Python Tutorial: Datetime Module - How to work with Dates, Times, Timedeltas, and Timezones

https://www.youtube.com/watch?v=eirjjyP2qcQ

There and Back Again a DateTime Journey

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

datacamp

From string to datetime

- The `datetime` module is part of the Python standard library
- Use the `datetime` type from inside the `datetime` module
- `.strptime()` method converts from a string to a `datetime` object

```
from datetime import datetime
print(parking_violations_date)
```

```
06/11/2016
```

Parsing strings into datetimes

```
date_dt = datetime.strptime(parking_violations_date,
                                              '%m/%d/%Y')
print(date_dt)
```

```
2016-06-11 00:00:00
```

Time Format Strings

Directive	Meaning	Example
%d	Day of the month as a zero-padded decimal number.	01, 02, ..., 31
%m	Month as a zero-padded decimal number.	01, 02, ..., 12
%Y	Year with century as a decimal number.	0001, 0002, ..., 2013, 2014, ..., 9998, 9999

Full list available in the Python documentation

Datetime to String

- `.strftime()` method uses a format string to convert a datetime object to a string

```
date_dt.strftime('%m/%d/%Y')
```

```
'06/11/2016'
```

- `isoformat()` method outputs a datetime as an ISO standard string

```
date_dt.isoformat()
```

```
'2016-06-11T00:00:00'
```

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON

Working with Datetime Components and current time

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

datacamp

Datetime Components

- `day`, `month`, `year`, `hour`, `minute`, `second`, and more are available from a datetime instance
- Great for grouping data

```
daily_violations = defaultdict(int)
for violation in parking_violations:
    violation_date = datetime.strptime(violation[4],
                                       '%m/%d/%Y')
    daily_violations[violation_date.day] += 1
```

2. Datetime Components
All the parts of a datetime object are available as attributes, such as day, month, year, hour, minute, second etc. These are often used to group data by a particular time frame. Let's count the NYC parking violations for 2016 and group by day. We'll start by using a defaultdict of ints to count the records by day. Next, I'll loop over the packing violations to and parse the date which is found in the fifth element of our list. Next I increment the appropriate day based on the violation.

Datetime Components - Results

```
print(sorted(daily_violations.items()))
```

```
[(1, 80986), (2, 79831), (3, 74610), (4, 69555),
(5, 68729), (6, 76232),(7, 82477), (8, 72472),
(9, 80415), (10, 75387), (11, 73287), (12, 74614),
(13, 75278), (14, 81803), (15, 79122), (16, 80692),
(17, 73677), (18, 75927), (19, 80813), (20, 80992),
(21, 78138), (22, 81872), (23, 78104), (24, 63490),
(25, 78898), (26, 78830), (27, 80164), (28, 81954),
(29, 80585), (30, 65864), (31, 44125)]
```

What is the deal with now

- `.now()` method returns the current local datetime
- `.utcnow()` method returns the current UTC datetime

```
from datetime import datetime
local_dt = datetime.now()
print(local_dt)
```

```
2017-05-05 12:30:00.740415
```

4. What is the deal with now
Often when working with datetime objects, you'll want to work on windows or ranges that start from the current date and time. We can do this using the datetime now functions. There is a dot now() method on the datetime object in the datetime module and a dot utcnow() method. The dot now() method returns the current local time on the machine on which it is run, and dot utcnow() does the same thing but returns the value in UTC timezone. We'll talk more about timezones in a second. Let's start by importing datetime type. Then we'll call the now method on it and it will return a new datetime representing the current time. Let's print it to see what we got.

What is the deal with utcnow

```
utc_dt = datetime.utcnow()
print(utc_dt)
```

```
2017-05-05 17:30:05.467221
```

5. What is the deal with utcnow
Next, let's call utcnow and get the current time in the UTC timezone. Finally, print it. The UTC timezone is only timezone with this special kind of method. Let's learn a bit more about timezones.

Timezones

- Naive datetime objects have no timezone data
- Aware datetime objects have a timezone
- Timezone data is available via the `pytz` module via the `timezone` object
- Aware objects have `.astimezone()` so you can get the time in another timezone

6. Timezones
In order to work effectively with other timezones, you can use the pytz module and use the timezone names from the Olsen database, the standard for timezone information. An "aware" datetime object has an dot astimezone() method that accepts a timezone object and returns a new datetime object in the desired timezone. If the tzinfo is not set for the datetime object it assumes the timezone of the computer you are working on.

Timezones in action

```
from pytz import timezone
record_dt = datetime.strptime('07/12/2016 04:39PM',
    ...: '%m/%d/%Y %H:%M%p')
ny_tz = timezone('US/Eastern')
a_tz = timezone('US/Pacific')
ny_dt = record_dt.replace(tzinfo=ny_tz)
la_dt = ny_dt.astimezone(la_tz)
```

7. Timezones in action
I'll begin by importing timezone from pytz. I've got the datetime of a violation, and parsed it into a naive datetime object. .. Next, I use the replace method to replace the empty timezone on my record datetime and save it as my datetime. Now that I have an aware datetime instance, I can use the as timezone method to get the records time in LA.

Timezones in action - results

```
print(ny_dt)
```

```
2016-07-12 04:39:00-04:00
```

```
print(la_dt)
```

```
2016-07-12 01:39:00-07:00
```

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON

Time Travel (Adding and Subtracting Time)

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

datacamp

Incrementing through time

- `timedelta` is used to represent an amount of change in time
- Used to add or subtract a set amount of time from a datetime object

```
from datetime import timedelta
flashback = timedelta(days=90)
print(record_dt)
```

```
2016-07-12 04:39:00
```

Adding and subtracting timedeltas

```
print(record_dt - flashback)
```

```
2016-04-13 04:39:00
```

```
print(record_dt + flashback)
```

```
2016-10-10 04:39:00
```

Datetime differences

- Use the `-` operator to calculate the difference
- Returns a timedelta with the difference

```
time_diff = record_dt - record2_dt
type(time_diff)
```

```
datetime.timedelta
```

```
print(time_diff)
```

```
0:00:04
```

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON

HELP! Libraries to make it easier

DATA TYPES FOR DATA SCIENCE IN PYTHON

Jason Myers
Instructor

datacamp

Parsing time with pendulum

- `.parse()` will attempt to convert a string to a pendulum datetime object without the need of the format string

```
import pendulum

occurred = violation[4] + ' ' + violation[5] +'M'

occurred_dt = pendulum.parse(occurred, tz='US/Eastern')

print(occured_dt)
```

```
'2016-06-11T14:38:00-04:00'
```

Timezone hopping with pendulum

- `.in_timezone()` method converts a pendulum time object to a desired timezone.
- `.now()` method accepts a timezone you want to get the current time in

```
print(violation_dts)
```

```
[<Pendulum [2016-06-11T14:38:00-04:00]>,
 <Pendulum [2016-04-25T14:09:00-04:00]>,
 <Pendulum [2016-04-23T07:49:00-04:00]>,
 <Pendulum [2016-04-26T07:09:00-04:00]>,
 <Pendulum [2016-01-04T09:52:00-05:00]>]
```

More timezone hopping

```
for violation_dt in violation_dts:
    print(violation_dt.in_timezone('Asia/Tokyo'))
```

```
2016-06-12T03:38:00+09:00
2016-04-26T03:09:00+09:00
2016-04-23T20:49:00+09:00
2016-04-26T20:09:00+09:00
2016-01-04T23:52:00+09:00
```

```
print(pendulum.now('Asia/Tokyo'))
```

```
<Pendulum [2017-05-06T08:20:40.104160+09:00]>
```

Humanizing differences

- `.in_XXX()` methods provide the difference in a chosen metric
- `.in_words()` provides the difference in a nice expressive form

```
diff = violation_dts[3] - violation_dts[2]
diff
```

```
<Period [2016-04-26T07:09:00-04:00 ->
                 2016-04-23T07:49:00-04:00]>
```

```
print(diff.in_words())
```

```
'2 days 23 hours 20 minutes'
```

More human than human

```
print(diff.in_days())
```

```
2
```

```
print(diff.in_hours())
```

```
71
```

Let's practice!

DATA TYPES FOR DATA SCIENCE IN PYTHON