参考

- Streamlit 기본 구조와 사용자 인터페이스 이해 (理解 Streamlit 的基本结构与用户界面)
- 입력창, 버튼, 슬라이더 등 위젯 활용 (熟练使用输入框、按钮、滑块等组件/控件)
- 데이터 시각화와 연계한 대화형 앱 만들기 (构建与数据可视化相结合的交互式应用)
- 미니 프로젝트 제작 (예: 성적 관리 대시보드, 주가 그래프 그리기, 데이터 분석 앱 등) (完成迷你项目如成绩管理仪表盘、绘制股票价格折线图, 数据分析应用等)

Streamlit이란?

2

- **Python만으로 웹 앱**(대시보드, 입력/출력 폼, 그래프 등)을 쉽게 만들 수 있게 기본 구조를 제공하는 틀
- 웹 앱 : 웹 브라우저에서 실행되며, 사용자가 직접 값을 입력하고 계산하거나 결과를 확인 할 수 있는 웹 프로그램
- 설치 : !pip install streamlit
- 실행 : !streamlit run app.py

CODE

```
%%writefile app.py
import streamlit as st

st.title("Streamlit 테스트 앱")
st.write("Hello World")

name = st.text_input("이름을 입력하세요:")
if st.button("인사하기"):
    st.success(f"안녕하세요, {name}님!")
    #st.write() 도 가능
```

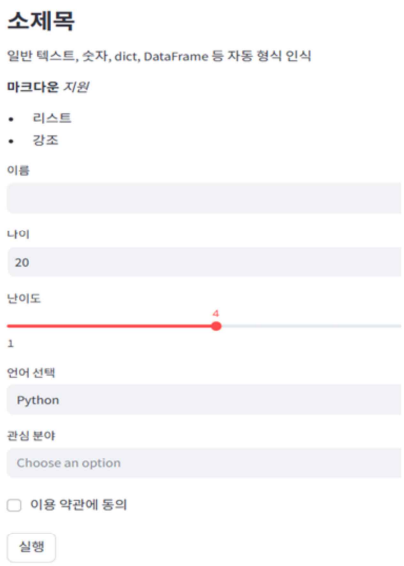
参考

- Streamlit 소개
  - ✓Streamlit 是一个只用 Python 就能轻松创建网页应用（如仪表板、输入/输出表单、图表等）的框架。
  - ✓网页应用：在浏览器中运行，用户可以直接输入数值、进行计算或查看结果的网络程序。
- 安装：!pip install streamlit
- 运行：!streamlit run app.py

streamlit 기본 명령어

3

- st.title("제목")
- st.header("부제목")
- st.subheader("소제목")
- st.**write**("일반 텍스트, 숫자, dict, DataFrame 등 자동 형식 인식")
- st.markdown("**마크다운** _지원_ ₩n- 리스트 ₩n- 강조")
- name = st.**text_input**("이름")
- age = st.**number_input**("나이", min_value=0, max_value=150, value=20)
- level = st.slider("난이도", 1, 10, 5)
- choice = st.**selectbox**("언어 선택", ["Python", "Java", "C++"])
- multi = st.multiselect("관심 분야", ["AI","Data","Web"])
- ok = st.**checkbox**("이용 약관에 동의")
- clicked = st.**button**("실행")

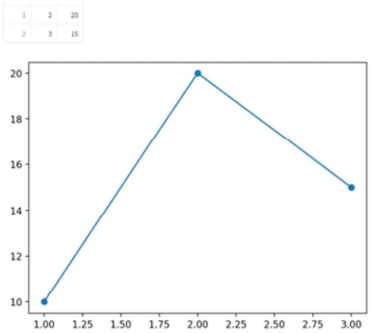


参考

- **Streamlit 주요 명령어 소개 (Streamlit 主要命令介绍)**
- **st.subheader("소제목")** 더 작은 단위의 소제목 설정(设置小标题，比副标题更小一级)
- **st.write("일반 텍스트, 숫자, dict, DataFrame 등 자동 형식 인식")**
  텍스트, 숫자, 딕셔너리, 데이터프레임 등 다양한 형식을 자동 인식하여 출력(可自动识别并输出多种格式的数据，如文本、数字、字典、数据框等)
- **st.markdown("마크다운 *지원* ₩n- 리스트 ₩n- 강조")**
  마크다운 문법을 지원하여 강조, 목록 등 형식 표현 가능(支持Markdown语法，可显示加粗、斜体、列表等格式)
- **st.text_input("이름")** 사용자가 문자열을 입력할 수 있는 입력창 생성(创建一个输入框，让用户输入文字内容)
- **st.number_input("나이", min_value=0, max_value=150, value=20)**
  숫자를 입력받을 수 있는 창 생성, 기본값과 범위 설정 가능(创建数字输入框，可设置默认值和范围)
- **st.slider("난이도", 1, 10, 5)** 슬라이더를 통해 범위 내 값 선택(创建滑动条，在指定范围内选择数值)
- **st.selectbox("언어 선택", ["Python", "Java", "C++"])**
  목록 중 하나를 선택할 수 있는 드롭다운 메뉴 생성(创建下拉菜单，让用户从列表中选择一项)
- **st.multiselect("관심 분야", ["AI", "Data", "Web"])**
  여러 항목을 동시에 선택할 수 있는 멀티 셀렉트 박스 생성(创建多选框，可同时选择多个选项)
- **st.checkbox("이용 약관에 동의")**체크박스를 통해 True/False 값 입력(创建复选框，用于输入True/False值)
- **st.button("실행")** 버튼 클릭 시 특정 동작 실행(创建按钮，用于执行特定操作)

streamlit 기본 명령어

4



CODE

```
%%writefile app2.py
import pandas as pd
import streamlit as st
import matplotlib.pyplot as plt
df = pd.DataFrame({"x":[1,2,3], "y":[10,20,15]})
st.table(df)
st.dataframe(df)      # 스크롤·정렬 가능한 표
#fig, ax = plt.subplots()
plt.plot(df["x"], df["y"], marker="o")
st.pyplot(fig)
st.image("cat1.png", caption="이미지 예시", width=400)
```

参考

- **Streamlit 데이터 출력 및 시각화 (Streamlit 数据输出与可视化)**
- **st.table(df)**
  데이터프레임을 표 형태로 출력(以表格形式输出数据框)
- **st.dataframe(df)**
  스크롤·정렬이 가능한 표 출력(可滚动并可排序的表格显示)
- **plt.plot(df["x"], df["y"], marker="o")**
  Matplotlib을 이용하여 선 그래프 그리기(使用 Matplotlib 绘制折线图)
- **st.pyplot(fig)**
  Matplotlib 그래프를 Streamlit 앱에 표시(Streamlit 中显示 Matplotlib 图形)
- **st.image("cat1.png", caption="이미지 예시", width=400)**
  이미지 파일을 화면에 표시하며, 캡션과 크기 설정 가능(显示图片文件，可设置标题和宽度)

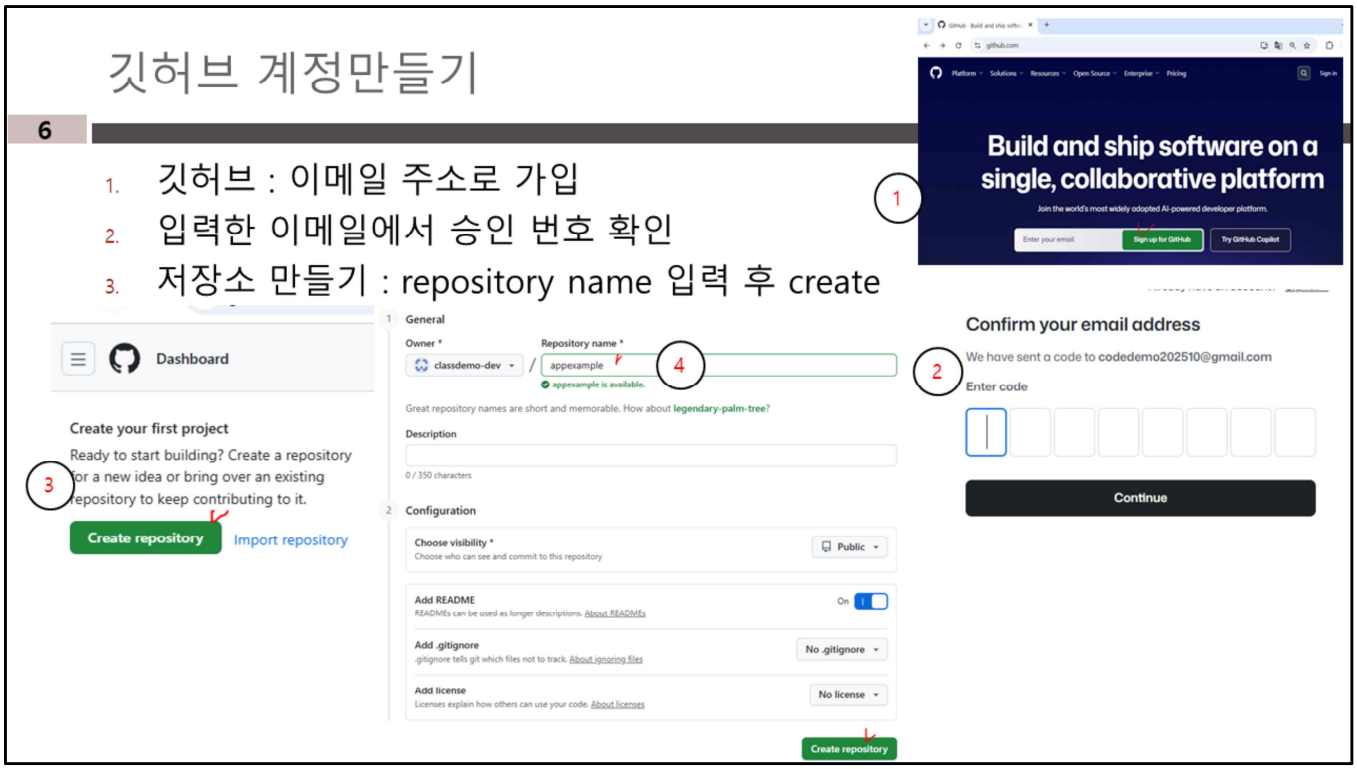
웹에 올려서 보기(배포)

5

- **Streamlit Community Cloud**로 무료 배포 가능: https://share.streamlit.io
- **절차**
  1. **GitHub 저장소 생성**** (예: `streamlit-examples`)
  2. `app.py` (앱 파일) 업로드
  3. `requirements.txt` 업로드 – 필요한 패키지 명시
  4. Streamlit Cloud 접속 → ****New app****
  5. GitHub 저장소/브랜치/파일 경로 선택 → ****Deploy****
  6. 배포 주소 예: `https://username-streamlit-examples.streamlit.app`

参考

- **절차 (步骤)**
- GitHub 저장소 생성 (예: streamlit-examples)
  创建 GitHub 仓库（例如：streamlit-examples）
- app.py (앱 파일) 업로드
  上传 app.py（应用程序文件）
- requirements.txt 업로드 – 필요한 패키지 명시
  上传 requirements.txt 文件，注明所需的依赖包
- Streamlit Cloud 접속 → **New app** 선택
  进入 Streamlit Cloud → 选择 **New app**
- GitHub 저장소 / 브랜치 / 파일 경로 선택 → **Deploy**
  选择 GitHub 仓库 / 分支 / 文件路径 → **Deploy（部署）**
- 배포 주소 예:
  发布地址示例：
  https://username-streamlit-examples.streamlit.app



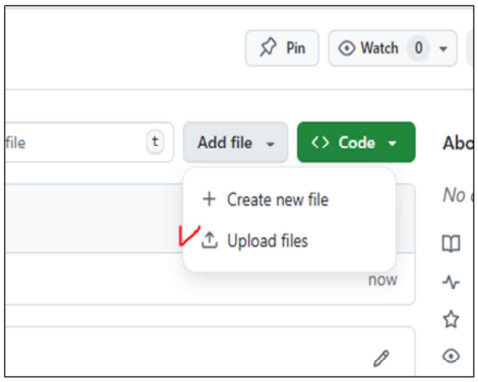
参考

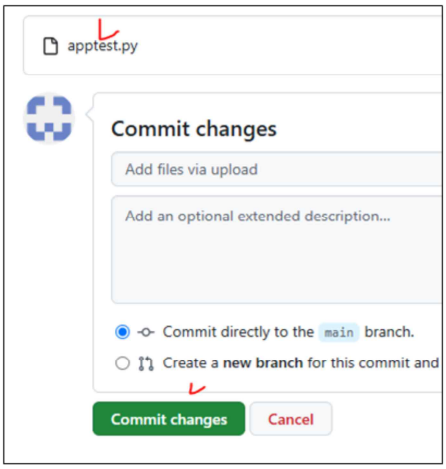
- **깃허브 가입 및 저장소 만들기 (GitHub 注册与创建仓库)**
- **GitHub은 코드와 프로젝트를 온라인으로 관리하고 공유할 수 있는 플랫폼**
  GitHub 是一个可以在线管理和共享代码与项目的平台。
- 깃허브: 이메일 주소로 가입
  GitHub：使用电子邮箱注册
- 입력한 이메일에서 승인 번호 확인
  在输入的邮箱中查看验证码并确认
- 저장소 만들기: repository name 입력 후 **Create**
  创建仓库：输入仓库名称（repository name）后点击 **Create（创建）**

깃허브에 앱 업로드

7

- Add file -> Upload files
- 파일 업로드 후 Commit changes 클릭





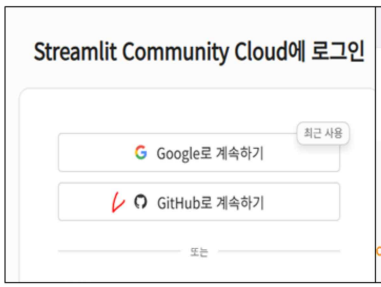
参考

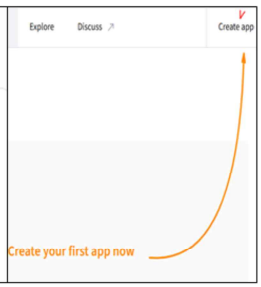
- **파일 업로드 (文件上传)**
- **Add file → Upload files**
  파일을 추가할 때 선택(添加文件 → 上传文件)
- **파일 업로드 후 Commit changes 클릭**
  파일 업로드가 완료되면 **Commit changes** 클릭하여 변경 내용 저장
  文件上传完成后，点击 **Commit changes（提交更改）** 保存修改内容

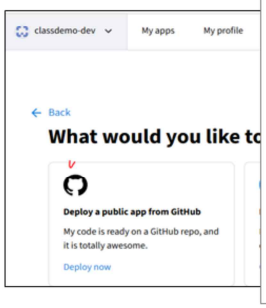
streamlit에 공유하기

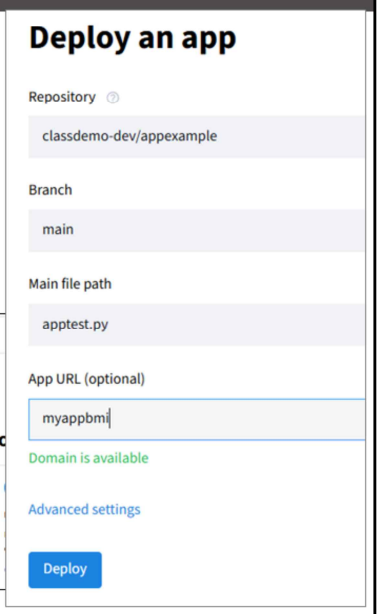
8

1. https://share.streamlit.io/-> GitHub로 로그인
2. Create app
3. Deploy a public app from GitHub
4. Repository 선택 -> app file 선택->URL 입력
5. url 확인 https://myappbmi.streamlit.app/









参考

Streamlit 앱 배포 과정 (Streamlit 应用发布流程)

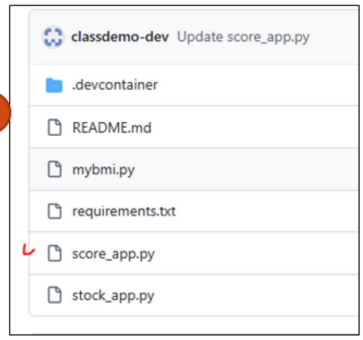
✓Streamlit 앱은 **코드를 작성한 후 GitHub에 업로드하고**, 그 저장소를 **Streamlit Community Cloud(https://share.streamlit.io)** 와 연결하여 웹에서 바로 실행할 수 있는 형태로 배포함
(Streamlit 应用可上传至 GitHub，并通过 Streamlit Community Cloud 在线运行)

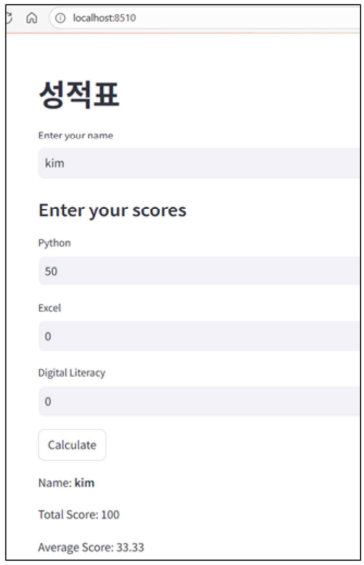
1. **GitHub 준비**
   ① GitHub 계정 생성 후 로그인함(注册并登录 GitHub 账号)
   ② 새 저장소(repository) 생성 후 Streamlit 앱 파일(**app.py**)과 필요한 패키지를 명시한 **requirements.txt** 업로드함
   (创建仓库，上传 app.py 与 requirements.txt 文件)
   ③ 파일 업로드 후 **Commit changes** 클릭하여 저장함(上传完成后点击 Commit changes 保存)
2. **Streamlit Cloud 접속**
   ① 웹사이트 **https://share.streamlit.io** 접속 후 GitHub 계정으로 로그인함(访问网站并使用 GitHub 登录)
   ② "**Create app**" 버튼 클릭하여 새 앱 생성함(点击 Create app 创建新应用)
3. **앱 선택 및 배포**
   ① GitHub 저장소와 브랜치(branch), 앱 파일(**app.py**) 경로 선택 후 **Deploy** 클릭함(选择仓库、分支和应用文件后点击 Deploy)
   ② Streamlit이 자동으로 배포 과정을 수행함(Streamlit 自动完成部署过程)
4. **배포 완료**
   ① 배포 완료 후 https://사용자명-저장소명.streamlit.app 형태의 주소가 생성됨
   (部署完成后生成网页地址)
   ② 예: **https://myappbmi.streamlit.app/**

성적 앱 만들기

9

1. score_app.py 파일 생성
2. 코드 작성
3. 로컬로 확인 !streamlit run score_app.py
4. GitHub에 업로드
5. https://share.streamlit.io 접속
   - New app → GitHub repo 선택 → Deploy





参考

参考

Streamlit 성적표 앱 실행 절차 (Streamlit 成绩报告应用流程)

1. 앱 실행 후 제목과 입력창을 화면에 표시함
   (运行应用后显示标题和输入框)

2. 이름과 세 과목 점수(Python, Excel, Digital Literacy)를 입력함
   (输入姓名与三门课程成绩：Python、Excel、数字素养)

3. Calculate 버튼을 클릭하여 계산을 수행함
   (点击 “Calculate” 按钮执行计算)

4. 입력된 점수를 이용해 총점과 평균을 계산함
   (根据输入成绩计算总分与平均分)

5. 이름, 총점, 평균을 화면에 출력함
   (在页面上显示姓名、总分与平均分)

6. 평균 점수에 따라 등급(Excellent, Good, Try Again)을 표시함
   (根据平均分显示等级：优秀、良好、需努力)

웹 크롤링

11

- 웹사이트의 정보를 자동으로 수집하여 필요한 데이터를 가져오는 과정임
- requests
  - 웹사이트에서 HTML 코드 가져오기 위한 라이브러리
- BeautifulSoup
  - HTML 구조를 분석해서 필요한 정보 추출
- requests → BeautifulSoup → select → get_text()
  1. 웹페이지 요청 ex) 네이버 금융 사이트
  2. HTML분석 ex) HTML분석하여 주가지수에 해당하는 부분 찾기
  3. 원하는 부분 가져오기 ex) 값 가져오기

参考

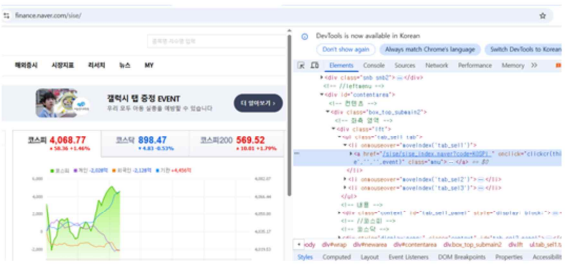
- **웹 크롤링이란?**
  ✓웹사이트의 정보를 자동으로 수집하여 필요한 데이터를 가져오는 과정임(网络爬虫是指自动从网站收集信息并提取所需数据的过程)

- **웹 크롤링 절차 (网页爬取流程)**
  1. **requests 라이브러리로 HTML 코드 가져옴**
     웹페이지에 요청을 보내고, HTML 문서를 불러옴(使用 requests 向网页发送请求并获取 HTML 内容)
  2. **BeautifulSoup으로 HTML 구조 분석함**
     가져온 HTML 코드를 분석하여 필요한 부분을 찾음(用 BeautifulSoup 分析 HTML 结构，找到需要的部分)
  3. **select()나 get_text()로 원하는 데이터 추출함**
     원하는 태그나 텍스트를 선택해 내용만 가져옴(使用 select() 或 get_text() 提取所需文本或数据)

웹 데이터 가져오기1(주가지수)

12

https://finance.naver.com/sise
원하는 위치에서 마우스 우클릭 하여'검사'
Elements(元素) 패널에서 하이라이트된 태그 확인



KOSPI : 4,002.22
KOSDAQ: 902.53
KOSPI200: 558.61

CODE

```
import requests
from bs4 import BeautifulSoup
url = "https://finance.naver.com/sise/"
html = requests.get(url, headers={"User-Agent":"Mozilla/5.0"}).text
soup = BeautifulSoup(html, "html.parser")
kospi  = soup.select_one("#KOSPI_now").get_text(strip=True)
kosdaq = soup.select_one("#KOSDAQ_now").get_text(strip=True)
kospi200 = soup.select_one("#KPI200_now").get_text(strip=True)
print("KOSPI  :", kospi)
print("KOSDAQ:", kosdaq)
print("KOSPI200:", kospi200)
```

参考

- **네이버 금융 지수 크롤링 (Naver 金融指数爬取)**
  1. requests와 BeautifulSoup 라이브러리 불러옴 (**导入 requests 与 BeautifulSoup 库**)
  2. 크롤링할 웹페이지 주소를 설정함 — "https://finance.naver.com/sise/" (**设置要爬取的网页地址**)
  3. User-Agent를 포함해 웹 요청을 보냄 (**发送带有 User-Agent 的网页请求**)
  4. 가져온 HTML 문서를 .text로 저장함 (**将获取到的 HTML 文档以 .text 形式保存**)
  5. BeautifulSoup을 이용해 HTML을 구조화하여 분석함 (**使用 BeautifulSoup 解析 HTML 结构**)
  6. select_one("#KOSPI_now")로 코스피 지수 위치를 찾고, get_text(strip=True)로 숫자만 추출함 (**用 select_one("#KOSPI_now") 定位 KOSPI 指数，并用 get_text(strip=True) 提取数字**)
  7. 코스닥(#KOSDAQ_now), 코스피200(#KPI200_now)도 같은 방식으로 가져옴 (**以相同方式获取 KOSDAQ 与 KOSPI200**)
  8. 추출한 결과를 print()로 출력함 (**用 print() 输出提取结果**)

웹 데이터 가져오기1(주가지수)

13

- popularItemList의 목록에서 상위 5개만 가져오기
  - [:5] -> 0~4번지 5개 가져옴

```
KOSPI : 3,995.33
KOSDAQ: 901.16
KOSPI200: 557.45
==============
인기 검색 종목
삼성SDI : 309,000
삼성전자 : 99,700
SK하이닉스 : 514,500
두산에너빌리티 : 84,100
삼성중공업 : 29,950
```

CODE

```
print("KOSPI :", kospi)
print("KOSDAQ:", kosdaq)
print("KOSPI200:", kospi200)
print("===============")
print("인기 검색 종목")
items = soup.select("#popularItemList li")[:5]  # 상위 5개
for item in items:
  name = item.select_one("a").get_text(strip=True)
  price = item.select_one("span").get_text(strip=True)
  print(f"{name} : {price}")
```

参考

- **리스트에서 상위 5개만 가져오기 (从列表中获取前5个项目)**
  - ✓**popularItemList의 목록에서 상위 5개만 가져오기**
    从 popularItemList 列表中提取前 5 个项目
  - ✓**[:5] → 0~4번지 5개 가져옴**
    [:5] 表示获取索引 0 到 4 的 5 个元素

웹 데이터 가져오기2(환율)

14

- pandas
  - 데이터를 표(표 형태, 즉 엑셀처럼)로 다루기 쉽게 해주는 파이썬 라이브러리
  - pd.read_html() → 웹페이지 안의 <table>(표)을 자동으로 찾아서 데이터프레임(DataFrame) 으로 바꿔줌
  - df.head() → 데이터의 앞부분 5행만 미리보기

CODE

```
import requests, pandas as pd
url = https://kr.investing.com/currencies/exchange-rates-table
html = requests.get(url, headers={"User-Agent": "Mozilla/5.0"}).text
df = pd.read_html(html)[0]
print(df.head())
```

	종목	USD	EUR	GBP	JPY	CHF	CAD	AUD	KRW
0	USD	1.0000	0.8579	0.7490	152.42	0.7943	1.3992	1.5252	1436.7400
1	EUR	1.1656	1.0000	0.8731	177.67	0.9258	1.6307	1.7777	1674.4300
2	GBP	1.3350	1.1454	1.0000	203.50	1.0604	1.8678	2.0361	1917.8700
3	JPY	0.0066	0.0056	0.0049	1.00	0.5211	0.0092	0.0100	9.4257
4	CHF	1.2590	1.0801	0.9430	191.91	1.0000	1.7615	1.9200	1808.8400

参考

주가 데이터 불러와서 그래프 그리는 웹앱

15

CODE

```
%%writefile stock_app.py
# 파일명: stock_app.py
# 주가 데이터를 불러와 최근 종가와 그래프를 보여주는 간단한 대시보드
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
st.title("주가 시각화 대시보드")
# 종목(symbol), 기간, 간격 입력
symbol = st.text_input(" 종목 입력 (예: AAPL, TSLA, NVDA, 005930.KS)", "AAPL")
period = st.selectbox("기간", ["1mo", "3mo", "6mo", "1y", "2y", "5y"], index=3)
interval = st.selectbox("간격", ["1d", "1wk", "1mo"], index=0)
st.caption("데이터 출처: Yahoo Finance (https://finance.yahoo.com/)")

# 주가 데이터 불러오기
data = yf.download(symbol, period=period, interval=interval)
# 최신 종가 표시
st.write(" 최신 종가:", data["Close"].iloc[-1])
# 그래프 그리기
plt.rcParams['font.family'] = 'Malgun Gothic'
plt.rcParams['axes.unicode_minus'] = False
plt.figure(figsize=(8, 5))
plt.plot(data.index, data["Close"])
plt.title(f"{symbol}({period}, {interval})")
plt.xlabel("Date")
plt.ylabel("Price")
st.pyplot(plt)
```

```
st.pyplot(plt)
Overwriting stock_app.py
[15]: !streamlit run stock_app.py
```

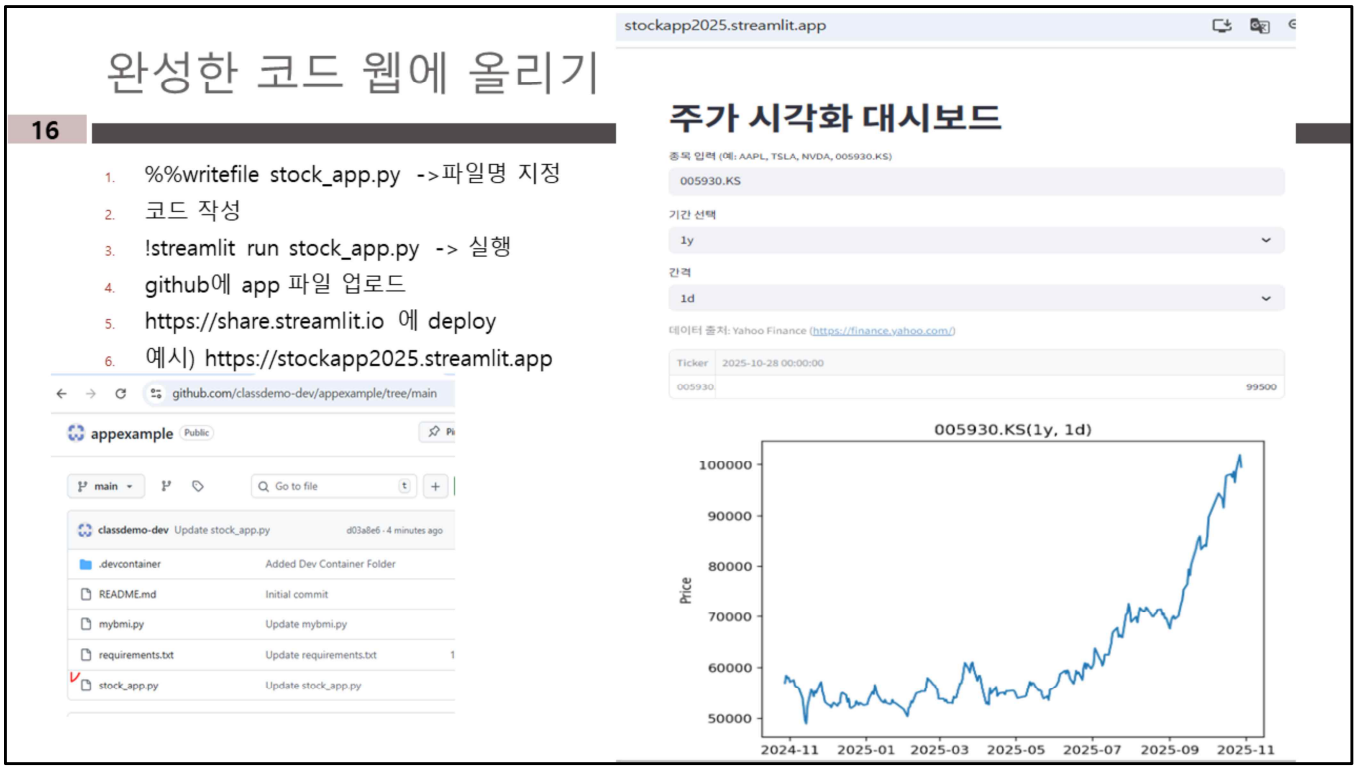
실행 후 다음셀에 "!streamlit run 파일명"으로 실행함

```
!streamlit run stock_app.py
```

参考

- **주가 시각화 대시보드 – 단계별 설명 (股票可视化仪表盘 – 分步说明)**

① Streamlit, Pandas, Matplotlib, yfinance 라이브러리를 임포트함 (导入 Streamlit、Pandas、Matplotlib、yfinance 库)
② 앱 제목을 "주가 시각화 대시보드"로 표시함 (将应用标题设置为"股票可视化仪表盘")
③ 사용자로부터 종목(symbol), 기간(period), 간격(interval)을 입력받는 위젯을 생성함 (创建输入控件以获取股票代码、时间周期与数据间隔)
④ 데이터 출처로 Yahoo Finance를 캡션으로 표기함 (以说明文字标注数据来源为 Yahoo Finance)
⑤ yfinance의 download()로 선택한 종목의 시세 데이터를 불러옴 (使用 yfinance 的 download() 获取所选股票的行情数据)
⑥ 불러온 데이터의 최신 종가(Close 열의 마지막 값)를 화면에 출력함 (在页面上显示下载数据中 Close 列的最新收盘价)
⑦ Matplotlib 한글/마이너스 표시 옵션을 설정함 (设置 Matplotlib 的韩文字体与负号显示选项)
⑧ 날짜 인덱스와 Close 값을 이용해 선 그래프를 그림 (使用日期索引与 Close 值绘制折线图)
⑨ 그래프 제목에 종목·기간·간격 정보를 포함해 가독성을 높임 (在图表标题中加入股票、周期与间隔信息以提升可读性)
⑩ X축은 날짜, Y축은 가격으로 라벨을 설정함 (将横轴设为日期、纵轴设为价格)
⑪ 완성된 Matplotlib 그래프를 st.pyplot()으로 Streamlit 화면에 출력함 (通过 st.pyplot() 将 Matplotlib 图形输出到 Streamlit 页面)
⑫ 실행: 터미널/노트북에서 아래 명령으로 앱을 구동함 !streamlit run stock_app.py
(运行：在终端/Notebook 中执行上述命令以启动应用)



参考

- %%writefile stock_app.py로 파일명 지정 후 코드 저장함
  （用 %%writefile stock_app.py 指定文件名并保存代码）
- 로컬에서 코드 작성·수정 완료함 (stock_app.py, 필요 시 requirements.txt 함께 준비함)
  （在本地完成代码编写与修改，必要时准备 requirements.txt）
- !streamlit run stock_app.py 로 실행해 로컬 동작 확인함
  （用 !streamlit run stock_app.py 本地运行并确认无误）
- GitHub에 앱 파일 업로드(또는 push)함 → 저장소에 app.py/stock_app.py 및 requirements.txt 포함함
  （将应用文件上传/推送到 GitHub；仓库中包含应用文件与 requirements.txt）
- https://share.streamlit.io 접속 후 **Create app → 저장소/브랜치/앱 파일 선택 → Deploy** 함
  배포 완료 후 생성된 URL로 접속함 (예: https://username-repo.streamlit.app)
  （在 Streamlit Community Cloud 选择仓库/分支/应用文件并部署；完成后使用生成的访问地址）
- https://stockapp2025.streamlit.app/