14.155 - Leaflet - Dates

July 23, 2018

1 Date values in Python

For the representation of date values there is the datetime module in Python (https://docs.python.org/3/library/datetime.html). With it you can represent date values and calculate with them.

```
In [1]: from datetime import datetime
```

With `datetime.now()` you have the possibility to create a date object for the current date:

```
In [2]: now = datetime.now()
        print(now)
```

```
2018-07-08 04:51:41.680081
```

Alternatively, you can enter a specific date:

```
In [3]: day = datetime(2017, 8, 20, 20, 0, 0)
        print(day)
```

```
2017-08-20 20:00:00
```

If you have created such a date object, you can access the year directly via `.year`, for example. So you have direct access to all individual data:

```
In [4]: print(day.year)
        print(day.month)
        print(day.day)
        print(day.hour)
        print(day.minute)
        print(day.second)
```

```
2017
8
20
20
0
0
```

The `.timestamp()` method returns the corresponding Unix timestamp at a given date value. Unix timestamp is simply a number that counts up the seconds since 1970-01-01.

The advantage of a Unix timestamp is that we can store it quite compactly; internally, the computer only needs to store a number to represent a date value.

But we can use it here to calculate the time difference in seconds. :-)

```
In [5]: print(now.timestamp() - day.timestamp())
```

```
27766301.68008089
```

1.0.1 `date` and `time` information

The `datetime` package also provides us with additional classes that we can use to work with dates.

For example, `date` represents a date, `time` a time.

- `datetime`: date + time
- `date`: Date only
- `time`: Only time specification

```
In [6]: from datetime import date, time
```

```
In [7]: d = date(2017, 8, 20)
        print(d)
```

```
2017-08-20
```

```
In [8]: t = time(20, 1, 4)
        print(t)
```

```
20:01:04
```

Of course you can also compare date values.

But beware! For `date`, `==` is fulfilled on the same date; for `datetime` objects, both the date and time must of course match.

In detail, then:

- `datetime`: date + time must match
- `date`: Date must match
- `time`: Time must match

```
In [9]: print(date(2017, 8, 20) == date(2017, 8, 20))
        print(datetime(2017, 8, 20, 20, 0, 0) == datetime(2017, 8, 20, 15, 0, 0))
```

```
True
False
```

1.0.2 Convert `datetime` to `date` and `time`

Of course, you can also split a `datetime` object into a `date` and a `time` object:

```
In [10]: dt = datetime(2017, 8, 20, 20, 0, 0)
         print(dt.time())
         print(dt.date())

20:00:00
2017-08-20
```

1.0.3 Convert `date` and `time` to `datetime`

And of course the other way around: :-)

```
In [11]: print(datetime.combine(date(2017, 8, 20), time(20, 30, 0)))

2017-08-20 20:30:00
```

2 Output date values

Of course it is also important that we want to output a formatted date. After all, we don't always want an output in the form `2017-08-20`.

Python offers various format options for this purpose.

Documentation: https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior

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

         now = datetime.now()

In [13]: print(now)

2018-07-08 04:56:50.320717


In [14]: print(now.strftime("%d.%m.%Y"))
         print(now.strftime("%Y-%m-%d"))
         print(now.strftime("%Y%m%d"))

08.07.2018
2018-07-08
20180708
```

2.0.1 Read date values

The other way around: You can also extract date values from a string, for example, if you want to use the Python functions to calculate the date later:

```
In [15]: d = "18.07.2017"

In [16]: print(datetime.strptime(d, "%d.%m.%Y"))

2017-07-18 00:00:00
```

3 Time differences

With an `timedelta` you have the possibility to calculate with time differences. An `timedelta` simply describes the time difference between two date values.

For example, you can add an `timedelta` to a date value to get to the new date:

```
In [17]: from datetime import datetime, timedelta
         now = datetime.now()
         print(now)
         print(now + timedelta(days = 20, hours = 4, minutes = 3, seconds = 1))

2018-07-08 04:57:26.960186
2018-07-28 09:00:27.960186
```

Also, if you subtract two dates from each other, the result is an `timedelta` object:

```
In [19]: day = datetime(2017, 8, 20)
         td = now - day
         print(td)

322 days, 4:57:26.960186


In [20]: print(datetime(2018, 1, 1) + td)

2018-11-19 04:57:26.960186
```