[3-1. Pandas 라이브러리 소개 및 설치/환경설정]

1. Pandas 라이브러리 소개

1-1. 개발 배경과 목적

'pandas'는 Python Data Analysis Library의 약자로, Wes McKinney에 의해 2008년에 개발이 시작되었습니다.

Wes Mckinney는 금융 데이터 분석을 위해 작업 중 효율적인 데이터 분석과 조작을 위한 도구의 필요성을 느꼈고, 이를 해결하기 위해 'pandas'를 만들었습니다.

'pandas'의 주요 목적은 빠르고, 효율적이며, 사용하기 쉬운 데이터 구조를 제공하여 Python에서 실제 데이터 분석 작업을 용이하게 하는 것입니다.

1-2. 주요 기능

- 데이터 조작 : 데이터 정제(clean), 변형(transform), 집계(aggregate) 등 다양한 데이터 조작 기능을 제공합니다.
- 데이터 분석 : 다양한 통계적 분석 기능을 내장하고 있어 데이터 분석을 용이하게 합니다.
- 데이터 입출력 : CSV, Excel, SQL, 데이터베이스 등 다양한 형식의 파일을 읽고 쓸 수 있는 기능을 제공합니다.

1-3. 사용자층

'pandas'는 데이터 과학, 금융 분석, 경제학, 통계학, 엔지니어링 등 다양한 분야에서 데이터를 분석하고 조작해야 하는 사람들이 널리 사용합니다. 특히 데이터 과학자, 데이터 분석가, 소프트웨어 엔지니어, 학계 연구원 등 데이터와 밀접하게 일하는 다양한 전문가들이 'pandas'를 사용합니다.

1-4. 참고 자료

- pandas 홈페이지 : https://pandas.pydata.org
- 대표적인 참고서적 : 파이썬 라이브러리를 활용한 데이터 분석(Python for Data Analysis), 저자 : Wes Mckinney

2. Pandas의 자료구조

'pandas'는 두 가지 핵심 자료구조를 사용합니다.

'Series'와 **'DataFrame'** 입니다. 이 두가지 자료형은 데이터 조작과 분석을 위해 다양한 기능을 제공하며, 'pandas' 라이브러리의 기본을 이룹니다.

2-1. Series

'Series'는 1차원 배열과 유사한 구조를 가지며, 단일 데이터 타입의 값을 갖습니다. 각 값에는 고유한 인덱스가 할당되어 있어, 인덱스를 통해 개별 데이터에 접근할 수 있습니다. 'Series' 객체는 다음과 같이 생성할 수 있습니다.

In [ ]:
```
import pandas as pd

data = [1, 2, 3, 4, 5]
series = pd.Series(data)

print(series)
```

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

'Series'는 기본적으로 정수 인덱스를 사용하며, 문자열 등 다른 타입의 인덱스를 명시적으로 설정할 수도 있습니다.

2-2. DataFrame

'DataFrame'은 2차원 레이블이 붙은 데이터 구조로, SQL 테이블이나 엑셀 스프레드시트와 비슷한 형태입니다. 서로 다른 데이터 타입의 열을 갖을 수 있으며, 각 열은 'Series'객체입니다.

'DataFrame'은 다음과 같이 생성할 수 있습니다.

In [ ]:
```
data = {
    'Name': ['John', 'Anna', 'Peter', 'Linda'],
    'Age': [28, 34, 29, 32],
    'City': ['New York', 'Paris', 'Berlin', 'London']
}
df = pd.DataFrame(data)

print(df)
```

```
    Name  Age      City
0   John   28  New York
1   Anna   34     Paris
2  Peter   29    Berlin
3  Linda   32    London
```

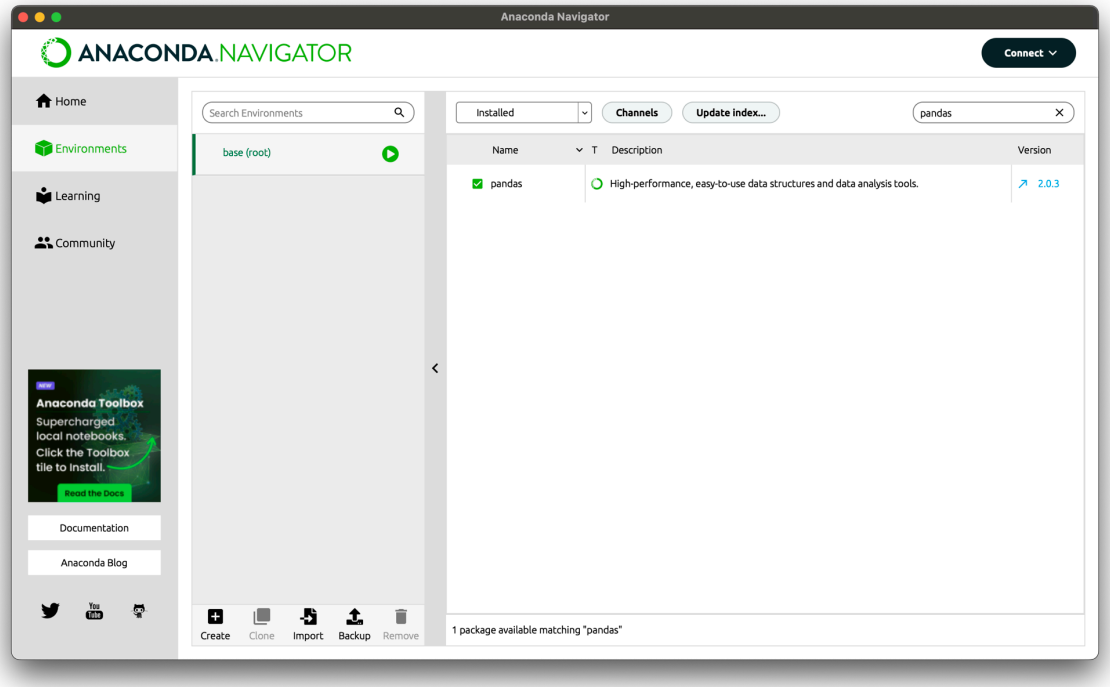
여기서 'data'는 딕셔너리로, 키는 열의 이름이 되고 값은 각 열의 데이터가 됩니다. 'DataFrame'에서도 인덱스를 사용해 행에 접근할 수 있으며, 인덱스는 기본적으로 정수 인덱스가 사용되지만, 명시적으로 설정할 수도 있습니다.

3. pandas의 설치 및 임포트

3-1. pandas 설치

3-1-1. 아나콘다를 이용한 설치

'pandas'는 아나콘다에서 기본 제공되는 라이브러리로 이미 설치되어 있습니다. 필요하다면, 아나콘다 Environments 탭에서 체크하는 방식으로 설치 및 업데이트 가능합니다.



3-1-2. Python의 패키지 관리자인 'pip'를 사용하여 설치할 수 있습니다.

윈도우즈 명령 프롬프트 또는 파워쉘에서나 맥OS 터미널에서 아래와 같이 명령어를 실행합니다.

```
pip install pandas
```

라이브러리를 최신 버전으로 업그레이드 하려면 아래와 같이 명령어를 실행합니다.

```
pip install --upgrade pandas
```

3-2. pandas 임포트하기

'pandas' 라이브러리를 Python 코드 내에서 사용하기 위해, 먼저 'import' 문을 사용하여 'pandas'를 임포트해야 합니다. 일반적으로 'pandas'는 'pd'라는 약칭으로 임포트 합니다. 이는 표준 관례이며, 이렇게 함으로써 코드를 좀 더 간결하게 작성할 수 있습니다. 'pandas' 라이브러리에서 'Series'와 'DataFrame' 자료형은 사용 빈도가 높으므로, 편의를 목적으로 별도로 임포트 하기도 합니다.

In [ ]:
```
import pandas as pd
from pandas import Series, DataFrame
```

In [ ]:
```
data = {
    'Name': ['John', 'Anna', 'Peter', 'Linda'],
    'Age': [28, 34, 29, 32],
    'City': ['New York', 'Paris', 'Berlin', 'London']
}
df1 = pd.DataFrame(data)
df2 = DataFrame(data)
```

In [ ]:
```
print(df1)
```

```
    Name  Age      City
0   John   28  New York
1   Anna   34     Paris
2  Peter   29    Berlin
3  Linda   32    London
```

In [ ]:
```
print(df2)
```

```
    Name  Age      City
0   John   28  New York
1   Anna   34     Paris
2  Peter   29    Berlin
3  Linda   32    London
```

3-3. pandas display 설정

'pandas'의 'set_option' 메서드를 사용하면 데이터프레임을 화면에 출력할 때 여러 가지 옵션을 설정할 수 있습니다. 이를 통해 큰 데이터셋을 다룰 때 출력 형태를 사용자의 필요에 맞게 조정할 수 있습니다.

- display.max_rows : 출력할 최대 행의 수를 설정합니다. 최대 행의 수를 넘어서 데이터가 있는 경우 데이터를 축약해서 보여줍니다.
- display.max_columns : 출력할 최대 열의 수를 설정합니다. 최대 열의 수를 넘어서 열이 있는 경우 열을 축약해서 보여줍니다.
- display.max_colwidth : 한 열 내에 표시할 수 있는 최대 문자의 수를 설정합니다.
- display.width : 콘솔에서 데이터프레임을 출력할 때의 전체 페이지 너비를 설정합니다.

In [ ]:
```
data = pd.read_csv('data/amazon.csv')
```

In [ ]:
```
pd.set_option('display.max_rows', 100)     # 1000, 2000
pd.set_option('display.max_columns', 10)   # 100
pd.set_option('display.max_colwidth', 10) # 100
pd.set_option('display.width', 100)        # 300
```

In [ ]:
```
data
```

Out[ ]:

	product_id	product_name	category	discounted_price	actual_price	...	review_
0	B07JW9...	Wayona...	Comput...	399	1,099	...	R3HXW
1	B098NS...	Ambran...	Comput...	199	349	...	RGIQEC
2	B096MS...	Sounce...	Comput...	199	1,899	...	R3J3EC
3	B08HDJ...	boAt D...	Comput...	329	699	...	R3EEUZ
4	B08CF3...	Portro...	Comput...	154	399	...	R1BP4L
...	...	...	...	...	...	...	
1460	B08L7J...	Noir A...	Home&K...	379	919	...	R3G3XI
1461	B01M64...	Presti...	Home&K...	2,280	3,045	...	R3DDL2
1462	B009P2...	Bajaj ...	Home&K...	2,219	3,080	...	R1TLR.
1463	B00J5D...	Havell...	Home&K...	1,399	1,890	...	R39Q2Y
1464	B01486...	Borosi...	Home&K...	2,863	3,690	...	R20RBF

1465 rows × 16 columns

In [ ]:
```
print(data)
```

```
     product_id product_name   category discounted_price actual_price
...  review_id  \
0     B07JW9...  Wayona...    Comput...        399            1,099
...  R3HXWT...
1     B098NS...  Ambran...    Comput...        199              349
...  RGIQEG...
2     B096MS...  Sounce...    Comput...        199            1,899
...  R3J3EQ...
3     B08HDJ...  boAt D...    Comput...        329              699
...  R3EEUZ...
4     B08CF3...  Portro...    Comput...        154              399
...  R1BP4L...
...         ...        ...          ...        ...              ...
...        ...
1460  B08L7J...  Noir A...    Home&K...        379              919
...  R3G3XF...
1461  B01M64...  Presti...    Home&K...      2,280            3,045
...  R3DDL2...
1462  B009P2...  Bajaj ...    Home&K...      2,219            3,080
...  R1TLRJ...
1463  B00J5D...  Havell...    Home&K...      1,399            1,890
...  R39Q2Y...
1464  B01486...  Borosi...    Home&K...      2,863            3,690
...  R20RBR...

     review_title review_content   img_link product_link
0     Satisf...    Looks ...      https:...  https:...
1     A Good...    I orde...      https:...  https:...
2     Good s...    Not qu...      https:...  https:...
3     Good p...    Good p...      https:...  https:...
4     As goo...    Bought...      https:...  https:...
...         ...          ...            ...        ...
1460  Receiv...    I rece...      https:...  https:...
1461  ok,eve...    ok,got...      https:...  https:...
1462  very g...    plasti...      https:...  https:...
1463  Fan Sp...    I have...      https:...  https:...
1464  Works ...    It doe...      https:...  https:...

[1465 rows x 16 columns]
```

3-4. Jupyter Notebook에서의 DataFrame 디스플레이

Jupyter Notebook에서 DataFrame을 디스플레이하는 방식은 두 가지가 있으며, 각각의 방식으로 출력되는 결과에 차이가 있습니다.

3-4-1. 셀에서 단순히 DataFrame객체를 실행하는 경우

Jupyter Notebook에서 셀에 단순히 DataFrame객체를 입력하고 실행하면, Jupyter는 DataFrame 객체를 HTML 테이블 형식으로 렌더링하여 보여줍니다. 이 방식은 다음과 같은 특징을 가집니다.

- 데이터가 보다 가독성 높게, 그리고 시각적으로 깔끔하게 표시됩니다.
- 인덱스와 열 이름이 굵은 글씨로 표시되어 구분이 쉽습니다.
- 대규모 데이터셋의 경우, 자동으로 일부 행만을 축약하여 보여주며, 전체 데이터를 로딩하지 않아도 전반적인 구조를 파악할 수 있습니다.

3-4-2. 'print()' 함수를 사용하여 출력하는 경우

print() 함수를 사용하여 DataFrame 객체를 출력하면, Python의 표준 출력 방식을 사용하여 데이터가 표시됩니다. 이 방식은 다음과 같은 특징을 가집니다.

- 데이터가 일반 텍스트 형태로 출력됩니다. HTML 테이블 형식으로 렌더링되지 않아, Jupyter Notebook에서 제공하는 시각적 형식보다는 가독성이 떨어질 수 있습니다.
- 큰 데이터셋을 출력할 때도 'pandas' 설정에 따라 일부 데이터만 보여주며, 이 경우 시작 부분과 끝 부분의 일부 데이터만 표시되고 중간은 생략됩니다.
- 열이 많아 한 화면에 모두 표시할 수 없는 경우, 열이 잘려서 출력될 수 있으며, 이를 통제하기 위해선 'pandas'의 출력 설정을 조절해야 합니다.

4. numpy 라이브러리 소개

4-1. numpy 라이브러리 개요

NumPy(Numerical Python의 약자)는 Python 프로그래밍 언어를 위한 핵심 라이브러리 중 하나로, 대규모 다차원 배열 및 행렬 연산에 필수적이며, 이러한 데이터 구조를 조작하기 위한 높은 수준의 수학 함수를 제공합니다.

과학 계산을 위한 Python 생태계의 기본 빌딩 블록 중 하나로 광범위하게 사용되며, 데이터 분석, 기계 학습, 공학 그리고 다양한 과학적 분야에서 필수적인 도구입니다.

4-2. numpy 임포트 하기

numpy는 일반적으로 'np'라는 약칭으로 임포트 합니다.

In [ ]:
```
import numpy as np

# numpy 배열 생성
a = np.array([1, 2, 3])

# 다차원 배열 생성
b = np.array([[1, 2, 3], [4, 5, 6]])

print(a)
print(b)
```

```
[1 2 3]
[[1 2 3]
 [4 5 6]]
```

4-3. numpy와 pandas의 관계

pandas는 numpy 배열을 기반으로 구축되었으며, numpy의 기능을 활용하여 데이터 분석에 특화된 기능을 추가하고 있습니다. pandas와 numpy의 관계는 아래와 같이 요약될 수 있습니다.

- 보완적 관계 : pandas는 내부적으로 numpy를 사용합니다. 이는 pandas의 데이터 구조들이 numpy의 배열 기능 위에 구축되어 있기 때문입니다. 따라서, numpy의 고성능 배열 연산 기능을 pandas에서도 활용할 수 있습니다.
- 특화된 기능 : numpy는 다차원 배열을 중심으로 한 수치 계산에 특화되어 있는 반면, pandas는 이러한 배열을 활용하여 데이터 분석과 처리에 특화된 기능을 제공합니다. 예를 들어, 시계열 데이터 처리, 결측치 처리, 데이터프레임 병합 및 그룹화 같은 고급 데이터 조작 기능이 이에 해당합니다.
- 상호 운용성 : pandas의 데이터 구조는 numpy 배열과 긴밀하게 연동되어, pandas와 numpy를 함께 사용하여 데이터 분석을 수행할 수 있습니다. 예를 들어, pandas 데이터프레임에서 numpy의 수학 함수를 직접 사용할 수 있습니다.