파이썬 프로그래밍

19. 딥러닝 파이썬 패키지(1)

00 Contents

❖ 수업 목표

- pandas 모듈의 기본 함수를 사용할 수 있다.
- numpy 모듈의 기본 함수를 사용할 수 있다.
- matplotlib 모듈의 기본 함수를 사용할 수 있다.

❖ 세부 목표

1. pandas 개요

❖ Pandas(판다스) 개요

- 데이터 분석과 조작을 위한 Python 라이브러리
  - CSV 파일 등의 데이터를 읽고 원하는 데이터 형식으로 변환
    - CSV(Comma Separated Values) 파일 : 데이터를 쉼표로 구분한 파일

- 주요 자료구조
  - Series: 1차원 배열, 라벨이 있는 데이터
  - DataFrame: 2차원 테이블 구조, 라벨이 있는 행과 열

  - 테이블이란? 가로(행)과 세로(열)로 정렬된 데이터 집합
    - 행(row, 로우)
    - 열(column, 컬럼)

1. pandas 개요

❖ Pandas 설치

- 파이썬 표준 모듈이 아니므로 추가 설치 필요
- pip 명령어로 설치 가능
  - pip install pandas
- Pandas 사용을 위해 import 수행
  - import pandas as pd

- pip 이란?
  - Python 패키지 관리자(Package Installer for Python)
  - Python 라이브러리를 설치 및 관리 위해 사용
  - 주요 명령어 (터미널 명령 프롬프트에서 실행)
    - 패키지 설치 : pip install <패키지명>
    - 패키지 제거 : pip uninstall <패키지명>
    - 설치된 패키지 목록 확인 : pip list

1. pandas 개요

❖ Series 자료구조

▪ Series 생성

- 리스트 데이터를 사용한 Series 객체 생성
  - 따로 인덱스를 지정하지 않았다면 0부터 시작하는 정수 값으로 인덱싱 됨

```
1   import pandas as pd
2
3   data = [10,20,30]
4   series = pd.Series(data)
5   print(series)
```

- 출력 결과
  - 0 10
  - 1 20
  - 2 30
  - dtype: int64

1. pandas 개요

❖ Series 자료구조

▪ Series 생성

- 딕셔너리 데이터를 사용한 Series 객체 생성
  - key를 인덱스 값으로 사용

```
1   import pandas as pd
2
3   data = {'a':10,'b':20,'c':30}
4   series = pd.Series(data)
5   print(series)
```

- 출력 결과
  - a 10
  - b 20
  - c 30
  - dtype: int64

1. pandas 개요

❖ DataFrame 자료구조

▪ DataFrame 생성

- 중첩 리스트 데이터를 사용한 DataFrame 객체 생성
  - 컬럼 명을 옵션으로 지정 가능

```
import pandas as pd

data = [
    [1, 'Alice', 30],
    [2, 'Bob', 35],
    [3, 'Charlie', 25]
]
df = pd.DataFrame(data, columns=['ID', 'Name', 'Age'])
print(df)
```

- 출력 결과

```
   ID     Name  Age
0   1    Alice   30
1   2      Bob   35
2   3  Charlie   25
```

1. pandas 개요

❖ DataFrame 자료구조

▪ DataFrame 생성

- 딕셔너리 데이터를 사용한 DataFrame 객체 생성
  - key를 컬럼 명으로 사용

```
import pandas as pd

data = {
    'ID': [1, 2, 3],
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [30, 35, 25]
}
df = pd.DataFrame(data)
print(df)
```

- 출력 결과

```
      ID     Name  Age
0   1    Alice    30
1   2      Bob    35
2   3  Charlie    25
```

1. pandas 개요

❖ 데이터 탐색

▪ 데이터 로드

• CSV 파일 읽기

- df_csv = pd.read_csv('D:/rokey/ch19/data.csv')

- CSV 파일 구조
  - » 줄 바꾼으로 구분되는 1개 이상의 행(레코드)으로 구성
  - » 각 행은 1개 이상의 열(필드)로 구성
  - » 필드는 큰따옴표로 둘러싸도 됨 (단, 통일성은 있어야 함)

- CSV 파일 기본 생성
  - » 텍스트 에디터에 아래와 같이 작성 후, 확장자를 (.csv)로 저장
- 예)
  - » ID,이름,가격
  - » 1,비누,300
  - » 2,장갑,150
  - » 3,마스크,230
  - » 4,손수건,400
  - » 5,볼펜,200
  - » 6,건전지,250

1. pandas 개요

❖ 데이터 탐색

▪ 데이터 로드

• Excel 파일 읽기

- openpyxl 모듈 필요
  - » 파이썬 표준 모듈이 아니므로 추가 설치 필요
- pip 명령어로 설치
  - » pip install openpyxl
- 실행 프로그램
  - » df_xl = pd.read_excel('D:/rokey/ch19/data.xlsx')

- Excel 파일 구조
  - » 파일을 북(book)이라고 함
  - » 하나의 북 내부에는 여러 개의 시트(sheet)가 존재
  - » 각 시트는 행(row)와 열(column)을 가진 2차원 셀(cell)로 구성

- Excel 파일 기본 생성
  - » 마이크로소프트 엑셀 프로그램을 통한 생성

1. pandas 개요

❖ 데이터 탐색

▪ 데이터 탐색

- 데이터 상위 5개 행 보기
  - print(df_csv.head())
  - 출력 결과
    - » ID 이름 가격
    - » 0 1 비누 300
    - » 1 2 장갑 150
    - » 2 3 마스크 230
    - » 3 4 손수건 400
    - » 4 5 볼펜 200

- 데이터 하위 5개 행 보기
  - print(df_csv.tail())
    - » ID 이름 가격
    - » 1 2 장갑 150
    - » 2 3 마스크 230
    - » 3 4 손수건 400
    - » 4 5 볼펜 200
    - » 5 6 건전지 250

1. pandas 개요

❖ 데이터 탐색

▪ 데이터 탐색

• 데이터 요약 정보

- print(df.info())

- 출력 결과
  - » <bound method DataFrame.info of ID Name Age
  - » 0 1 Alice 30
  - » 1 2 Bob 35
  - » 2 3 Charlie 25>

1. pandas 개요

❖ 데이터 탐색

▪ 데이터 탐색

• 기술 통계

- print(df.describe())

- 출력 결과
  - » ID Age
  - » count 3.0 3.0
  - » mean 2.0 30.0
  - » std 1.0 5.0
  - » min 1.0 25.0
  - » 25% 1.5 27.5
  - » 50% 2.0 30.0
  - » 75% 2.5 32.5
  - » max 3.0 35.0

1. pandas 개요

❖ 데이터 활용

▪ 데이터 샘플링

- 랜덤 샘플링:
  - print(df.sample(2)) # 2개
  - 출력 결과
    - » ID 이름 가격
    - » 0 1 비누 300
    - » 2 3 마스크 230
- 특정 비율로 샘플링:
  - print(df.sample(frac=0.5)) # 50%
  - 출력 결과
    - » ID 이름 가격
    - » 5 6 건전지 250
    - » 4 5 볼펜 200
    - » 0 1 비누 300

2. pandas 활용

❖ 데이터 조작

▪ 데이터 선택 및 필터링

- 열 선택
  - print(df['Name'])
  - 실행 결과
    - » 0 Alice
    - » 1 Bob
    - » 2 Charlie
    - » Name: Name, dtype: object
- 조건 필터링
  - filtered = df[df['Age'] > 30]
  - print(filtered)
  - 실행 결과
    - » ID Name Age
    - » 1 2 Bob 35

2. pandas 활용

❖ 데이터 조작

▪ 데이터 정렬

• 값으로 정렬

- sorted_df = df.sort_values(by='Age')
- print(sorted_df)

- 실행 결과
  - » ID Name Age
  - » 2 3 Charlie 25
  - » 0 1 Alice 30
  - » 1 2 Bob 35

2. pandas 활용

❖ 데이터 조작

▪ 데이터 추가 및 삭제

• 열 추가

- df['Salary'] = [50000, 60000, 70000]
- print(df)

- 실행 결과

```
»     ID     Name  Age  Salary
» 0   1     Alice   30   50000
» 1   2      Bob   35   60000
» 2   3  Charlie   25   70000
```

2. pandas 활용

❖ 데이터 조작

▪ 데이터 추가 및 삭제

- 행 추가
  - df.loc[len(df)] = [4, 'David', 40, 80000]
  - print(df)

  - 실행 결과
    - » ID Name Age Salary
    - » 0 1 Alice 30 50000
    - » 1 2 Bob 35 60000
    - » 2 3 Charlie 25 70000
    - » 3 4 David 40 80000

2. pandas 활용

❖ 데이터 조작

▪ 데이터 추가 및 삭제

• 행 삭제

- df = df.drop(1) # 1번 인덱스 행 삭제
- print(df)

- 실행 결과

```
»    ID   Name  Age  Salary
» 0   1    Alice   30   50000
» 2   3  Charlie   25   70000
» 3   4    David   40   80000
```

2. pandas 활용

❖ 데이터 조작

▪ 데이터 연결

• DataFrame 행 연결

- concat() 함수 매개변수는 데이터 프레임 리스트형

```
data2 = {
    'ID': [5, 6],
    'Name': ['Eve', 'Frank'],
    'Age': [28, 33]
}
df2 = pd.DataFrame(data2)
concated = pd.concat([df, df2])
print(concated)
```

- 실행 결과

```
»    ID    Name  Age   Salary
» 0   1    Alice   30  50000.0
» 2   3  Charlie   25  70000.0
» 3   4    David   40  80000.0
» 0   5      Eve   28      NaN
» 1   6    Frank   33      NaN
```

• NaN(Not a Number)

- 결측값(NA, Not Available) 표현
- 정의되지 않은 값 또는 표현할 수 없는 값 또는 누락된 값
- 값이 없는 경우, None 으로 데이터를 입력 가능

2. pandas 활용

❖ 데이터 조작

▪ 데이터 병합

• DataFrame 열 병합

```
data3 = {
    'ID': [1,2,3,4,5,6],
    'Department': ['HR','Engineering','Sales','R&D','Finance','Planning']
}
df3 = pd.DataFrame(data3)
merged = pd.merge(concated, df3)
print(merged)
```

- 실행 결과

```
»    ID   Name  Age   Salary  Department
» 0   1    Alice   30  50000.0        HR
» 1   3  Charlie   25  70000.0     Sales
» 2   4    David   40  80000.0       R&D
» 3   5      Eve   28     NaN   Finance
» 4   6    Frank   33     NaN  Planning
```

2. pandas 활용

❖ 데이터 처리

▪ 결측치 처리

• 결측치 확인

- print(df.isnull().sum())

- 실행 결과
  - » ID 0
  - » Name 0
  - » Age 0
  - » Salary 2
  - » Department 0
  - » dtype: int64

2. pandas 활용

❖ 데이터 처리

▪ 결측치 처리

• 결측치 채우기

- meanVal = merged['Salary'].mean()
- merged['Salary'] = merged['Salary'].fillna(meanVal)
- print(merged)

- 실행 결과

```
»    ID    Name  Age       Salary    Department
» 0   1    Alice   25  50000.000000         HR
» 1   3  Charlie   35  70000.000000      Sales
» 2   4    David   40  80000.000000        R&D
» 3   5      Eve   28  66666.666667    Finance
» 4   6    Frank   33  66666.666667   Planning
```

2. pandas 활용

❖ 데이터 처리

▪ 중복 데이터 처리

• 중복 확인

```
data1 = {
    'ID': [1, 3],
    'Name': ['Alice', 'Charlie'],
    'Age': [30, 25],
    'Salary': [50000, 70000],
    'Department': ['HR', 'Sales']
}
df1 = pd.DataFrame(data1)
df1 = pd.concat([merged, df1])
print(df1)
print(df1.duplicated())
```

- 실행 결과
  - » 0 False
  - » 1 False
  - » 2 False
  - » 3 False
  - » 4 False
  - » 0 True
  - » 1 True
  - » dtype: bool

2. pandas 활용

❖ 데이터 처리

▪ 중복 데이터 처리

• 중복 제거

- df1_1 = df1.drop_duplicates()
- print(df1_1)

- 실행 결과

```
»    ID   Name  Age   Salary Department
» 0   1   Alice   30  50000.0         HR
» 1   3 Charlie   25  70000.0      Sales
» 2   4   David   40  80000.0        R&D
» 3   5     Eve   28  66666.0    Finance
» 4   6   Frank   33  66666.0      Sales
```

3. numpy 개요

❖ Numpy(넘파이, Numerical Python) 개요

- 다차원 배열 및 행렬 연산 지원 라이브러리
- 과학 계산을 위한 다양한 함수 제공

❖ Numpy 설치

- 파이썬 표준 모듈이 아니므로 추가 설치 필요
  - 단, Pandas 설치 시, 자동 설치 됨
- pip 명령어로 설치 가능
  - pip install numpy
- Numpy 사용을 위해 import 수행
  - import numpy as np

3. numpy 개요

❖ Numpy 배열의 기본

▪ 배열 생성

- arr = np.array([1, 2, 3, 4, 5])
- print(arr)

  - 실행 결과
    - » [1 2 3 4 5]

▪ 배열 속성 확인

- print(arr.shape) # 배열의 모양
- print(arr.dtype) # 데이터 타입

  - 실행 결과
    - » (5,)
    - » int64

3. numpy 개요

❖ 배열 생성 및 초기화

▪ 다양한 배열 생성 방법

- 0으로 초기화된 배열
  - zeros = np.zeros((3, 3))
  - print(zeros)

  - 실행 결과
    - » [[0. 0. 0.]
    - » [0. 0. 0.]
    - » [0. 0. 0.]]

- 1로 초기화된 배열:
  - ones = np.ones((2, 4))
  - print(ones)

  - 실행 결과
    - » [[1. 1. 1. 1.]
    - » [1. 1. 1. 1.]]

3. numpy 개요

❖ 배열 생성 및 초기화

▪ 다양한 배열 생성 방법

- 특정 값으로 채운 배열:
  - full = np.full((2, 2), 7)
  - print(full)

  - 실행 결과
    - » [[7 7]
    - » [7 7]]

3. numpy 개요

❖ 배열 생성 및 초기화

▪ 다양한 배열 생성 방법

- 단위 행렬: 대각선 상의 원소가 1이고 나머지 원소는 모두 0인 행렬
  - identity1 = np.eye(2, 3)
  - identity2 = np.identity(3)
  - print(identity1)
  - print(identity2)

  - 실행 결과
    - » [[1. 0. 0.]
    - » [0. 1. 0.]]

    - » [[1. 0. 0.]
    - » [0. 1. 0.]
    - » [0. 0. 1.]]

3. numpy 개요

❖ 배열 생성 및 초기화

▪ 랜덤 배열 생성

- 난수 배열
  - random = np.random.rand(3, 3)
  - print(random)

  - 실행 결과
    - » [[0.98926147 0.25809324 0.38728431]
    - » [0.72484548 0.54439084 0.42520048]
    - » [0.10259329 0.32706793 0.83867207]]

- 정수 난수 배열
  - randint = np.random.randint(1, 10, (3, 3))
  - print(randint)

  - 실행 결과
    - » [[7 7 7]
    - » [2 6 5]
    - » [1 9 7]]

4. numpy 활용

❖ 배열 연산

▪ 기본 산술 연산

• 요소별 연산

- arr = np.array([1, 2, 3, 4])
- print(arr + 5)
- print(arr * 2)

- 실행 결과
  - » [6 7 8 9]
  - » [2 4 6 8]

4. numpy 활용

❖ 배열 연산

▪ 통계 함수

• 합계 및 평균

- print(arr.sum())
- print(arr.mean())

- 실행 결과
  - » 10
  - » 2.5

• 최대값과 최소값

- print(arr.max())
- print(arr.min())

- 실행 결과
  - » 4
  - » 1

4. numpy 활용

❖ 배열 연산

▪ 브로드캐스팅

• 배열 간 연산

- arr1 = np.array([1, 2, 3])
- arr2 = np.array([[1], [2], [3]])
- result = arr1 + arr2
- print(result)

- 실행 결과
  - » [[2 3 4]
  - » [3 4 5]
  - » [4 5 6]]

4. numpy 활용

❖ 배열 연산

▪ 선형 대수 연산

- 행렬 곱:
  - matrix1 = np.array([[1, 2], [3, 4]])
  - matrix2 = np.array([[5, 6], [7, 8]])
  - result = np.dot(matrix1, matrix2)
  - print(result)

  - 실행 결과
    - » [[19 22]
    - » [43 50]]

4. numpy 활용

❖ 배열 인덱싱 및 슬라이싱

▪ 기본 인덱싱

- 1차원 배열:
  - arr = np.array([10, 20, 30, 40])
  - print(arr[2])
  - 실행 결과
    - » 30
- 다차원 배열:
  - arr = np.array([[1, 2, 3], [4, 5, 6]])
  - print(arr[1, 2])
  - 실행 결과
    - » 6

4. numpy 활용

❖ 배열 인덱싱 및 슬라이싱

▪ 슬라이싱

- **배열 부분 선택:**
  - **print(arr[0, :]) # 첫 번째 행**
  - **print(arr[:, 1]) # 두 번째 열**

  - **실행 결과**
    - » **[1 2 3]**
    - » **[2 5]**

4. numpy 활용

❖ 고급 기능

▪ 조건부 연산

- 조건 필터링:
  - arr = np.array([1, 2, 3, 4, 5])
  - filtered = arr[arr > 3]
  - print(filtered)

  - 실행 결과
    - » [4 5]

4. numpy 활용

❖ 고급 기능

▪ Numpy와 Pandas 통합 사용

• Pandas DataFrame으로 변환

- import pandas as pd

- df = pd.DataFrame(arr, columns=['Value'])
- print(df)

- 실행 결과
  - » Value
  - » 0 1
  - » 1 2
  - » 2 3
  - » 3 4
  - » 4 5

5. matplotlib 개요

❖ 데이터 시각화 개요

- 데이터를 그래프로 표현하여 이해를 돕는 과정
- 의사 결정 및 통찰력 발견에 필수적

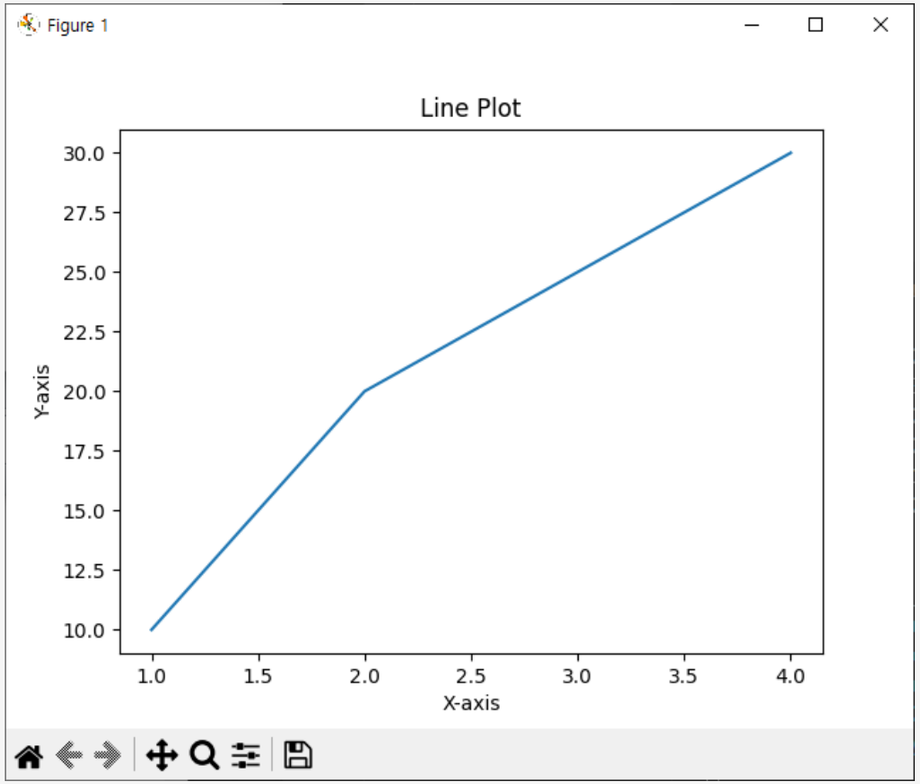
- matplotlib 란?
  - 정적, 애니메이션, 대화형 시각화를 만드는 포괄적 라이브러리
    - 플롯(plot): 두 개 이상의 변수간의 관계를 보여주는 그래프

- 주요 Python 시각화 라이브러리 비교
  - Matplotlib : 기본적이고 유연한 시각화 도구
  - Seaborn : 통계적 데이터 시각화에 적합

5. matplotlib 개요

❖ Matplotlib 기본 사용법

- 설치
  - pip install matplotlib
- 임포트
  - import matplotlib.pyplot as plt
- 기본 플롯 그리기
  - 선 그래프
    - x = [1, 2, 3, 4]
    - y = [10, 20, 25, 30]
    - plt.plot(x, y)
    - plt.title("Line Plot")
    - plt.xlabel("X-axis")
    - plt.ylabel("Y-axis")
    - plt.show()

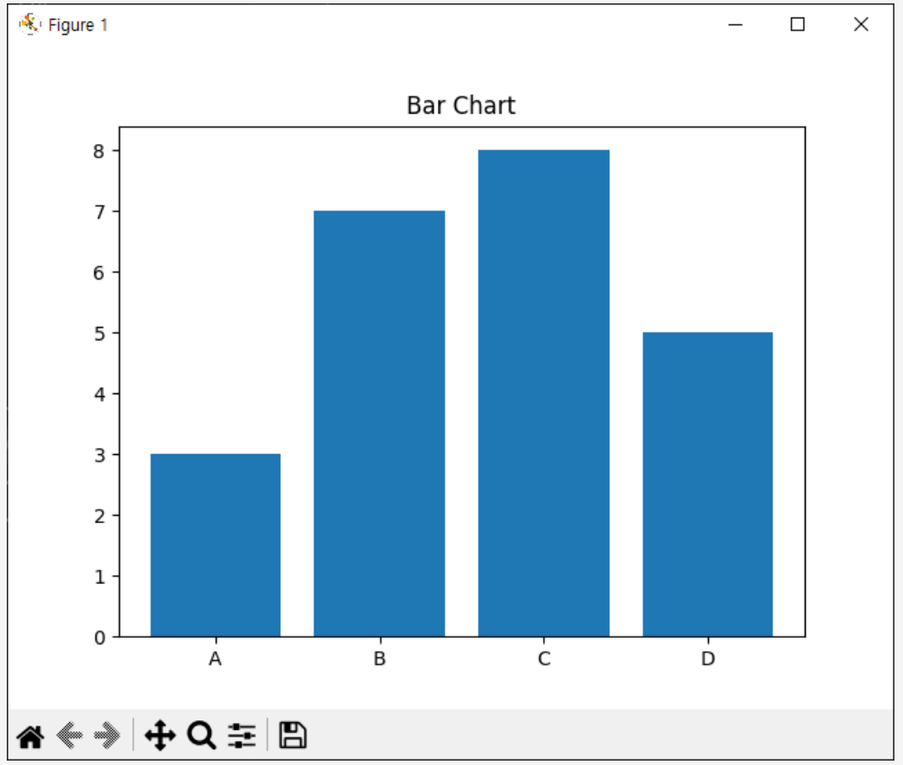


5. matplotlib 개요

❖ 다양한 그래프 종류

▪ Bar Chart (막대 그래프)

- 범주가 있는 데이터 값을 직사각형의 막대로 표현하는 그래프
  - categories = ["A", "B", "C", "D"]
  - values = [3, 7, 8, 5]
  - plt.bar(categories, values)
  - plt.title("Bar Chart")
  - plt.show()



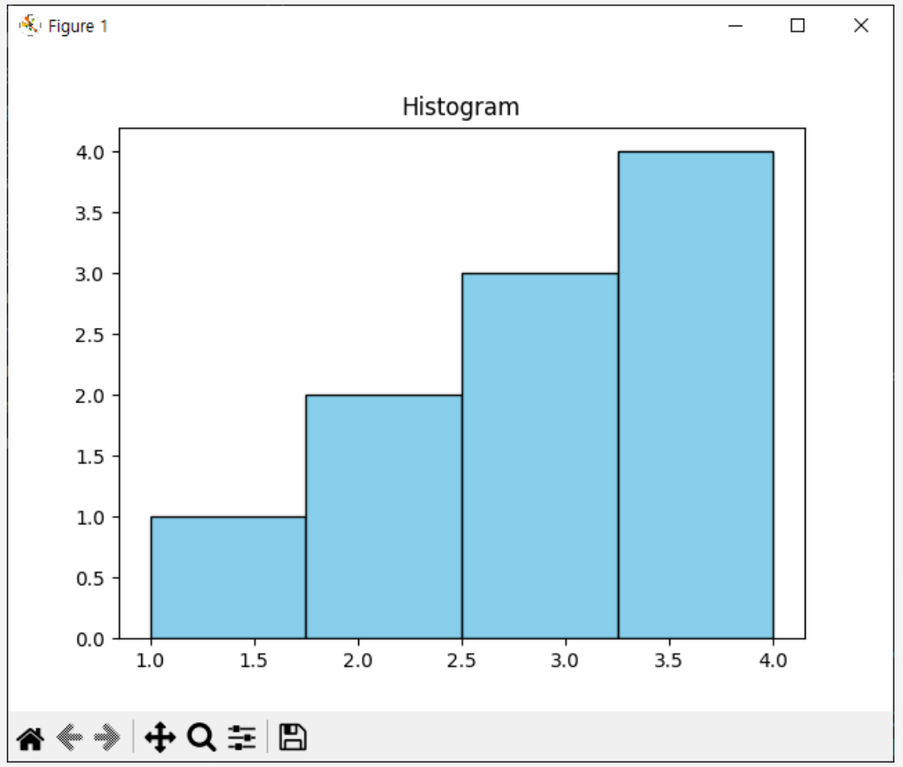
5. matplotlib 개요

❖ 다양한 그래프 종류

▪ Histogram (히스토그램)

- 도수분포표를 그래프로 표현. 가로축은 계급, 세로축은 도수(횟수나 개수)
  - data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
  - plt.hist(data, bins=4, color="skyblue", edgecolor="black")
  - plt.title("Histogram")
  - plt.show()

» bins : 전체 범위를 몇 등분할지 표현

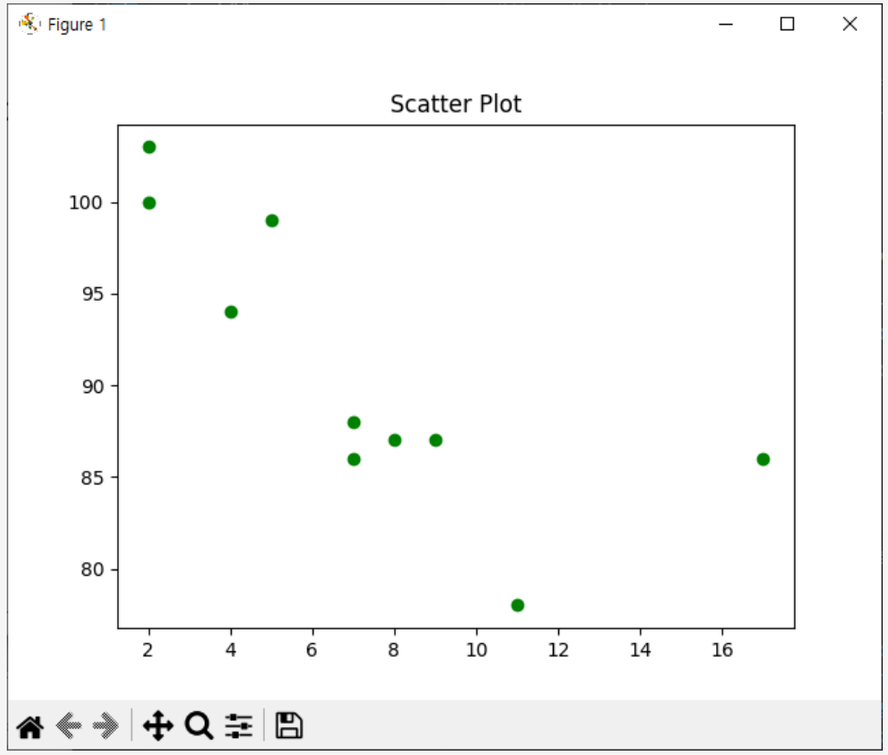


5. matplotlib 개요

❖ 다양한 그래프 종류

▪ Scatter Plot (산점도)

- 두 변수의 상관 관계를 직교 좌표계의 평면에 점으로 표현하는 그래프
  - x = [5, 7, 8, 7, 2, 17, 2, 9, 4, 11]
  - y = [99, 86, 87, 88, 100, 86, 103, 87, 94, 78]
  - plt.scatter(x, y, color="green")
  - plt.title("Scatter Plot")
  - plt.show()



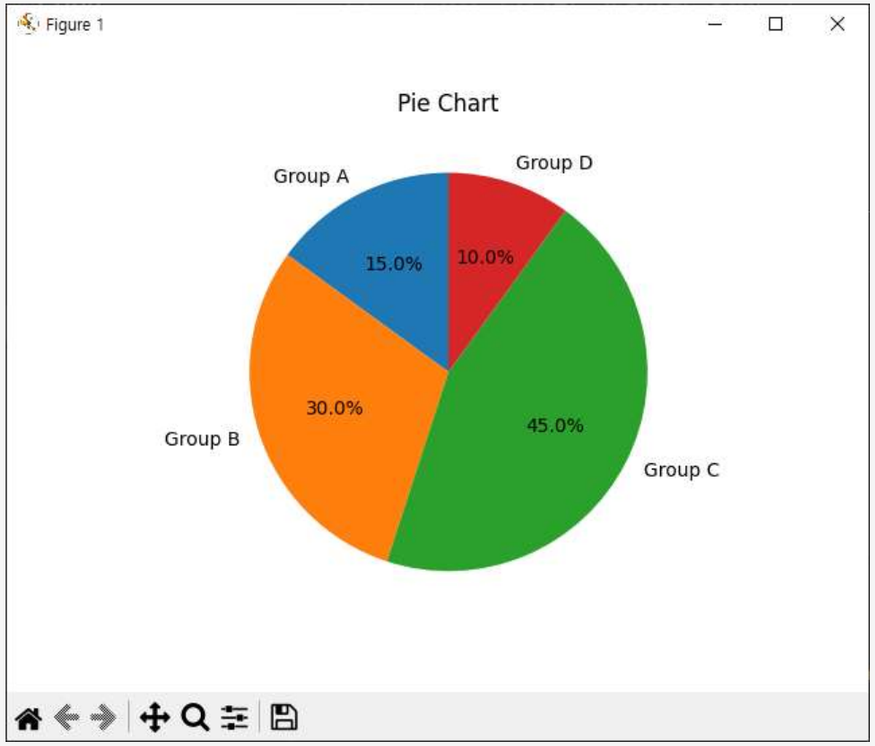
5. matplotlib 개요

❖ 다양한 그래프 종류

▪ Pie Chart (파이 차트, 원 그래프)

- 범주별 구성 비율을 원형으로 표현한 그래프
  - sizes = [15, 30, 45, 10]
  - labels = ["Group A", "Group B", "Group C", "Group D"]
  - plt.pie(sizes, labels=labels, autopct="%1.1f%%", startangle=90)
  - plt.title("Pie Chart")
  - plt.show()

» autopct : 부채꼴 안 표시 숫자 형식 지정
» startangle : 시작 각도(기본 x축)



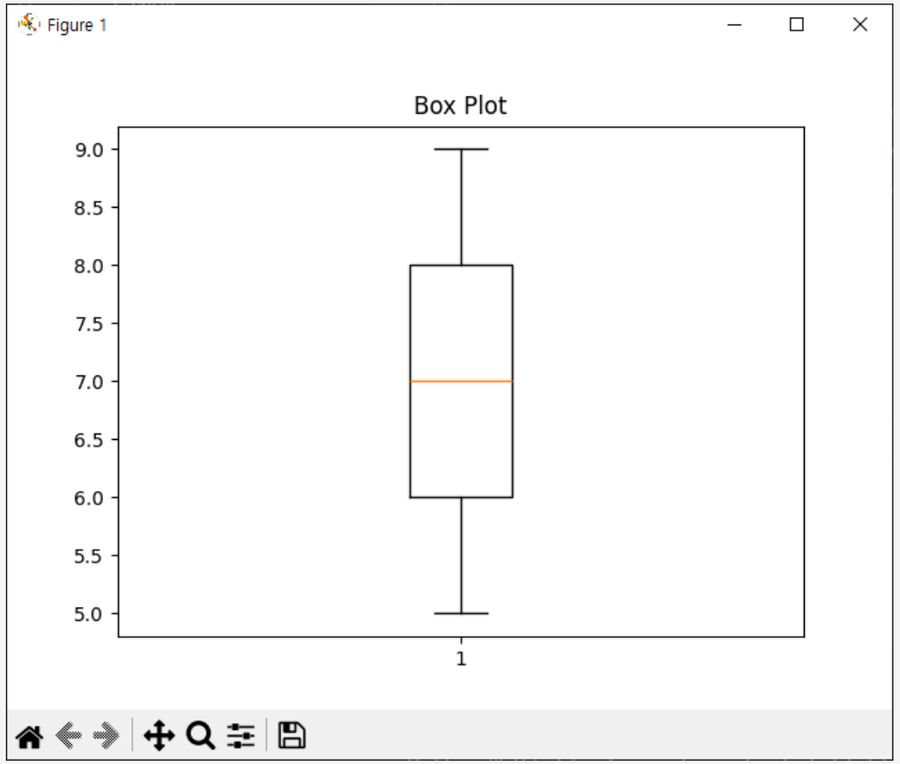
5. matplotlib 개요

❖ 다양한 그래프 종류

▪ Box Plot (박스 플롯)

- 수치 데이터를 표현하는 하나의 방식
- 전체 데이터로부터 얻어진 다섯 가지 요약 수치를 사용
  - 최소값
  - 제 1사분위 수 (Q1)
  - 제 2사분위 수 또는 중위수 (Q2)
  - 제 3사분위 수 (Q3)
  - 최대값

```
data = [7, 8, 5, 6, 8, 9, 6, 7, 5, 8]
plt.boxplot(data)
plt.title("Box Plot")
plt.show()
```

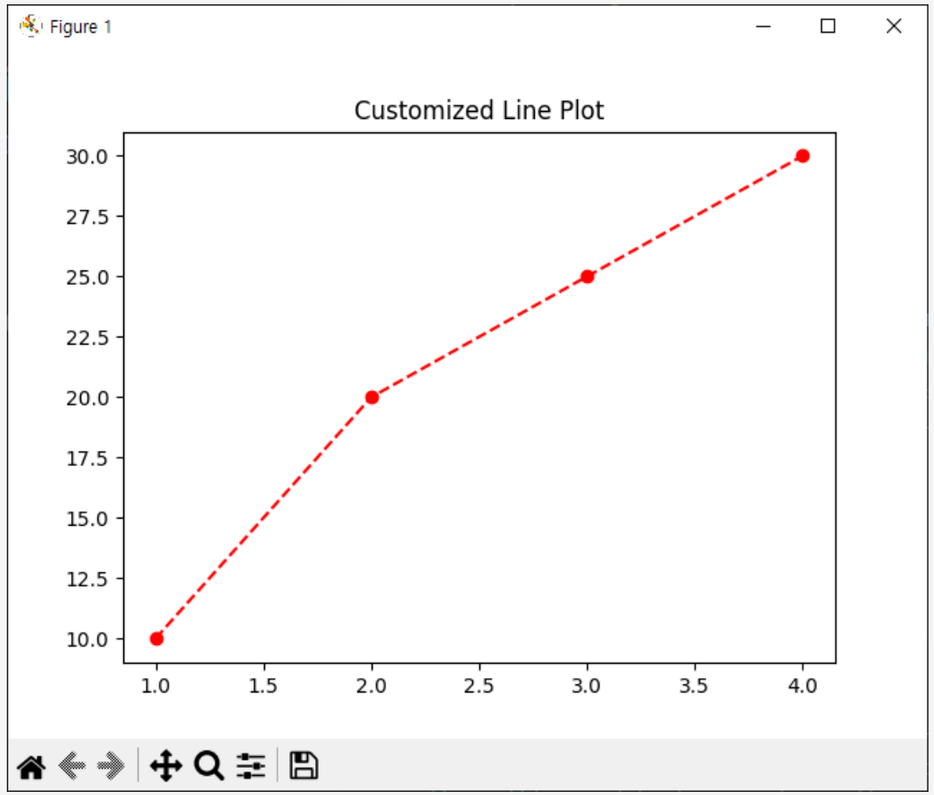


6. matplotlib 활용

❖ 그래프 커스터마이징

▪ 색상, 선 스타일, 마커

- plt.plot(x, y, color="red", linestyle="--", marker="o")
- plt.title("Customized Line Plot")
- plt.show()



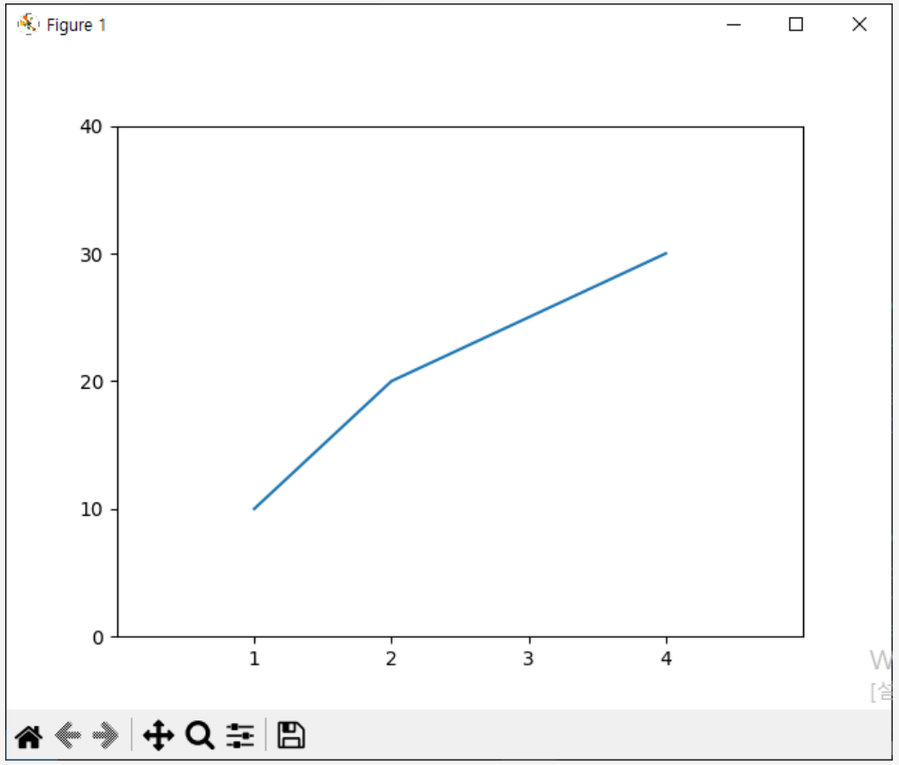
6. matplotlib 활용

❖ 그래프 커스터마이징

▪ 축 범위와 눈금 설정

- plt.plot(x, y)
- plt.xlim(0, 5)
- plt.ylim(0, 40)
- plt.xticks(range(1, 5))
- plt.yticks(range(0, 41, 10))
- plt.show()

  - » xlim : x축 표시 범위
  - » ylim : y축 표시 범위
  - » xticks : x축 눈금
  - » yticks : y축 눈금

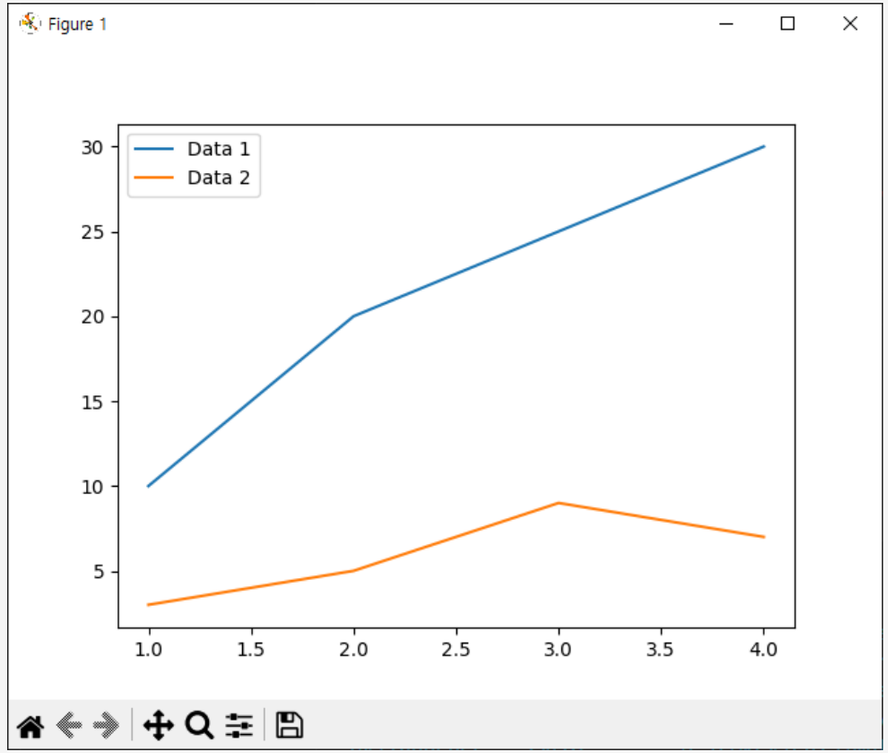


6. matplotlib 활용

❖ 그래프 커스터마이징

▪ 범례 추가 및 위치 설정

- x = [1, 2, 3, 4]
- y = [10, 20, 25, 30]
- x1 = [1, 2, 3, 4]
- y2 = [3, 5, 9, 7]
- plt.plot(x, y, label="Data 1")
- plt.plot(x1, y2, label="Data 2")
- plt.legend(loc="upper left")
- plt.show()



▪ 그래프 저장

- plt.savefig("ch19/matplotlib/my_plot.png")

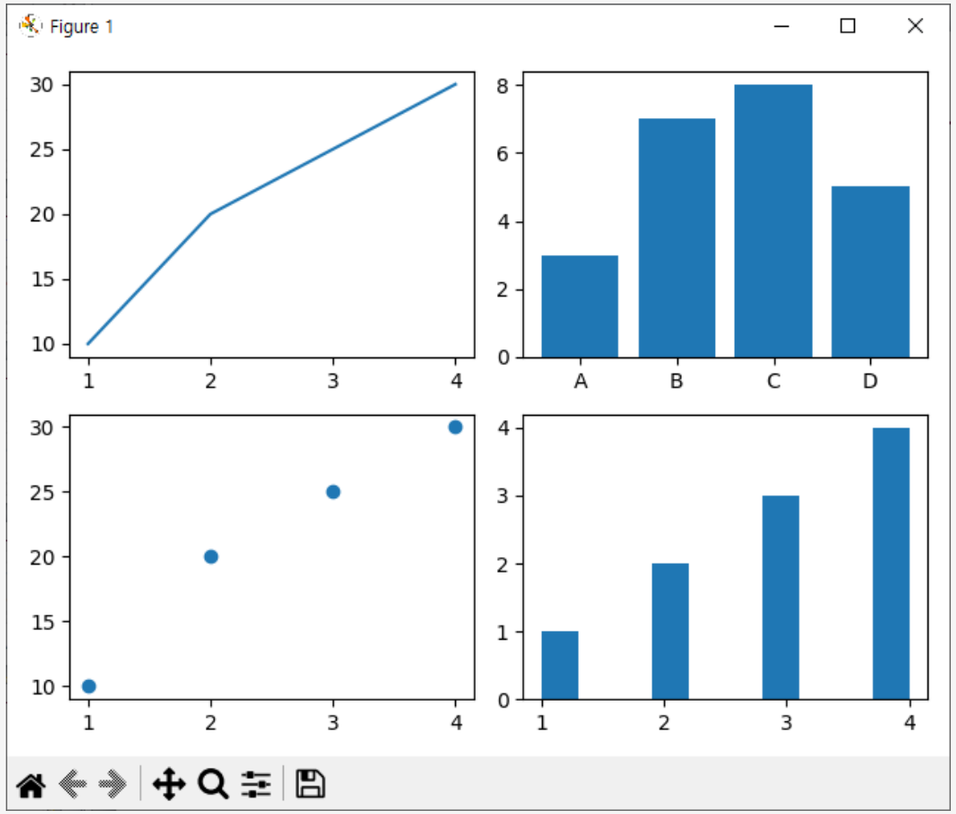
» 그래프 저장 시, plt.show() 를 호출하면 이미지가 해제되므로 호출 전 사용해야 함

6. matplotlib 활용

❖ 고급 기능

▪ Subplot 활용

- fig, axs = plt.subplots(2, 2)
- axs[0, 0].plot(x, y)
- axs[0, 1].bar(categories, values)
- axs[1, 0].scatter(x, y)
- axs[1, 1].hist(data)
- plt.tight_layout()
- plt.show()



00

Notice

❖ 과제

- 1. pandas 모듈의 기본 함수 사용하여 코드 작성하기
- 2. numpy 모듈의 기본 함수를 사용하여 코드 작성하기
- 3. matplotlib 모듈의 기본 함수를 사용하여 그래프 생성하기

❖ 다음 수업 내용

- 딥러닝 파이썬 패키지(2)
  - statsmodels, scikit-learn, scipy