Python Data Products
Course 1: Basics

Lecture: Time and date data
Learning objectives

In this lecture we will...
- Consider various **formats and structures** to represent time and date data
- Demonstrate the main **methods** to manipulate time data in python
- **Convert** time and date data between various formats
Dealing with time and date data can be difficult as string-formatted data doesn't admit easy comparison or feature representation:

• Which date occurs first, 4/7/2003 or 3/8/2003?
• How many days between 4/5/2003 - 7/15/2018?
• e.g. how many hours between 2/6/2013 23:02:38 - 2/7/2013 08:32:35?
Time and date data

Most of the data we've seen so far include plain-text time data, that we need to carefully manipulate:

```python
{'business_id': 'FYWN1wneV18bWNgQjJ2GNg', 'attributes':
{'BusinessAcceptsCreditCards': True, 'AcceptsInsurance': True, 'ByAppointmentOnly': True},
'longitude': -111.9785992, 'state': 'AZ',
'address': '4855 E Warner Rd, Ste B9',
'neighborhood': '', 'city': 'Ahwatukee',
'hours': {'Tuesday': '7:30-17:00',
'Wednesday': '7:30-17:00', 'Thursday': '7:30-17:00', 'Friday': '7:30-17:00', 'Monday': '7:30-17:00'},
'postal_code': '85044', 'review_count': 22, 'stars': 4.0,
'categories': ['Dentists', 'General Dentistry', 'Health & Medical', 'Oral Surgeons', 'Cosmetic Dentists', 'Orthodontists'], 'is_open': 1, 'name': 'Dental by Design',
'latitude': 33.3306902}
```
In this lecture we'll cover a few functions:

- `Time.strptime`: convert a time string to a structured time object
- `Time.strftime`: convert a time object to a string
- `Time.mktime` / `calendar.timegm`: convert a time object to a number
- `Time.gmtime`: convert a number to a time object
Time and date data

In this lecture we'll cover a few functions:

- **Time string**
  - `strptime`
  - `strftime`

- **Structured time object**
  - `time.struct_time(tm_year=2019, tm_mon=5, tm_mday=28, tm_hour=21, tm_min=36, tm_sec=18, tm_wday=1, tm_yday=148, tm_isdst=-1)`

- **Number**
  - `mktime` / `timegm`
  - `gmtime`
  - `1464418800.0`
Concept: Unix time

Internally, time is often represented as a number, which allows for easy manipulation and arithmetic.

- The value (Unix time) is the number of seconds since Jan 1, 1970 in the UTC timezone.
- So I made this slide at 1532568962 = 2018-07-26 01:36:02 UTC (or 18:36:02 in my timezone).
- But real datasets generally have time as a "human readable" string.
- Our goal here is to convert between these two formats.
First, let's look at converting a string to a structured object (strptime)

Time string: 21:36:18, 28/5/2019

Structured time object:
```
strptime(21:36:18, 28/5/2019)
```

```
te.time.struct_time(tm_year=2019, tm_mon=5, tm_mday=28, tm_hour=21, tm_min=36, tm_sec=18, tm_wday=1, tm_yday=148, tm_isdst=-1)
```
Code: time.strptime()

In [1]:
```python
import time
import calendar
```

String-formatted time data

In [2]:
```python
timeString = "2018-07-26 01:36:02"
```

In [3]:
```python
timeStruct = time.strptime(timeString, "%Y-%m-%d %H:%M:%S")
```

In [4]:
```python
timeStruct
```

```
```

Out[4]:
```python
struct_time(tm_year=2018, tm_mon=7, tm_mday=26, tm_hour=1, tm_min=36, tm_sec=2, tm_wday=3, tm_yday=207, tm_isdst=-1)
```

In [5]:
```python
timeStruct.tm_wday
```

Note: this day is a Wednesday!

Out[5]:
```python
3
```

In [6]:
```python
help(time.strptime)
```

```
Help on built-in function strptime in module time:

```
strptime(...) -> struct_time
```

Parse a string to a time tuple according to a format specification.

Note: different time formatting options in the help page
Strptime is convenient when we want to extract features from data

- E.g. does a date correspond to a weekday or a weekend?
- Converting month names or abbreviations (e.g. "Jan") to month numbers
- Dealing with mixed-format data by converting it to a common format
- But if we want to perform arithmetic on timestamps, converting to a number may be easier
For this we'll use mktime to convert our structured time object to a number:

time.struct_time(tm_year=2019, tm_mon=5, tm_mday=28, tm_hour=21, tm_min=36, tm_sec=18, tm_wday=1, tm_yday=148, tm_isdst=-1)

1464418800.0
• time.mktime() allows us to convert our structured time object to a number
• **NOTE:** mktime assumes the structure is a *local* time whereas timegm assumes the structure is a *UTC* time
• This allows for easy manipulation, arithmetic, and comparison (e.g. sorting) of time data
Finally, both of these operations can be reversed, should we wish to format time data as a string or structure.
These methods can be used to put adjusted times back into string format.
Summary of concepts

• Understand the idea and motivation behind **unix time**
• Understand the methods `strptime`, `strftime`, `mktime`, and `gmtime`
• Be able to convert between various time formats
• Be able to read and manipulate string-formatted time data from real datasets

On your own...

• Try converting the dates in Yelp or Amazon reviews to unix time, and sorting the reviews by their date