Introduction

Handling date, time and delays can be required in a scientific progress, especially in an experimental work. For instance:

- schedule data acquisition
- handle experimental databases

Python comes with the `datetime` library to address these issues. Scientific oriented libraries such as `numpy` and `pandas` have a different, **but compatible**, implementation of date/time management.

Timestamp

A timestamp is an accurate description of a moment in time.

`date`

The `date` library (built in) is used to describe accurately a date: year, month, day of month.

In [1]:
```python
from datetime import date
d1 = date(2023, 3, 1)
print(d1.day, d1.month, d1.year)
```

```
1 3 2023
```

The current date is obtained using `date.today()`.

In [2]:
```python
d2 = date.today()
print(d2.day, d2.month, d2.year)
```

```
20 11 2024
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

As many Python objects, dates can be **compared**:

```
In [3]:
if d1 < d2:
    print(f"{d1} is anterior of {d2}")
else:
    print(f"{d1} is posterior of {d2}")
```

```
2023-03-01 is anterior of 2024-11-20
```

Yet, there is no meaning to do the sum of two dates:

```
In [4]:
d1 + d2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 d1 + d2

TypeError: unsupported operand type(s) for +: 'datetime.date' and 'datetime.
date'
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

A `datetime` instance comes with its own representation:

```
In [5]: d1
```

```
Out[5]:   datetime.date(2023, 3, 1)
```

To get a string representation, one must use `date.strftime`, i.e. 'string from time'. This method takes as an argument the wanted **format**. This format must be specified following the special characters described here.

```
In [6]: print(d1.strftime("%d %B, %Y (%A)"))     # custom representation
        print(d1.strftime("%x"))                # official representation for your country
```

```
01 March, 2023 (Wednesday)
03/01/23
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Without using `strftime`, the previously introduced formatting methods (using `f'{var}'`) can be used:

```
In [7]: print(f"The event happened on the {d1:%d}th of {d1:%B} {d1:%Y}.")
```

```
The event happened on the 01th of March 2023.
```

`time`

Following the same idea, python can describe an exact hour: from hour to microseconds:

```
In [8]:
from datetime import time
t1 = time(13, 34, 28, microsecond=156545)
print(t1)
print(t1.second)
```

```
13:34:28.156545
28
```

Comparison of `time` instances is also possible:

```
In [9]: t2 = time(11, 14, 54)
        print(t2.microsecond)
        print(t2 < t1)
```

```
0
True
```

But won't work between date and time instances:

```
In [10]: d1 < t2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 d1 < t2

TypeError: '<' not supported between instances of 'datetime.date' and 'datet
ime.time'
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Formatting is possible too:

```
In [11]: print(t2.strftime('%I:%M %p, %S seconds'))
         print(f'It is currently {t2:%I}:{t2:%M} {t2:%p} (and {t2:%S} seconds)')
```

```
11:14 AM, 54 seconds
It is currently 11:14 AM (and 54 seconds)
```

`datetime`

`datetime` objects bring together the functionalities of both `date` and `time` objects (still with a microsecond resolution). Beware of not mistaking the `datetime` module (the one of `date` and `time`) and its submodule `datetime` (siblings of `date` and `time`).

In [12]:
```python
from datetime import datetime

now = datetime.now()
print(now.strftime("%H:%M:%S:%f in %B %Y"))
```

```
15:37:26:385717 in November 2024
```

Feature: a datetime instance can be built from a `str` using the `strptime` function (which is **not** `strftime`):

```
In [13]: var = datetime.strptime("16:50:24:194724 in January 2029", "%H:%M:%S:%f in %B %Y")
         var
```

```
Out[13]: datetime.datetime(2029, 1, 1, 16, 50, 24, 194724)
```

Note that there are some limits to the creation of dates:

```
In [14]: from datetime import MINYEAR, MAXYEAR
         print(MINYEAR, MAXYEAR)
```

```
1 9999
```

Thus, be careful when your experimental data acquisition may last longer than 8000 years.

Time periods: `timedelta`

A time period describes the temporal length of an event. It can be represented by objects of type `timedelta`:

In [15]:
```
from datetime import timedelta
dt = timedelta(days=1, seconds=2, microseconds=3, milliseconds=4, minutes=5, hours=6,
dt
```

Out[15]:
```
datetime.timedelta(days=50, seconds=21902, microseconds=4003)
```

In [16]:
```
print('Total number of seconds: ', dt.total_seconds())
```

```
Total number of seconds:  4341902.004003
```

Parameters can be **negative**. In the example below, a duration of 55 minutes is equal to a duration of 1 hour minus 5 minutes.

In [17]:
```
dt1 = timedelta(minutes=55)
dt2 = timedelta(hours=1, minutes=-5)
dt1 == dt2
```

Out[17]:
```
True
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

One can substract, sum and even divide these instances:

```
In [18]:
dt1 = timedelta(days=1, seconds=2, microseconds=3, milliseconds=4, minutes=5, hours=6
dt2 = timedelta(days=41, seconds=265, microseconds=123, milliseconds=41, minutes=51,
print('Sum:         ', dt1 + dt2)
print('Difference: ', dt1 - dt2)
ratio = dt2 / dt1
print(f'Division:    {ratio:.2f} ({type(ratio)})')
```

```
Sum:         140 days, 13:00:27.045126
Difference:  -41 days, 23:09:36.962880
Division:    1.80 (<class 'float'>)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Important: a `date` (or `datetime`) instance can be added to a `timedelta` instance to get a new `date` (or `datetime`).

```
In [19]: print(d1, dt1, d1 - dt1, type(d1 - dt1), sep="\n")
```

```
2023-03-01
50 days, 6:05:02.004003
2023-01-10
<class 'datetime.date'>
```

```
In [20]: print(var, dt1, var + dt1, type(var + dt1), sep="\n")
```

```
2029-01-01 16:50:24.194724
50 days, 6:05:02.004003
2029-02-20 22:55:26.198727
<class 'datetime.datetime'>
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js