Lecture7_Dates

February 15, 2024

1 Dates, times and datetimes in Python

1.1 Dates in Python

- Dates and times contain so much information!
- Python dates should be in the form YYYY-MM-DD, and they should be a datetime object so that Python recognises them as dates.
- For example, 2024-02-21 is the correct format for today's date.
- In Pandas, we use the function pd.to_datetime to convert a date into the correct format.
- There is a datetime package in Python dedicated to dealing with datetime objects. It has lots of useful features!

1.2 Datetime package

[17]:
```
from datetime import date

date_today = date(2024,2,21)

print(date_today)

date_today = date.today()

print(date_today)
```

```
2024-02-21
2024-02-15
```

[18]:
```
print(type(date_today))
```

```
<class 'datetime.date'>
```

[19]:
```
date_today
```

[19]:
```
datetime.date(2024, 2, 15)
```

[20]:
```
print(date_today.day)
```

```
15
```

[21]:
```
print(date_today.month)
```

2

[22]:
```
print(date_today.year)
```

2024

1.3 Times in datetime

Time is in the format hour, minute, seconds, microseconds.

(1 microsecond = 1*10^(-6) seconds)

[23]:
```
from datetime import time

time1 = time(14,0,22,10)

print(time1)
```

14:00:22.000010

[24]:
```
print(type(time1))
```

<class 'datetime.time'>

[25]:
```
print(time1.hour)
```

14

[26]:
```
print(time1.minute)
```

0

[27]:
```
print(time1.second)
```

22

[28]:
```
print(time1.microsecond)
```

10

1.4 Datetimes in datetime

[29]:
```
from datetime import datetime
datetime1 = datetime(2024,2,21,19,30,5,10)

datetime1
```

[29]: datetime.datetime(2024, 2, 21, 19, 30, 5, 10)

[30]:
```
print(datetime1)
```

2024-02-21 19:30:05.000010

[31]:
```
print(type(datetime1))
```

```
<class 'datetime.datetime'>
```

1.5 Separating, joining datetime objects

[32]:
```
print(datetime1.date())
```

```
2024-02-21
```

[33]:
```
print(datetime1.time())
```

```
19:30:05.000010
```

[34]:
```
print(datetime1.replace(day=11, hour=15))
```

```
2024-02-11 15:30:05.000010
```

1.6 Changing format of datetime objects

- strptime() creates a DateTime object from a string representing date and time.
- It takes two arguments: the date and the format in which your date is present.
- Formatting code link: https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes

[35]:
```
date = '21 February, 2024 19:30:00'
python_date = datetime.strptime(date,'%d %B, %Y %H:%M:%S')
print(python_date)
```

```
2024-02-21 19:30:00
```

Use strftime() to format the date into a string representing date and time.

[36]:
```
formatted_date = python_date.strftime('%d/%m/%Y %H:%M')
print(formatted_date)
```

```
21/02/2024 19:30
```

Can use strftime() to extract information from the datetime object and print it to the console:

[37]:
```
print('Weekday :', python_date.strftime('%A'))
print('Month :', python_date.strftime('%B'))
print('Week number :', python_date.strftime('%W'))
print("Locale's date and time representation :", python_date.strftime('%c'))
```

```
Weekday : Wednesday
Month : February
Week number : 08
Locale's date and time representation : Wed Feb 21 19:30:00 2024
```

1.7 Interesting functionality of datetime objects

For example: * What day was it? datetime1.weekday()

Weekday indexed from 0 for Monday to 6 for Sunday.

- Leap year or not?
- What week number is it in the year?
- Duration between two dates

2 Dates in pandas

2.1 Datetime in Pandas

[38]:
```
import pandas as pd

date_pandas = pd.to_datetime('21/02/2024', format = '%d/%m/%Y')

print(date_pandas)
```

```
2024-02-21 00:00:00
```

[39]:
```
date_pandas
```

[39]:
```
Timestamp('2024-02-21 00:00:00')
```

[40]:
```
type(date_pandas)
```

[40]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

Timestamp is the Pandas equivalent of Python's datetime.

2.2 Formatting date code link

https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes

This contains details on all of the codes for specifying date formats in Python.

pd.to_datetime often works without specifying the format, but it is safest to specify the format.

[41]:
```
date = pd.to_datetime("21st of feb, 2024")
print(date)
```

```
2024-02-21 00:00:00
```

[42]:
```
date = pd.to_datetime("21 Feb 2023")
print(date)
```

```
2023-02-21 00:00:00
```

What is the problem below?

[43]:
```
date = pd.to_datetime("11/10/2023")
print(date)
```

```
2023-11-10 00:00:00
```

[44]:
```
print(pd.to_datetime('11/10/2023', format = '%d/%m/%Y'))
```

```
2023-10-11 00:00:00
```

pd.to_timedelta

Use to_timedelta to add or subtract time from a datetime object:

[45]:
```
# Date after 1 day
print(date_pandas + pd.to_timedelta(1,unit='D'))
```

```
2024-02-22 00:00:00
```

[46]:
```
# Date after 12 hours
print(date_pandas + pd.to_timedelta(12,unit='h'))
```

```
2024-02-21 12:00:00
```

[47]:
```
# Date after 6 weeks
print(date_pandas + pd.to_timedelta(6,unit='W'))
```

```
2024-04-03 00:00:00
```

2.3 pd.date_range

date_range gives all dates in the specified range eg. pd.date_range(start='2020-11-10', end='2020-12-25', freq='D') gives every date between start and end inclusive:

[48]:
```
pd.date_range(start='2020-11-10', end='2020-12-25', freq='D')
```

[48]:
```
DatetimeIndex(['2020-11-10', '2020-11-11', '2020-11-12', '2020-11-13',
               '2020-11-14', '2020-11-15', '2020-11-16', '2020-11-17',
               '2020-11-18', '2020-11-19', '2020-11-20', '2020-11-21',
               '2020-11-22', '2020-11-23', '2020-11-24', '2020-11-25',
               '2020-11-26', '2020-11-27', '2020-11-28', '2020-11-29',
               '2020-11-30', '2020-12-01', '2020-12-02', '2020-12-03',
               '2020-12-04', '2020-12-05', '2020-12-06', '2020-12-07',
               '2020-12-08', '2020-12-09', '2020-12-10', '2020-12-11',
               '2020-12-12', '2020-12-13', '2020-12-14', '2020-12-15',
               '2020-12-16', '2020-12-17', '2020-12-18', '2020-12-19',
               '2020-12-20', '2020-12-21', '2020-12-22', '2020-12-23',
               '2020-12-24', '2020-12-25'],
              dtype='datetime64[ns]', freq='D')
```

[49]:
```
pd.date_range(start='2020-11-10', end='2021-12-25', freq='M') # final date of␣
↪every month between start and end.
```

[49]:
```
DatetimeIndex(['2020-11-30', '2020-12-31', '2021-01-31', '2021-02-28',
               '2021-03-31', '2021-04-30', '2021-05-31', '2021-06-30',
               '2021-07-31', '2021-08-31', '2021-09-30', '2021-10-31',
               '2021-11-30'],
              dtype='datetime64[ns]', freq='M')
```

[50]:
```
pd.date_range(start='2020-11-10', periods=20, freq='D') # 20 consecutive dates␣
↪beginning at start.
```

[50]:
```
DatetimeIndex(['2020-11-10', '2020-11-11', '2020-11-12', '2020-11-13',
               '2020-11-14', '2020-11-15', '2020-11-16', '2020-11-17',
               '2020-11-18', '2020-11-19', '2020-11-20', '2020-11-21',
               '2020-11-22', '2020-11-23', '2020-11-24', '2020-11-25',
               '2020-11-26', '2020-11-27', '2020-11-28', '2020-11-29'],
              dtype='datetime64[ns]', freq='D')
```

[51]:
```
# 20 consecutive datetimes going up in minutes from start (2020-11-10 00:00:00)
pd.date_range(start='2020-11-10', periods=20, freq='T')
```

[51]:
```
DatetimeIndex(['2020-11-10 00:00:00', '2020-11-10 00:01:00',
               '2020-11-10 00:02:00', '2020-11-10 00:03:00',
               '2020-11-10 00:04:00', '2020-11-10 00:05:00',
               '2020-11-10 00:06:00', '2020-11-10 00:07:00',
               '2020-11-10 00:08:00', '2020-11-10 00:09:00',
               '2020-11-10 00:10:00', '2020-11-10 00:11:00',
               '2020-11-10 00:12:00', '2020-11-10 00:13:00',
               '2020-11-10 00:14:00', '2020-11-10 00:15:00',
               '2020-11-10 00:16:00', '2020-11-10 00:17:00',
               '2020-11-10 00:18:00', '2020-11-10 00:19:00'],
              dtype='datetime64[ns]', freq='T')
```

2.4 Extracting the elements from a date column in a data frame

[52]:
```
df= pd.DataFrame()

print(df)

df['Start_Date'] = pd.date_range(start='2020-11-10', end='2020-12-25', freq='D')

df['End_Date'] = pd.date_range(start='2020-11-30', periods = 46, freq='M')

df
```

```
Empty DataFrame
Columns: []
Index: []
```

[52]:

	Start_Date	End_Date
0	2020-11-10	2020-11-30
1	2020-11-11	2020-12-31
2	2020-11-12	2021-01-31
3	2020-11-13	2021-02-28
4	2020-11-14	2021-03-31
5	2020-11-15	2021-04-30
6	2020-11-16	2021-05-31
7	2020-11-17	2021-06-30
8	2020-11-18	2021-07-31
9	2020-11-19	2021-08-31
10	2020-11-20	2021-09-30
11	2020-11-21	2021-10-31
12	2020-11-22	2021-11-30
13	2020-11-23	2021-12-31
14	2020-11-24	2022-01-31
15	2020-11-25	2022-02-28
16	2020-11-26	2022-03-31
17	2020-11-27	2022-04-30
18	2020-11-28	2022-05-31
19	2020-11-29	2022-06-30
20	2020-11-30	2022-07-31
21	2020-12-01	2022-08-31
22	2020-12-02	2022-09-30
23	2020-12-03	2022-10-31
24	2020-12-04	2022-11-30
25	2020-12-05	2022-12-31
26	2020-12-06	2023-01-31
27	2020-12-07	2023-02-28
28	2020-12-08	2023-03-31
29	2020-12-09	2023-04-30
30	2020-12-10	2023-05-31
31	2020-12-11	2023-06-30
32	2020-12-12	2023-07-31
33	2020-12-13	2023-08-31
34	2020-12-14	2023-09-30
35	2020-12-15	2023-10-31
36	2020-12-16	2023-11-30
37	2020-12-17	2023-12-31
38	2020-12-18	2024-01-31
39	2020-12-19	2024-02-29
40	2020-12-20	2024-03-31
41	2020-12-21	2024-04-30
42	2020-12-22	2024-05-31
43	2020-12-23	2024-06-30
44	2020-12-24	2024-07-31
45	2020-12-25	2024-08-31

[53]:
```
df['Day'] = df['Start_Date'].dt.day

df['Month'] = df['Start_Date'].dt.month

df['Year'] = df['Start_Date'].dt.year

df
```

[53]:

	Start_Date	End_Date	Day	Month	Year
0	2020-11-10	2020-11-30	10	11	2020
1	2020-11-11	2020-12-31	11	11	2020
2	2020-11-12	2021-01-31	12	11	2020
3	2020-11-13	2021-02-28	13	11	2020
4	2020-11-14	2021-03-31	14	11	2020
5	2020-11-15	2021-04-30	15	11	2020
6	2020-11-16	2021-05-31	16	11	2020
7	2020-11-17	2021-06-30	17	11	2020
8	2020-11-18	2021-07-31	18	11	2020
9	2020-11-19	2021-08-31	19	11	2020
10	2020-11-20	2021-09-30	20	11	2020
11	2020-11-21	2021-10-31	21	11	2020
12	2020-11-22	2021-11-30	22	11	2020
13	2020-11-23	2021-12-31	23	11	2020
14	2020-11-24	2022-01-31	24	11	2020
15	2020-11-25	2022-02-28	25	11	2020
16	2020-11-26	2022-03-31	26	11	2020
17	2020-11-27	2022-04-30	27	11	2020
18	2020-11-28	2022-05-31	28	11	2020
19	2020-11-29	2022-06-30	29	11	2020
20	2020-11-30	2022-07-31	30	11	2020
21	2020-12-01	2022-08-31	1	12	2020
22	2020-12-02	2022-09-30	2	12	2020
23	2020-12-03	2022-10-31	3	12	2020
24	2020-12-04	2022-11-30	4	12	2020
25	2020-12-05	2022-12-31	5	12	2020
26	2020-12-06	2023-01-31	6	12	2020
27	2020-12-07	2023-02-28	7	12	2020
28	2020-12-08	2023-03-31	8	12	2020
29	2020-12-09	2023-04-30	9	12	2020
30	2020-12-10	2023-05-31	10	12	2020
31	2020-12-11	2023-06-30	11	12	2020
32	2020-12-12	2023-07-31	12	12	2020
33	2020-12-13	2023-08-31	13	12	2020
34	2020-12-14	2023-09-30	14	12	2020
35	2020-12-15	2023-10-31	15	12	2020
36	2020-12-16	2023-11-30	16	12	2020
37	2020-12-17	2023-12-31	17	12	2020

```
38 2020-12-18 2024-01-31   18     12  2020
39 2020-12-19 2024-02-29   19     12  2020
40 2020-12-20 2024-03-31   20     12  2020
41 2020-12-21 2024-04-30   21     12  2020
42 2020-12-22 2024-05-31   22     12  2020
43 2020-12-23 2024-06-30   23     12  2020
44 2020-12-24 2024-07-31   24     12  2020
45 2020-12-25 2024-08-31   25     12  2020
```

[54]:
```
df['Start_hour'] = df['Start_Date'].dt.hour

df['Start_minute'] = df['Start_Date'].dt.minute

df['Start_second'] = df['Start_Date'].dt.second

df
```

[54]:

	Start_Date	End_Date	Day	Month	Year	Start_hour	Start_minute	\
0	2020-11-10	2020-11-30	10	11	2020	0	0	
1	2020-11-11	2020-12-31	11	11	2020	0	0	
2	2020-11-12	2021-01-31	12	11	2020	0	0	
3	2020-11-13	2021-02-28	13	11	2020	0	0	
4	2020-11-14	2021-03-31	14	11	2020	0	0	
5	2020-11-15	2021-04-30	15	11	2020	0	0	
6	2020-11-16	2021-05-31	16	11	2020	0	0	
7	2020-11-17	2021-06-30	17	11	2020	0	0	
8	2020-11-18	2021-07-31	18	11	2020	0	0	
9	2020-11-19	2021-08-31	19	11	2020	0	0	
10	2020-11-20	2021-09-30	20	11	2020	0	0	
11	2020-11-21	2021-10-31	21	11	2020	0	0	
12	2020-11-22	2021-11-30	22	11	2020	0	0	
13	2020-11-23	2021-12-31	23	11	2020	0	0	
14	2020-11-24	2022-01-31	24	11	2020	0	0	
15	2020-11-25	2022-02-28	25	11	2020	0	0	
16	2020-11-26	2022-03-31	26	11	2020	0	0	
17	2020-11-27	2022-04-30	27	11	2020	0	0	
18	2020-11-28	2022-05-31	28	11	2020	0	0	
19	2020-11-29	2022-06-30	29	11	2020	0	0	
20	2020-11-30	2022-07-31	30	11	2020	0	0	
21	2020-12-01	2022-08-31	1	12	2020	0	0	
22	2020-12-02	2022-09-30	2	12	2020	0	0	
23	2020-12-03	2022-10-31	3	12	2020	0	0	
24	2020-12-04	2022-11-30	4	12	2020	0	0	
25	2020-12-05	2022-12-31	5	12	2020	0	0	
26	2020-12-06	2023-01-31	6	12	2020	0	0	
27	2020-12-07	2023-02-28	7	12	2020	0	0	
28	2020-12-08	2023-03-31	8	12	2020	0	0	

29	2020-12-09	2023-04-30	9	12	2020	0	0
30	2020-12-10	2023-05-31	10	12	2020	0	0
31	2020-12-11	2023-06-30	11	12	2020	0	0
32	2020-12-12	2023-07-31	12	12	2020	0	0
33	2020-12-13	2023-08-31	13	12	2020	0	0
34	2020-12-14	2023-09-30	14	12	2020	0	0
35	2020-12-15	2023-10-31	15	12	2020	0	0
36	2020-12-16	2023-11-30	16	12	2020	0	0
37	2020-12-17	2023-12-31	17	12	2020	0	0
38	2020-12-18	2024-01-31	18	12	2020	0	0
39	2020-12-19	2024-02-29	19	12	2020	0	0
40	2020-12-20	2024-03-31	20	12	2020	0	0
41	2020-12-21	2024-04-30	21	12	2020	0	0
42	2020-12-22	2024-05-31	22	12	2020	0	0
43	2020-12-23	2024-06-30	23	12	2020	0	0
44	2020-12-24	2024-07-31	24	12	2020	0	0
45	2020-12-25	2024-08-31	25	12	2020	0	0

	Start_second
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0

```
28              0
29              0
30              0
31              0
32              0
33              0
34              0
35              0
36              0
37              0
38              0
39              0
40              0
41              0
42              0
43              0
44              0
45              0
```

Weekday: Monday is 0 and Sunday is 6.

[55]:
```
df['Start_weekday'] = df['Start_Date'].dt.weekday

df['Start_dayofyear'] = df['Start_Date'].dt.dayofyear

df['Start_week_of_year'] = df['Start_Date'].dt.isocalendar().week

df['Duration'] = df['End_Date'] - df['Start_Date']

df
```

[55]:

	Start_Date	End_Date	Day	Month	Year	Start_hour	Start_minute	\
0	2020-11-10	2020-11-30	10	11	2020	0	0	
1	2020-11-11	2020-12-31	11	11	2020	0	0	
2	2020-11-12	2021-01-31	12	11	2020	0	0	
3	2020-11-13	2021-02-28	13	11	2020	0	0	
4	2020-11-14	2021-03-31	14	11	2020	0	0	
5	2020-11-15	2021-04-30	15	11	2020	0	0	
6	2020-11-16	2021-05-31	16	11	2020	0	0	
7	2020-11-17	2021-06-30	17	11	2020	0	0	
8	2020-11-18	2021-07-31	18	11	2020	0	0	
9	2020-11-19	2021-08-31	19	11	2020	0	0	
10	2020-11-20	2021-09-30	20	11	2020	0	0	
11	2020-11-21	2021-10-31	21	11	2020	0	0	
12	2020-11-22	2021-11-30	22	11	2020	0	0	
13	2020-11-23	2021-12-31	23	11	2020	0	0	
14	2020-11-24	2022-01-31	24	11	2020	0	0	
15	2020-11-25	2022-02-28	25	11	2020	0	0	

16	2020-11-26	2022-03-31	26	11	2020	0	0
17	2020-11-27	2022-04-30	27	11	2020	0	0
18	2020-11-28	2022-05-31	28	11	2020	0	0
19	2020-11-29	2022-06-30	29	11	2020	0	0
20	2020-11-30	2022-07-31	30	11	2020	0	0
21	2020-12-01	2022-08-31	1	12	2020	0	0
22	2020-12-02	2022-09-30	2	12	2020	0	0
23	2020-12-03	2022-10-31	3	12	2020	0	0
24	2020-12-04	2022-11-30	4	12	2020	0	0
25	2020-12-05	2022-12-31	5	12	2020	0	0
26	2020-12-06	2023-01-31	6	12	2020	0	0
27	2020-12-07	2023-02-28	7	12	2020	0	0
28	2020-12-08	2023-03-31	8	12	2020	0	0
29	2020-12-09	2023-04-30	9	12	2020	0	0
30	2020-12-10	2023-05-31	10	12	2020	0	0
31	2020-12-11	2023-06-30	11	12	2020	0	0
32	2020-12-12	2023-07-31	12	12	2020	0	0
33	2020-12-13	2023-08-31	13	12	2020	0	0
34	2020-12-14	2023-09-30	14	12	2020	0	0
35	2020-12-15	2023-10-31	15	12	2020	0	0
36	2020-12-16	2023-11-30	16	12	2020	0	0
37	2020-12-17	2023-12-31	17	12	2020	0	0
38	2020-12-18	2024-01-31	18	12	2020	0	0
39	2020-12-19	2024-02-29	19	12	2020	0	0
40	2020-12-20	2024-03-31	20	12	2020	0	0
41	2020-12-21	2024-04-30	21	12	2020	0	0
42	2020-12-22	2024-05-31	22	12	2020	0	0
43	2020-12-23	2024-06-30	23	12	2020	0	0
44	2020-12-24	2024-07-31	24	12	2020	0	0
45	2020-12-25	2024-08-31	25	12	2020	0	0

	Start_second	Start_weekday	Start_dayofyear	Start_week_of_year	Duration
0	0	1	315	46	20 days
1	0	2	316	46	50 days
2	0	3	317	46	80 days
3	0	4	318	46	107 days
4	0	5	319	46	137 days
5	0	6	320	46	166 days
6	0	0	321	47	196 days
7	0	1	322	47	225 days
8	0	2	323	47	255 days
9	0	3	324	47	285 days
10	0	4	325	47	314 days
11	0	5	326	47	344 days
12	0	6	327	47	373 days
13	0	0	328	48	403 days
14	0	1	329	48	433 days

15	0	2	330	48	460 days
16	0	3	331	48	490 days
17	0	4	332	48	519 days
18	0	5	333	48	549 days
19	0	6	334	48	578 days
20	0	0	335	49	608 days
21	0	1	336	49	638 days
22	0	2	337	49	667 days
23	0	3	338	49	697 days
24	0	4	339	49	726 days
25	0	5	340	49	756 days
26	0	6	341	49	786 days
27	0	0	342	50	813 days
28	0	1	343	50	843 days
29	0	2	344	50	872 days
30	0	3	345	50	902 days
31	0	4	346	50	931 days
32	0	5	347	50	961 days
33	0	6	348	50	991 days
34	0	0	349	51	1020 days
35	0	1	350	51	1050 days
36	0	2	351	51	1079 days
37	0	3	352	51	1109 days
38	0	4	353	51	1139 days
39	0	5	354	51	1167 days
40	0	6	355	51	1197 days
41	0	0	356	52	1226 days
42	0	1	357	52	1256 days
43	0	2	358	52	1285 days
44	0	3	359	52	1315 days
45	0	4	360	52	1345 days

2.5 Using pd.to_timedelta to change the units

[56]:
```
# Get duration in days by dividing by 1 day
df['Duration_days'] = df['Duration']/pd.to_timedelta(1, unit='D')

# Get duration in hours by dividing by 1 hour
df['Duration_hours'] = df['Duration']/pd.to_timedelta(1, unit='h')

# Get duration in minutes by dividing by 1 minute
df['Duration_minutes'] = df['Duration']/pd.to_timedelta(1, unit='m')

# Get duration in seconds by dividing by 1 second
df['Duration_seconds'] = df['Duration']/pd.to_timedelta(1, unit='s')

df.iloc[0:10, 11:]
```

[56]:

	Duration	Duration_days	Duration_hours	Duration_minutes	Duration_seconds
0	20 days	20.0	480.0	28800.0	1728000.0
1	50 days	50.0	1200.0	72000.0	4320000.0
2	80 days	80.0	1920.0	115200.0	6912000.0
3	107 days	107.0	2568.0	154080.0	9244800.0
4	137 days	137.0	3288.0	197280.0	11836800.0
5	166 days	166.0	3984.0	239040.0	14342400.0
6	196 days	196.0	4704.0	282240.0	16934400.0
7	225 days	225.0	5400.0	324000.0	19440000.0
8	255 days	255.0	6120.0	367200.0	22032000.0
9	285 days	285.0	6840.0	410400.0	24624000.0

[ ]: