빅데이터 수집시스템 개발

시계열 데이터 다루기

한국기술교육대학교
온라인평생교육원

학습목표

- 날짜 데이터와 시간 데이터를 문자열 데이터로 전환하고, 데이터로부터 날짜와 시간 등의 데이터를 추출할 수 있다.
- 날짜와 시간에 적절한 연산을 수행하여 과거 또는 미래의 시간을 알아낼 수 있다.

학습내용

- 날짜와 시간 API
- 시계열 데이터 다루기

날짜와 시간 API

1 DateTime 모듈

- 파이썬에서 날짜와 시간을 다루기 위해 제공
- 파이썬이 기본으로 지원하는 모듈
- 날짜 및 시간에 대한 산술연산 지원

DateTime을 이용하여 현재 날짜와 시간을 얻는 방법

- import datetime ← 모듈 임포트
- datetime_object = datetime.datetime.now()
  ← 현재 시간을 얻음
- print(datetime_object) ← 객체 출력
- 결과 : 2050-10-16 15:50:07.625273

- dir 함수를 이용해 클래스 내부의 구조를 확인할 수 있음

```
import datetime          ← 모듈 임포트
print(dir(datetime))     ← 모듈 내부의 내용 확인
```

```
['MAXYEAR', 'MINYEAR', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__
', '__name__', '__package__', '__spec__', 'date', 'datetime', 'datetime_CAPI', 'sys', '
time', 'timedelta', 'timezone', 'tzinfo']
```

날짜와 시간 API

1 DateTime 모듈

- DataTime 모듈은 다음 4개의 클래스로 구성됨

Date Class	▪ 날짜에 대한 정보를 다룸(연도, 월, 일)
Timestamp Class	▪ 시간에 대한 정보를 다룸
DateTime Class	▪ 날짜와 시간에 대한 모든 정보를 다룸
Timedelta Class	▪ 두 날짜와 시간 사이의 차이를 다룸

날짜와 시간 API

1 DateTime 모듈

1 Date Class

- 현재 날짜를 구하려면 Date 클래스의 today 함수 사용

```
import datetime

date_object = datetime.date.today()
print(date_object)

결과 : 2050-10-19
```

- 생성자에 특정 연도, 월, 일을 전달하여 Date 객체를 생성하기도 함

```
import datetime
d = datetime.date(2050, 10, 13)

#생성자가 연도, 월, 일을 입력받음. 세 개의 매개변수가 있어야 함
print(d)

결과 : 2050-10-13
```

날짜와 시간 API

1 DateTime 모듈

1 Date Class

- from절을 이용하여 객체를 로딩할 수 있음
- from절에 선언된 모듈로부터 date 클래스 객체만 임포트 함
- 이 경우 모듈명을 생략하고 객체를 생성할 수 있음

```
from datetime import date
d = date(2050, 10, 13)
print(d)
```

날짜와 시간 API

1 DateTime 모듈

2 Timestamp Class

- 유닉스/리눅스 계열에서 사용하는 시간을 나타내는 정수
- 1970년 1월 1일 0시 0분 0초부터 몇 초가 지났는지 나타내는 정수값
- 지구자전축의 움직임을 고려하는 윤초(Leap Second)까지 반영하여 엄격하게 정의한 숫자가 아님
  ➡ 단순히 하루를 86,400초로 계산하여 순차적으로 증가시킨 것

```
from datetime import date
value = 1567345678
timestamp = date.fromtimestamp(1567345678)
print("timestamp =", 1567345678)
print("Date =", timestamp)

결과 :
timestamp = 1567345678
Date = 2050-09-01
```

날짜와 시간 API

1 DateTime 모듈

3 DateTime Class

- DateTime 클래스의 now 함수는 현재의 날짜와 시간에 대한 DateTime 객체를 반환
- year, month, day 필드를 통해 연도와 월, 일 정보를 추출할 수 있음

```
import datetime
today = datetime.datetime.now()
print("년도 : ", today.year)
print("월 : ",today.month)
print("일 : ",today.day)

결과 :
년도 : 2050
월 : 10
일 : 19
```

날짜와 시간 API

1 DateTime 모듈

3 DateTime Class

- hour, minute, second 필드를 이용해 시간과 분, 초를 추출할 수 있음

```
import datetime
today = datetime.datetime.now()
print("시간 : ", today.hour)
print("분 : ",today.minute)
print("초 : ",today.second)
```

결과 :
시간 : 11
분 : 10
초 : 35

날짜와 시간 API

1 DateTime 모듈

3 DateTime Class

- weekday() 함수 제공 ➡ 요일에 대한 정보 반환

월요일 : 0

화요일 : 1

수요일 : 2

목요일 : 3

금요일 : 4

토요일 : 5

일요일 : 6

```
weekday = ["월요일", "화요일", "수요일", "목요일",
"금요일", "토요일", "일요일"]
today = datetime.datetime.now()
print("요일 : ",today.weekday())
day =  today.weekday()
print("{0}-{1}-{2}는 {3}입니다"
.format(today.year,       today.month, today.day, week
day[day]))
```

날짜와 시간 API

1 DateTime 모듈

3 DateTime Class

```
#파일명 : exam14_1.py

import datetime

print("현재 날짜와 시간 알아보기")
today = datetime.datetime.now()
print(today)

#내부 구조 확인 - 클래스의 내부 구조 확인 가능
print()
print(dir(datetime))
print()

#현재 날짜
print()
date_object = datetime.date.today()
print(date_object)

#연도, 월, 일을 입력하여 date 객체 생성
d = datetime.date(2050, 10, 13)
print(d)
```

(계속)

날짜와 시간 API

1 DateTime 모듈

3 DateTime Class

```
#from절을 이용하여 모듈을 임포트 함
#이럴 경우 모듈명을 사용하지 않고 클래스명만으로 객체 생성 가능
from datetime import date
a = date(2050, 10, 13)
print(a)

#timestamp로부터 날짜 추출
value = 1567345678
timestamp = date.fromtimestamp(1567345678)
print("timestamp =", 1567345678)
print("Date =", timestamp)

#각 필드를 이용해 날짜와 시간 추출
today = datetime.datetime.now()
print(type(today))
print("년도 : ", today.year)
print("월 : ",today.month)
print("일 : ",today.day)

today = datetime.datetime.now()
print("시간 : ", today.hour)
print("분 : ",today.minute)
print("초 : ",today.second)
```

(계속)

날짜와 시간 API

1 DateTime 모듈

3 DateTime Class

```
#요일  월요일 : 0, 화요일 : 1, 수요일 : 2, 목요일 : 3, 금요일 : 4,
토요일 : 5,  일요일 : 6
weekday = ["월요일", "화요일", "수요일", "목요일", "금요일",
"토요일", "일요일"]
today = datetime.datetime.now()
print("요일 : ",today.weekday())
day =  today.weekday()
print("{0}-{1} {2}는 {3}임"
.format(today.year,    today.month, today.day, weekday[day]))
```

날짜와 시간 API

1 DateTime 모듈

4 Timedelta Class

- 두 날짜와 시간 사이의 차이를 저장하는 클래스
- Date 객체와 date 객체 간의 연산의 결과는 Timedelta객체로 반환됨

```
from datetime import datetime, date
d1 = date(2050,10, 13)
d2 = date(2050,12, 31)

d3 = d2 - d1

print("올해 마지막 일자까지 남은 기간은 " , d3, "입니다")
```

날짜와 시간 API

1 DateTime 모듈

4 Timedelta Class

Timedelta의 인자값

- 1주 : datetime.timedelta(weeks=1)
- 1일 : datetime.timedelta(days=1)
- 1시간 : datetime.timedelta(hours=1)
- 1분 : datetime.timedelta(minutes=1)
- 1초 : datetime.timedelta(seconds=1)
- 1밀리초 : datetime.timedelta(milliseconds=1)
- 1마이크로초 : datetime.timedelta(microseconds=1)

날짜와 시간 API

1 DateTime 모듈

4 Timedelta Class

#파일명 : exam14_2.py

```
from datetime import datetime, date,  timedelta
d1 = date(2050,10, 13)
d2 = date(2050,12, 31)
#날짜 연산
d3 = d2 - d1
print("올해 남은 날", d3)
print("type of d3 =", type(d3))

t1 = datetime(year = 2050, month = 10, day = 19, hour = 7,
minute = 9, second = 33)
t2 = datetime(year = 2050, month = 10, day = 19, hour = 5,
minute = 1, second = 13)
t3 = t1 - t2
print("t3 =", t3)
print("type of t3 =", type(t3))
```

```
올해 남은날 79 days, 0:00:00
type of d3 = <class 'datetime.timedelta'>
t3 = 2:08:20
type of t3 = <class 'datetime.timedelta'>
```

(계속)

날짜와 시간 API

1 DateTime 모듈

4 Timedelta Class

```
#timedelta 사이의 연산
t1 = timedelta(weeks = 2, days = 5, hours = 1, seconds = 33)
t2 = timedelta(days = 4, hours = 11, minutes = 4,
seconds = 54)
t3 = t1 - t2
print("t3 =", t3)

t1 = timedelta(seconds = 33)  #음수도 가능
t2 = timedelta(seconds = 54)
t3 = t1 - t2
print( " t3 = " , t3)
print( " t3 = " , abs(t3))

#기간을 초로 나타내기
t = timedelta(days = 1, hours = 2, seconds = 33,
microseconds = 233)
print("total seconds =", t.total_seconds())

#지금부터 3시간 뒤
current = datetime.today()
after = current + timedelta(hours=3)
print(current)
print(after)
```

날짜와 시간 API

1 DateTime 모듈

4 Timedelta Class

```
t3 = 14 days, 13:55:39
t3 = -1 day, 23:59:39
t3 = 0:00:21
total seconds = 93633.000233
2050-10-20 01:10:24.044908
2050-10-20 04:10:24.044908
```

날짜와 시간 API

2 DateTime 포매팅

- datetime을 원하는 서식으로 변경하여 문자열로 출력할 수 있음
- strftime()은 datetime 오브젝트를 서식 문자열을 이용해 문자열로 전환할 수 있음

```
import datetime

#서식 날짜 데이터 바꾸기
today = datetime.datetime.now()
print(today)

print(today.strftime('%Y-%m-%d'))
print(today.strftime('%H:%M:%S'))
print(today.strftime('%Y-%m-%d %H:%M:%S'))
```

```
2050-10-20 18:21:39.962898
2050-10-20
18:21:39
2050-10-20 18:21:39
```

날짜와 시간 API

2 DateTime 포매팅

서식	의미
%Y	year[0000~9999]
%m	month[01, 02, 03, ~ ,12]
%d	day[01, 02, 03, ~ 3, 1]
%H	hour[00, 02, ~ , 23]
%M	minute[00, 02, ~ , 59]
%S	second[00 ~ 59]
%B	month[January, Febuary~ December]

날짜와 시간 API

3 Timezone

- pytz 모듈을 이용하여 특정 지역의 시간을 확인할 수 있음
- 특별히 지정하지 않으면 날짜는 OS에 지정된 timezone을 사용
- pytz 모듈의 timezone 함수에 (지역 이름)을 전달하면 그 지역의 시간 개체를 반환

```
from datetime import datetime
import pytz
format = "%Y-%m-%d, %H:%M:%S"
local = datetime.now()
print("현재지역 :", local.strftime(format))

tz_NY = pytz.timezone('America/New_York')
datetime_NY = datetime.now(tz_NY)
print("뉴욕시간 :", datetime_NY.strftime(format))
```

날짜와 시간 API

3 Timezone

```
#파일명 :exam14_3.py

import datetime

#서식 날짜 데이터 바꾸기
today = datetime.datetime.now()
print(today)

print(today.strftime('%Y-%m-%d'))
print(today.strftime('%H:%M:%S'))
print(today.strftime('%Y-%m-%d %H:%M:%S'))

print(today.strftime('%Y-%m-%d %B'))
sDate = '2050-10-10 14:22:34'
#str 형태의 날짜를 date 타입으로 전환하기
date1 = datetime.datetime.strptime(sDate, '%Y-%m-
%d %H:%M:%S')
print(type(date1))
print(date1)
```

(계속)

날짜와 시간 API

3 Timezone

```
#타임존
from datetime import datetime
import pytz
format = "%Y-%m-%d, %H:%M:%S"
local = datetime.now()
print("현재 지역 :", local.strftime(format))
tz_NY = pytz.timezone('America/New_York')
datetime_NY = datetime.now(tz_NY)
print("뉴욕시간 :", datetime_NY.strftime(format))
tz_London = pytz.timezone('Europe/London')
datetime_London = datetime.now(tz_London)
print("런던시간 :", datetime_London.strftime(format))
```

날짜와 시간 API

4 달의 마지막 날 구하기

- calendar 모듈의 monthrange 함수를 사용
  ➡ 해당 달의 시작일과 마지막일을 구할 수 있음
- 해당 연도와 월을 두면 튜플 형태로 시작일과 마지막일을 반환

```
import datetime

import calendar
start_day, last_day = calendar.monthrange(2050, 10)
print("시작일 : ", start_day)
print("마지막일 : ", last_day)
```

날짜와 시간 API

4 달의 마지막 날 구하기

```
#파일명 : exam14_4.py

#날짜 연산하기
import datetime

import calendar
start_day, last_day = calendar.monthrange(2050, 10)
print(start_day)
print(last_day)

#2050년 한 해의 말일을 출력
for i in range(1,13):
    start_day, last_day = calendar.monthrange(2050, i)
    print(i, "월의 마지막 날은 ", last_day, "입니다")
```

시계열 데이터 다루기

1 Pandas 라이브러리의 시계열 데이터

- Pandas 라이브러리에서도 시계열 데이터 타입을 제공
- Pandas에서는 인덱스를 DateTimeIndex로 만들어야 함

DateTimeIndex	▪ 특정한 순간에 기록된 타임스탬프(Timestamp) 형식의 시계열 자료를 다루기 위한 인덱스 ▪ 라벨값이 반드시 일정한 간격일 필요는 없음
to_datetime 함수	▪ 문자열 형태의 날짜와 시간을 DateTimeIndex로 전환
date_range 함수	▪ 특정범위의 날짜와 시간 데이터를 자동 생성

시계열 데이터 다루기

1 Pandas 라이브러리의 시계열 데이터

1 to_datetime 함수

#파일명 : exam14_5.py

```
import pandas as pd
import numpy as np

#날짜 문자열은 연도, 월, 일이 구분되면 됨
date_str = ["2050-10-01", "2050-10-
2", "2050/10/3", "20501004"]
idx = pd.to_datetime(date_str)
print(type(idx))
print(idx)

np.random.seed(0)
s = pd.Series(np.random.randn(4), index=idx)
print(s)

#인덱스를 이용하여 출력할 수 있음
print(s['2050-10-01'])
print(s['2050-10-02'])
```

- 랜덤값은 수행할 때마다 수집값이 랜덤하게 바뀜
- seed 함수를 호출하여 같은 정수를 지정하면 같은 그룹의 랜덤값을 계속 가져옴

- DatetimeIndex의 라벨명으로 Series 객체의 데이터에 접근

```
<class 'pandas.core.indexes.datetimes.DatetimeIndex'>
DatetimeIndex(['2050-10-01', '2050-10-02', '2050-10-03', '2050-10-04'], dtype='datetime64[ns]', freq=None)
2050-10-01    1.764052
2050-10-02    0.400157
2050-10-03    0.978738
2050-10-04    2.240893
dtype: float64
1.764052345967664
0.4001572083672233
```

시계열 데이터 다루기

1 Pandas 라이브러리의 시계열 데이터

2 date_range 함수

- 특정 기간의 날짜 데이터를 반복하여 생성하도록 도와줌

date_range(start=시작일, 종료일)

- 시작일부터 종료일까지 반복하여 날짜 데이터 생성

date_range(시작일, periods=숫자)

- 시작일부터 periods에 지정된 숫자만큼 데이터 생성

date_range(시작일, 종료일 또는 periods, freq)

- 시작일부터 종료일 또는 지정된 숫자만큼을 freq에서 지정한 인수에 따라, 특정 요일 등 지정된 일자에 대한 데이터 생성

시계열 데이터 다루기

1 Pandas 라이브러리의 시계열 데이터

3 freq 인수

인수	의미	인수	의미
S	초	M	각 달(Month)의 마지막 날
T	분	MS	각 달의 첫 날
H	시간	BM	주말이 아닌 평일 중 각 달의 마지막 날
D	일(Day)	BMS	주말이 아닌 평일 중에서 각 달의 첫 날
B	주말이 아닌 평일	WOM- 2THU	각 달의 두 번째 목요일
W	주(일요일)	Q-JAN	각 분기의 첫 달의 마지막 날
W-MON	주(월요일)	Q-DEC	각 분기의 마지막 달의 마지막 날

시계열 데이터 다루기

1 Pandas 라이브러리의 시계열 데이터

4 resample 연산

- resample 연산을 쓰면 **시간 간격을 재조정**하는 리샘플링(Resampling)이 가능함
- 업-샘플링(Up-sampling) : 시간 구간이 작아지면 데이터 양이 증가
- 다운-샘플링(Down-sampling) : 시간 구간이 커지면 데이터 양이 감소

```
import pandas as pd
import numpy as np

ts = pd.Series(np.random.randn(100), index=
        pd.date_range( "2050-1-
1", periods=100, freq="D"))
print(ts.tail(20))

print(ts.resample('W').mean())
```

시계열 데이터 다루기

1 Pandas 라이브러리의 시계열 데이터

4 resample 연산

```
#파일명 : exam14_6.py

import pandas as pd
import numpy as np

#2050년 9월 1일부터 2050년 9월 30일까지 생성
d = pd.date_range(start="2050-9-1", end="2050-9-30")
print(d)

d = pd.date_range(start="2050-1-1", periods=60)
print(d)

#2050년 각 달의 마지막 날만 데이터 생성
np.random.seed(0)
ts = pd.Series(np.random.randn(12),
    index=pd.date_range(start="2050-01-
01", periods=12, freq="M"))
print(ts)

#resample 연산
ts = pd.Series(np.random.randn(100),
                index=pd.date_range("2050-1-
1", periods=100, freq="D"))
print(ts.tail(20))  #마지막 20개만 출력

print(ts.resample('W').mean()) #D(일일) -> W(주)형식으로 전환
```

학습정리

1. 날짜와 시간 API

- DateTime 모듈에는 Date, Time, DateTime, TimeDelta 클래스가 있음
- 현재 날짜와 시간은 DateTime클래스의 now 함수를 이용해 얻을 수 있음
- TimeDelta 클래스는 두 날짜와의 차이나 두 시간과의 차이를 얻고자 할 때 사용함, 날짜와 날짜는 서로 산술연산이 가능하고 음수도 가질 수 있음
- 각 달의 마지막 날은 calandar 객체의 monthrange 함수를 이용해 얻을 수 있음

학습정리

2. 시계열 데이터 다루기

- Pandas 라이브러리는 to_datetime 함수를 이용해 날짜와 시간 타입을 인덱스로 만들 수 있음
- 날짜 인덱스를 이용해 데이터에 대한 접근 가능
- 날짜 인덱스는 date_range 함수를 이용해 특정 기간 동안의 날짜 데이터,
  특정 시점부터 원하는 일수만큼(periods), 원하는 특정일에 대한 연속된 데이터를 생성할 수 있음
- resampling을 통해 시간 간격을 재조정하는 리샘플링이 가능함