# Theory: Datetime module



11 minutes 0 / 5 problems solved



## §1. datetime module

Often, in our projects, we need to work with dates and time. For example, we may want to track the current date and time or see how long our code runs. For these purposes, we can use the `datetime` module.

The `datetime` module has several classes that make working with time easy:

- `datetime.date` represents standard date;
- `datetime.time` represents standard time, independent from the date;
- `datetime.timedelta` represents the difference between two points in time;
- `datetime.tzinfo` represents timezones;
- `datetime.datetime` represents both time and date together.

In this topic, we'll focus on the `datetime.datetime` class.

## §2. datetime.datetime

The `datetime.datetime` class is a sort of combination of the `date` and `time` classes. Similarly to those two, it assumes the current Gregorian calendar and that there are exactly 86,400 seconds in each day.

The constructor of the `datetime.datetime` objects takes the following parameters:

```
import datetime

# necessary parameters

datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None)
```

The `year`, `month` and `day` parameters are required, others are optional. All arguments (except `tzinfo`) should be integers and just like in real life, their values are restricted:

- `datetime.MINYEAR` (1) ≤ `year` ≤ `datetime.MAXYEAR` (9999);
- 1 ≤ `month` ≤ 12;
- 1 ≤ `day` ≤ number of days in this month and year;
- 0 ≤ `hour` < 24;
- 0 ≤ `minute` < 60;
- 0 ≤ `second` < 60;
- 0 ≤ `microsecond` < 1,000,000.

The `tzinfo` argument can be an instance of the `datetime.tzinfo` class, but its default value is `None`, so we don't need to worry about it here.

To see how this all works, let's create a `datetime.datetime` object. For example, the date and time of the first human going to space which took place on April 12, 1961, at 6:07 UTC:

```
import datetime

vostok_1 = datetime.datetime(1961, 4, 12, 6, 7)
print(vostok_1)  # 1961-04-12 06:07:00
```

## §3. datetime methods

The `datetime.datetime` class has several very handy methods.

If you need to get the current time and date, there are two methods you can use: `datetime.datetime.today()` and `datetime.datetime.now(tz=None)`. They are very similar and the only difference between these two methods is that

Current topic:

Datetime module Stage 2

Topic depends on:

- ✓ Load module 5 Stage 1
- ✓ Class instances 3 Stage 1

Topic is required for:

- Datetime parsing and formatting
- Django template language Stage 2
- Django ORM

Table of contents:

`datetime.datetime.now()` has a keyword argument `tz`. If you don't specify it, the two methods work the same. However, in some cases or on some platforms, the `datetime.datetime.now()` method may be more precise.

This is an example of how they perform:

```
print(datetime.datetime.now())  # 2019-09-12 17:18:23.620734
print(datetime.datetime.today())  # 2019-09-12 17:18:23.625716
```

You can also transform a `datetime.datetime` object to a `datetime.time` or `datetime.date` objects using `datetime.datetime.time()` or `datetime.datetime.date()` methods respectively:

```
print(vostok_1.time())  # 06:07:00
print(vostok_1.date())  # 1961-04-12
```

Those were just a couple of methods available in the `datetime.datetime` class. There are many more: the ones that deal with timestamps or timezones or the ones that help parse and convert datetime objects. Don't worry, you'll have a chance to work with them in the next topics!

Report a typo

**195** users liked this theory. **1** didn't like it. **What about you?**

Start practicing

Comments (4) Hints (0) Useful links (0) Show discussion