DATA TYPES FOR DATA SCIENCE

There and Back Again a DateTime Journey

Jason Myers
Instructor

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

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

In [2]: print(parking_violations_date)
06/11/2016

In [3]: date_dt = datetime.strptime(parking_violations_date, '%m/%d/%Y')

In [4]: 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

```
In [1]: date_dt.strftime('%m/%d/%Y')
Out[1]: '06/11/2016'
```

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

```
In [1]: date_dt.isoformat()
Out[1]: '2016-06-11T00:00:00'
```

DATA TYPES FOR DATA SCIENCE

Let's practice!

DATA TYPES FOR DATA SCIENCE

Working with Datetime Components and current time

Jason Myers
Instructor

Datetime Components

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

```
In [1]: daily_violations = defaultdict(int)

In [2]: for violation in parking_violations:
   ...:     violation_date = datetime.strptime(violation[4], '%m/%d/%Y')

   ...:     daily_violations[violation_date.day] += 1

In [3]: 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

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

In [2]: local_dt = datetime.now()

In [3]: print(local_dt)
2017-05-05 12:30:00.740415

In [4]: utc_dt = datetime.utcnow()

In [5]: print(utc_dt)
2017-05-05 17:30:05.467221
```

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

Timezones in action

```
In [1]: from pytz import timezone

In [2]: record_dt = datetime.strptime('07/12/2016 04:39PM',
   ...: '%m/%d/%Y %H:%M%p')

In [3]: ny_tz = timezone('US/Eastern')

In [4]: la_tz = timezone('US/Pacific')

In [5]: ny_dt = record_dt.replace(tzinfo=ny_tz)

In [6]: la_dt = ny_dt.astimezone(la_tz)

In [7]: print(ny_dt)
2016-07-12 04:39:00-04:00

In [8]: print(la_dt)
2016-07-12 01:39:00-07:00
```

DATA TYPES FOR DATA SCIENCE

Let's practice!

DATA TYPES FOR DATA SCIENCE

Time Travel (Adding and Subtracting Time)

Jason Myers
Instructor

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

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

In [2]: flashback = timedelta(days=90)

In [3]: print(record_dt)
2016-07-12 04:39:00

In [4]: print(record_dt - flashback)
2016-04-13 04:39:00

In [5]: 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

```
In [1]: time_diff = record_dt - record2_dt

In [2]: type(time_diff)
Out[2]: datetime.timedelta

In [3]: print(time_diff)
0:00:04
```

DATA TYPES FOR DATA SCIENCE

Let's practice!

DATA TYPES FOR DATA SCIENCE

HELP! Libraries to make it easier

Jason Myers
Instructor

Parsing time with pendulum

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

```
In [1]: import pendulum

In [2]: occurred = violation[4] + ' ' + violation[5] +'M'

In [3]: occurred_dt = pendulum.parse(occurred, tz='US/Eastern')

In [4]: 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

```
In [1]: 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]>]

In [2]: 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

In [3]: 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 expresive form

```
In [1]: diff = violation_dts[3] - violation_dts[2]

In [2]: diff
Out[2]: <Period [2016-04-26T07:09:00-04:00 -> 2016-04-23T07:49:00-04:00]>

In [3]: print(diff.in_words())
'2 days 23 hours 20 minutes'

In [4]: print(diff.in_days())
2

In [5]: print(diff.in_hours())
71
```

DATA TYPES FOR DATA SCIENCE

Let's practice!