LECTURE 37

DATES AND TIMES

MCS 260 Fall 2021

David Dumas

REMINDERS

- Project 3 solution posted, grades soon
- Homework 13 coming tomorrow, due Tue 23 Nov
- Homework 14, due Tue 30 Nov, will be the last one
- No labs next week (23, 25 Nov)
- Lab times on 23 Nov become TA office hours
- Kylash also has extra office hours next week
- No synchronous lecture on Wed 24 Nov
- Worksheet 15 and lab 15 will happen!

TIME

Python's `time` module can tell you the current *timestamp*, i.e. the time in second since a certain base point, the *epoch*. It can also do some other things.

The epoch is usually 0:00 on January 1, 1970 (GMT), but officially it can be different in each Python installation.

- `time.time()` — return current timestamp (float).
- `time.gmtime(0)` — return some data about the epoch for this Python installation.
- `time.sleep(seconds)` — pause execution for `seconds` seconds.

TIME

The main thing I think the `time` module is good for is measuring the *elapsed time between two events*, e.g.

```
import time

t0 = time.time()
for x in huge_list:
    complicated_function(x)
t1 = time.time()

print("That took {:.2f} seconds".format(t1-t0))
```

Keep in mind this measures "wall clock" time, not the total CPU time spent doing actual work.

DATETIME

Module includes class `datetime.datetime` for representing a Gregorian calendar time as month, day, hour, minute, second, microsecond.

- `datetime.datetime.now()` — The current local time (as reported by the OS)
- `datetime.datetime.utcnow()` — The current time in UTC (equal to GMT)
- `datetime.datetime(2021,8,23,10,3,27)` — object representing 27 seconds after 10:03am on Aug 23, 2021

There are also `datetime.date` objects, representing dates in the Gregorian calendar, and `datetime.time` objects, representing a time of day.

These have similar behavior, so we will focus on `datetime.datetime`.

TIME ZONE HANDLING

Everything we've covered so far uses *naive* datetime objects, not labeled by a specific time zone.

Real world applications typically need to account for time zones (and their complexity).

We won't cover this in MCS 260 but I want to suggest:

- If you need to work with time zones in Python, know that built-in support for this is limited.
- There are a number of add-on modules that can make time zone handling easier.

STRING TO DATETIME

The datetime module can take a string and convert it to a datetime object, which is one of its most powerful features.

- `datetime.datetime.strptime(date_string,format)` — Convert a string to a datetime, assuming it uses the format described in `format` (`%`-codes indicate datetime parts).
- `datetime.datetime.strptime("2021-12-03","%Y-%m-%d")` — Parse a year-month-day string and make a datetime object out of it.

Format codes for `strptime` include (see full list):

- `%Y` = year
- `%m` = month (two digit)
- `%B` = full month name
- `%d` = day (two digit)
- `%A` = weekday name (e.g. Friday)
- `%a` = weekday abbreviation (e.g. Fri)
- `%H` = hour (two digit, 24 hour format)
- `%I` = hour (two digit, 12 hour format)
- `%M` = minute (two digit)
- `%S` = second
- `%p` = AM/PM

DATETIME TO STRING

If `dt` is a datetime object:

- `dt.strftime(format)` — converts `dt` to a string in the given format.

DATETIME ⇆ TIMESTAMP

If `dt` is a datetime object:

- `datetime.datetime.fromtimestamp(ts)` — Convert from a timestamp to a local date and time
- `dt.timestamp()` — Convert from datetime to a timestamp

COMPARISON

For datetime objects, the comparison operator < means "is earlier in time than".

```
datetime.datetime(1999,11,19) < datetime.datetime.now() # True
```

TIMEDELTA

Subtracting two datetime objects gives a `datetime.timedelta` object.

- `datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)` — build a new timedelta object
- `delta.total_seconds()` — convert an existing timedelta object to units of seconds

Internally, timedelta stores days, seconds, and microseconds. It supports division by other timedelta objects, and multiplication/division by numbers.

RECOMMENDATIONS

- For past events, store timestamp or UTC datetime
- Convert to a datetime object when displaying
- For future events, it's really complicated! (e.g. what if time zone rules change between now and then?)

DATEUTIL

`dateutil` is another module not in the standard library that is often used for handling dates and times in Python.

(Ask pip to install `python-dateutil`.)

A nice feature of `dateutil` is that it has a function `dateutil.parser.parse(s)` to make a "best guess" at the meaning of a date string `s` of unknown format.

Not required in MCS 260.

REFERENCES

- datetime module official docs
- pytz docs
- dateutil docs

REVISION HISTORY

- 2021-11-17 Initial publication
- 2021-11-17 Corrected typos