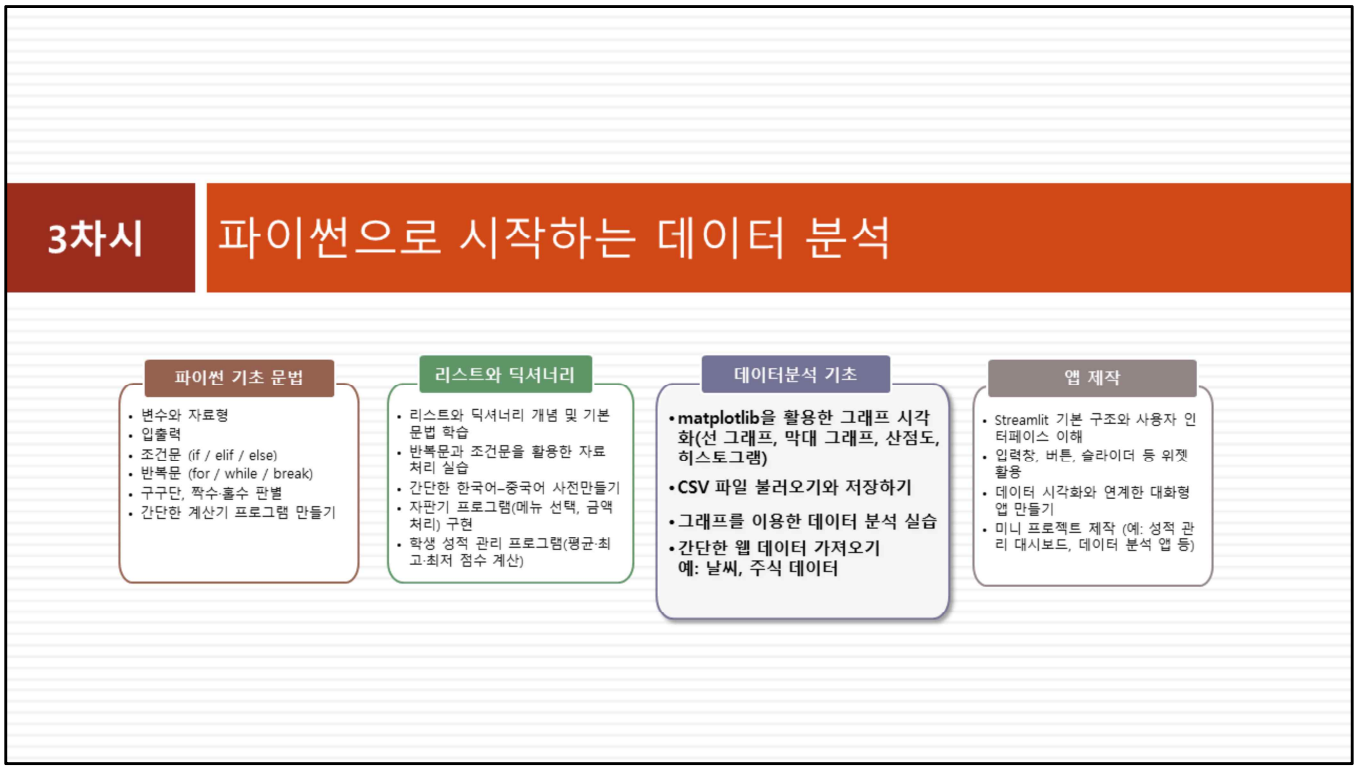


参考

그래프의 필요성 (为什么要画图？)

✓데이터를 시각화하면 더 쉽게 이해할 수 있다.(数据可视化可以让我们更容易理解)

✓예: 성적 비교, 매출 변화, 날씨 변화(例如：成绩比较、销售额变化、天气变化)

- 使用 matplotlib 进行图表可视化（折线图、柱状图、散点图、直方图）
- 读取与保存 CSV 文件
- 利用图表进行数据分析实操
- 简单获取网页数据（例如：天气、股票数据）

파이썬으로 데이터 분석하기

31

- **쉽고 강력한 라이브러리**
  - 파이썬은 Pandas, Matplotlib, Seaborn 등 데이터 분석에 최적화된 라이브러리들이 풍부하게 제공되어 복잡한 데이터 처리도 간단한 코드로 해결이 가능 함
- Matplotlib
  - 파이썬의 대표적인 시각화 라이브러리로, 초보자도 쉽게 접근할 수 있으면서도 전문적인 결과물을 만들 수 있음
- CSV
  - Comma Separated Values의 줄임말로, 콤마로 구분된 텍스트 형태의 데이터 파일로 가장 보편적이고 간단한 데이터 저장 방식 중 하나
- **matplotlib로 CSV 데이터 시각화하기**

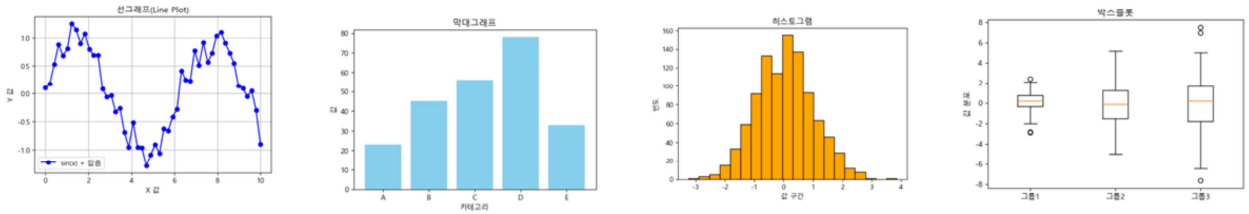
参考

- **简单而强大的库**
  - ✓Python 提供了丰富的适用于数据分析的库，如 Pandas、Matplotlib、Seaborn 等，即使复杂的数据处理也能通过简洁的代码解决
- **Matplotlib**
  - ✓Python 中最具代表性的可视化库，初学者容易上手，同时也能生成专业的可视化结果
- **CSV**
  - ✓**CSV** 是 Comma Separated Values（逗号分隔值）的缩写，是一种以逗号分隔的文本格式数据文件，是最常见、最简单的数据存储方式之一

그래프 유형

32

선 그래프 (Line Plot)	막대 그래프 (Bar Chart)	히스토그램(Histogram)	박스플롯(Box plot)
• 시간에 따른 데이터 변화나 연속적인 트렌드를 표현할 때 사용 • 주식 가격 변동이나 온도 변화 등에 적합	• 범주별 데이터를 비교할 때 사용 • 지역별 매출, 제품별 판매량 등 범주형 데이터 시각화에 효과적	• 연속형 데이터의 분포와 빈도를 시각화하는 도구 • 시험 점수 분포, 연령 분포, 소득 분포 등을 파악하는 데 유용	• 데이터의 중앙값, 사분위수, 이상치 등 통계적 정보를 한번에 보여주는 시각화 기법 • 데이터의 분포와 이상값을 동시에 파악



参考

1. **선 그래프 (Line Plot | 折线图)**
   - ✓시간에 따른 데이터 변화나 연속적인 트렌드를 표현할 때 사용 (用于表示随时间变化的数据或连续趋势)
   - ✓주식 가격 변동이나 온도 변화 등에 적합(适合股票价格变动、气温变化等)
2. **막대 그래프 (Bar Chart | 条形图)**
   - ✓범주별 데이터를 비교할 때 사용(用于比较不同类别的数据)
   - ✓지역별 매출, 제품별 판매량 등 범주형 데이터 시각화에 효과적(对地区销售额、产品销量等分类数据的可视化效果显著)
3. **히스토그램 (Histogram | 直方图)**
   - ✓연속형 데이터의 분포와 빈도를 시각화하는 도구(可视化连续数据的分布与频率的工具)
   - ✓시험 점수 분포, 연령 분포, 소득 분포 등을 파악하는 데 유용(适用于分析考试成绩、年龄、收入等的分布情况)
4. **박스플롯 (Box Plot | 箱线图)**
   - ✓데이터의 중앙값, 사분위수, 이상치 등 통계 정보를 한 번에 보여주는 기법(一次性展示数据的中位数、四分位数、异常值等统计信息的方法)
   - ✓데이터의 분포와 이상값을 동시에 파악 가능(可同时掌握数据的分布与异常值)

matplotlib의 주요 그래프

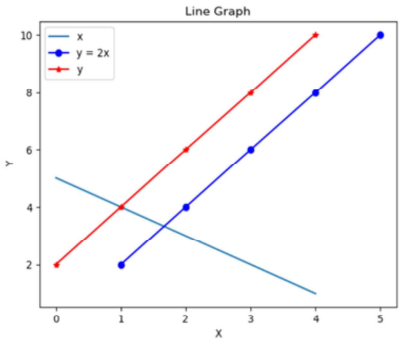
33

- plt.plot() : 선그래프
  - plt.plot(x, **y**, color="blue", linestyle="--", marker="o", label="Line1")
- plt.bar() : 막대그래프
  - plt.bar(**["A","B","C"], [3,7,5]**, color="green", width=0.5)
- plt.hist() : 히스토그램
  - plt.hist(**data**, bins=10, color="orange", alpha=0.7)
- plt.boxplot() : 박스플롯
  - plt.boxplot(**data**, labels=["Score"], patch_artist=True)
- plt.title() : 그래프 제목
- plt.xlabel(),plt.ylabel() : 축 이름
- plt.legend() : 범례 표시

参考

선 그래프

34



기호	예시
-	———
--	– – –
-.	– · – · –
:	·····

색상	약어	설명
b	blue	파랑
g	green	초록
r	red	빨강
c	cyan	청록
m	magenta	자홍
y	yellow	노랑
k	black	검정 (k = black)
w	white	흰색

기호	모양	예시
o	원(circle)	●
s	네모(square)	■
^	위 삼각형	▲
v	아래 삼각형	▼
<	왼쪽 삼각형	◀
>	오른쪽 삼각형	▶
*	별	★
+	플러스	+
x	엑스	×
D	다이아몬드	◆
p	오각형	⬟

CODE

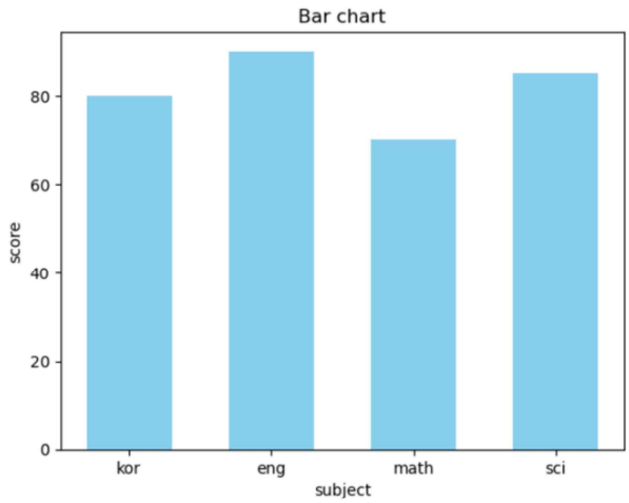
```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
plt.plot([5,4,3,2,1], label='x') #x축 데이터 생략하여 0,1,2,3,4로 자동 생성
plt.plot(x, y, marker='o', color='b', label="y = 2x")
plt.plot(y, 'ro-') #color/marker/linestyle
plt.title("Line Graph")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend() #레이블 표시
plt.show()
```

参考

- **plt.plot(x, y, color='r', linestyle='--', marker='o')**
- **x, y** → 데이터 좌표 | 数据坐标
- **color** → 선 색상 (예: 'r' 빨강) | 线条颜色（例: 'r' 红色）
- **linestyle** → 선 모양 (예: '-' 실선, '--' 점선) | 线条样式（例: '-' 实线, '--' 虚线）
- **marker** → 데이터 점 모양 (예: 'o' 원형) | 数据点形状（例: 'o' 圆点）
- 使用 plt.plot([5,4,3,2,1], label='x') 时，x 轴会自动从 0 开始生成整数序列。
- **세 가지 요소를 축약 가능**
  → color(색상), marker(마커), linestyle(라인스타일)을 한 줄로 표현할 수 있다.
  → 可以把 颜色(color)、标记(marker)、线型(linestyle) 简写在一起。
- **조합 순서**
  → 보통 "컬러 → 마커 → 라인" 순서로 쓰지만, 순서는 크게 상관없다.
  → 通常顺序是 "颜色 → 标记 → 线型"，但顺序没有严格要求。
- **축약 문자열 예시**
- 'ro--' → 빨강(r) + 원 마커(o) + 파선(--)
- 'gs:' → 초록(g) + 네모 마커(s) + 점선(:)
  → 示例: 'ro--' = 红色(r) + 圆点(o) + 虚线(--)
- **키워드 인자 사용도 가능**
  → 더 명확히 쓰려면 color='red', marker='o', linestyle='--' 형태로 지정할 수 있다.
  → 如果想更清晰，可以写成 color='red', marker='o', linestyle='--'。

막대 그래프

35



CODE

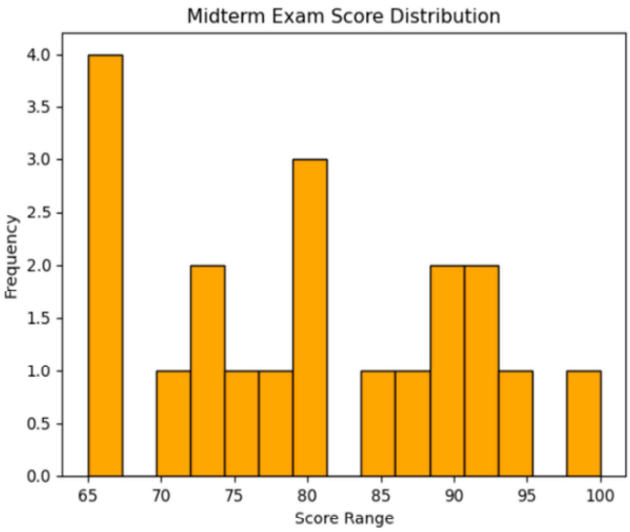
```
# 2. 막대그래프 (Bar Chart)
categories = ["kor", "eng", "math", "sci"]
scores = [80, 90, 70, 85]
plt.bar(categories, scores, color="skyblue")
plt.title("Bar chart")
plt.xlabel("subject")
plt.ylabel("score")
plt.show()
```

参考

- **plt.bar(x, height, color='b', width=0.6)**
- **x** → 막대의 위치 | 条形的位置
- **height** → 막대의 높이(값) | 条形的高度（数值）
- **color** → 막대 색상 | 条形颜色
- **width** → 막대 두께 (기본값 0.8) | 条形宽度（默认 0.8）

히스토그램

36



CODE

```
# 3. 히스토그램 (Histogram)
data = [60, 62, 63, 65, 70, 72, 72, 75, 78, 80,
        82, 83, 85, 88, 90, 90, 91, 92, 95, 100]

plt.hist(data, bins=15, color="orange", edgecolor="black")
plt.title("Midterm Exam Score Distribution")
plt.xlabel("Score Range")
plt.ylabel("Frequency")
plt.show()

#랜덤 숫자
import numpy as np
data = np.random.randint(0, 101, 10) #0~100까지 정수 10개 생성
```

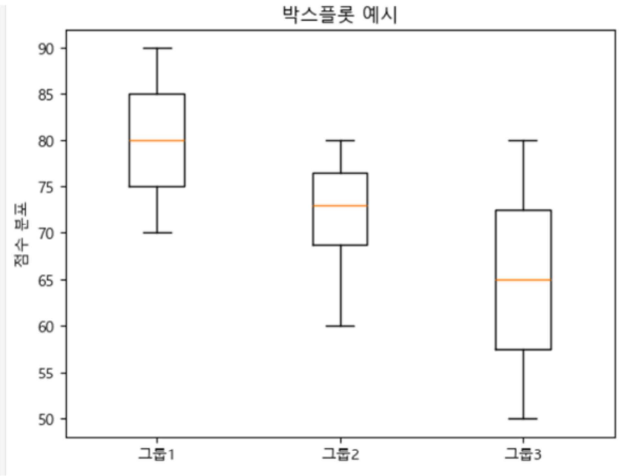
参考

import numpy as np
np.random.randint(0, 101, 100)
→ 0 이상 101 미만의 정수 중에서 **100개** 난수를 생성. 즉, 0~100 점 사이의 임의의 시험 점수 100개가 만들어 짐
→ 在 **0 到 100（含 0，不含 101）** 之间随机生成 **100 个整数**。相当于模拟 0~100 分之间的考试成绩。

- bins
  - ✓히스토그램에서 bins는 데이터를 나눌 **구간(Interval)** 개수를 의미(在直方图中，bins 表示把数据划分为多少个**区间**)

박스 플롯

37



CODE

```
# 4. 박스플롯 (Boxplot)
group1 = [70, 75, 80, 85, 90]
group2 = [60, 65, 70, 72, 74, 76, 78, 80]
group3 = [50, 55, 60, 65, 70, 75, 80]
plt.boxplot([group1, group2, group3], tick_labels
=["g1", "g2", "g3"])
plt.title("Boxplot")
plt.ylabel("Score")
plt.show()
```

参考

- 박스플롯은 데이터의 **분포(Distribution)** 를 한눈에 볼 수 있도록 만든 그래프 (箱线图是一种可以直观显示**数据分布**的图表)
- **구성 요소 (组成部分)**
  - ✓**최솟값 (最小值)**
    - 데이터에서 가장 작은 값(图中最底部的"胡须")
  - ✓**제1사분위수 Q1 (下四分位数)**
    - 데이터의 하위 25% 지점(盒子的下边缘)
  - ✓**중앙값 Median (中位数)**
    - 데이터의 가운데 값 (50%)(盒子中间的横线)
  - ✓**제3사분위수 Q3 (上四分位数)**
    - 데이터의 상위 25% 지점(盒子的上边缘)
  - ✓**최댓값 (最大值)**
    - 데이터에서 가장 큰 값(图中最上面的"胡须")
  - ✓**이상치 Outlier (异常值)**
    - 다른 값들과 동떨어진 값(以单独点的形式显示)

넘파이(numpy)를 이용하여 데이터 다루기

38

- 넘파이란? Numerical Python의 줄임말
  - 수치 계산을 빠르고 편리하게 할 수 있는 라이브러리
  - 리스트로는 대량의 데이터를 다루는데 어려움이 있지만, 넘파이는 벡터(1차원), 행렬(2차원) 연산을 효율적으로 지원
  - 주요 기능
    - 배열(array) 생성: np.array, np.arange, np.linspace
    - 랜덤 데이터 생성: np.random.randint, np.random.normal
    - 수학 연산: +, -, *, /, 제곱 등 빠른 연산 가능

参考

- 넘파이(Numpy)란? (什么是Numpy?)
  - ✓Numpy 是 **Numerical Python** 的缩写
  - ✓一个可以快速、方便地进行数值计算的库
  - ✓Python 的列表处理大规模数据时效率低，而 Numpy 可以高效地支持 **向量(一维)、矩阵(二维)** 运算
- **주요 기능 (主要功能)**
  - ✓**배열(array) 생성 (创建数组)**
    - np.array, np.arange, np.linspace
  - ✓**랜덤 데이터 생성 (生成随机数据)**
    - np.random.randint, np.random.normal
  - ✓**수학 연산 (数学运算)**
    - +, -, *, /, 거듭제곱 등 빠른 연산 가능
    - 可以进行 +, -, *, /, 平方等快速运算

넘파이 기본 실습

39

- 리스트에 각 10 더하기
  - list_a = [1, 2, 3, 4, 5]
  - [x+10 for x in list_a] 또는
  - for i in range(5):

    list_a[i]= list_a[i]+10
- 넘파이를 이용하여 10씩 더하기
  - np_a = np.array([1, 2, 3, 4, 5])
  - np_a+10
- 배열 만들기
  - np_a = np.array([1, 2, 3, 4, 5])
  - np.arange(1, 6) #1~5
  - x = np.arange(1, 11) # 1~10
  - y = x * 2 # y=2x
- 랜덤 배열 만들기
  - scores = np.random.randint(60, 100, size=5)
  - print(scores)
  - group1 = np.random.randint(60, 100, 20)
  - group2 = np.random.randint(50, 95, 20)

参考

기온데이터 다운로드

参考

- https://data.kma.go.kr/ -> 데이터 -> 세계기상전문
- 국가명, 도시명 선택
- 회원가입 필요

CSV 파일 열기

41

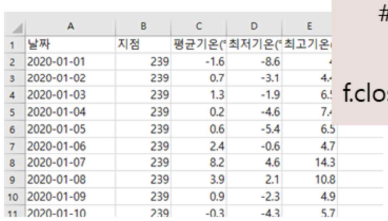
- f = open("파일이름", "모드", encoding="cp949")
  - 파일이름 : 열고 싶은 파일의 경로 (예: "data.csv")
  - 모드 : 파일을 어떻게 열지 지정(r, w, a 등)
  - encoding : 문자를 컴퓨터가 이해할 수 있는 **숫자(이진 코드)**로 바꿔 저장하는 방식
    - utf-8이 기본, 안 되면 cp949
  - reader = csv.reader(f)
  - f 파일 객체를 한 줄씩 읽어서, 각 줄을 **리스트 형태**로 변환해 줌
  - 이때 reader자체는 리스트가 아니라 하나씩 순차적으로 꺼낼 수 있는 이터레이터(iterator) 객체
- 반복을 통해 각 행을 출력
  - for row in reader:
    print(row)

CODE

```
import csv
f = open("temdata2010.csv", "r", encoding="cp949")
reader = csv.reader(f)
next(reader)  # 첫 줄(헤더) 건너뛰기

for row in reader:
    print(row)
    #dates.append(row[0])          # 날짜
    #temps.append(float(row[-1])) # 기온

f.close()  # 파일 닫기
```



	A	B	C	D	E
1	날짜	지점	평균기온(℃	최저기온(℃	최고기온(
2	2020-01-01	239	-1.6	-8.6	
3	2020-01-02	239	0.7	-3.1	
4	2020-01-03	239	1.3	-1.9	
5	2020-01-04	239	0.2	-4.6	
6	2020-01-05	239	0.6	-5.4	6.5
7	2020-01-06	239	2.4	-0.6	4.7
8	2020-01-07	239	8.2	4.6	14.3
9	2020-01-08	239	3.9	2.1	10.8
10	2020-01-09	239	0.9	-2.3	4.9
11	2020-01-10	239	-0.3	-4.3	5.7

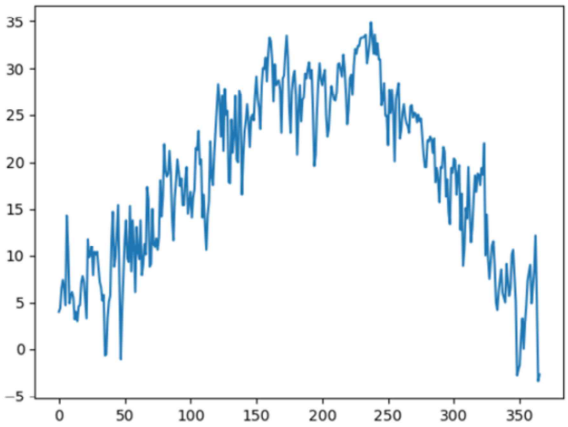
参考

- **파일 열기 (打开文件)**
  - ✓f = open("파일이름", "모드", encoding="utf-8")
  - ✓**파일이름 (文件名)**: 열고 싶은 파일의 경로
    예: "data.csv"
    例如: "data.csv"
  - ✓**모드 (模式)**: 파일을 어떻게 열지 지정
    - "r" : 읽기 (只读)
    - "w" : 쓰기 (写入, 기존 내용 삭제)
    - "a" : 추가 (追加, 기존 내용 뒤에 씀)
- **encoding**: 한글 깨짐 방지를 위해 "utf-8" 권장
  - ✓推荐使用 "utf-8" 防止韩文乱码
  - ✓UTF-8 是国际标准，大多数情况下默认使用。
  - ✓但是在韩国 Windows 上创建的旧文件，很多是用 CP949 保存的。
  - ✓所以记住：**"先试 UTF-8，不行再用 CP949"**。
- **CSV 읽기 (读取 CSV 文件)**
  - ✓reader 는 f 파일 객체를 한 줄씩 읽어서 → **리스트(list)** 형태로 변환
  - ✓단, reader 자체는 리스트가 아니라 **이터레이터(iterator)** → 하나씩 순차적으로 꺼낼 수 있음
  - ✓**中文**: reader 会逐行读取文件，并把每一行转成 **列表**。
    但 reader 本身不是列表，而是一个 **迭代器**，可以逐个取出数据。

CSV 파일에서 날짜와 기온 데이터를 읽어 리스트에 저장

42

- 마지막 값(row[-1])이 비어 있지 않으면 → 실수형의 숫자로 변환해 temps 리스트에 저장
- 기온리스트(temps)로 그래프 그리기



CODE

```
import csv

dates = []
temps = []

# CSV 파일 열기
f = open("temdata2010.csv", "r", encoding="cp949")
reader = csv.reader(f)

next(reader) # 첫 줄(헤더) 건너뛰기

for row in reader:
    if row[-1] != '':                # 마지막 열(기온)이 비어 있지 않으면
        dates.append(row[0])         # 첫 번째 열(날짜)을 dates 리스트에 저장
        temps.append(float(row[-1])) # 마지막 열(기온)을 temps 리스트에 저장

f.close() # 파일 닫기
print(temps) # 기온 리스트 출력
plt.plot(temps)
plt.show()
```

参考

- 从 temdata2010.csv 文件中逐行读取数据。
- 如果最后一列 (row[-1]) 不为空，就把它转换为 **浮点数 (float)**，并保存到 **气温列表 (temps)** 中。
- 同时把第一列 (row[0]) 保存为 **日期列表 (dates)**。
- 最后使用 **气温列表 (temps)** 绘制折线图，横轴是日期，纵轴是气温。

세 도시의 온도 비교

43

```
#대전 (Daejeon)
years_dae = []
temps_dae = []
f = open( " data1_dae.csv " , " r " , encoding= " cp949 " )
data = csv.reader(f)
next(data) # 헤더 건너뛰기
for row in data:
    year = row[2].split( ' - ' )[0]  # 연도 추출
    month = row[2].split( ' - ' )[1]  # 월 추출
    if month ==  ' 10 '  and 2010 <= int(year) <= 2019:
        years_dae.append(int(year))
        temps_dae.append(float(row[3])) # 월평균 기온
f.close()
```

```
#광저우 (Guangzhou)
years_gua = []
temps_gua = []

f = open("data2_guang.csv", "r", encodi
data = csv.reader(f)
next(data)
for row in data:
    year = row[2].split('-')[0]
    month = row[2].split('-')[1]
    if month == '10' and 2010 <= int(year) <= 2019:
        years_gua.append(int(year))
        temps_gua.append(float(row[3]))
f.close()
```

```
#상하이 (Shanghai)
years_sha = []
temps_sha = []

f = open("data3_shang.csv", "r",
encoding="cp949")
data = csv.reader(f)
next(data)
for row in data:
    year = row[2].split('-')[0]
    month = row[2].split('-')[1]
    if month == '10' and 2010 <= int(year)
<= 2019:
        years_sha.append(int(year))
        temps_sha.append(float(row[3]))
f.close()
```

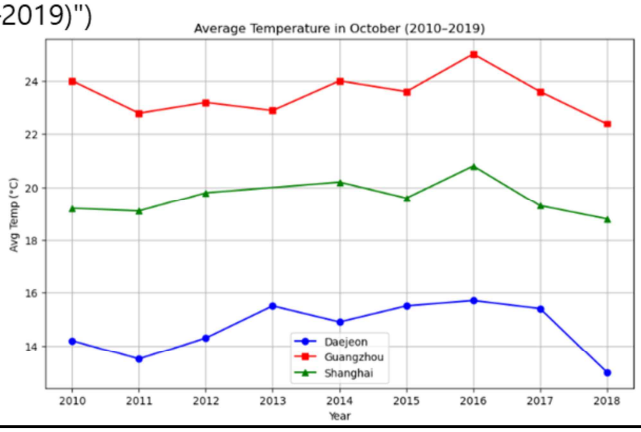
参考

1. CSV 파일 읽기 (读取 CSV 文件)
   - ✓각 도시(대전, 광저우, 상하이)의 CSV 파일을 열고 데이터를 읽는다.(打开每个城市（大田、广州、上海）的 CSV 文件并读取数据。)
2. 필요한 데이터 선택 (选择需要的数据)
   - ✓날짜에서 연도와 월을 분리한다. (从日期中分离出年份和月份)
   - ✓조건: 2010~2019년, 10월 데이터만 선택한다.(条件：只选择 2010~2019 年 的 10 月 数据)
   - ✓기온 데이터를 실수형(float)으로 변환하여 리스트에 저장한다.(把气温数据转换为浮点数 (float)，保存到列表中)
3. 도시별 리스트 준비 (准备每个城市的列表)
   - ✓years_dae, temps_dae (대전)
   - ✓years_gua, temps_gua (광저우)
   - ✓years_sha, temps_sha (상하이)
4. 2010~2019년 10월 동안 대전·광저우·상하이의 평균 기온을 CSV에서 읽어와 그래프로 비교한다(从 CSV 文件中读取 2010~2019 年 10 月大田、广州、上海的平均气温，并绘制对比图)

세 도시의 온도 그래프로 표현하기

44

```
plt.figure(figsize=(10,6))
plt.plot(years_dae, temps_dae, label="Daejeon", color="blue", marker="o")
plt.plot(years_gua, temps_gua, label="Guangzhou", color="red", marker="s")
plt.plot(years_sha, temps_sha, label="Shanghai", color="green", marker="^")
plt.title("Average Temperature in October (2010–2019)")
plt.xlabel("Year")
plt.ylabel("Avg Temp (°C)")
plt.legend()
plt.grid(True)
plt.show()
```



参考