LECTURE 37

DATES AND TIMES

MCS 260 Fall 2020

Emily Dumas

REMINDERS

- Work on Worksheet 13
- Quiz 13 will be posted tomorrow

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).

- `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.

(The `time` module has many other functions.)

DATETIME

Primary module for working with dates and times. The main class is `datetime.datetime` representing a date and time (Gregorian calendar) broken into year, 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)

These return "naive" datetimes; no time zone information is attached.

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`.

Datetime from string:

- `datetime.datetime.strptime(date_string,format)` — Convert a string to a datetime, assuming it uses the format described in `format` (`%`-codes indicate datetime parts).

Format codes include (see full list):

- `%Y` = year
- `%m` = month (two digit)
- `%B` = full month name
- `%d` = day (two digit)
- `%H` = hour (two digit, 24 hour)
- `%I` = hour(two digit, 12 hour)
- `%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 to/from 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

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.

TIME ZONE HANDLING

Everything we've covered so far uses naive datetime objects, assuming the OS-reported local time zone when necessary.

Often, this isn't good enough.

But Python's built-in time zone handling is very limited. It can only represent a fixed offset from GMT.

PYTZ

The `pytz` module is one of the ways of working with time zones I recommend.

It is not in the standard library; install it with pip.

In `pytz` you build timezone objects and then *localize* naive datetimes to them (add time zone info), or *normalize* datetimes (convert from one zone to another).

- `pytz.timezone(zone_name)` — Build new time zone object representing a named zone like "US/Eastern"
- `pytz.all_timezones` — List of all recognized time zone names (>500)
- `tzobj.localize(naive_dt)` — Convert a naive datetime to the time zone represented by `tzobj`
- `tzobj.normalize(dt)` — Convert datetime that already contains time zone info to one in the time zone represented by `tzobj`

RECOMMENDATIONS

- For past events, store timestamp or UTC datetime
- Convert to user's preferred time zone when displaying
- For future events, it's complicated! (e.g. what if time zone rules change between now and then?)
  - Generally need to store the local time specification and the user's time zone.

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`.)

Like `pytz` it augments the functionality of `datetime`.

A nice feature of `dateutil` is that it has a function `dateutil.parser.parse(s)` to make a "best

REFERENCES

- datetime module official docs
- pytz docs
- dateutil docs

REVISION HISTORY

- 2020-11-18 Gregorian calendar note; dateutil; more links
- 2020-11-17 Initial publication