

3 주차

Numpy

› Numpy 기본 개념

- Python에서의 사용하기 위한 라이브러리
- 벡터, 행렬을 위한 다차원 배열 구조로 표현하기 위한 객체 제공
- 수학, 과학 분야의 연산을 위한 Python 모듈
- Pandas, Matplotlib, TensorFlow, scikit-learn 등 데이터 분석 도구의 기반 라이브러리

Python 리스트 vs Numpy 배열

구분	Python	Numpy
데이터 타입	다른 데이터 타입 가질 수 있음	모두 같음
항목 개수 변경	변경 가능	변경 불가능
연산 속도	<	

함수

› 배열 생성 함수

- `np.arange(start, stop, step)` : start ~ (stop-1) step 간격으로 배열 생성
- `np.array(data , dtype=데이터타입)` : data 를 numpy 배열로 생성
- `np.zeros(shape, dtype=데이터타입)` : shape 크기 0 으로 채워진 배열 생성
- `np.ones(shape, dtype=데이터타입)` : shape 크기 1 로 채워진 배열 생성

› 통계량 출력 함수

- `np.min(numpy 배열)` : 최솟값
- `np.max(numpy 배열)` : 최댓값
- `np.sum(numpy 배열)` : 합계
- `np.mean(numpy 배열)` : 평균
- `np.var(numpy 배열)` : 분산
- `np.std(numpy 배열)` : 표준편차

› 난수 생성

- `np.random.rand(shape)` : 0 ~ 1 사이 균일한 분포의 난수 배열 생성
- `np.random.randint(start, end, size=shape)` : start ~ end 사이의 정수인 난수 배열 생성

› 인덱싱 / 슬라이싱

ex) `G = [1,2,3,4]` 에서 4 를 출력하고 싶다

→ `G[3]`

ex) `G = [1,2,3,4]` 에서 2, 3 을 출력하고 싶다

→ `G[1:3]`

- 배열의 인덱스 → `arr = [a(0), b(1), c(2), d(3)]` 으로 0 부터 시작

› 배열 변형

- `(배열이름).transpose()` / `(배열이름).T` : 배열의 행과 열 변경

4 주차 ~ 5 주차

Pandas

> 개념

- Python에서 데이터 분석, 조작을 위해 널리 쓰이는 라이브러리
- 결측치 처리, 데이터 변환, 데이터 결합 등 편리하게 수행
- Numpy 기반 라이브러리로 연산 속도 우수
- Matplotlib 과 함께 사용되어 DataFrame 시각적 표현 기능 제공
- 1 차원 구조 (Series), 2 차원 구조(DataFrame)

> Pandas 특징

1. 가변적 데이터 구조
2. 쉬운 데이터 처리(결측치 처리, 집계, 슬라이싱 등)

> Series 의 구조

- 복수의 행(row)으로 이루어진 하나의 열(column) 구조
- 인덱스 접근 가능

	②	①
0	7	
1	3	
2	5	
3	8	

① 칼럼(Column)
② 행 번호(index, 색인)

> DataFrame 구조

- ① 레코드 (Record) ② 칼럼(Column) ③ 행(Row) ④ 열 이름

	이름	나이	지역
0	Kim	20	서울
1	Lee	22	경기
2	Park	25	제주
3	Choi	19	강원
4	Song	23	인천

함수

〉 생성 함수

- `pd.Series(Python 배열, index = [열이름 1, 열이름 2, ...])`
: 1 차원 리스트를 활용하여 Series 생성, index 인자로 열 이름 지정 가능
- `pd.DataFrame([[값 11, 값 12, ...], [값 21, 값 22, ...], [값 31, 값 32, ...], ...])`
: 각 배열이 하나의 행을 이루는 DataFrame 생성, 딕셔너리 형태로 입력시 Key 값이 열 이름이 됨

〉 외부 데이터 가져오기

- `pd.read_excel('파일경로')` : .xlsx 확장자의 Excel 파일을 불러와 데이터 프레임 생성
- `pd.read_csv('파일경로', encoding='인코딩 타입')` : .csv 파일을 불러와 데이터 프레임 생성

〉 데이터 관련 함수

- `(DataFrame 이름).info()` : DataFrame 의 열, 열의 데이터 타입, 개수 등 기본정보 출력
- `(DataFrame 이름).describe()` : DataFrame 에서 숫자값을 갖는 열의 기본 통계량 출력
- `(DataFrame 이름).head(개수)/.tail(개수)` : DataFrame 상위 / 하위 개수 만큼 출력 (기본 5)
- `(DataFrame 이름)['열이름'].value_counts()` : 지정한 열에서 값 별 데이터 개수 출력

- `(DataFrame 이름)['열이름']` : 하나의 열
- `(DataFrame 이름)['열이름'][idx1:idx2]` : 하나의 열에서 idx1 행부터 idx2 행 까지
- `(DataFrame 이름)[idx1:idx2]` : idx1 행부터 idx2 행 까지
- `(DataFrame 이름)[['열이름 1', '열이름 2']]` : 여러 개의 열

- `(DataFrame 이름)['열이름'] (조건문)` : 열 내에서 조건문의 결과 (True/False 출력)
→ `ex_ df['math'] >= 80` : 80 이상인 점수 True / 80 미만 False

- `(DataFrame 이름)[(DataFrame 이름)['열이름'] 조건문]` : 조건에 맞는 행
- `(DataFrame 이름).query('열에 대한 조건문 문자열')` : 조건에 맞는 행 (위의 함수와 동일)

- `(DataFrame 이름)['열이름'].sum() / count() / mean() / std() / max() / min() / median()`
: 통계함수 / Numpy 에서의 통계량과 동일

- `(DataFrame 이름).groupby('열이름').(통계함수명)()` : 지정한 열을 기준 통계량 출력

〉 정렬

- `sorted(정렬 기준값)` : 정렬 기준값으로 정렬(ex_ `x.index / x.values`)
- `(DataFrame 이름).sort_values(by=['열이름'], ascending=True/False, inplace=True/False)`
: 지정한 열 기준으로 ascending 옵션에 따라 오름차순/내림차순 정렬 (기본은 오름차순)

〉 결측값 처리

- (DataFrame 이름).isna() : 열의 값이 있으면 True / 없으면 False
- (DataFrame 이름).isna().sum() : 열 별로 결측치 집계
- (DataFrame 이름).fillna(값, inplace=True/False) : 모든 열의 결측치를 값으로 채움
- (DataFrame 이름).fillna({'열이름' : '값'}, inplace=True/False)
: 지정열의 결측치 값으로 채움
- (DataFrame 이름).dropna(axis=0/1, subset=['열이름'], inplace=True/False)
 - ➔ axis = 1 : 결측치가 존재하는 열 삭제 / axis = 0 : 결측치가 존재하는 행 삭제
 - ➔ subset 설정 시, 지정한 열에서 결측치가 존재하는 경우 행 삭제

6 주차

Seaborn

› 개념

- Matplotlib 라이브러리 기반으로 통계 그래프 그리기 도구 제공 라이브러리
- Titanic, iris, tips, penguins 등 샘플 데이터셋 제공

함수

› 그래프 생성

- `seaborn.pairplot(DataFrame, hue='범주를 갖는 열이름')`: 상관관계 그래프 생성
- `seaborn.regplot(x='x 축 이름', y='y 축 이름', DataFrame)`: 추세선 그래프 생성
- `seaborn.scatterplot()`: 산점도 그래프 생성
- `seaborn.countplot()`: 범주별 데이터 개수 막대 그래프 생성
- `seaborn.barplot()`: 막대 그래프 생성

- `show()`: 생성한 그래프 출력

Titanic

› 함수

- `.describe()`: 숫자형 자료의 기술 통계량 출력 (Pandas 에서의 describe 와 유사)

› 구분에 따른 생존율

- DataFrame 의 이름이 d 라고 가정

ex) 성별에 따른 생존자 집계

→ `d.groupby('sex')['survived'].count()`

ex) 성별에 따른 생존율 연산

→ `d.groupby('sex')['survived'].mean()`

ex) 나이에 따른 생존율

→ `d.groupby('age')['survived'].mean()`

→ 같은 출력을 보이는 다른 코드가 존재함 유의(강의 자료 참고)

› 결측치 처리

- `d.loc[조건문, 값]`: 조건문이 참이 되는 데이터를 값으로 채움
- `d['열이름'].isnull()`: 지정한 열 내에서 결측치가 존재할 경우 True
- 둘 조합 해서 결측값 처리에 사용
- ex) `d.loc[d['age'].isnull() & (d['who']=='child'), 'age'] = 6`
: 결측치가 있고, 어린아이면 결측값을 6 으로 채움

11 주차

WordCloud

› 라이브러리 불러오기

```
pip install wikipedia # wikipedia 라이브러리 설치
```

```
import wikipedia : 라이브러리 import
```

› STOPWORD(중지어) 설정

```
ex)
from wordcloud import WordCloud, STOPWORDS

s_words = STOPWORDS.union( {'중지어 1', '중지어 2', '중지어 3', '중지어 4', ... } )
wordCloud = WordCloud(width=2000, height=1500, stopwords = s_words).generate(text)
```

13 주차

Web Crawling (웹 크롤링)

> 관련 라이브러리

- requests : HTTP 요청(GET 등) 처리 라이브러리
- BeautifulSoup4 : HTML, XML으로 된 웹 페이지 파싱(Parsing) 하는 파서(Parser) 라이브러리

> BeautifulSoup4 사용법

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(webpage.content, "html.parser")

# 타이틀 가져오기
print(soup.head.title)
# p 태그 출력
print(soup.p)
# p 태그에서 텍스트만 출력
print(soup.p.string)
```

- soup.(태그명) : 태그를 여러 개 이어 붙여서 원하는 부분을 가져올 수 있음

Web Scraping (웹 스크래핑)

> 결측치 처리

- (Table 명).drop('행/열 이름',axis=0/1) : 행/열을 지정하여 삭제 (axis 가 1 이면 열, 0 이면 행)
- (Table 명).dropna(subset=['열이름']) : 지정한 열에서 값이 NaN(결측치)인 행 삭제

> 데이터 저장

- (Table 명).to_csv('파일경로', encoding='인코딩 타입', index=True/False) : csv 파일 저장
- (Table 명).to_excel('파일경로', encoding='인코딩 타입', index=True/False) : xlsx 파일 저장

14 주차

folium 지도 시각화 라이브러리

› 개념

- Python 에서 제공해주는 지도를 다루는 라이브러리
- 지도위에 점, 선, 원 등 원하는 정보 시각적으로 나타내는 기능 존재
- 다른 패키지에 비교해 안정적
- Pandas 와 쉽게 연동 가능

› 주요 기능

- 지도 생성
- 마커 추가
- 원, 선, 다각형 추가
- 레이어 및 컨트롤 추가
- 인터랙티브
- HTML 파일로 내보내기

› 사용법

```
import folium

mymap = folium.Map(location=[37.5, 127], zoom_start=17)
mymap
```

- folium.Map() 으로 지도 생성
- location 옵션으로 [위도, 경도] 지정
- zoom_start 옵션으로 초기 지도 확대/축소

› 함수

- folium.Marker([위도, 경도], popup='문구').add_to(지도 이름)
: 지도 위에서 위도, 경도 위치에 문구가 표시되는 마커 생성

› Pandas 를 활용한 결측치 제거

- Pandas 와 동일
- df.isna().sum() / df.dropna() 등