

# streamlit 대시보드 강의

with  python™

# 배 포 금 지

# 강의 대상 및 목적

with  python™

**입문자**

**국비교육생**

**취업준비생**

**포트폴리오**

**Google Cloud Platform**

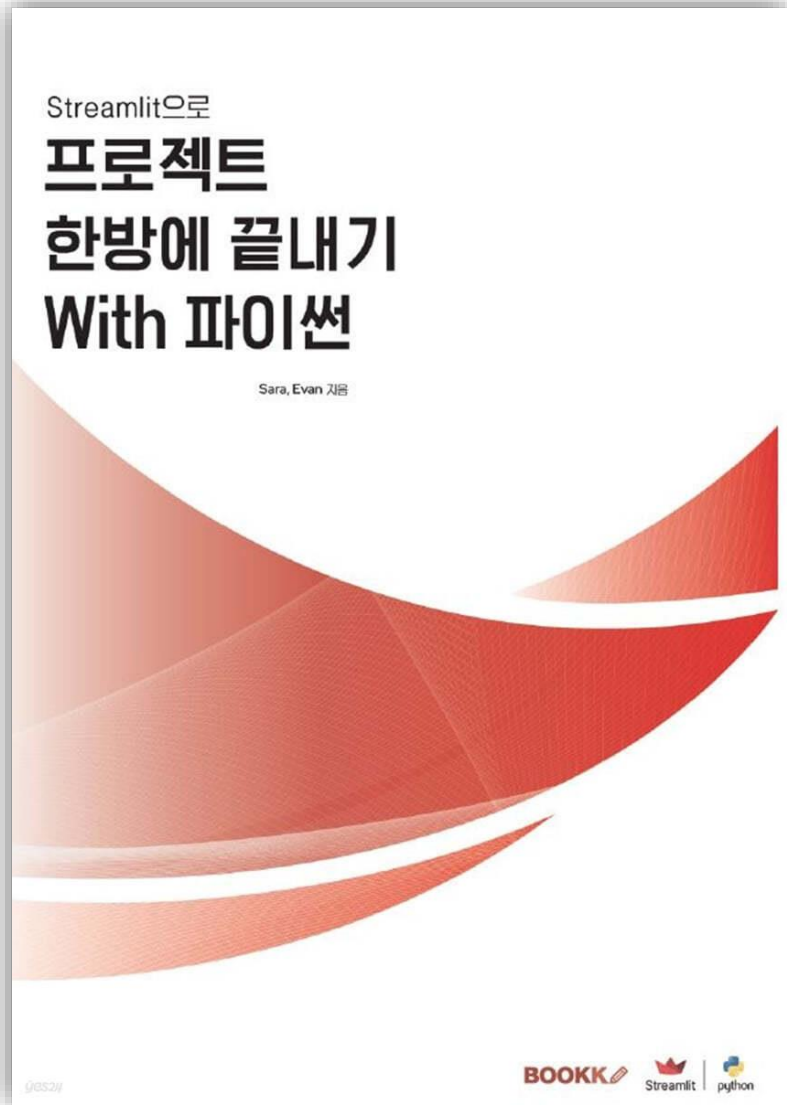
**Streamlit 대시보드**

**기초통계, 머신러닝**

**파이썬 기초 문법**

## 03. 주요 교재

(2023년 11월 기준)




- ✓ 2023년 5월 30일 출간 (초판)
  - 파이썬 기초문법
  - 기초통계 및 머신러닝
  - Streamlit 대시보드
- ✓ 2024년 1월 개정판
  - Google Cloud Platform 배포
  - Github Actions

# 강사 소개

with  python™

# 01. 강사 소개

(2023년 11월 기준)

- ✓ INTJ (혼자 있을 때 힘이 충전됨)
- ✓ 학점은행 경영학사 (2009)
- ✓ 한동대학교 국제개발협력대학원 (2021)
- ✓ 국민대학교 비즈니스IT전문대학원 박사과정 (2023 ~ )
- ✓ 2023년 기준 강의경력 4년차
  - 공기관 강의
  - 금융권 강의
  - 재직자 대상 강의
  - 취업준비생 대상 강의 (★ ★ ★)
- ✓ 오프라인 누적 강의 시간(2023년 12월 기준 5000시간 )
  - 월평균 150시간 x 3년



## 02. 저서

(2023년 11월 기준)

- ✓ (KCI 등재) 필리핀 스타트업의 기업가적 지향성과 기업성장에 관한 연구:  
사회적 자본의 매개 효과(2021, 한국벤처창업학회)
  - 주요 분석 방법론 : 구조방정식(R, PLS-SEM)
- ✓ 파이썬 캐글 뽀개기(2021, 비제이퍼블릭)
- ✓ Streamlit으로 프로젝트 한방에 끝내기 with 파이썬(2023, 부크크)

# 개발 환경 설정

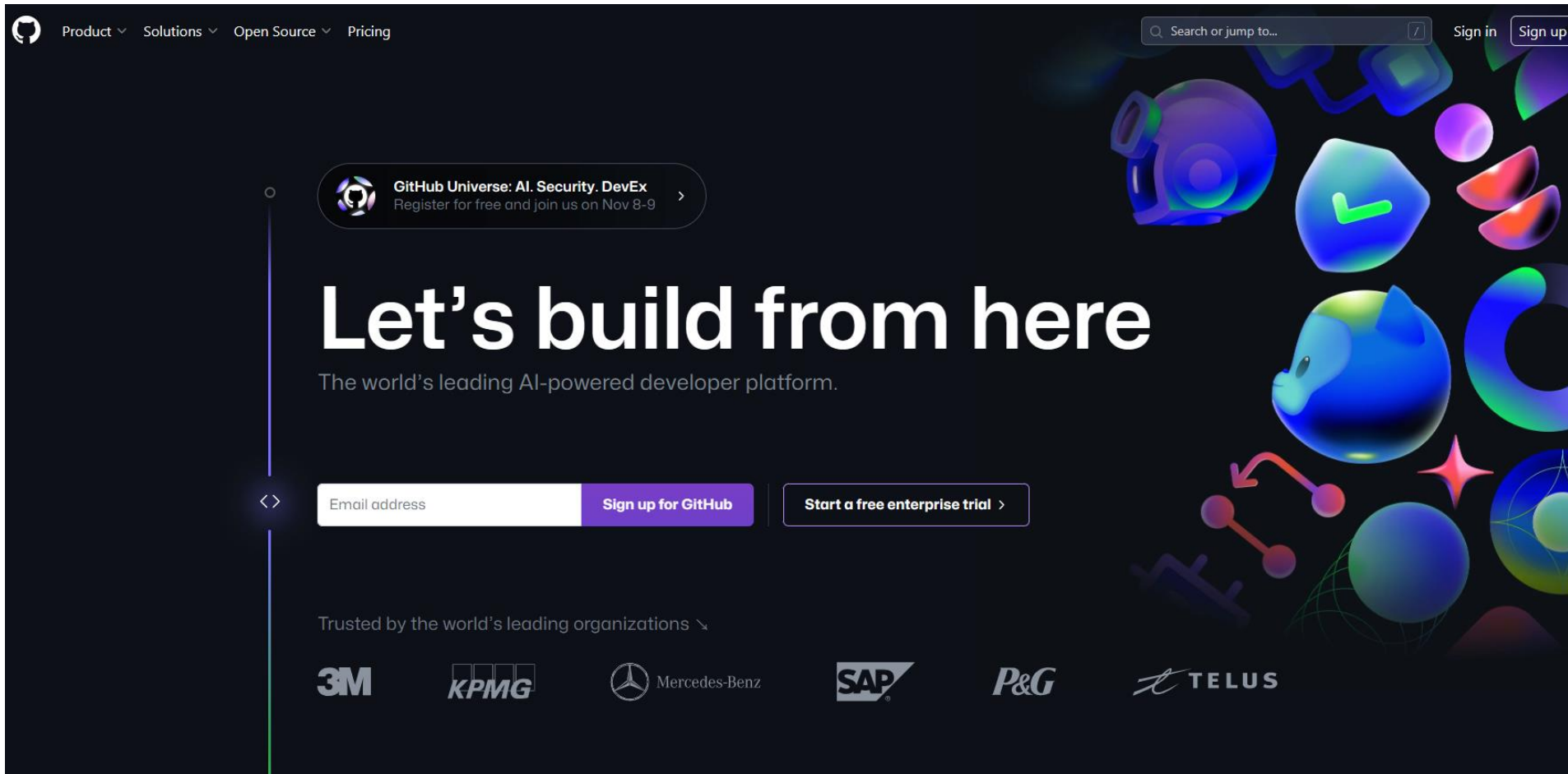
with  python™

# 01. Github 설정

(2023년 11월 기준)

◆ Github : Git을 사용한 버전 관리 및 협업을 위한 웹 기반 플랫폼

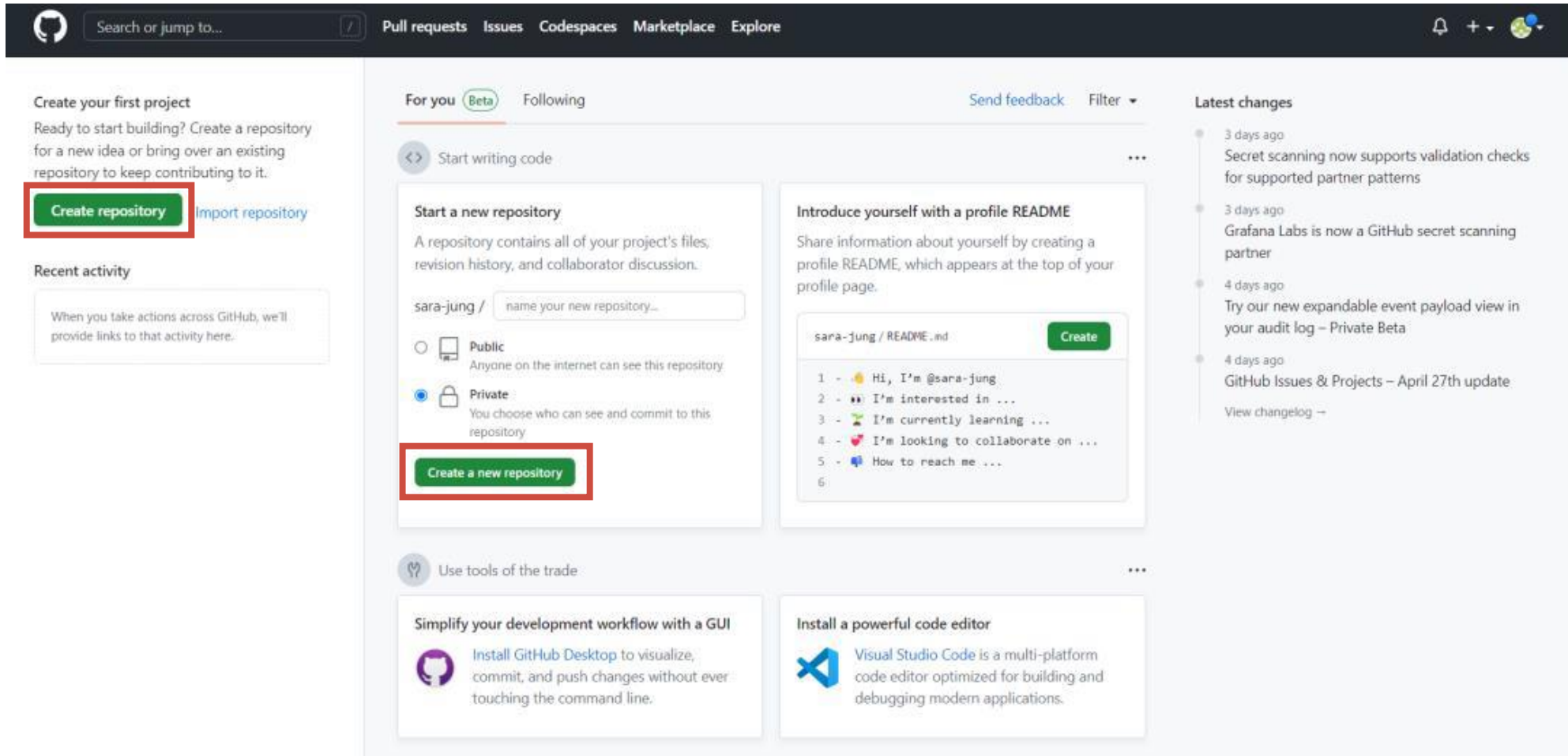
- 사이트 : <https://github.com/>



# 01. Github 설정

(2023년 11월 기준)

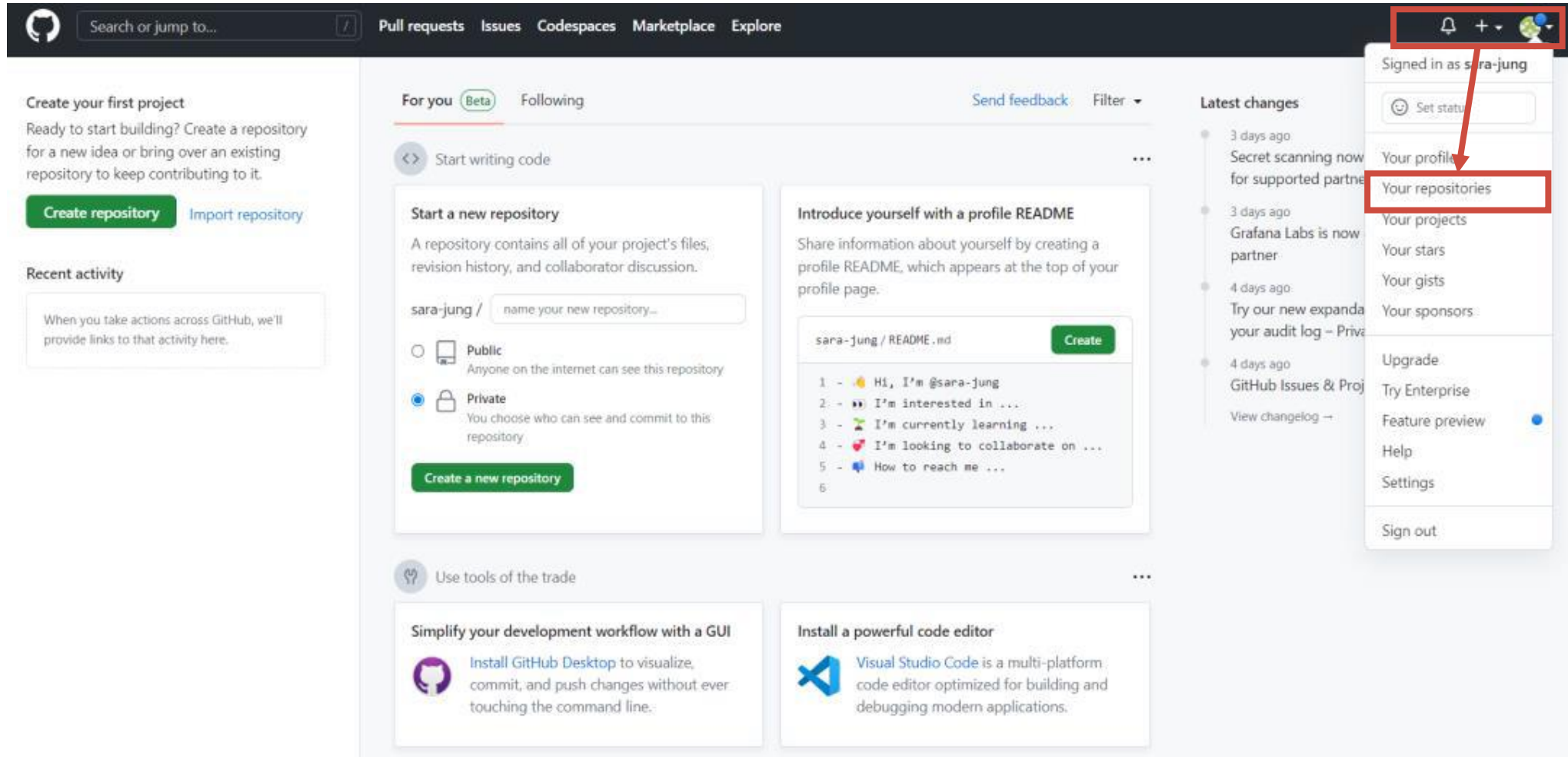
## ◆ 회원가입이 완료된 후, Create Repository 클릭



# 01. Github 설정

(2023년 11월 기준)

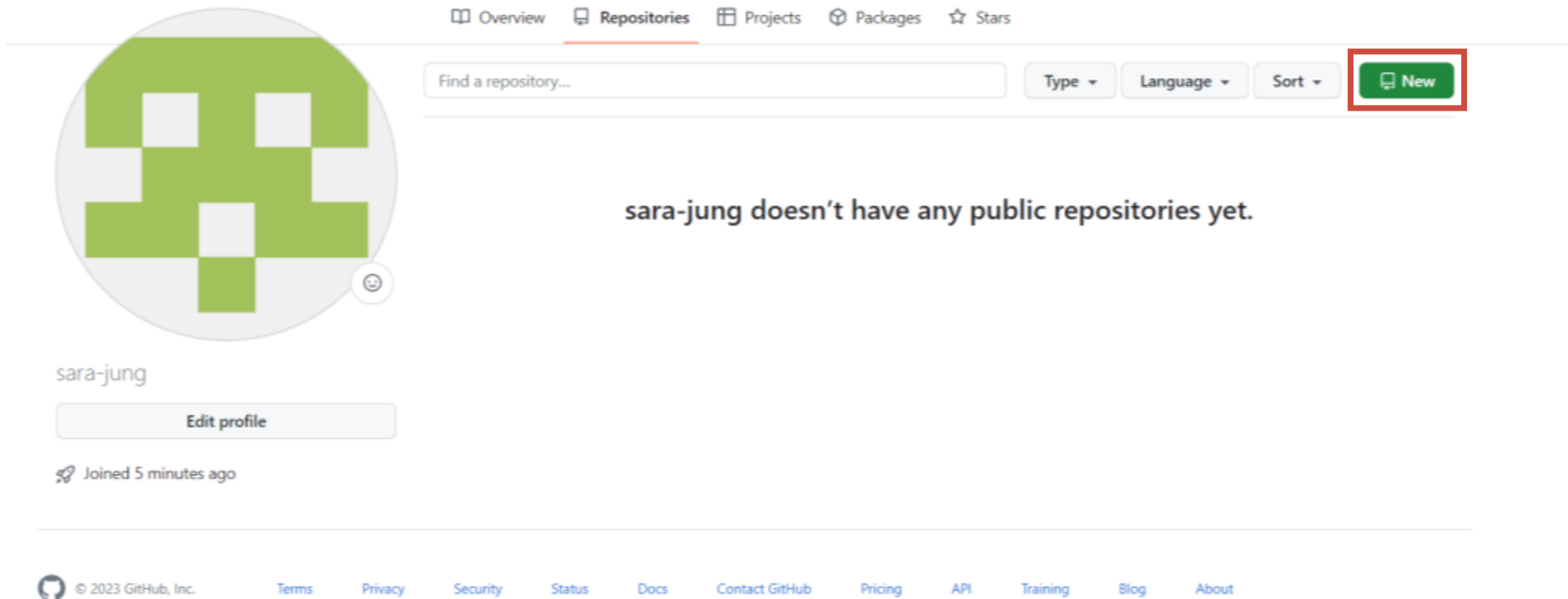
## ◆ (또는) 우측 상단의 Profile icon 클릭 > Your repositories 클릭



# 01. Github 설정

(2023년 11월 기준)

◆ (또는) 우측 상단의 Profile icon 클릭 > Your repositories 클릭



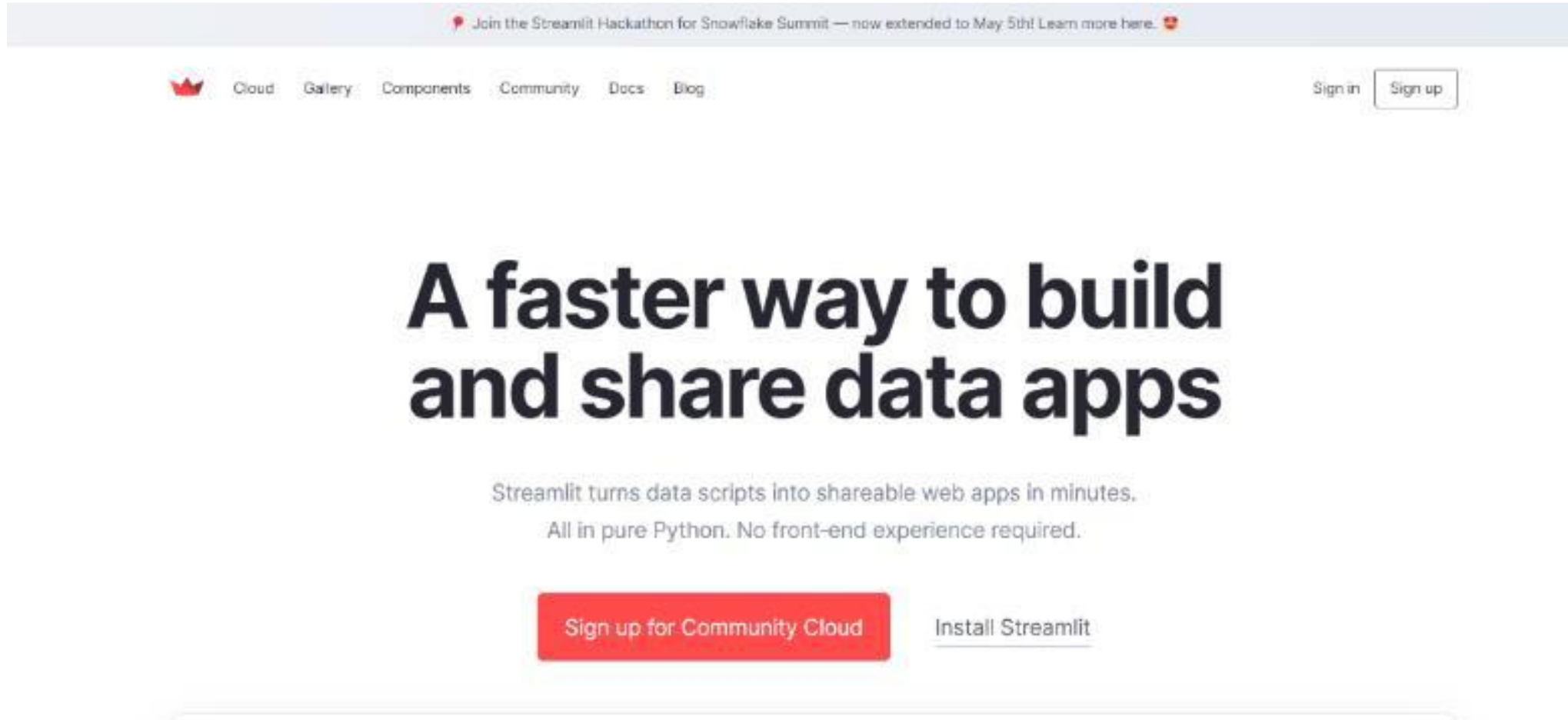
## 강의 실습 영상 참고

## 02. Streamlit 회원가입

(2023년 11월 기준)

◆ Streamlit은 머신러닝 및 데이터 사이언스 프로젝트 위한 대시보드 오픈소스 라이브러리

- 사이트 : <https://streamlit.io/>



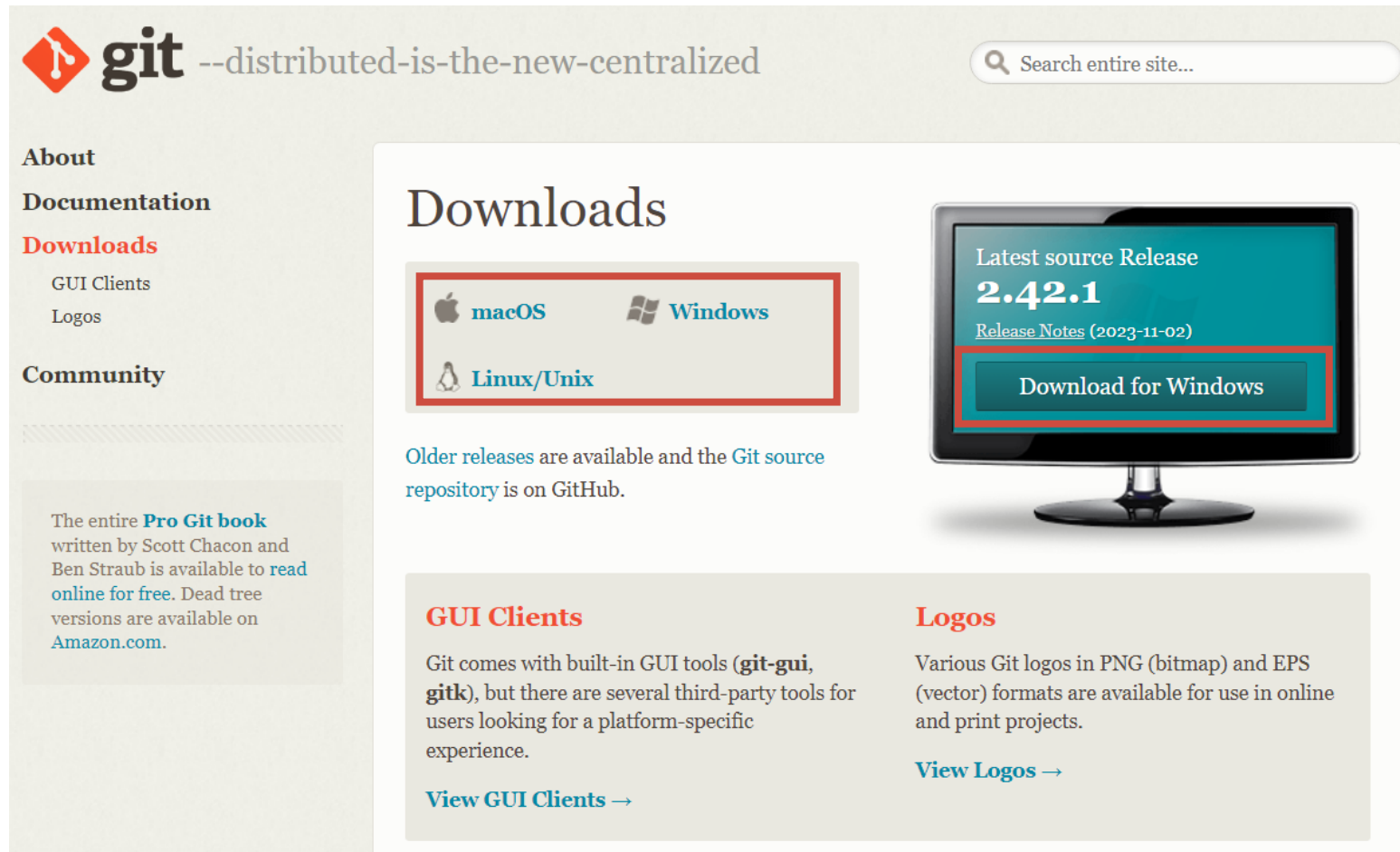


# 03. Git 설치

(2023년 11월 기준)

## ◆ Git은 소스 코드의 변경 사항을 추적하기 위한 분산 버전 제어 시스템

- 사이트 : <https://git-scm.com/downloads>



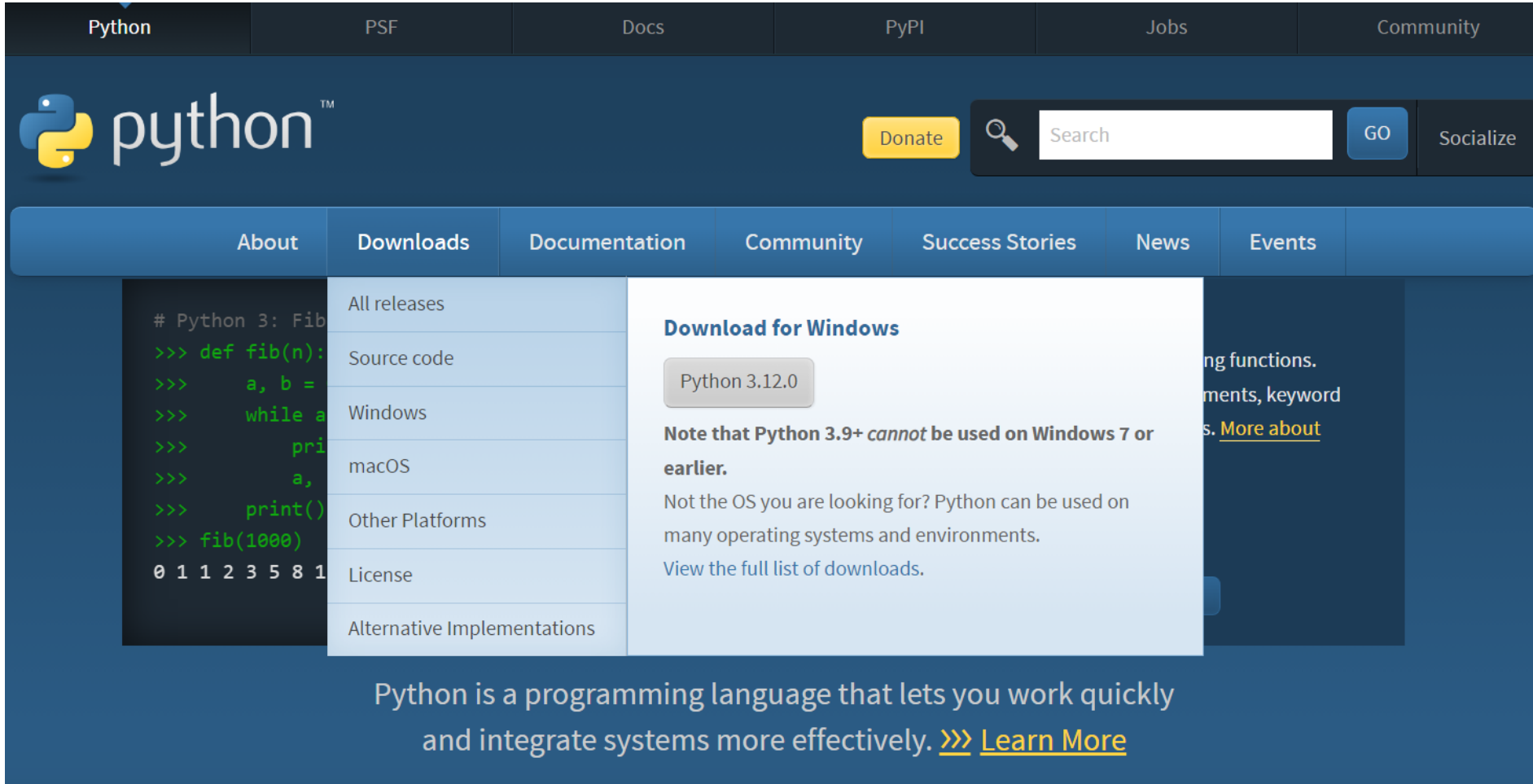
# 강의 실습 영상 참고

# 04. Python 설치

(2023년 11월 기준)

◆ Python은 웹 개발, 데이터 분석, 머신 러닝 등 다양한 작업에 사용되는 프로그래밍 언어

- 사이트 : <https://www.python.org/>



## 04. Python 설치

(2023년 11월 기준)

- ◆ Python은 웹 개발, 데이터 분석, 머신 러닝 등 다양한 작업에 사용되는 프로그래밍 언어
  - 사이트 : <https://www.python.org/>

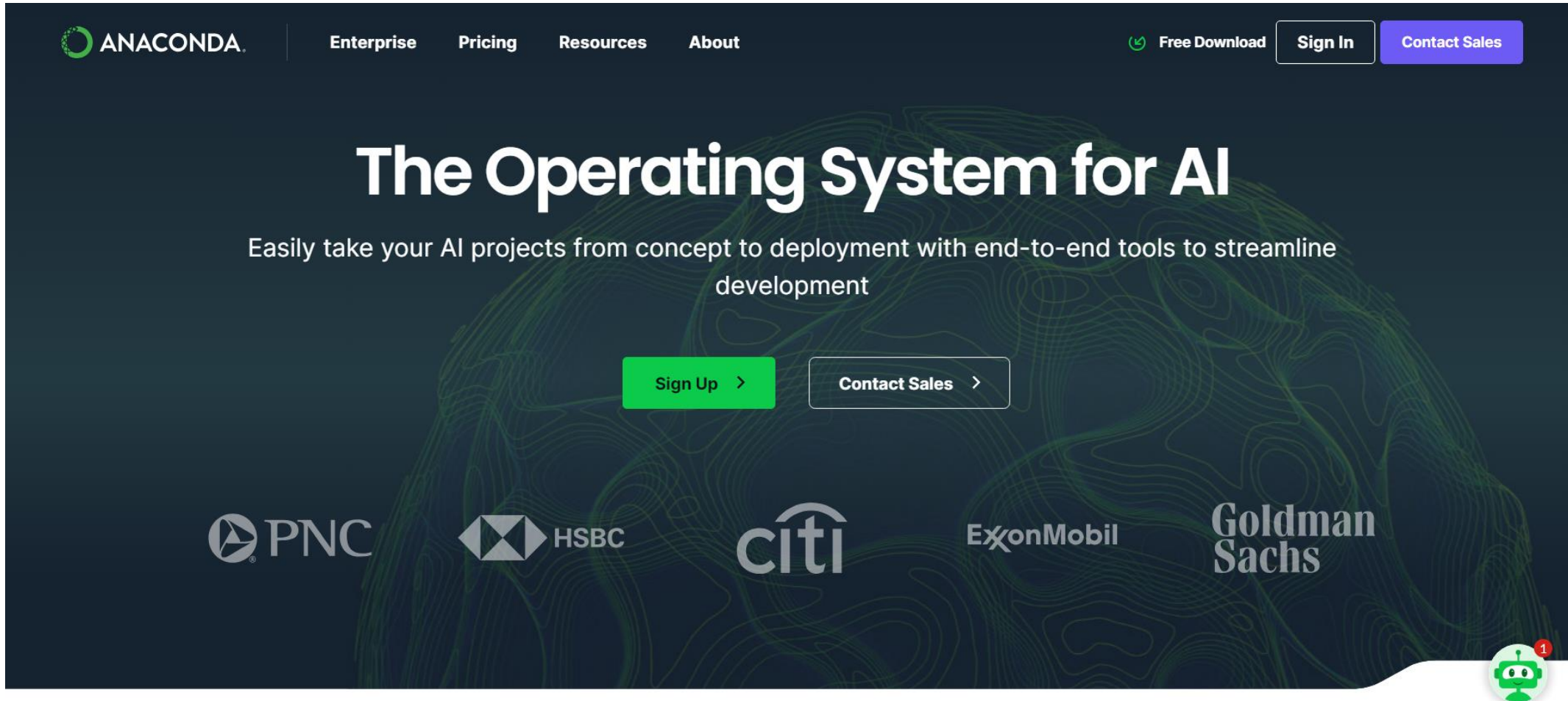
# 강의 실습 영상 참고

## 05. Anaconda 설치

(2023년 11월 기준)

### ◆ Anaconda 데이터 과학 및 머신 러닝 작업을 위한 오픈 소스 플랫폼

- 사이트 : <https://www.anaconda.com/>



## 05. Anaconda 설치

(2023년 11월 기준)

- ◆ Anaconda 데이터 과학 및 머신 러닝 작업을 위한 오픈 소스 플랫폼
  - 사이트 : <https://www.anaconda.com/>

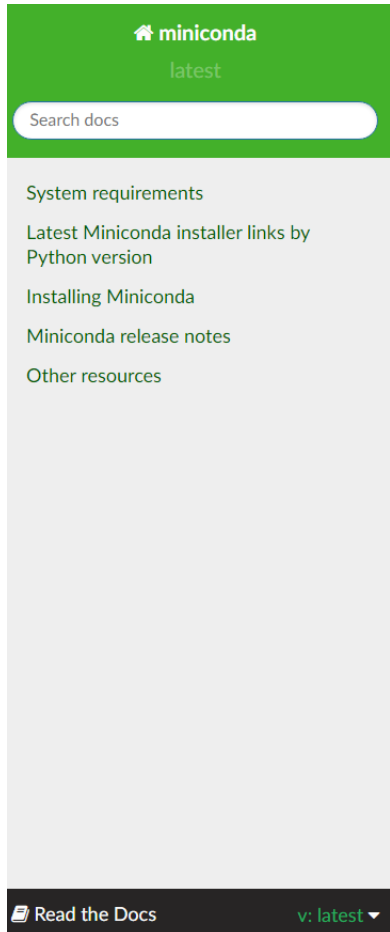
# 강의 실습 영상 참고

# 06. miniconda 설치

(2023년 11월 기준)

◆ miniconda는 아나콘다의 경량 버전으로, 데이터 과학 및 패키지 관리를 위한 필수 도구를 제공하는 작은 배포 버전

- 사이트 : <https://docs.conda.io/projects/miniconda/en/latest/>



🏠 / Miniconda

## Miniconda

Miniconda is a free minimal installer for conda. It is a small bootstrap version of Anaconda that includes only conda, Python, the packages they both depend on, and a small number of other useful packages (like pip, zlib, and a few others). If you need more packages, use the `conda install` command to install from thousands of packages available by default in Anaconda's public repo, or from other channels, like conda-forge or bioconda.

Is Miniconda the right conda install for you? The [Anaconda or Miniconda](#) page lists some reasons why you might want one installation over the other.

- [System requirements](#)
- [Latest Miniconda installer links by Python version](#)
- [Installing Miniconda](#)
- [Miniconda release notes](#)
- [Other resources](#)

## Latest Miniconda installer links

This list of installers is for the latest release of Python: 3.11.5. For installers for older versions of Python, see [Other installer links](#). For an archive of Miniconda versions, see <https://repo.anaconda.com/miniconda/>.

*Latest - Conda 23.9.0 Python 3.11.5 released October 19, 2023*

| Platform | Name                                   | SHA256 hash  |
|----------|--|--|
| Windows  | Miniconda3 Windows 64-bit              | 29e008bcaa3970bdf4f21362e545533d55a0f04c7ae99b93f20cb2b42f9250b1 |
| macOS    | Miniconda3 macOS Intel x86 64-bit bash | 4b60eb49cf8fea6272bd2060878ab02cbab187dff2fd732685c3c92a60b62ed  |

- ◆ miniconda는 아나콘다의 경량 버전으로, 데이터 과학 및 패키지 관리를 위한 필수 도구를 제공하는 작은 배포 버전
  - 사이트 : <https://docs.conda.io/projects/miniconda/en/latest/>

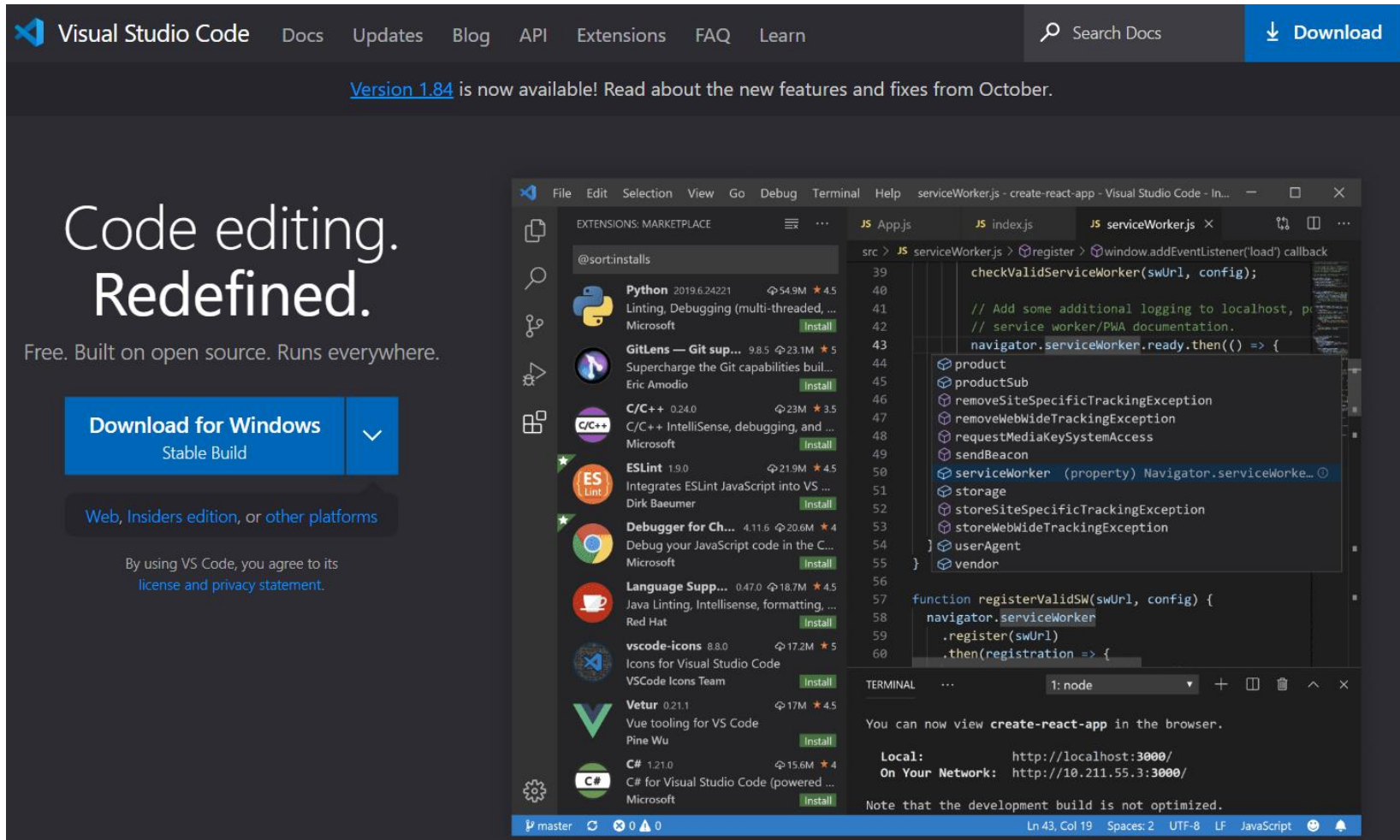
# 강의 실습 영상 참고



# 07. Visual Studio Code 설치

(2023년 11월 기준)

- ◆ Microsoft에서 개발한 인기 있는 무료 소스 코드 편집기로, Windows, Linux, MacOS에서 실행 가능
  - 사이트 : <https://code.visualstudio.com/>



## 07. Visual Studio Code 설치

(2023년 11월 기준)

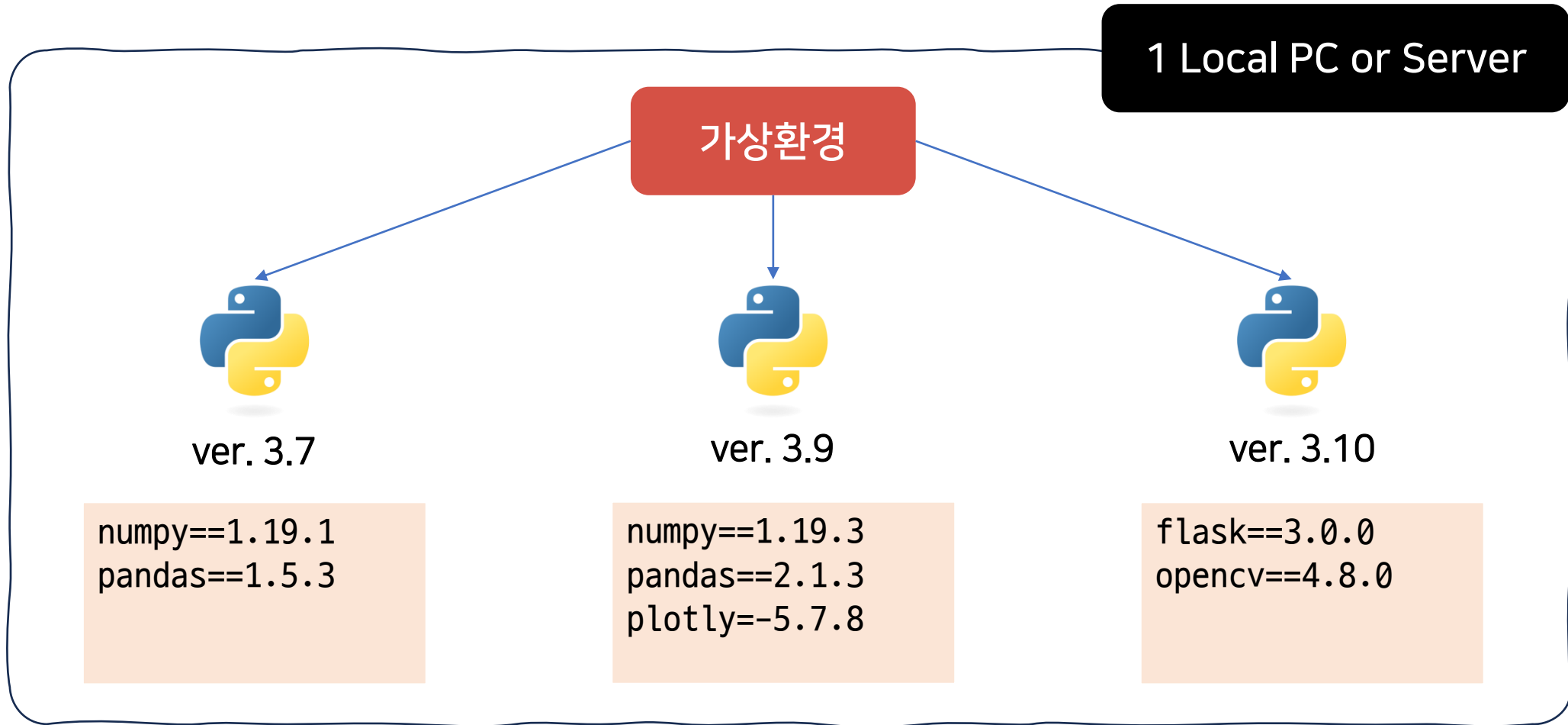
- ◆ Microsoft에서 개발한 인기 있는 무료 소스 코드 편집기로, Windows, Linux, MacOS에서 실행 가능
  - 사이트 : <https://code.visualstudio.com/>

# 강의 실습 영상 참고

## 08. Python 가상환경

(2023년 11월 기준)

- ◆ 가상환경은 프로젝트별로 독립적인 환경을 만들어주는 도구(virtualenv, conda, ...)
- ◆ 프로젝트 간 라이브러리 버전 충돌을 피하고 라이브러리들을 유연하게 관리할 수 있도록 해줌



## 08. Python 가상환경

(2023년 11월 기준)

- ◆ 가상환경은 프로젝트별로 독립적인 환경을 만들어주는 도구
- ◆ 프로젝트 간 라이브러리 버전 충돌을 피하고 라이브러리들을 유연하게 관리할 수 있도록 해줌

# 강의 실습 영상 참고

◆ 주요 명령어는 다음과 같다.

---

```
git config --global user.email "your_email@email.com"
```

```
git config --global user.name "yourusername"
```

---

- git 설정을 변경하는 것으로 전역으로 사용자의 이름과 이메일 주소를 설정 (일종의 로그인 기능)

---

```
git add .
```

```
git commit -m "your message"
```

```
git push
```

---

| 주요 명령어     | 설명  |
|------------|---|
| git add .  | 코드 수정 후, 변경 사항을 모두 준비하는 단계를 말하며, `.`은 현재 디렉토리를 의미 |
| git commit | 이전 단계에서 변경 사항으로 새로운 변경 사항으로 새 커밋                  |
| git push   | 커밋된 변경 사항을 원격 Git저장소(Github)에 업로드 함               |

# 강의 실습 영상 참고

# 10. Python 라이브러리 설치

(2023년 11월 기준)

## ◆ Python 라이브러리 설치방법 (pip)

---

```
pip install name_of_library
```

---

## ◆ Python 라이브러리 설치방법 (특정버전)

---

```
pip install name_of_library==버전번호
```

---

## ◆ Python 라이브러리 설치방법 (requirements.txt)

- requirements.txt 파일에 설치할 라이브러리 목록을 작성

---

```
pip install -r requirements.txt
```

---

# 강의 실습 영상 참고



# 파이썬 기초 문법

with  python™

## ◆ 숫자형(Number)

| 항목         | 표현 예시                               |
|------------|-------------------------------------|
| 정수 (int)   | 1, 2, 3, 123, 1345, 0, -123, -1, -2 |
| 실수 (float) | 12.34, 456.189, 1.2e+5*, 1.2e-5**   |

☒ Scientific Notation  
\*  $1.2e+5 = 1.2 \times 10^5 = 120000$   
\*\*  $1.2e-5 = 1.2 \times 10^{-5} = 0.000012$

## ◆ 문자열(String)

| 주요 기능 | 정의   |
|-------|--|
| 더하기   | 더하기 순서대로 문자열을 하나로 연결한다는 것을 의미                |
| 곱하기   | 문자열을 반복해서 연결하는 것을 의미                         |
| 인덱싱   | 특정 위치에 있는 문자만 지정하여 일부를 추출함. 인덱스번호는 0번째부터 시작함 |
| 슬라이싱  | 특정 위치에 있는 범위를 지정하여 전체 또는 1개 이상의 문자를 추출함      |

## 02. 문자열 주요 메서드

(2023년 11월 기준)

◆ 메서드, a는 임의의 문자열이 저장된 객체

| 주요 메서드                           | 메서드 설명                                       |
|----------------------------------|--|
| <code>a.count('p')</code>        | 특정 문자 ('p')가 몇 개가 있는지 확인                     |
| <code>a.find('p')</code>         | 특정 문자 ('p')가 첫번째로 등장한 위치(인덱스) 번호를 확인         |
| <code>a.upper()</code>           | 영어 문자를 대문자로 변환                               |
| <code>a.lower()</code>           | 영어 문자를 소문자로 변환                               |
| <code>a.lstrip()</code>          | 문자열 왼쪽에 있는 공백 제거                             |
| <code>a.rstrip()</code>          | 문자열 오른쪽에 있는 공백 제거                            |
| <code>a.strip()</code>           | 문자열 양쪽에 있는 공백 제거                             |
| <code>a.replace("x", "y")</code> | 문자열에 포함된 x를 y로 변경                            |
| <code>b = ', '.join(a)</code>    | 기존 문자열(a)에 특정 문자(예: ,)를 문자열 사이마다 삽입할 수 있음    |
| <code>a.split(조건)</code>         | 문자열을 특정 조건에 따라 나눠서 다수의 문자열로 구성된 리스트 자료형으로 변경 |

◆ 리스트는 숫자형, 문자열 자료형을 하나의 집합으로 구성할 수 있음

---

```
1 : []  
2 : [2,4,6,8,10]  
3 : ['빅데이터', '분석', '기사']  
4 : [1, 2, 3, ['빅데이터', '분석', '기사']] # 중첩리스트(Nested List)  
5 : ['빅데이터', '분석', '기사', 1, 2, 3]
```

---

◆ 문자열과 마찬가지로 더하기, 곱하기 같은 사칙연산을 활용해서 리스트를 합치거나 반복 가능

---

```
1 : a = [1, 2, 3]  
2 : b = [4, 5, 6]  
3 : c = a + b  
4 : d = c * 3  
5 : print(d)
```

---

## 03. 리스트

(2023년 11월 기준)

- ◆ 문자열과 마찬가지로 인덱싱과 슬라이싱 가능
- ◆ 중첩된 리스트에 대해서도 인덱싱과 슬라이싱 가능

---

```
1 : a = [1, 2, 3, ["a", "b", "c", "d"]]  
2 : print(a)  
3 : print(a[3][0]) # 중첩된 리스트 인덱싱 및 슬라이싱
```

---

```
[1, 2, 3, ['a', 'b', 'c', 'd']]  
a
```

◆ a는 임의의 리스트가 저장된 객체

| 주요 메서드         | 메서드 설명  |
|----------------|---|
| a.append(x)    | 리스트의 맨 마지막 요소 위치에 x라는 요소를 추가한다.                   |
| a.sort()       | 리스트의 요소들을 정렬하는 함수가 있다.                            |
| a.index(x)     | 리스트 내의 x값의 위치에 해당하는 인덱스 값을 반환한다.                  |
| a.insert(x, y) | x값은 인덱스 번호, y값은 특정 값으로 x 인덱스 위치에 y 값을 추가한다.       |
| a.remove(x)    | x는 특정값을 의미하며, 리스트에서 x 값을 제거한다.                    |
| a.pop(x)       | x는 인덱스 번호를 의미, 리스트에서 해당 인덱스의 값을 반환하며, 리스트에서 제거한다. |
| a.count(x)     | 특정요소 x의 개수를 계산한다.                                 |

## 04. 튜플(Tuple) 자료형

(2023년 11월 기준)

◆ 튜플과 리스트는 비슷한 역할을 수행함. 대괄호가 아닌 소괄호를 사용해야 함.

---

```
1 : (1, 2, 3)
2 : (1, )
3 : 1, 2, 3
4 : (1, 2, 3, ('빅데이터', '분석', '기사')) # 중첩튜플(Nested Tuple)
5 : ('빅데이터', '분석', '기사', 1, 2, 3)
```

---

◆ 리스트와 마찬가지로 더하기, 곱하기 같은 사칙연산을 활용해서 튜플을 합치거나 반복 가능

◆ 인덱싱과 슬라이싱 가능

◆ 리스트와의 가장 큰 차이점

- 리스트 : 요소의 생성, 삭제, 수정 가능
- 튜플 : **요소 변경 불가능**

## 05. 딕셔너리(Dictionary 자료형)

(2023년 11월 기준)

### ◆ 키(Key)와 값(Value)으로 이루어진 자료형

---

```
1 : a = {'name' : 'evan', 'age' : 30, 'birth' : [4, 30]}
```

---

### ◆ Value 값을 구하기 위해서는 Key를 활용해야 함.

---

```
1 : a = {'name' : 'evan', 'age' : 30, 'birth' : [4, 30]}
2 : print(a['name'])
3 : print(a['birth'])
```

---

```
evan
[4, 30]
```



## 05. 딕셔너리(Dictionary 자료형) - 메서드

(2023년 11월 기준)

◆ 주요 메서드는 다음과 같음, a는 임의의 딕셔너리로 저장된 객체

| 주요 메서드      | 메서드 설명                                     |
|-------------|--|
| a.keys()    | 딕셔너리의 키(Key)의 리스트를 추출할 수 있음                |
| a.values()  | 딕셔너리의 값(Value)의 리스트를 추출할 수 있음              |
| a.items()   | 딕셔너리의 Key, Value를 튜플 구조로 묶고, 리스트로 추출할 수 있음 |
| a.get(x, y) | 딕셔너리의 x의 키가 존재하지 않을 때 y값 반환하도록 처리          |

◆ If문 : 조건문1을 테스트하여 참이면 if문, 아니면 elif 조건문2를 테스트하여 참이면 코드

---

```
1  :  if 조건문1:
2  :      코드 실행 1 # 들여쓰기는 공백(Spacebar) 또는 탭(Tab) 사용
3  :      코드 실행 2
4  :  elif 조건문2:
5  :      코드 실행 1
5  :      코드 실행 2
6  :  else:
7  :      코드 실행 1
8  :      코드 실행 2
9  :  A = [1, 2, 3]
10 :  ...
```

---

◆ 조건문 작성 방법

| 조건문 표현 방법 | 조건문 의미         |
|-----------|----------------|
| $x < y$   | x가 y보다 작다면     |
| $x > y$   | x가 y보다 크다면     |
| $x == y$  | x와 y가 같다면      |
| $x != y$  | x와 y가 같지 않다면   |
| $x >= y$  | x가 y보다 크거나 같다면 |
| $x <= y$  | x가 y보다 작거나 같다면 |

## 06. 파이썬 제어문

(2023년 11월 기준)

### ◆ 조건문 비교 연산자 (AND)

| X     | Y     | Result |
|-------|-------|--------|
| True  | True  | True   |
| True  | False | False  |
| False | True  | False  |
| False | False | False  |

### ◆ 조건문 비교 연산자 (OR)

| X     | Y     | Result |
|-------|-------|--------|
| True  | True  | True   |
| True  | False | True   |
| False | True  | True   |
| False | False | False  |

◆ not x : x가 거짓이면 참이다

## 06. 파이썬 제어문

(2023년 11월 기준)

◆ while 반복문 : 조건문이 참인 동안에 while 아래의 문장을 반복해서 수행함

```
1  : a = 0
2  :
3  : while a < 3:
4  :     print(f'현재 a값은 {a} 입니다.')
5  :     a = a + 1
6  :
7  : print("종료")
```

```
현재 a값은 0 입니다.
현재 a값은 1 입니다.
현재 a값은 2 입니다.
종료
```

## 06. 파이썬 제어문

(2023년 11월 기준)

◆ for 반복문 : 리스트, 문자열, 튜플 등 0번째 인덱스부터 마지막 인덱스까지 차례로 변수에 대입되어 명령문이 실행

```
1 : numbers = [100, 200, 300]
2 :
3 : for num in numbers:
4 :     print(num)
5 :
6 : print("종료")
```

```
100
200
300
종료
```

## 07. 사용자 정의 함수

(2023년 11월 기준)

◆ def문 : 함수 이름을 임의로 만들고, 이름 뒤 괄호 안의 매개변수는 입력으로 전달되는 값을 받는 변수

---

```
1  :  def 함수이름(매개변수1, 매개변수2, ...)  
2  :      코드 1  
3  :      코드 2  
4  :      ...  
5  :      코드 N  
6  :      return 결과값
```

---

# 강의 실습 영상 참고



# 파이썬 데이터 분석 라이브러리

with  python™

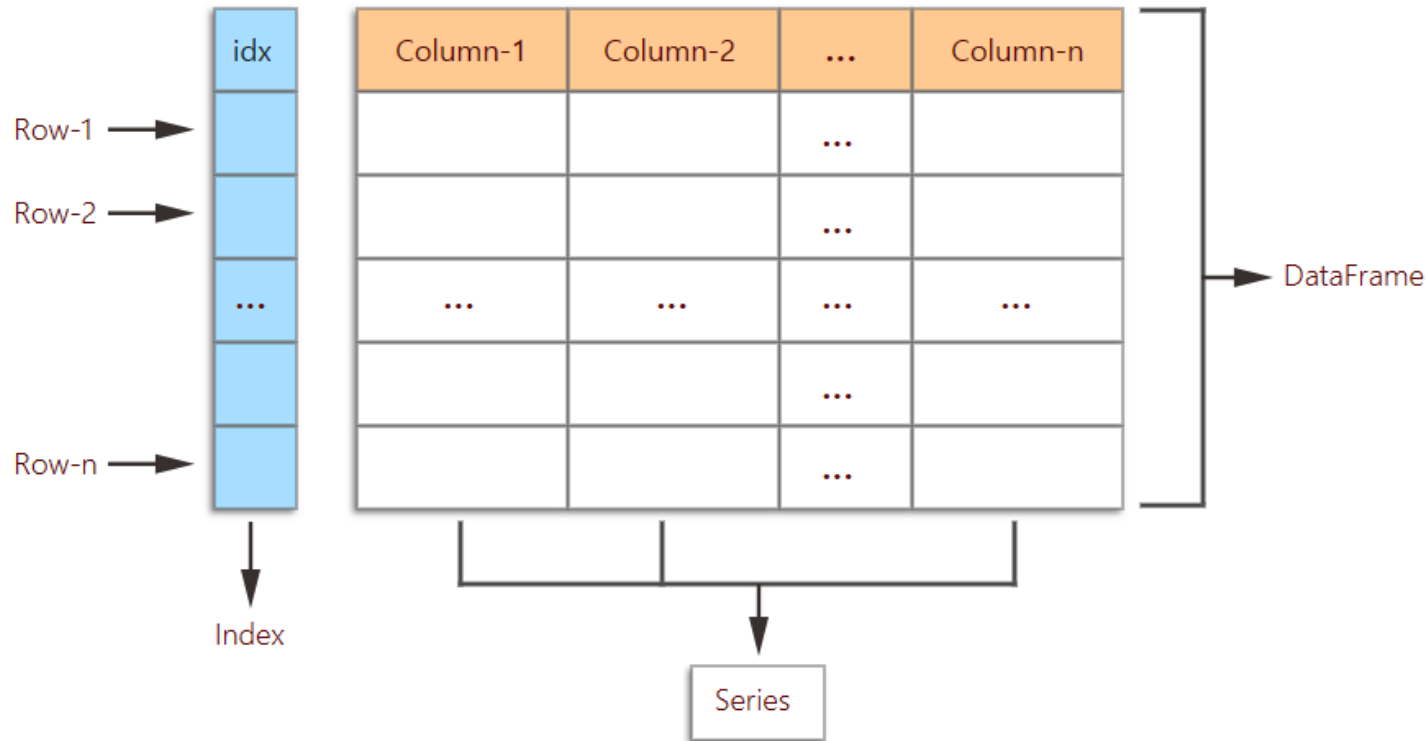
# 03. pandas

(2023년 11월 기준)

## ◆ Series vs DataFrame

- ✓ Series : 1차원 배열의 형태를 가진 자료 구조, index와 value 확인 가능
- ✓ DataFrame : 2차원 행렬 구조의 테이블 형태로 구성, index와 복수의 컬럼(Column)이 존재함

Pandas Data structure



출처 : [https://www.w3resource.com/w3r\\_images/pandas-data-structure.svg](https://www.w3resource.com/w3r_images/pandas-data-structure.svg)

## ◆ DataFrame 컬럼의 데이터 확인 및 변경

| 구분       | 내용   |
|----------|--|
| int      | 정수형, 소수점을 가지지 않은 숫자                            |
| float    | 실수형, 소수점 이하의 값을 가진 숫자                          |
| bool     | 부울형, True 혹은 False로 이루어짐                       |
| datetime | 날짜와 시간 표현                                      |
| category | 카테고리, 범주형 변수일 경우 사용                            |
| object   | 문자열 & 복합형, 위의 형식으로 정할 수 없거나 여러 형식이 섞여 있는 경우 사용 |

## ◆ Series vs DataFrame

- ✓ Series : 1차원 배열의 형태를 가진 자료 구조, index와 value 확인 가능
- ✓ DataFrame : 2차원 행렬 구조의 테이블 형태로 구성, index와 복수의 컬럼(Column)이 존재함

## ◆ 주요 메서드 - 데이터 살펴보기, data는 pandas DataFrame을 의미한다.

| 주요 메서드          | 설명   |
|-----------------|--|
| data.head(n)    | 데이터프레임의 첫번째 행부터 순차적으로 n개까지의 행을 반환                |
| data.tail(n)    | 데이터프레임의 마지막 행부터 역순으로 n개까지의 행을 반환                 |
| data.shape      | 데이터프레임의 행과 컬럼 정보를 튜플 형태로 반환                      |
| data.info()     | 데이터프레임의 컬럼, Non-Null 데이터 개수, 컬럼의 타입              |
| data.describe() | 컬럼별 숫자형 데이터의 개수, 평균값, 표준편차, 최솟값, 사분위수값, 최댓값을 표현함 |
| value_counts()  | 해당 컬럼 값의 유형과 건수를 확인할 수 있음. 데이터의 분포도를 확인하는 데 유용함  |

## ◆ Series vs DataFrame

- ✓ Series : 1차원 배열의 형태를 가진 자료 구조, index와 value 확인 가능
- ✓ DataFrame : 2차원 행렬 구조의 테이블 형태로 구성, index와 복수의 컬럼(Column)이 존재함

## ◆ 주요 메서드 - 데이터 살펴보기, data는 pandas DataFrame을 의미한다.

## ◆ 주요 작업

- ✓ 컬럼 생성과 수정
- ✓ 데이터프레임 데이터 삭제

---

```
1 : data.drop("column1", axis=1) # column1 삭제
2 : data.drop(["column1", "column2"], axis=1) # column1, column2 삭제
3 : data.drop([0, 1, 2], axis=0) # 행 인덱스 0, 1, 2 삭제
```

---

## ◆ Series vs DataFrame

- ✓ Series : 1차원 배열의 형태를 가진 자료 구조, index와 value 확인 가능
- ✓ DataFrame : 2차원 행렬 구조의 테이블 형태로 구성, index와 복수의 컬럼(Column)이 존재함

## ◆ 주요 메서드 - 데이터 살펴보기, data는 pandas DataFrame을 의미한다.

## ◆ 주요 작업

- ✓ 컬럼 생성과 수정
- ✓ 데이터프레임 데이터 삭제
- ✓ 데이터 조회 (컬럼명, 슬라이싱, 논리형 인덱싱)

# 03. pandas

(2023년 11월 기준)

## ◆ Input/Output

- ✓ 다양한 파일 읽기 및 쓰기 제공(CSV, JSON, HTML 등)
- ✓ SAS, SPSS, SQL, Google BigQuery, STATA 등도 제공

| 파일구분    | Reading Data                   | Writing Data                        |
|---------|--------------------------------|-------------------------------------|
| CSV     | <code>pd.read_csv()</code>     | <code>DataFrame.to_csv()</code>     |
| EXCEL   | <code>pd.read_excel()</code>   | <code>DataFrame.to_excel()</code>   |
| JSON    | <code>pd.read_json()</code>    | <code>DataFrame.to_json()</code>    |
| HTML    | <code>pd.read_html()</code>    | <code>DataFrame.to_html()</code>    |
| SQL     | <code>pd.read_sql()</code>     | <code>DataFrame.to_sql()</code>     |
| Parquet | <code>pd.read_parquet()</code> | <code>DataFrame.to_parquet()</code> |
| HDFT    | <code>pd.read_hdf()</code>     | <code>DataFrame.to_hdf()</code>     |

◆ iloc vs loc 차이

| iloc   | loc  |
|--|--|
| Integer-location based (위치 기반)   | Label(s) or Boolean Array based (명칭 기반)  |
| <code>data.iloc[row_index, column_index]</code>  | <code>data.loc[row_label, column_label]</code>   |
| input 형태 <ul style="list-style-type: none"><li>- integer</li><li>- List or Array [4, 3, 0]</li><li>- Slicing 1:7</li><li>- Boolean Array</li></ul> | input 형태 <ul style="list-style-type: none"><li>- Single Label 5, 'a'</li><li>- List or Array of Label, ["a", "b", "c"]</li><li>- Slicing 'a' : 'c'</li></ul> |



◆ 데이터 정렬 및 집계함수

| 주요 메서드             | 설명   |
|--------------------|--|
| data.sort_values() | DataFrame, Series의 정렬을 할 때 사용한다. SQL의 order by 키워드와 유사함. |
| data.sum()         | DataFrame의 모든 컬럼 각각 합계가 나타남.                             |

```
1 : Import seaborn as sns
2 : iris = sns.load_dataset('iris')
3 : iris[['sepal_length', 'sepal_width', 'petal_length']].sum()
```

```
sepal_length    876.5
sepal_width      458.6
petal_length     563.7
dtype: float64
```

## ◆ 데이터 정렬 및 집계함수

| 주요 메서드                          | 설명   |
|---------------------------------|--|
| <code>data.sort_values()</code> | DataFrame, Series의 정렬을 할 때 사용한다. SQL의 order by 키워드와 유사함. |
| <code>data.sum()</code>         | DataFrame의 모든 컬럼에 대하여 각각 합계가 나타남.                        |
| <code>data.min()</code>         | DataFrame의 모든 컬럼에 대하여 최솟값이 나타남.                          |
| <code>data.max()</code>         | DataFrame의 모든 컬럼에 대하여 최댓값이 나타남.                          |
| <code>data.count()</code>       | DataFrame의 모든 컬럼에 대하여 행의 개수가 나타남                         |
| <code>data.mean()</code>        | DataFrame의 모든 컬럼에 대하여 평균값이 나타남                           |
| <code>data.median()</code>      | DataFrame의 모든 컬럼에 대하여 중간값이 나타남                           |

◆ 데이터 요약 및 기술통계량

| 주요 메서드                     | 설명   |
|----------------------------|--|
| describe()                 | 데이터의 기초 통계량(평균, 표준편차 각 컬럼의 사분위수 등) 함수      |
| describe(include=[object]) | Object 컬럼에 count, unique, top, freq 값을 출력함 |

```
1 : import seaborn as sns
2 : iris = sns.load_dataset("iris")
3 : iris.describe(include=[object])
```

|       | sepal_length | sepal_width | petal_length | petal_width |
|-------|--------------|-------------|--------------|-------------|
| count | 150.000000   | 150.000000  | 150.000000   | 150.000000  |
| mean  | 5.843333     | 3.057333    | 3.758000     | 1.199333    |
| std   | 0.828066     | 0.435866    | 1.765298     | 0.762238    |
| min   | 4.300000     | 2.000000    | 1.000000     | 0.100000    |
| 25%   | 5.100000     | 2.800000    | 1.600000     | 0.300000    |
| 50%   | 5.800000     | 3.000000    | 4.350000     | 1.300000    |
| 75%   | 6.400000     | 3.300000    | 5.100000     | 1.800000    |
| max   | 7.900000     | 4.400000    | 6.900000     | 2.500000    |

```
1 : import seaborn as sns
2 : iris = sns.load_dataset("iris")
3 : iris.describe(include=[object])
```

|        | species |
|--------|---------|
| count  | 150     |
| unique | 3       |
| top    | setosa  |
| freq   | 50      |

◆ rename

| 주요 메서드                                      | 설명                  |
|---|---------------------|
| df.rename(columns={'old name': 'new name'}) | 컬럼명을 변경할 때 사용하는 메서드 |

```
1 : sample_df = sample_df.rename(columns={'ZN': 'landZone'})
2 : sample_df.head()
```

|   | CRIM    | ZN   | INDUS | CHAS |
|---|---------|------|-------|------|
| 0 | 0.00632 | 18.0 | 2.31  | 0.0  |
| 1 | 0.02731 | 0.0  | 7.07  | 0.0  |
| 2 | 0.02729 | 0.0  | 7.07  | 0.0  |
| 3 | 0.03237 | 0.0  | 2.18  | 0.0  |
| 4 | 0.06905 | 0.0  | 2.18  | 0.0  |

|   | CRIM    | landZone | INDUS | CHAS |
|---|---------|----------|-------|------|
| 0 | 0.00632 | 18.0     | 2.31  | 0.0  |
| 1 | 0.02731 | 0.0      | 7.07  | 0.0  |
| 2 | 0.02729 | 0.0      | 7.07  | 0.0  |
| 3 | 0.03237 | 0.0      | 2.18  | 0.0  |
| 4 | 0.06905 | 0.0      | 2.18  | 0.0  |

## ◆ value\_counts() : 고유 값의 개수를 반환함

```
df.value_counts(normalize=False)
```

- normalize True로 설정 시, 각 객체는 고유값의 상대적인 비율로 조회됨

```
1 : df_boston['RAD'].value_counts()
```

```
24.0    132
5.0      115
4.0      110
3.0       38
6.0       26
2.0       24
8.0       24
1.0       20
7.0       17
```

```
Name: RAD, dtype: int64
```

```
1 : df_boston['RAD'].value_counts(normalize=True)
```

```
24.0    0.260870
5.0      0.227273
4.0      0.217391
3.0      0.075099
6.0      0.051383
2.0      0.047431
8.0      0.047431
1.0      0.039526
7.0      0.033597
```

```
Name: RAD, dtype: float64
```

◆ `isin()` : 데이터 필터링, 데이터 프레임의 각 요소가 값에 포함되어 있는지 판단

`df.isin(values)`

- `values`      iterables(i.e., List), Series(index), DataFrame(index & column labels) or dict(keys)

```
1 : numbers = [1.0, 7.0]
2 : filtered_df = df_boston[df_boston['RAD'].isin(numbers)]
3 : filtered_df[['CRIM', 'RAD']].head(6)
```

|     | CRIM    | RAD |
|-----|---------|-----|
| 0   | 0.00632 | 1.0 |
| 193 | 0.02187 | 1.0 |
| 194 | 0.01439 | 1.0 |
| 244 | 0.20608 | 7.0 |
| 245 | 0.19133 | 7.0 |
| 246 | 0.33983 | 7.0 |

# 03. pandas

(2023년 11월 기준)

## ◆ 날짜 데이터 처리

- ✓ 날짜와 시간을 다루기 위해서는 datetime 라이브러리 활용
- ✓ 날짜 형식으로 변환방법 (pandas DataFrame)

```
df['date'] = pd.to_datetime(df['date'], format="%Y-%m-%d %H:%M:%S")
```

## ◆ datetime 객체 사용 예시

```
df['date'].dt.year
```

| 구분      | 내용   |
|---------|--|
| year    | 객체 datetime의 연도를 추출함                           |
| month   | 객체 datetime의 월을 추출함                            |
| day     | 객체 datetime의 일을 추출함                            |
| hour    | 객체 datetime의 시간을 추출함                           |
| weekday | 객체 datetime 날짜의 요일을 추출함 (Monday = 0, Sunday=6) |

그 외 : <https://pandas.pydata.org/docs/reference/series.html#datetimelike-properties>

# 03. pandas

(2023년 11월 기준)

## ◆ Timedelta

- ✓ 두 날짜 또는 시간 간의 차이, 즉 기간을 표현함
- ✓ 두개의 datetime 컬럼(i.e., 출발시간 - 도착시간) 연산 시, **Timedelta 객체**로 자동 변환

---

```
df['이동시간'] = df['도착시간'] - df['출발시간']
```

---

```
import pandas as pd

df = pd.DataFrame({
    '출발시간' : [pd.to_datetime('2023-01-01 08:00:00')],
    '도착시간' : [pd.to_datetime('2023-01-01 09:30:00')],
})

df['이동시간'] = df['도착시간'] - df['출발시간']
print(df.info())
```

```
df.head(1)
```

|   | 출발시간                | 도착시간                | 이동시간            |
|---|---------------------|---------------------|-----------------|
| 0 | 2023-01-01 08:00:00 | 2023-01-01 09:30:00 | 0 days 01:30:00 |

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 3 columns):
#   Column   Non-Null Count  Dtype  
---  -
0   출발시간   1 non-null      datetime64[ns]
1   도착시간   1 non-null      datetime64[ns]
2   이동시간   1 non-null      timedelta64[ns]
dtypes: datetime64[ns](2), timedelta64[ns](1)
memory usage: 152.0 bytes
None
```



# 03. pandas

(2023년 11월 기준)

## ◆ Timedelta

- ✓ 두 날짜 또는 시간 간의 차이, 즉 기간을 표현함
- ✓ 두개의 datetime 컬럼(i.e., 출발시간 - 도착시간) 연산 시, **Timedelta 객체**로 자동 변환

```
df.head(1)
```

|   | 출발시간                | 도착시간                | 이동시간            |
|---|---------------------|---------------------|-----------------|
| 0 | 2023-01-01 08:00:00 | 2023-01-01 09:30:00 | 0 days 01:30:00 |

df['이동시간'].dt.days

| 구분                             | 내용  |
|--------------------------------|---|
| days                           | 0   |
| total_seconds()                | 5400(초) = hours * 3600 + minutes * 60 + seconds |
| dt.total_seconds() / 60        | 90(분) = 5400 / 60                               |
| dt.total_seconds() / (60 * 60) | 1.5(시간) = 5400 / (60 * 60)                      |

그 외 : <https://pandas.pydata.org/docs/reference/api/pandas.Timedelta.html>

## ◆ shift()

- ✓ 인덱스는 그대로 두고 데이터만 이동이 가능함
- ✓ 현재 기준 앞으로 또는 지정된 기간만큼 이동함

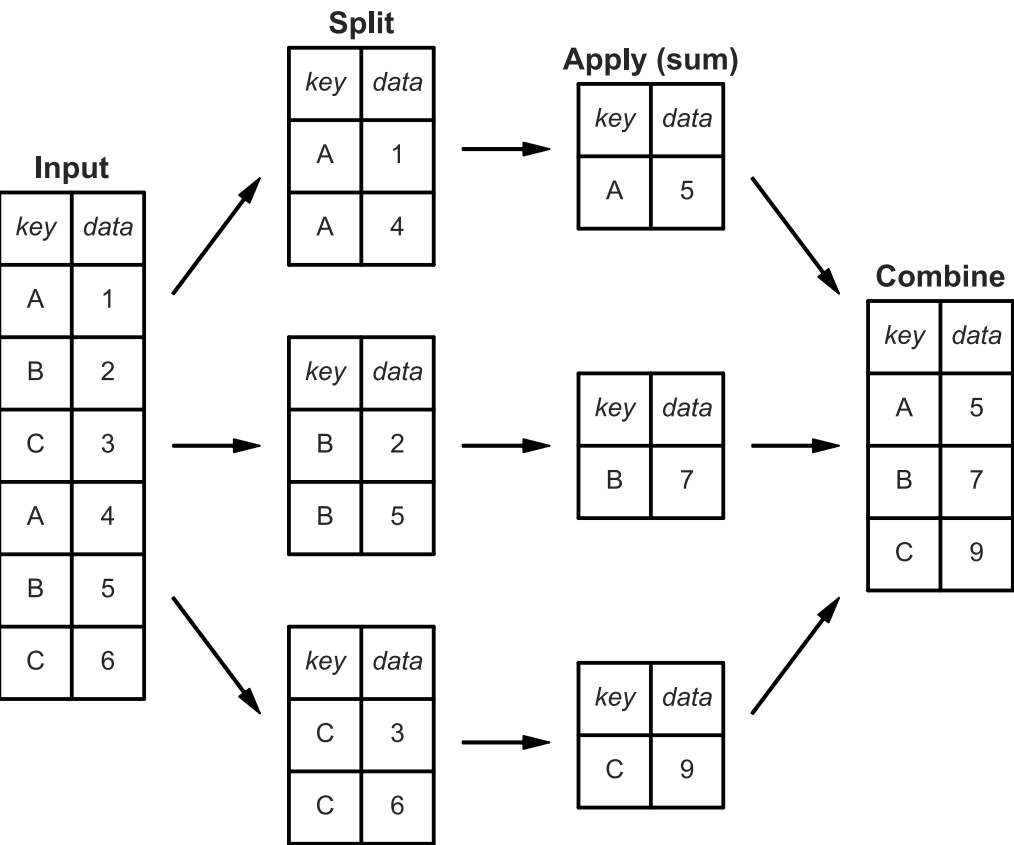
```
df.shift(periods=1, fill_value=object, optional)
```

- periods      데이터가 shift 이동할 기간의 숫자, 양수 또는 음수가 올 수 있음
- fill\_value    shift 실행 시, 발생할 결측값을 채워주는 scala 값

```
temp_df['shifted_v1'] = temp_df['price'].shift(periods=1, fill_value=0).astype(int)
temp_df['shifted_v2'] = temp_df['price'].shift(periods=2, fill_value=0).astype(int)
```

|   | datesold   | price  | shifted_v1 | shifted_v2 |
|---|------------|--------|------------|------------|
| 0 | 2007-02-07 | 525000 | 0          | 0          |
| 1 | 2007-02-27 | 290000 | 525000     | 0          |
| 2 | 2007-03-07 | 328000 | 290000     | 525000     |
| 3 | 2007-03-09 | 380000 | 328000     | 290000     |
| 4 | 2007-03-21 | 310000 | 380000     | 328000     |

◆ groupby 원리



◆ 집계함수 종류

| 주요 메서드     | 설명    |
|------------|-------|
| count()    | 값의 개수 |
| sum()      | 값들의 합 |
| min()      | 최솟값   |
| max()      | 최댓값   |
| mean()     | 평균    |
| median()   | 중앙값   |
| std()      | 표준편차  |
| var()      | 분산    |
| quantile() | 분위수   |
| first()    | 첫번째 값 |
| last()     | 마지막 값 |

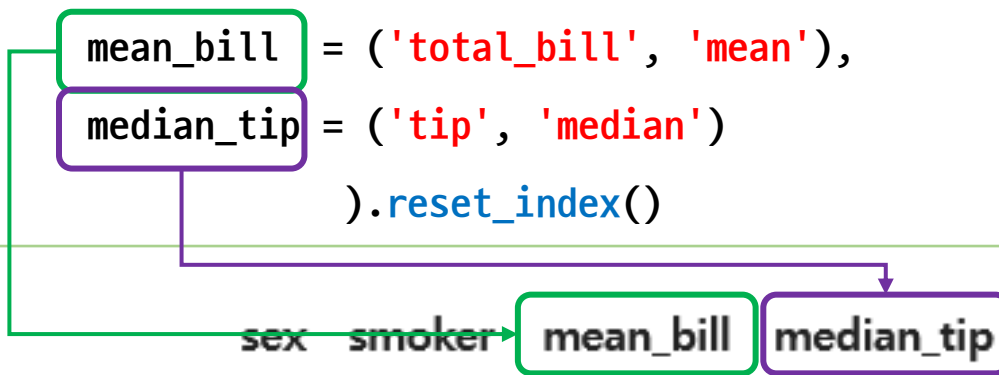
<https://jakevdp.github.io/blog/2017/03/22/group-by-from-scratch/>

# 03. pandas

(2023년 11월 기준)

## ◆ groupby 원리

```
1 : tips.groupby(  
2 :     ['sex', 'smoker']  
3 :     ).agg(  
4 :     mean_bill = ('total_bill', 'mean'),  
5 :     median_tip = ('tip', 'median')  
6 :     ).reset_index()
```



|   | sex    | smoker | mean_bill | median_tip |
|---|--------|--------|-----------|------------|
| 0 | Male   | Yes    | 22.284500 | 3.00       |
| 1 | Male   | No     | 19.791237 | 2.74       |
| 2 | Female | Yes    | 17.977879 | 2.88       |
| 3 | Female | No     | 18.105185 | 2.68       |

|   | total_bill | tip  | sex    | smoker | day | time   | size |
|---|------------|------|--------|--------|-----|--------|------|
| 0 | 16.99      | 1.01 | Female | No     | Sun | Dinner | 2    |
| 1 | 10.34      | 1.66 | Male   | No     | Sun | Dinner | 3    |
| 2 | 21.01      | 3.50 | Male   | No     | Sun | Dinner | 3    |
| 3 | 23.68      | 3.31 | Male   | No     | Sun | Dinner | 2    |
| 4 | 24.59      | 3.61 | Female | No     | Sun | Dinner | 4    |

# 03. pandas

(2023년 11월 기준)

## ◆ 데이터프레임 결측치 처리

- ✓ 컬럼에 값이 없는 Null을 의미한다.
- ✓ 결측치를 처리하지 않으면 작동하지 않으므로 이 값을 반드시 다른 값으로 대체해야 함

| 주요 메서드        | 설명                               |
|---------------|----------------------------------|
| data.isna()   | 결측치 여부를 확인하는 함수, 반환값은 True/False |
| data.fillna() | 결측치를 채우는 함수                      |
| data.dropna() | 결측값이 포함된 모든 행을 삭제                |

```
df['몸무게'] = df['몸무게'].fillna(df['몸무게'].mean())
```

|   | 연도   | 키     | 몸무게  | 시력  | 병결  |
|---|------|-------|------|-----|-----|
| 0 | 2017 | 160.0 | 53.0 | 1.2 | NaN |
| 1 | 2018 | 162.0 | 52.0 | NaN | NaN |
| 2 | 2019 | 165.0 | NaN  | 1.2 | NaN |
| 3 | 2020 | NaN   | 50.0 | 1.2 | 2.0 |
| 4 | 2021 | NaN   | 51.0 | 1.1 | NaN |
| 5 | 2022 | 166.0 | 54.0 | 0.8 | 1.0 |

|   | 연도   | 키     | 몸무게  | 시력  | 병결  |
|---|------|-------|------|-----|-----|
| 0 | 2017 | 160.0 | 53.0 | 1.2 | NaN |
| 1 | 2018 | 162.0 | 52.0 | NaN | NaN |
| 2 | 2019 | 165.0 | 52.0 | 1.2 | NaN |
| 3 | 2020 | NaN   | 50.0 | 1.2 | 2.0 |
| 4 | 2021 | NaN   | 51.0 | 1.1 | NaN |
| 5 | 2022 | 166.0 | 54.0 | 0.8 | 1.0 |

# 03. pandas

(2023년 11월 기준)

## ◆ 데이터 재구조화

✓ pivot\_table : 많은 양의 데이터에서 필요한 자료만을 뽑아 데이터를 재구조화 할 수 있음

```
df.pivot_table(index=["ID"], columns=["반"], values = "성적", aggfunc="sum")
```

- index 피벗테이블에서 인덱스로 지정할 컬럼의 이름(두 개 이상이면 리스트로 입력할 것)
- columns 피벗테이블에서 컬럼으로 지정할 컬럼의 이름(범주형 변수 활용)
- values 피벗테이블에서 columns의 값이 될 컬럼의 이름(수치형 변수 활용)
- aggfunc 집계함수를 사용할 경우 지정

|   | ID | 반 | 성적  |
|---|----|---|-----|
| 0 | 1  | A | 100 |
| 1 | 1  | B | 88  |
| 2 | 1  | A | 85  |
| 3 | 2  | B | 75  |
| 4 | 2  | A | 100 |
| 5 | 2  | B | 80  |



|    | 반   | A   | B |
|----|-----|-----|---|
| ID |     |     |   |
| 1  | 185 | 88  |   |
| 2  | 100 | 155 |   |

# 03. pandas

(2023년 11월 기준)


## ◆ 데이터 재구조화

✓ melt : 피벗테이블의 반대 개념으로 생각하면 된다

```
df.melt(id_vars = ["ID"], var_name = "반", value_name = "성적")
```

- id\_vars          피벗 테이블에서 인덱스가 될 컬럼의 이름
- var\_name        variable 변수의 이름으로 지정할 문자열(선택)
- value\_name      value 변수의 이름으로 지정할 문자열(선택)

| 반  | A   | B   |
|----|-----|-----|
| ID |     |     |
| 1  | 185 | 88  |
| 2  | 100 | 155 |



|   | ID | 반 | 성적    |
|---|----|---|-------|
| 0 | 1  | A | 92.5  |
| 1 | 2  | A | 100.0 |
| 2 | 1  | B | 88.0  |
| 3 | 2  | B | 77.5  |

## ◆ 주의

✓ 기존 테이블에서 집계된 값을 원래값으로 재분리하는 것은 아님

# 강의 실습 영상 참고



- ◆ 데이터 전처리의 의미
  - ✓ 분석에 적합하게 데이터를 가공하는 작업 의미

| 구분         | 내용                  |
|------------|---------------------|
| 데이터 클리닝    | 결측치 처리, 이상치 확인 및 정제 |
| 데이터 통합     | 다양한 데이터 파일의 결합      |
| 데이터 변환     | 스케일링, 요약            |
| 데이터 축소     | 변수 축소, 라벨링          |
| 불균형 데이터 처리 | 언더 샘플링, 오버 샘플링      |
| 데이터 분할     | train, test 데이터 분할  |