 되는가?  
A-B, B-C, A-C를 각각 비교하면, 검정을 할 때마다 5%의 오류 확률이 누적되어 전체적인 제1종 오류(위양성) 확률이 급격히 증가한다. (다중 비교의 문제)
- 원리(F-검정): (집단 간의 분산 / 집단 내의 분산) 비율을 본다. 그룹 간의 차이가 그룹 내부의 개인차보다 월등히 크다면 "그룹 간 효과가 있다"고 판단한다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.3. 세 집단 이상의 평균 비교: ANOVA (분산분석)

### - 사후 검정 (Post-hoc Analysis)

- ANOVA의 P-value가 0.05보다 작다는 것은 "적어도 한 그룹은 다른 그룹과 다르다"는 것만 알려줄 뿐, 구체적으로 A와 B가 다른지, B와 C가 다른지는 알려주지 않는다. 따라서 반드시 사후 검정을 수행해야 한다.
- Tukey's HSD: 모든 그룹 쌍을 비교할 때 가장 보편적으로 사용된다.
- Bonferroni: 매우 보수적인 방법으로, 오류 가능성을 엄격하게 통제할 때 사용한다.

### - 헬스케어 예시

- "3가지 종류의 항암제(Drug A, Drug B, Drug C)를 투여한 환자 그룹 간의 종양 크기 감소율 비교."

### 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

#### 3.3.4. 범주형 변수의 비율 비교: Chi-square Test (카이제곱 검정)

##### - 개념 및 원리

- 평균을 낼 수 없는 명목형 변수(성별, 혈액형, 질병 유무) 간의 관계를 분석한다. 두 변수가 서로 독립적인지(관련이 없는지) 아니면 연관성이 있는지를 검정한다. 이를 '독립성 검정'이라고도 한다.

##### - 논리 구조 (관찰 빈도 vs 기대 빈도)

- 관찰 빈도 (Observed): 실제 데이터에서 집계된 환자 수.
- 기대 빈도 (Expected): 두 변수가 아무런 관련이 없다고 가정했을 때 이론적으로 나와야 하는 환자 수.

- 카이제곱 통계량: 
$$\sum \frac{(\text{관찰값} - \text{기대값})^2}{\text{기대값}}$$

- 관찰값과 기대값의 차이가 클수록 통계량이 커지고, 두 변수는 관련이 있다고 판단한다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.3. 세 집단 이상의 평균 비교: ANOVA (분산분석)

### - 헬스케어 예시

- 상황: "흡연 여부(흡연자/비흡연자)에 따라 폐암 발생(발생/정상) 비율이 다른가?"

- 분석:

2x2 교차표(Contingency Table)를 작성한다.

카이제곱 검정을 수행한다.

P-value < 0.05라면, "흡연과 폐암 발생은 서로 독립적이지 않다(관련이 있다)"고 결론 내린다.

### - 주의사항 (Fisher's Exact Test)

- 카이제곱 검정은 표본 수가 충분히 클 때(보통 각 칸의 기대 빈도가 5 이상) 유효하다. 희귀 질환 연구처럼

표본 수가 매우 적은 경우에는 피셔의 정확 검정(Fisher's Exact Test)을 사용해야 정확한 결과를 얻을 수 있다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.4. 상관분석 (Correlation Analysis)

### 1. 상관분석의 개요와 목적

#### 1.1. 정의

상관분석은 두 개의 연속형 변수(Continuous Variable)가 서로 얼마나 밀접하게 관련되어 움직이는지를 통계적으로 파악하는 기법이다. 한 변수가 증가할 때 다른 변수도 함께 증가하는지, 아니면 감소하는지, 혹은 아무런 관련이 없는지를 수치로 정량화하여 보여준다.

#### 1.2. 헬스케어에서의 활용 목적

**변수 간 관계 탐색:** "BMI가 높은 환자는 혈압도 높은 경향이 있는가?", "운동 시간이 길수록 공복 혈당은 낮아지는가?"와 같은 임상적 궁금증을 해소한다.

**예측 변수 선정 (Feature Selection):** 질병 발생을 예측하는 모델을 만들 때, 질병 유무(타겟)와 상관성이 높은 변수를 찾아내어 모델의 입력값으로 사용한다.

**다중공선성(Multicollinearity) 확인:** 독립변수들끼리 너무 강한 상관관계가 있으면(예: 복부 둘레와 BMI) 통계 모델의 신뢰도를 떨어뜨리므로 이를 사전에 감지하는 데 사용한다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.4. 상관분석 (Correlation Analysis)

### 2. 피어슨 상관계수 (Pearson Correlation Coefficient)

#### 2.1. 개념 및 $r$ 값의 의미

가장 보편적으로 사용되는 모수적(Parametric) 방법이다. 두 변수 간의 '선형(Linear) 관계'의 강도를 측정하며,

$r$  또는  $\rho$  (rho)로 표기한다.  $r$  값은 항상 -1에서 +1 사이의 값을 가진다.

- $r = +1$  (완벽한 양의 상관관계): 한 변수가 증가하면 다른 변수도 일정한 비율로 정확히 증가한다. (현실 데이터에서는 거의 존재하지 않음)
- $r = -1$  (완벽한 음의 상관관계): 한 변수가 증가하면 다른 변수는 일정한 비율로 정확히 감소한다.
- $r = 0$  (상관관계 없음): 두 변수는 서로 아무런 선형적 관계가 없다. (무작위로 퍼져 있음)

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.4. 상관분석 (Correlation Analysis)

### 2.2. 해석 가이드라인 (Cohen's Standard 등)

- 일반적으로 통계학에서는 다음과 같은 기준으로 상관관계의 강도를 해석한다. 단, 헬스케어 분야는 생체 데이터의 변동성이 크기 때문에 조금 더 낮은 수치라도 의미 있게 받아들여질 수 있다.

r 값의 범위 (절댓값)	해석	헬스케어 예시
0.0 ~ 0.2	관계가 거의 없다 (무시 가능)	신발 사이즈와 지능지수(IQ)
0.2 ~ 0.4	약한 상관관계 (Weak)	커피 섭취량과 수면 시간
0.4 ~ 0.6	뚜렷한 상관관계 (Moderate)	BMI와 수축기 혈압
0.6 ~ 0.8	강한 상관관계 (Strong)	키와 몸무게
0.8 ~ 1.0	매우 강한 상관관계 (Very Strong)	왼쪽 시력과 오른쪽 시력, 수축기 혈압과 이완기 혈압

### 2.3. 전제 조건

- 피어슨 상관분석을 수행하기 위해서는 두 변수가 모두 정규분포를 따라야 하며, 선형성을 가져야 한다. 이상치(Outlier)에 매우 민감하게 반응하므로, 전처리 단계에서 이상치 제거가 필수적이다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.4. 상관분석 (Correlation Analysis)

### 3. 비모수적 상관분석: 스피어만 & 켄달

헬스케어 데이터는 정규분포를 따르지 않는 경우가 많다(예: 입원 기간, 의료비 등은 오른쪽으로 꼬리가 긴 분포를 가짐). 또한, 명확한 수치보다는 '순위(Rank)'가 중요한 데이터도 있다. 이럴 때는 피어슨 대신 비모수적 방법을 사용한다.

### 3.1. 스피어만 순위 상관계수 (Spearman's Rank Correlation)

원리: 실제 데이터 값 대신, 값의 '순위(Rank)'를 매겨서 그 순위 간의 상관관계를 계산한다.

특징:

데이터의 분포가 정규분포가 아니어도 사용 가능하다.

선형 관계뿐만 아니라 곡선 관계(비선형적이지만 단조 증가/감소하는 경우)도 잘 잡아낸다.

이상치의 영향을 덜 받는다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.4. 상관분석 (Correlation Analysis)

### 3.2. 켄달의 타우 (Kendall's Tau)

원리: 두 변수 간의 순위 쌍(Pair)이 일치하는지 불일치하는지 비율을 본다.

특징: 표본 수(Sample Size)가 적을 때 스피어만보다 더 신뢰할 수 있는 결과를 보여준다. 회귀 질환 연구에서 종종 사용된다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.4. 상관분석 (Correlation Analysis)

### 3.3 모수 vs 비모수 비교

구분	모수적 (Parametric)	비모수적 (Non-parametric)
핵심 전제	데이터가 정규분포를 따른다.	정규분포를 따르지 않거나 모른다.
대표 기법	피어슨 (Pearson)	스피어만 (Spearman), 켄달 (Kendall)
사용하는 값	실제 수치 (Value)	등수 / 순위 (Rank)
이상치 영향	매우 민감함 (취약함)	덜 민감함 (강함)
헬스케어 예시	"키와 몸무게의 관계" (대부분의 생체 신호는 정규분포를 따름)	"암 기수(Stage)와 통증 점수의 관계" (1기, 2기는 숫자가 아니라 순서임) "재원 기간과 병원비의 관계" (한쪽으로 치우친 데이터)

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.4. 상관분석 (Correlation Analysis)

### 4. 시각화를 통한 분석

숫자( $r$ 값)만 확인하는 것은 위험하다. 반드시 그래프를 통해 눈으로 확인해야 한다. (Anscombe's Quartet 사례:  $r$ 값은 똑같은데 그래프 모양은 완전히 다른 경우가 있다.)

#### 4.1. 산점도 (Scatter Plot)

- $x$ 축: 원인 변수 (독립변수)
- $y$ 축: 결과 변수 (종속변수)
- 데이터 포인트들이 우상향하면 양의 상관관계, 우하향하면 음의 상관관계이다.
- 점이 직선 주위에 뿔뿔하게 모여 있을수록 상관관계가 강한 것이다.
- 비선형 관계(U자형, J자형 곡선)를 발견하는 데 탁월하다. 피어슨 상관계수는 U자형 관계에서 '0'에 가까운 값이 나와 관계가 없다고 오판할 수 있지만, 산점도를 보면 명확한 패턴을 알 수 있다.

#### 4.2. 히트맵 (Heatmap)

- 수십 개의 변수(검사 항목)를 동시에 분석할 때 유용하다.
- 색상의 농도(진한 빨강=강한 양의 상관, 진한 파랑=강한 음의 상관)를 통해 변수들 간의 관계를 한눈에 파악한다.
- 주로 탐색적 데이터 분석(EDA) 단계에서 변수 간의 '클러스터(덩어리)'를 찾는 데 쓰인다.

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.4. 상관분석 (Correlation Analysis)

## 5. 헬스케어 분석 시 주의사항 (Critical Pitfalls)

### 5.1. 상관관계는 인과관계가 아니다 (Correlation $\neq$ Causation)

가장 중요하고 흔한 오류이다. 두 변수가 같이 움직인다고 해서, 하나가 다른 하나의 원인이라고 단정 지을 수 없다.

예시: "병원의 규모가 클수록 환자 사망률이 높다."

- *상관관계*: 양의 상관관계가 나타날 수 있다.
- *잘못된 해석*: "큰 병원이 환자를 더 많이 죽게 한다." (인과관계 오류)
- *올바른 해석*: "큰 병원에는 중증도가 높은 위독한 환자들이 많이 몰리기(제3의 요인) 때문에 사망률이 높게 나타난다."

### 5.2. 교란 변수 (Confounding Variable)

위의 예시에서 '환자의 중증도'처럼, 두 변수 모두에게 영향을 미쳐서 마치 둘 사이에 관계가 있는 것처럼 보이게 만드는 제3의 변수를 **교란 변수**라고 한다.

순수한 상관관계를 파악하기 위해서는 이 교란 변수를 통제(Control)해야 한다. (예: 중증도가 비슷한 환자끼리 그룹을 나누어 비교)

# 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

## 3.3.4. 상관분석 (Correlation Analysis)

## 5.3. 허위 상관 (Spurious Correlation)

실제로는 아무런 논리적 관계가 없는데, 우연히 데이터의 패턴이 맞아떨어져 상관관계가 높게 나오는 경우이다.

빅데이터 분석에서 변수가 많아질수록 우연히 상관계수가 높은 변수 쌍이 발견될 확률이 높아진다.

따라서 통계적 수치뿐만 아니라 '의학적/생물학적 기전(Mechanism)'으로 설명이 가능한지 반드시 검토해야 한다.

## 5.4. 생태학적 오류 (Ecological Fallacy)

집단(국가, 병원)의 통계치를 개인에게 그대로 적용할 때 발생하는 오류이다.

- "지방 섭취량이 높은 국가는 유방암 발병률이 높다" (국가 단위 상관관계)
- → "따라서 지방을 많이 먹는 철수는 유방암에 걸릴 것이다." (개인 단위 적용 불가)

### 3. 비시계열 데이터 분석 1 : 통계적 가설검정기법

실습:

NB3\_1.ipynb

NB3\_2.ipynb

NB3\_3.ipynb

NB\_4\_1.ipynb

NB\_4\_2.ipynb

NB\_4\_3.ipynb

**수고하셨습니다.**

