[2-1. 데이터분석도구 Pandas 라이브러리 소개]

1. pandas 라이브러리 소개

- pandas는 파이썬의 여러 라이브러리 중 하나로 마이크로소프트 엑셀과 같이 데이터 분석을 더 효율적으로 편하게 할 수 있도록 다양한 자료 구조와 함수들을 제공한다.
- pandas를 개발한 웨스 맥키니(Wes Mckinney)는 금융회사에서 일하면서 기존 데이터 분석 도구에 한계를 느끼고, 금융 데이터를 분석할 목적으로 pandas를 설계했다고 한다.
- 이러한 이유로 pandas는 금융데이터를 분석하는데 적합한 도구(자료구조와 함수)들을 많이 보유하고 있다.
- pandas 홈페이지 : https://pandas.pydata.org (https://pandas.pydata.org)
- 대표적인 참고서적 : 파이썬 라이브러리를 활용한 데이터 분석(Python for Data Analysis), 저자: 웨스 맥키니

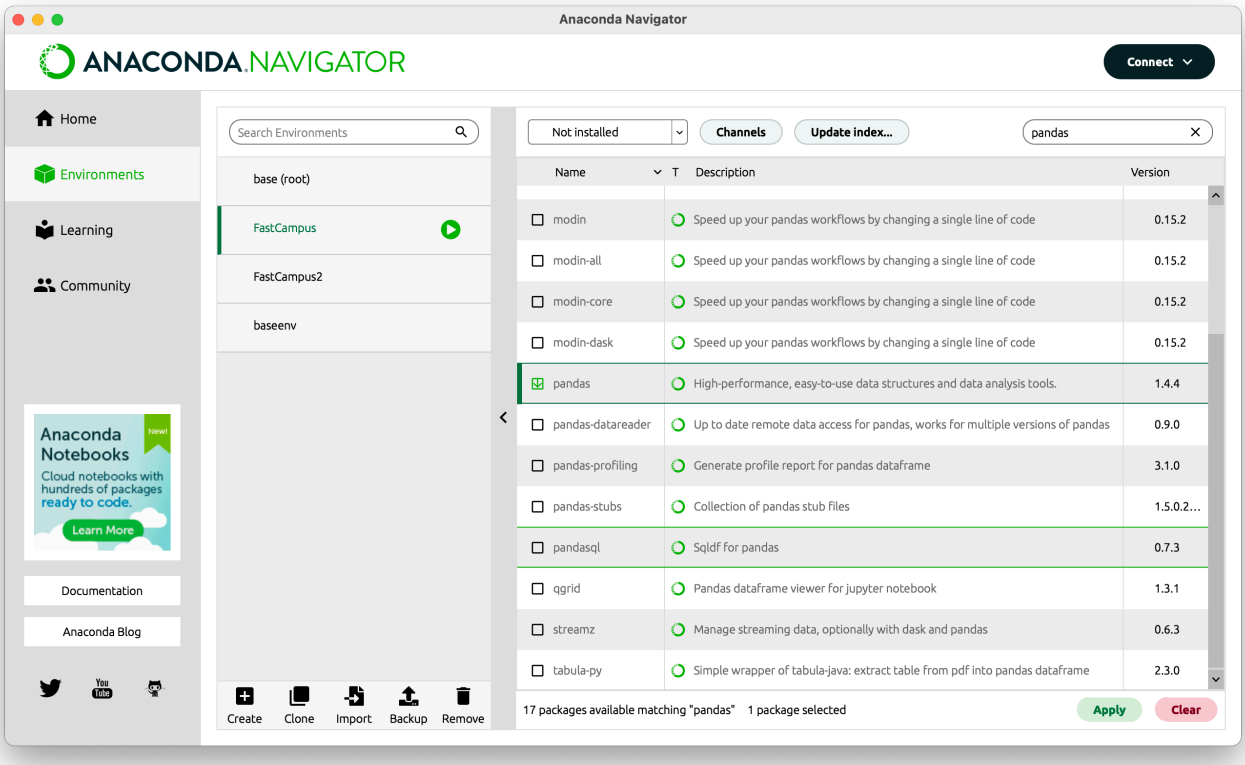
2. pandas의 자료구조 : Series, DataFrame

- pandas에는 Series와 DataFrame의 두가지 자료구조가 있으며, 두 자료구조에 데이터를 저장하고, 자료구조 내에서 데이터를 다듬고, 변형/병합하면서 데이터를 분석한다.
- Series : 1차원의 데이터 배열. index가 있어 각각의 데이터에 index를 통해 접근이 가능.
- DataFrame : 2차원의 데이터 표(스프레드시트). 행과 열에 대해 각각의 index가 있어, index를 통해 데이터 접근이 가능.

3. pandas 설치 및 임포트

(1) pandas 설치

- pandas는 아나콘다에서 기본 제공되는 라이브러리 이므로, 아나콘다 Environments 탭에서 체크하는 방식으로 설치 가능



(2) pandas 임포트

- 일반적으로 pandas 임포트 시에는 "pd"로 축약한 이름을 사용한다.

```
import pandas as pd
```

- DataFrame과 Series는 사용 빈도가 높으므로 직접 임포트해서 사용하기도 한다.

```
from pandas import DataFrame, Series
```

- 결과적으로 pandas를 사용하고자 할 때는, 처음에 아래와 같은 임포트 문을 작성한 후 코드를 작성하는 것이 편하다.

```
import pandas as pd
from pandas import Series, DataFrame
```

4. Series, DataFrame 생성하기

(1) 임포트 하기

In [1]:
```
import pandas as pd
from pandas import Series, DataFrame
```

(2) Series 생성

1) 데이터의 리스트를 전달하여 Series 생성

In [2]:
```
a = Series([1, 2, 3, 4])
```

In [3]:
```
a
```

Out[3]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

=> [1, 2, 3, 4]의 데이터 값과 자동으로 생성된 index [0, 1, 2, 3]을 확인할 수 있다.

In [5]:
```
a[1] #index를 이용해서 데이터에 접근
```

Out[5]:
```
2
```

2) index 값을 직접 설정하여 Series 생성

In [6]:
```
b = Series([1, 2, 3, 4], index=[10, 20, 30, 40])
```

In [7]:
```
b
```

Out[7]:
```
10    1
20    2
30    3
40    4
dtype: int64
```

=> index 값을 별도로 지정할 수 있다.

In [8]:
```
b[20] #index를 이용해서 데이터에 접근
```

Out[8]:
```
2
```

3) 파이썬 딕셔너리를 사용하여 Series 객체 생성

In [9]:
```
dct = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
```

In [10]:
```
dct
```

Out[10]:
```
{'a': 10, 'b': 20, 'c': 30, 'd': 40}
```

In [11]:
```
dct['a']
```

Out[11]:
```
10
```

In [15]:
```
c = Series(dct)  #딕셔너리를 사용하여 Series 객체 생성
```

In [13]:
```
c
```

Out[13]:
```
a    10
b    20
c    30
d    40
dtype: int64
```

=> 딕셔너리의 key값은 Series의 index로 설정됨.

In [14]:
```
c['b']
```

Out[14]:
```
20
```

In [ ]:
```

```

(3) DataFrame 생성

- 2차원의 데이터 표(스프레드시트)
- 각각의 열은 동일한 길이를 가지고 있으며, 하나의 열에는 같은 종류의 데이터를 담을 수 있다.
- 각각의 열은 서로 다른 종류의 데이터를 담을 수 있다.

1) 같은 길이의 리스트가 담긴 딕셔너리를 이용하여 DataFrame 생성

In [16]:
```
data = {'a': [1, 2, 3, 4],
        'b': [10, 20, 30, 40],
        'c': [100, 200, 300,400]}
```

In [17]:
```
data
```

Out[17]:
```
{'a': [1, 2, 3, 4], 'b': [10, 20, 30, 40], 'c': [100, 200, 300, 40
0]}
```

In [18]:
```
a = DataFrame(data)
```

In [19]:
```
a
```

Out[19]:

	a	b	c
0	1	10	100
1	2	20	200
2	3	30	300
3	4	40	400

=> dictionary의 key는 DataFrame의 컬럼(열) 이름이 되었으며,
=> index는 자동 생성되어 [0, 1, 2, 3]이 적용되었음.

In [ ]:
```

```

2) DataFrame 생성시 index 값 설정하기

In [17]:
```
b = DataFrame(data, index=['k', 'l', 'm', 'n'])
```

In [18]:
```
b
```

Out[18]:

	a	b	c
k	1	10	100
l	2	20	200
m	3	30	300
n	4	40	400

In [ ]:
```

```

3) 배열을 이용하여 DataFrame 생성하기

In [23]:
```
data = [[2022, 100, 1000],
        [2023, 200, 2000],
        [2024, 300, 3000],
        [2025, 400, 4000]]
```

In [24]:
```
c = DataFrame(data, columns=['year', 'data1', 'data2'],
              index=['one', 'two', 'three', 'four'])
```

In [25]:
```
c
```

Out[25]:

	year	data1	data2
one	2022	100	1000
two	2023	200	2000
three	2024	300	3000
four	2025	400	4000

In [ ]:
```

```

4) DataFrame에서 데이터 추출하기

In [26]:
```
c['year'] #컬럼명을 이용하여 컬럼 데이터 추출
```

Out[26]:
```
one      2022
two      2023
three    2024
four     2025
Name: year, dtype: int64
```

In [27]:
```
c.loc['two'] #로우 데이터를 추출하려면 loc 메서드를 사용함
```

Out[27]:
```
year     2023
data1     200
data2    2000
Name: two, dtype: int64
```

In [28]:
```
c.loc['two', 'data1'] #하나의 데이터에 접근하기 위해서는 loc 메서드를 사용하여 로우명과 컬럼명을 같이 지정해 줌.
```

Out[28]:
```
200
```

In [29]:
```
c.loc['four', :]
```

Out[29]:
```
year     2025
data1     400
data2    4000
Name: four, dtype: int64
```

In [30]:
```
c.loc[:, 'data2']
```

Out[30]:
```
one      1000
two      2000
three    3000
four     4000
Name: data2, dtype: int64
```