[6-1. 시계열데이터의 기본 처리방법(datetime)]

1. datetime 라이브러리 소개

Python 표준 라이브러리인 datetime 라이브러리는 날짜와 시간을 다루기 위한 다양한 모듈을 제공합니다. 이를 사용하여 날자와 시간을 생성, 조작, 연산, 포맷팅하는 등의 작업을 수행할 수 있습니다.

1-1. 주요 모듈과 기능

- date : 연, 월, 일을 표현합니다.
- time : 시간, 분, 초, 마이크로초를 표현합니다.
- datetime : 날짜와 시간을 함께 표현합니다.
- timedelta : 두 날짜나 시간 사이의 차이를 표현합니다.
- tzinfo : 시간대 관련 정보를 표현하는 기본 클래스 입니다.

1-2. datetime 객체 생성 예시

1-2-1. datetime 객체 생성하기

In [ ]:
```
# datetime 라이브러리에서 datetime 모듈 임포트하기
from datetime import datetime

# 현재 시간 가져오기(datetime 모듈의 now() 메서드 사용)
now = datetime.now()
print("현재 시간 : ", now)
```

```
현재 시간 :  2024-03-01 17:21:24.748900
```

In [ ]:
```
# 직접 연월일을 입력하여 datetime 객체 생성하기
dt = datetime(2024, 3, 1, 10, 32, 10)
print(dt)
```

```
2024-03-01 10:32:10
```

In [ ]:
```
dt = datetime(2024, 3, 1)
print(dt)
```

```
2024-03-01 00:00:00
```

In [ ]:
```
# date 객체로 변환하기
print(dt.date())
```

```
2024-03-01
```

1-2-2. date, time 객체 생성하기

In [ ]:
```
# datetime 라이브러리에서 date, time 모듈 임포트하기
from datetime import date, time

# date 객체 생성하기
d = date(2020, 1, 1)
print("날짜 : ", d)

# time 객체 생성하기
t = time(12, 30)
print("시간 : ", t)
```

```
날짜 :  2020-01-01
시간 :  12:30:00
```

1-3. 날짜와 시간 연산하기

1-3-1. 특정 날짜로부터 일정 기간 후의 날짜 계산하기

In [ ]:
```
from datetime import datetime, timedelta

# 2023년 1월 1일로부터 10일 후
start_date = datetime(2023, 1, 1)
days_after = start_date + timedelta(days=10)
print("10일 후:", days_after)
```

```
10일 후: 2023-01-11 00:00:00
```

In [ ]:
```
# date객체에 대해서 날짜 계산하기
start_date = date(2023, 1, 1)
days_after = start_date + timedelta(days=10)
print("10일 후:", days_after)
```

```
10일 후: 2023-01-11
```

1-3-2. 두 날짜 간의 차이 계산하기

In [ ]:
```
date1 = datetime(2023, 1, 15)
date2 = datetime(2023, 2, 1)
difference = date2 - date1
print("두 날짜의 차이 (일):", difference.days)
```

```
두 날짜의 차이 (일): 17
```

In [ ]:
```
date1 = date(2023, 1, 15)
date2 = date(2023, 2, 1)
difference = date2 - date1
print("두 날짜의 차이 (일):", difference.days)
```

```
두 날짜의 차이 (일): 17
```

1-3-3. 날짜에서 일정 기간 연산

In [ ]:
```
# 2023년 3월 1일로부터 2주 전
target_date = datetime(2023, 3, 1)
weeks_before = target_date - timedelta(weeks=2)
print("2주 전:", weeks_before)
```

```
2주 전: 2023-02-15 00:00:00
```

In [ ]:
```
# 2023년 3월 1일로부터 2주 전
target_date = date(2023, 3, 1)
weeks_before = target_date - timedelta(weeks=2)
print("2주 전:", weeks_before)
```

```
2주 전: 2023-02-15
```

1-3-4. 현재 시각으로부터 특정 시간 더하기

In [ ]:
```
# 현재 시각으로부터 5시간 30분 후
now = datetime.now()
hours_later = now + timedelta(hours=5, minutes=30)
print("5시간 30분 후:", hours_later)
```

```
5시간 30분 후: 2024-03-01 23:07:51.530961
```

1-3-5. 초 단위로 날짜와 시간 연산하기

In [ ]:
```
# 현재 시각으로부터 3600초(1시간) 후
now = datetime.now()
one_hour_later = now + timedelta(seconds=3600)
print("3600초(1시간) 후:", one_hour_later)
```

```
3600초(1시간) 후: 2024-03-01 18:37:41.347988
```

1-4. timedelta에서 사용 가능한 매개변수

timedelta 객체는 두 날짜나 시간 사이의 차이를 표현할 때 사용되며, timedelta를 사용할 때 다음과 같은 매개변수를 지정할 수 있습니다:

- weeks: 주를 나타내며, 정수 또는 부동소수점 수가 될 수 있습니다. 주는 일(days)로 변환되어 저장됩니다.
- days: 일 수를 나타내며, 정수 또는 부동소수점 수가 될 수 있습니다.
- hours: 시간을 나타내며, 정수 또는 부동소수점 수가 될 수 있습니다. 시간은 초(seconds)로 변환되어 저장됩니다.
- minutes: 분을 나타내며, 정수 또는 부동소수점 수가 될 수 있습니다. 분은 초(seconds)로 변환되어 저장됩니다.
- seconds: 초를 나타내며, 정수 또는 부동소수점 수가 될 수 있습니다. (0에서 86399 사이)
- milliseconds: 밀리초를 나타내며, 정수 또는 부동소수점 수가 될 수 있습니다. 밀리초는 초(seconds)로 변환되어 저장됩니다.
- microseconds: 마이크로초를 나타내며, 정수 또는 부동소수점 수가 될 수 있습니다.

2. 날짜와 시간 formatting 하기

2-1. strftime 메서드 사용하기

datetime 모듈의 strftime() 메서드를 사용하여 날짜와 시간을 다양한 형식의 문자열로 formatting 할 수 있습니다.

2-1-1. 예시 1: 연-월-일 포맷

In [ ]:
```
now = datetime.now()
formatted_date = now.strftime("%Y-%m-%d")
print("현재 날짜 (연-월-일):", formatted_date)
```

```
현재 날짜 (연-월-일): 2024-03-01
```

In [ ]:
```
dt = now.date()
formatted_date = dt.strftime("%Y-%m-%d")
print("현재 날짜 (연-월-일):", formatted_date)
```

```
현재 날짜 (연-월-일): 2024-03-01
```

2-1-2. 예시 2: 12시간 형식의 시간과 AM/PM 표시

In [ ]:
```
formatted_time = now.strftime("%I:%M:%S %p")
print("현재 시간 (12시간 형식, AM/PM):", formatted_time)
```

```
현재 시간 (12시간 형식, AM/PM): 05:44:46 PM
```

2-1-3. 예시 3: 24시간 형식의 시간

In [ ]:
```
formatted_time_24hr = now.strftime("%H:%M:%S")
print("현재 시간 (24시간 형식):", formatted_time_24hr)
```

```
현재 시간 (24시간 형식): 17:44:46
```

2-1-4. 예시 4: 요일, 월, 일, 연도 포맷

In [ ]:
```
formatted_full_date = now.strftime("%A, %B %d, %Y")
print("현재 날짜 (요일, 월 일, 연도):", formatted_full_date)
```

```
현재 날짜 (요일, 월 일, 연도): Friday, March 01, 2024
```

In [ ]:
```
formatted_full_date = dt.strftime("%A, %B %d, %Y")
print("현재 날짜 (요일, 월 일, 연도):", formatted_full_date)
```

```
현재 날짜 (요일, 월 일, 연도): Friday, March 01, 2024
```

2-1-5. 예시 5: ISO 8601 형식의 날짜와 시간

In [ ]:
```
formatted_iso8601 = now.strftime("%Y-%m-%dT%H:%M:%S")
print("현재 날짜와 시간 (ISO 8601 형식):", formatted_iso8601)
```

```
현재 날짜와 시간 (ISO 8601 형식): 2024-03-01T17:44:46
```

In [ ]:
```
formatted_iso8601 = dt.strftime("%Y-%m-%dT%H:%M:%S")
print("현재 날짜와 시간 (ISO 8601 형식):", formatted_iso8601)
```

```
현재 날짜와 시간 (ISO 8601 형식): 2024-03-01T00:00:00
```

2-2. strptime 메서드 사용하기

datetime 모듈의 strptime() 메서드를 사용하여 문자열을 datetime 객체로 변환 가능합니다.

2-2-1. 예시 1: "연-월-일" 포맷

In [ ]:
```
date_str1 = "2023-01-25"
date_obj1 = datetime.strptime(date_str1, "%Y-%m-%d")
print("예시 1:", date_obj1)
```

```
예시 1: 2023-01-25 00:00:00
```

In [ ]:
```
# date 모듈에는 strptime 메서드가 없음
date_obj1 = date.strptime(date_str1, "%Y-%m-%d")
print("예시 1:", date_obj1)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[47], line 2
      1 # date 모듈에는 strptime 메서드가 없음
----> 2 date_obj1 = date.strptime(date_str1, "%Y-%m-%d")
      3 print("예시 1:", date_obj1)

AttributeError: type object 'datetime.date' has no attribute 'strptime'
```

In [ ]:
```
# date 객체를 얻고자 하는 경우 date() 메서드를 사용
date_obj1 = datetime.strptime(date_str1, "%Y-%m-%d").date()
print("예시 1:", date_obj1)
```

```
예시 1: 2023-01-25
```

2-2-2. 예시 2: "월/일/연 시:분" 포맷

In [ ]:
```
date_str2 = "01/25/2023 14:30"
date_obj2 = datetime.strptime(date_str2, "%m/%d/%Y %H:%M")
print("예시 2:", date_obj2)
```

```
예시 2: 2023-01-25 14:30:00
```

2-2-3. 예시 3: "일-월-연 시간(12시간) AM/PM" 포맷

In [ ]:
```
date_str3 = "25-Jan-2023 2:30 PM"
date_obj3 = datetime.strptime(date_str3, "%d-%b-%Y %I:%M %p")
print("예시 3:", date_obj3)
```

```
예시 3: 2023-01-25 14:30:00
```

2-2-4. 예시 4: "연도월일시분초" 포맷

In [ ]:
```
date_str4 = "20230125143000"
date_obj4 = datetime.strptime(date_str4, "%Y%m%d%H%M%S")
print("예시 4:", date_obj4)
```

```
예시 4: 2023-01-25 14:30:00
```

2-2-5. 예시 5: "요일, 일 월 연도 시:분:초" 포맷

In [ ]:
```
date_str5 = "Wednesday, 25 Jan 2023 14:30:00"
date_obj5 = datetime.strptime(date_str5, "%A, %d %b %Y %H:%M:%S")
print("예시 5:", date_obj5)
```

```
예시 5: 2023-01-25 14:30:00
```

2-3. datetime 포맷 규칙

- %Y: 4자리 연도
- %y: 2자리 연도
- %m: 0으로 채워진 2자리 월
- %d: 0으로 채워진 2자리 일
- %I: 0으로 채워진 12시간 형식의 시
- %H: 0으로 채워진 24시간 형식의 시
- %M: 0으로 채워진 2자리 분
- %S: 0으로 채워진 2자리 초
- %p: AM 또는 PM
- %w: 정수로 나타낸 요일[0 (일요일), 6)
- %W: 연중 주차'00', '53'
  - 연도의 첫번재 일요일이 속한 주를 첫번째 주('01')로 정의하고, 그 전의 날짜들은 '00'으로 표시
- %U: 연중 주차'00', '53'
  - 연도의 첫번재 월요일이 속한 주를 첫번째 주('01')로 정의하고, 그 전의 날짜들은 '00'으로 표시
- %a: 축약된 요일 이름
- %A: 요일의 전체 이름
- %b: 축약된 월 이름
- %B: 월의 전체 이름

3. dateutil.parser 사용하기

3-1. dateutil.parser 소개

dateutil.parser의 parse 메서드는 문자열 형식의 날짜를 자동으로 파싱하여 Python datetime 객체로 변환해주는 도구입니다.

이 메서드는 다양한 날짜 및 시간 포맷을 자동으로 인식하고 처리할 수 있어, 특정 포맷을 명시하지 않아도 사용 가능합니다.

dateutil을 사용하기 위해서는 dateutil 라이브러리가 설치되어 있어야 합니다. 아나콘다 네비게이터 environments에서 설치 가능합니다.

3-2. 사용방법

In [ ]:
```python
from dateutil.parser import parse

# 기본적인 날짜 문자열 파싱
dt = parse("2024-01-01")
print(dt)

# 더 복잡한 날짜와 시간 문자열
dt = parse("1st of January, 2024 at 12:30 PM")
print(dt)

# 일/월/연 포맷 파싱 (dayfirst=True를 사용하여 날짜가 먼저 오는 것을 명시)
dt = parse("01/01/2024", dayfirst=True)
print(dt)

# 연/월/일 포맷 파싱 (yearfirst=True를 사용하여 연도가 먼저 오는 것을 명시)
dt = parse("2024/01/01", yearfirst=True)
print(dt)

# ISO 8601 포맷 파싱
dt = parse("2024-01-01T12:30:00Z")
print(dt)
```

```
2024-01-01 00:00:00
2024-01-01 12:30:00
2024-01-01 00:00:00
2024-01-01 00:00:00
2024-01-01 12:30:00+00:00
```

In [ ]:
```python

```