# 외모 데이터 취!

WEEK3 Pandas 기초

# Pandas

외모 데이터 쳌!

- Pandas 모듈
  - Pandas는 데이터 처리를 위한 기능을 제공하는 Python 모듈이에요.
  - 처리할 데이터를 조작하는 데이터프레임 기능부터, 수학 처리 모듈 Numpy와 연계한 통계 처리 기능까지 제공해요.

```
# pandas 설치 (최초 1회만 하면 됨)
!pip install pandas
```

```
# S03-01
import pandas as pd
```

# 데이터프레임

외모 데이터 췍!

- 데이터프레임
  - 표 형태를 가진
    데이터 처리를 위한 자료형이에요.
  - 각 열은 이름을 가지며,
    특정 열에는 하나의 자료형
    (수, 문자 등)만 담길 수 있어요.

| | name [문자] | class [문자] | math_sc [정수] |
|---|---|---|---|
| 0 | 'winter' | 'senior' | 90 |
| 1 | 'isa' | 'senior' | 95 |
| 2 | 'liz' | 'junior' | 85 |
| 3 | 'lily' | 'senior' | 70 |
| 4 | 'daniel' | 'junior' | 75 |

※ 행이름은 별도 지정이 없으면
인덱스 값으로 자동 설정돼요.

# 데이터프레임

외모 데이터 췍!

- 데이터프레임 만들기
  - pd.DataFrame을 통해 데이터프레임을 만들 수 있어요.
  - 이미 만들어진 데이터프레임을 새변수 = 기존변수.copy()로 다른 변수에 복제할 수 있어요.

```
# S03-02
# 데이터프레임 만들기
s_data = {'name':['winter', 'isa', 'liz',
                  'lily', 'daniel'],
          'class':['senior', 'senior', 'junior',
                   'senior', 'junior'],
          'math_sc':[90, 95, 85, 70, 75]}
stu = pd.DataFrame(data = s_data)
print(stu)
```

```
     name   class  math_sc
0  winter  senior       90
1     isa  senior       95
2     liz  junior       85
3    lily  senior       70
4  daniel  junior       75
```

# 데이터프레임

외모 데이터 췍!

- 행/열 정보
  - DF.columns 열이름
  - DF.index 행이름

데이터프레임이 저장된 변수를 DF라 표현할게요.

columns와 index로 이름을 확인하거나 설정해요.

  - DF.shape는 (행개수, 열개수)를 보여줘요.

```
# S03-03
# 행 이름
print(stu.columns)

# 열 이름
print(stu.index)

# 행/열 개수
print(stu.shape)
```

```
Index(['name', 'class', 'math_sc'], dtype='object')
RangeIndex(start=0, stop=5, step=1)
(5, 3)
```

# 데이터프레임

외모 데이터 췍!

## 췍췍췍! EXERCISE 03-01

아래와 같은 데이터프레임 cafe를 만든 후, cafe를 출력해봅시다.

| menu [문자] | type [문자] | price [정수] |
|---|---|---|
| 'Americano' | 'coffee' | 2100 |
| 'Cafe Latte' | 'coffee' | 2800 |
| 'Iced Tea' | 'non coffee' | 2300 |
| 'Mint Latte' | 'non coffee' | 3000 |

## 췍췍췍! EXERCISE 03-02

앞서 만든 cafe에 대해 다음 정보를 출력하는 코드를 작성해봅시다.

- 행의 개수
- 두번째 열의 이름

# 행/열/셀 선택

외모 데이터 쳌!

- 열 선택
  - DF['열이름']
    DF.열이름
    으로 특정 열을 선택할 수 있어요.
  - DF[['열이름', '열이름', …]]
    으로 여러 열을 선택할 수 있어요.

```
# S03-04
# name 열 선택
print(stu['name'])
print()

# math_sc 열 선택
print(stu.math_sc)
print()

# name, math_sc 동시 선택
print(stu[['name', 'math_sc']])
```

```
0    winter
1       isa
2       liz
3      lily
4    daniel
Name: name, dtype: object
```

```
0    90
1    95
2    85
3    70
4    75
Name: math_sc, dtype: int64
```

```
      name  math_sc
0   winter       90
1      isa       95
2      liz       85
3     lily       70
4   daniel       75
```

# 행/열/셀 선택

외모 데이터 쳌!

- 행/셀 선택
  - DF.iloc[행 번호, 열 번호]

    행 번호, 열 번호에 해당하는 셀 선택
    (0부터 시작)
    (:을 이용한 슬라이싱도 가능하지만, 끝 번호 포함)
  - DF.iloc[행 번호]

    행 번호에 해당하는 모든 열 선택
    (0부터 시작)
    (:을 이용한 슬라이싱도 가능하지만, 끝 번호 포함)
  - 선택하여 출력하거나,
    선택하여 수정할 수 있어요.

```
# S03-05
# 첫번째 행 두번째 열 선택
print(stu.iloc[0, 1])
print()

# 세번째 행부터 마지막 행까지 전체 출력
print(stu.iloc[2:])
print()

# 네번째 행 세번째 열 수정
stu.iloc[3, 2] = 100
# 수정 확인
print(stu)
```

```
senior

     name   class  math_sc
2     liz  junior       85
3    lily  senior      100
4  daniel  junior       75
```

```
     name   class  math_sc
0  winter  senior       90
1     isa  senior       95
2     liz  junior       85
3    lily  senior      100
4  daniel  junior       75
```

# 행/열/셀 선택

외모 데이터 쉑!

- 행/셀 선택
  - DF.loc[행이름, 열이름]

    행이름, 열이름에 해당하는 셀 선택
    (각 이름 자리에 리스트로 여러 행 또는 열 지정 가능)
  - DF.loc[행이름]

    행이름에 해당하는 모든 열 선택
    (각 이름 자리에 리스트로 여러 행 지정 가능)
  - 선택하여 출력하거나,
    선택하여 수정할 수 있어요.

```
# S03-06
# 행이름 0, 열이름 name 선택
print(stu.loc[0, 'name'])
print()

# 행이름 2의 name 열과 math_sc 열 출력
print(stu.loc[2, ['name', 'math_sc']])
print()

# 행이름 3의 math_sc 열 수정
stu.loc[3, 'math_sc'] = 70
# 수정 확인
print(stu)
```

```
winter

name       liz
math_sc     85
Name: 2, dtype: object
```

| | name | class | math_sc |
|---|---|---|---|
| 0 | winter | senior | 90 |
| 1 | isa | senior | 95 |
| 2 | liz | junior | 85 |
| 3 | lily | senior | 70 |
| 4 | daniel | junior | 75 |

# 행/열/셀 선택

외모 데이터 쳌!

- 열 새로 추가하기
  - DF['열이름'] = ~~
    에서 기존에 없던 열이름이면,
    해당 열이름으로 된
    새로운 열이 추가돼요.

```
# S03-07
# eng_sc 열 추가
stu['eng_sc'] = [80, 90, 90, 95, 80]

# 추가 확인
print(stu)
```

```
     name   class  math_sc  eng_sc
0  winter  senior       90      80
1     isa  senior       95      90
2     liz  junior       85      90
3    lily  senior       70      95
4  daniel  junior       75      80
```

# 행/열/셀 선택

외모 데이터 쉑!

## 쉑쉑쉑! EXERCISE 03-03

앞서 만든 cafe에 대해 다음 정보를 출력하는 코드를 작성해봅시다.

- menu 열 전체
- 세번째 행 전체
- 두번째 행의 price 열

## 쉑쉑쉑! EXERCISE 03-04

앞서 만든 cafe에 대해 다음을 수행하는 코드를 작성해봅시다.

- 세번째 행의 가격을 변수 price3에 저장한다.
- price3를 활용하여,
  세번째 행의 가격이 기본보다 200원
  높아지도록 cafe 데이터프레임 수정한다.
- cafe를 출력하여 잘 수정됐는지 확인한다.

# 기초 연산

외모 데이터 쳌!

- 산술 연산
  - 데이터프레임의 열과 열 또는 열과 값을 대상으로 +, -, *, /, //, % 등 산술 연산을 수행하면, 열 내 각 값을 대상으로 연산이 수행돼요.

```
DF['열이름'] + 값
DF['열이름'] + DF['열이름']
```

```
# S03-08
# math_sc를 2로 나누기
print(stu['math_sc'] / 2)
print()

# 수학 점수와 영어 점수의 평균 열 만들기
stu['avg_sc'] = (stu['math_sc'] + stu['eng_sc']) / 2

# 수정 확인
print(stu)
```

```
0    45.0
1    47.5
2    42.5
3    35.0
4    37.5
Name: math_sc, dtype: float64
```

| | name | class | math_sc | eng_sc | avg_sc |
|---|---|---|---|---|---|
| 0 | winter | senior | 90 | 80 | 85.0 |
| 1 | isa | senior | 95 | 90 | 92.5 |
| 2 | liz | junior | 85 | 90 | 87.5 |
| 3 | lily | senior | 70 | 95 | 82.5 |
| 4 | daniel | junior | 75 | 80 | 77.5 |

# 기초 연산

외모 데이터 쳌!

- 비교 연산
  - ==, !=, <, <=, >, >= 등 비교 연산을 수행하면, 열 내 각 값을 대상으로 연산이 수행돼요.

```
DF['열이름'] == 값
DF['열이름'] == DF['열이름']
```

```
# S03-09
# avg_sc가 85 이상인지 여부
print(stu['avg_sc'] >= 85)
print()

# 수학 점수가 영어 점수보다 높은지 여부를 담은
# mt_ov_eg 열 만들기
stu['mt_ov_eg'] = stu['math_sc'] > stu['eng_sc']

# 수정 확인
print(stu)
```

```
0     True
1     True
2     True
3    False
4    False
Name: avg_sc, dtype: bool
```

| | name | class | math_sc | eng_sc | avg_sc | mt_ov_eg |
|---|---|---|---|---|---|---|
| 0 | winter | senior | 90 | 80 | 85.0 | True |
| 1 | isa | senior | 95 | 90 | 92.5 | True |
| 2 | liz | junior | 85 | 90 | 87.5 | False |
| 3 | lily | senior | 70 | 95 | 82.5 | False |
| 4 | daniel | junior | 75 | 80 | 77.5 | False |

# 기초 연산

외모 데이터 쉑!

- 논리 연산
  - 앞선 연산과 유사하게
    논리 연산도 수행할 수 있지만,
    and 대신 &를,
    or 대신 |를, not 대신 ~를
    사용해야 해요.

조건 & 조건

```
# S03-10
# 수학과 영어 점수가 모두 85 이상인지 여부
(stu['math_sc'] >= 85) & (stu['eng_sc'] >= 85)
```

```
0     True
1     True
2     True
3    False
4    False
dtype: bool
```

※ ipynb 파일은 print가 없더라도 마지막 줄이 출력돼요!

# 조건 선택

외모 데이터 쳌!

- 행 조건 선택
  - 각종 연산을 활용하여,
    DF[ 연산 ]을 통해
    연산 결과가 True인 행만
    선택할 수 있어요.

```
# S03-11
# 점수 평균이 85 이상인 행만 선택
stu[ stu['avg_sc'] >= 85 ]
```

| | name | class | math_sc | eng_sc | avg_sc | mt_ov_eg |
|---|---|---|---|---|---|---|
| 0 | winter | senior | 90 | 80 | 85.0 | True |
| 1 | isa | senior | 95 | 90 | 92.5 | True |
| 2 | liz | junior | 85 | 90 | 87.5 | False |

※ 주피터 노트북에서 마지막 줄에 데이터프레임을 둘 경우,
위와 같이 표 형태로 가공해서 보여줘요!

```
# S03-12
# 점수 평균이 85 이상인 학생 수 구하기
stu[ stu['avg_sc'] >= 85 ].shape[0]
```

```
3
```

# 조건 선택

외모 데이터 췍!

## 췍췍췍! EXERCISE 03-05

앞서 만든 cafe에 대해 다음을 수행하는 코드를 작성해봅시다.

- 판매량을 뜻하는 volume 열을 만든다. (volume: [300, 200, 250, 180])
- 각 메뉴의 매출을 뜻하는 revenue 열을 만든다. (revenue = price × volume)
- cafe를 출력하여 잘 수정됐는지 확인한다.

## 췍췍췍! EXERCISE 03-06

앞서 만든 cafe에 대해 다음을 수행하는 코드를 작성해봅시다.

- 전체 메뉴의 매출 합을 구해 출력합니다.
- 매출이 600,000 이상인 메뉴 행만 골라 출력합니다.
- 매출이 600,000 이상인 메뉴의 개수를 구해 출력합니다.

# 데이터 불러오기/내보내기

외모 데이터 쉑!

- CSV 파일
  - CSV(comma separated value)란 표 형태의 데이터를 저장하는 파일 확장자로, 엑셀로 열어 쉽게 확인할 수 있으면서도 용량이 작아 흔히 사용돼요.

※ 엑셀에서 '다른 이름으로 저장'으로 CSV 파일을 만들 수 있어요.
한글을 사용했다면 'CSV UTF-8'을 선택하여 저장해야 해요.

※ 메모장으로 연 csv 파일

```
날짜,영화관,스크린수,상영횟수
2024-08-07,CGV,495,1135
2024-08-07,롯데시네마,318,899
2024-08-07,메가박스,182,517
2024-08-08,CGV,459,1130
2024-08-08,롯데시네마,318,863
2024-08-08,메가박스,184,500
```

※ 엑셀로 연 csv 파일

| | A | B | C | D |
|---|---|---|---|---|
| 1 | 날짜 | 영화관 | 스크린수 | 상영횟수 |
| 2 | 2024-08-07 | CGV | 495 | 1135 |
| 3 | 2024-08-07 | 롯데시네마 | 318 | 899 |
| 4 | 2024-08-07 | 메가박스 | 182 | 517 |
| 5 | 2024-08-08 | CGV | 459 | 1130 |
| 6 | 2024-08-08 | 롯데시네마 | 318 | 863 |
| 7 | 2024-08-08 | 메가박스 | 184 | 500 |

# 데이터 불러오기/내보내기

외모 데이터 쳌!

- CSV 파일 불러오기
  - pd.read_csv('파일명.csv')로 CSV 파일을 읽어 데이터프레임을 만들 수 있어요.
  - 한글을 사용한 데이터는 encoding 설정을 해줘야 해요.

※ 데이터를 만들 때 어떤 방식으로 만들었는지에 따라 다르지만, 보통 encoding='utf-8'로 하면 돼요.


※ 사용할 데이터

영화 '사랑의 하츄핑'의 개봉 후 14일 간
3대 영화관에서 일자별로 상영된 스크린수, 상영횟수

※ 코드를 작성하고 있는 ipynb 파일과 같은 디렉토리(폴더)에 해당 csv 파일을 저장해주세요.

```
# S03-13
ping = pd.read_csv('사랑의하츄핑_영화관별.csv',
                   encoding = 'utf-8')
ping.head()  # DF.head()는 첫 5행
```

| | 날짜 | 영화관 | 스크린수 | 상영횟수 |
|---|---|---|---|---|
| 0 | 2024-08-07 | CGV | 495 | 1135 |
| 1 | 2024-08-07 | 롯데시네마 | 318 | 899 |
| 2 | 2024-08-07 | 메가박스 | 182 | 517 |
| 3 | 2024-08-08 | CGV | 459 | 1130 |
| 4 | 2024-08-08 | 롯데시네마 | 318 | 863 |

# 데이터 불러오기/내보내기

외모 데이터 쳌!

- CSV 파일 내보내기
  - DF.to_csv('파일명.csv')로 데이터프레임을 CSV 파일로 내보낼 수 있어요.
  - 한글을 사용한 데이터는 encoding 설정을 해줘야 해요.

※ 보통 encoding='utf-8-sig'로 하면 돼요.
('utf-8'로 하면 엑셀에서 글자가 깨질 수 있어요.)

```
# S03-14
# 첫 5행을 CSV 파일로 내보내기
ping_f5 = ping.head()

# index = False를 해주어야 행이름 없이 저장돼요.
ping_f5.to_csv('사랑의하츄핑_5행.csv',
               index = False,
               encoding = 'utf-8-sig')
```

# 다양한 활용

외모 데이터 쳌!

- np.where()
  - 데이터를 처리하다 보면 조건에 따라 값을 처리해야 하는 경우가 많아요.
  - 이럴 때 Numpy 모듈의 np.where()을 사용할 수 있어요.

```
np.where(조건, 참일 때 값, 거짓일 때 값)
```

```
!pip install numpy
```

```
# S03-15
# 스크린 당 상영횟수 구하기
ping['평균상영'] = ping['상영횟수'] / ping['스크린수']

# numpy 불러오기
import numpy as np

# 평균상영이 3 이상인지 확인하는 열 만들기
ping['수준'] = np.where(ping['평균상영'] >= 2.5,
                      '좋음', '아쉽')

# 확인
ping.head()
```

# 다양한 활용

외모 데이터 쉑!

- itertuples(), iterrows()
  - 각 행 별로 연산이 필요하다면, itertuples()각 행을 튜플로 반복와 iterrows()각 행을 순번과 튜플로 반복를 for와 함께 사용할 수 있어요.

```
for row in DF.itertuples():
    ~~
```

```
for idx, row in DF.iterrows():
    ~~
```

```
# S03-16
# "일자_영화관_수준"을 담은 요약 열 만들기
ping['요약'] = ''

for idx, row in ping.iterrows():
    string = ''
    string += row['날짜'][-2:] + '_'
    string += row['영화관'] + '_'
    string += row['수준']
    ping.loc[idx, '요약'] = string

# 확인
ping.head()
```

| | 날짜 | 영화관 | 스크린수 | 상영횟수 | 평균상영 | 수준 | 요약 |
|---|---|---|---|---|---|---|---|
| 0 | 2024-08-07 | CGV | 495 | 1135 | 2.292929 | 아쉽 | 07_CGV_아쉽 |
| 1 | 2024-08-07 | 롯데시네마 | 318 | 899 | 2.827044 | 좋음 | 07_롯데시네마_좋음 |
| 2 | 2024-08-07 | 메가박스 | 182 | 517 | 2.840659 | 좋음 | 07_메가박스_좋음 |
| 3 | 2024-08-08 | CGV | 459 | 1130 | 2.461874 | 아쉽 | 08_CGV_아쉽 |
| 4 | 2024-08-08 | 롯데시네마 | 318 | 863 | 2.713836 | 좋음 | 08_롯데시네마_좋음 |

- '사랑의 하츄핑' 영화관 상영 분석

① '사랑의하츄핑_영화관별.csv'를 새로이 불러와 데이터프레임 heart에 저장한다.

② heart의 행 개수를 구해 출력한다.

③ '영화관' 열의 첫 3개 행을 출력한다.

④ 행 별로 스크린 당 상영횟수를 나타내는 '평균상영' 열을 만든다.

⑤ '평균상영' 값이 2.5 이상인 행의 개수를 구해 출력한다.

⑥ CGV와 롯데시네마 각각의 상영횟수 합계를 구해 출력한다.

⑦ CGV의 상영횟수가 롯데시네마의 상영횟수의 몇 배인지 구해 출력한다.

※ 예시 결과

```
(2) 42

(3)
0       CGV
1    롯데시네마
2     메가박스
Name: 영화관, dtype: object

(5) 18

(6) CGV 12913, 롯데시네마 9071

(7) 1.4235475691764965
```

# 외모 데이터 쒝!

WEEK3 Pandas 기초



## Further Topics

Numpy 모듈

전처리(결측치, 이상치, 파생변수)

## References

영화진흥위원회 통합전산망 (http://www.kobis.or.kr)

# 췍췍췍! LEARN MORE : 행 추가하기

외모 데이터 췍!

- 행 추가
  - 데이터프레임에 행을 추가하려면, 같은 열 구조를 가진 별도의 데이터프레임에 추가 행을 두고, 두 데이터프레임을 합치는 방법으로 해야 돼요.
- pd.concat()
  - 두 데이터프레임의 행을 이어줘요.

```
DF1 = pd.concat([DF1, DF2])
```

```
# S03-17
# 새로운 행을 담은 별도의 데이터프레임
new_row = pd.DataFrame({
  'name' : ['karina'],
  'class' : ['senior'],
  'math_sc' : [80],
  'eng_sc' : [100],
  'avg_sc' : [90],
  'mt_ov_eg' : [False]
})

# 데이터프레임 합치기
stu = pd.concat([stu, new_row])

# 추가 확인
stu
```

# 쳌쳌쳌! LEARN MORE : 중복 제외

외모 데이터 쳌!

- drop_duplicates()
  - DF의 행에 중복행이 있을 때, 하나씩만 남겨줘요.

```
DF.drop_duplicates()
```

```
# S03-18
# class 열과 mt_ov_eg의 중복 제외 분류
stu[['class', 'mt_ov_eg']].drop_duplicates()
```

| | class | mt_ov_eg |
|---|---|---|
| 0 | senior | True |
| 2 | junior | False |
| 3 | senior | False |

# 쵁쵁쵁! LEARN MORE : 테이블 합치기

외모 데이터 쵁!

- merge()
  - 두 데이터프레임을 특정 열을 기준으로 병합할 때 merge()을 사용할 수 있어요.

```
# DF1과 DF2를 '열이름' 열을 기준으로 병합
DF1.merge(DF2, on = '열이름')
```

※ 기본적으로 DF1과 DF2에 둘 다 포함된 기준 열 값만 다루는 inner join이 수행돼요.
만약 DF1의 기준열 값에 해당하는 DF2의 기준열 값이 없더라도 해당 행을 남기려면
how = 'left'라는 조건을 merge() 안에 추가해주세요.

```
# S03-19
# 출생연도
birth = pd.DataFrame({
  'name' : ['karina', 'winter'],
  'birth' : [2000, 2001]
})

# 합치기
stu = stu.merge(birth, on = 'name')
```

# 쵝쵝쵝! LEARN MORE : 데이터 전처리

외모 데이터 쵝!

- 전처리
  - 분석에 들어가기에 앞서, 적절한 분석이 되기 위해 전처리를 수행해요.
  - 전처리 유형에는 대표적으로
    결측치 처리, 이상치 처리, 파생변수 생성이 있어요.
  - 결측치 처리: 데이터의 특정 부분을 기술적으로 수집하지 못할 때
    이를 대체하는 방안 (단순대치법, 평균대치법, ...)
  - 이상치 처리: 어떠한 이유에서 너무 일반 추이에서 벗어난 값을
    조치하는 방안 (IQR 기반 이상치 처리, ESD 기반 이상치 처리, Winsorize, ...)
  - 파생변수 생성: 기존 열 하나를 변형하거나 둘 이상의 열을 조합하여
    추가 변수를 생성하는 방안 (단위 변환, 변수 간 연산, 증가율, 원핫인코딩, 스케일링, ...)
  - 관련 내용을 미리 알아보고 오면 프로젝트 때 도움이 될 거에요.
    [별도 자료 제공 예정]