Dates, times, and timezones: Python vs the real world

Andrew Neitsch

2017-09-28

Outline

Introduction

Theory

Python

Pendulum

Conclusions

Basic use cases for date and time computation

- What time is it now?
- What time is it in _____?
- What time is it here when it's _____in _____?
- What does the date 9/10/11 mean?
- How long ago did _____ happen?
- Remind me when _____ is about to happen

Introduction

Theory

Python

Pendulum

Conclusions

Dates and times for humans

- Times
- Dates
- Time zones

Times for humans

- Times

  Pretty straightforward

  Partition the day into regular intervals:
- Dates
- Time zones

Times for humans

- Times

  Pretty straightforward

  Partition the day into regular intervals:

  24 hours of 60 minutes of 60 seconds
- Dates
- Time zones

Dates for humans

- Times
- Dates

  Partition the year into *irregular* intervals
- Time zones

Dates for humans

- Times
- Dates

  Partition the year into *irregular* intervals

  12 months of varying lengths

  Pattern depends on year
- Time zones

Dates for humans

- Times

- Dates

  Partition the year into *irregular* intervals

  Root problem:
  365.24220 earth rotations per orbit of sun

  ```
  $ gfactor 365
  365: 5 73
  ```

  Not a convenient number,
  so months of varying lengths

- Time zones

Dates for humans

- Times
- Dates

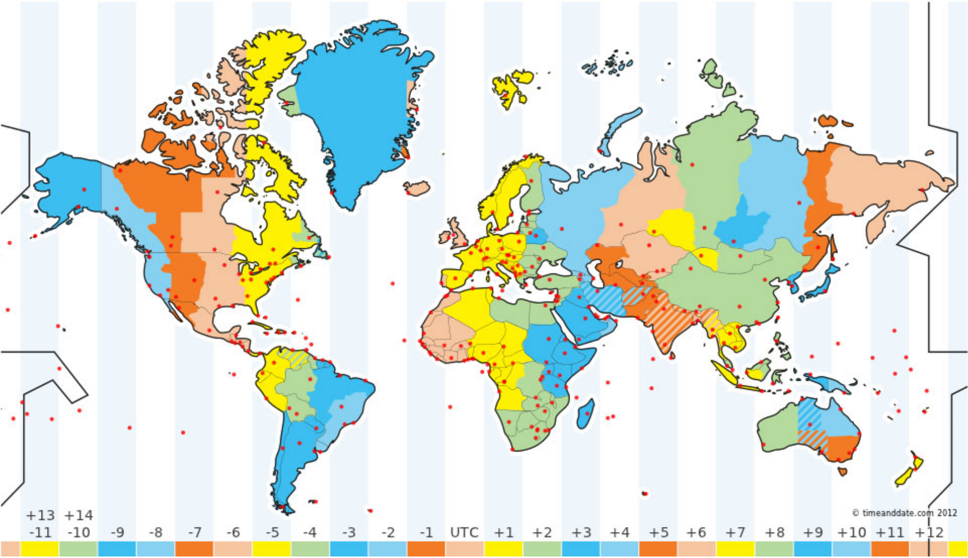
  365.24220 earth rotations per orbit of sun

  Leap days to deal with the fractional part

  ```
  >>> 3 / ((365 + 1/4 - 1/100 + 1/400) - 365.24220)
  10000.000000615424
  ```

- Time zones

Time zones for humans


© timeanddate.com 2012
+13 +14
-11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 UTC +1 +2 +3 +4 +5 +6 +7 +8 +9 +10 +11 +12

Time zones for humans

- Noon is slightly different in different parts of the world

Time zones for humans

- Time zones are a balance between precision of having clocks close to the sun, and the convenience of uniform time over large parts of the earth

Time zones for humans

- Defined by local laws, can change with little or no notice

Time zones for humans

- Defined by local laws, can change with little or no notice
- Defined relative to Universal Coordinated Time: UTC (Similar to historical Greenwich Mean Time)

Time zones for humans

- Defined by local laws, can change with little or no notice
- Defined relative to Universal Coordinated Time: UTC
- Today, Calgary is UTC-0600, which means exactly 6 hours behind UTC

Time zones for humans

- Defined by local laws, can change with little or no notice
- Defined relative to Universal Coordinated Time: UTC
- Today, Calgary is UTC-0600, which means exactly 6 hours behind UTC
- Changes to UTC-0700 at 2:00 a.m. on November 5

Time zones for humans

- Defined by local laws, can change with little or no notice
- Defined relative to Universal Coordinated Time: UTC
- Egypt 2014: Daylight savings transitions in May, June, July, and September

Time zones for humans

- Defined by local laws, can change with little or no notice
- Defined relative to Universal Coordinated Time: UTC
- Not a lot of theory here: just need to be able to look it up

Conclusions for humans

- Times

  Divided into regular intervals, relatively straightforward

- Dates

- Time zones

Conclusions for humans

- Times

  Divided into regular intervals, relatively straightforward

- Dates

  Divided into irregular intervals, complicated but unchanging

- Time zones

Conclusions for humans

- Times

  Divided into regular intervals, relatively straightforward

- Dates

  Divided into irregular intervals, complicated but unchanging

- Time zones

  Super-complicated, arbitrary, subject to change without notice

Conclusions for humans

- Times

  Divided into regular intervals, relatively straightforward

- Dates

  Divided into irregular intervals, complicated but unchanging

- Time zones

  Super-complicated, arbitrary, subject to change without notice

Time and date are meaningless without timezone

Time zones for humans

You can't know how far away a date or time is from any other date or time without knowing what timezones they are referring to

Time zones for humans

You can’t know how far away a date or time is from any other date or time without knowing what timezones they are referring to

Given that time is what keeps everything from happening at once ...

Time zones for humans

You can't know how far away a date or time is from any other date or time without knowing what timezones they are referring to

Given that time is what keeps everything from happening at once ...

That's pretty important.

Dates and times for computers

Computers do not care about what we care about

Dates and times for computers

Computers do not care about what we care about

- Times

  We can just count seconds since some arbitrary point

  Do math by subtracting

- Dates
- Time zones

Dates and times for computers

Computers do not care about what we care about

- Times

  We can just count seconds since some arbitrary point

  Do math by subtracting

  Seconds since start of January 1, 1970, UTC: 1506624282
- Dates
- Time zones

Dates and times for computers

Computers do not care about what we care about

- Times
- Dates
- Time zones

  Only matter for I/O with humans

Theoretical conclusions

Humans and computers have very different needs when it comes to dates and times

Introduction

Theory

Python

Pendulum

Conclusions

Basic use cases for date and time computation

- What time is it now?
- What time is it in ______?
- What time is it here when it's ______in ______?
- What does the date 9/10/11 mean?
- How long ago did ______ happen?
- Remind me when ______ is about to happen

Python's standard library

Create datetime objects and convert them to seconds since the epoch:

Python's standard library

Create datetime objects and convert them to seconds since the epoch:

```
>>> datetime.datetime.utcnow().strftime("%s")
'1506662308'
>>> datetime.datetime.now().strftime("%s")
'1506640708'
```

Python's standard library

Create datetime objects and convert them to seconds since the epoch:

```
>>> datetime.datetime.utcnow().strftime("%s")
'1506662308'
>>> datetime.datetime.now().strftime("%s")
'1506640708'
```

This is substantially more dangerous than using MySQL:

```
mysql> SELECT 0 = 'banana';
+--------------+
| 0 = 'banana' |
+--------------+
|            1 |
+--------------+
1 row in set, 1 warning (0.00 sec)
```

Python's standard library

Create datetime objects and convert them to seconds since the epoch:

```
>>> datetime.datetime.utcnow().strftime("%s")
'1506662308'
>>> datetime.datetime.now().strftime("%s")
'1506640708'
```

Python's standard library does not understand time zones and is not generally safe to use for date or time computations.

Introduction

Theory

Python

Pendulum

Conclusions

Pendulum: “Python datetimes made easy”

```
pip3 install pendulum
```

Pendulum: “Python datetimes made easy”

```
pip3 install pendulum
```

Automatically uses reliable time zone database

Olson database ▪ https://github.com/eggert/tz

What time is it now?

```
>>> now = pendulum.now()
>>> now
<Pendulum [2017-09-28T17:18:28.975259-06:00]>
>>> now.timezone
<Timezone [America/Edmonton]>
>>> now.timestamp()
1506640708.975259
>>> now.day_of_week == pendulum.THURSDAY
True
```

What time is it now?

```
>>> now = pendulum.now()
>>> now
<Pendulum [2017-09-28T17:18:29.191388-06:00]>
>>> now.timezone
<Timezone [America/Edmonton]>
>>> now.timestamp()
1506640709.191388
>>> now.day_of_week == pendulum.THURSDAY
True

>>> utcnow = pendulum.utcnow()
>>> utcnow
<Pendulum [2017-09-28T23:18:29.303454+00:00]>
>>> utcnow.timestamp()
1506640709.303454
```

What time is it in ______?

```
>>> pendulum.now().in_timezone("Europe/Berlin")
<Pendulum [2017-09-29T01:18:29.411153+02:00]>
```

What time is it here when it's ______ in ______?

```
>>> there_time = pendulum.create(
>>>     hour=9, minute=15, tz="Europe/Berlin")
>>> there_time
<Pendulum [2017-09-29T09:15:00+02:00]>
>>> here_time = there_time.in_tz(
>>>     pendulum.local_timezone())
>>> here_time
<Pendulum [2017-09-29T01:15:00-06:00]>
>>> there_time == here_time
True
>>> here_time.strftime("%H:%M")
'01:15'
```

What does the date 9/10/11 mean?

```
>>> t = pendulum.parse("9/10/11")
>>> t
<Pendulum [2009-10-11T00:00:00+00:00]>
```

What does the date 9/10/11 mean?

```
>>> t = pendulum.parse("9/10/11")
>>> t
<Pendulum [2009-10-11T00:00:00+00:00]>
```

Pendulum comes up with something
Pendulum isn't perfect and
should probably raise exceptions than it does

What does the date 9/10/11 mean?

```
>>> t = pendulum.parse("9/10/11")
>>> t
<Pendulum [2009-10-11T00:00:00+00:00]>
```

YYYY-MM-DD

Commonly called ISO8601:

- Unambiguous
- If you have a list of dates/times, sorting alphabetically sorts by date/time

How long ago did ______ happen?

```
>>> d = pendulum.now()\
>>>     - pendulum.parse("1969-07-21 2:39")
>>> d
<Period [1969-07-21T02:39:00+00:00 -> 2017-09-2
>>> d.total_seconds()
1520800769.965581
>>> str(d)
'48 years 2 months 1 week 14 hours 39 minutes 2
```

Remind me when _______ is about to happen

```
>>> i = pendulum.create(2021, 1, 20, 12,
>>>     tz="America/New_York")
>>> i
<Pendulum [2021-01-20T12:00:00-05:00]>
>>> d = i - pendulum.now()
>>> d
<Period [2017-09-28T17:18:30.084259-06:00 -> 20
>>> str(d)
'3 years 3 months 3 weeks 1 day 18 hours 41 mi
```

Remind me when ______ is about to happen

```
>>> i = pendulum.create(2021, 1, 20, 12,
>>>       tz="America/New_York")
>>> i
<Pendulum [2021-01-20T12:00:00-05:00]>
>>> d = i - pendulum.now()
>>> d
<Period [2017-09-28T17:18:30.206070-06:00 -> 20
>>> str(d)
'3 years 3 months 3 weeks 1 day 18 hours 41 mi
```

But what if the time zone changes?
Better to store it as datetime and timezone,

Introduction

Theory

Python

Pendulum

Conclusions

What we haven't talked about

Pendulum may or not help you here; you're on your own.

What we haven’t talked about

Pendulum may or not help you here; you’re on your own.

- Other calendars: Buddhist, Coptic, Hebrew, Islamic ...
- Localization
- Leap seconds
- Time synchronization
- ...

Conclusion

- Dates and times are most complicatied because of time zones
- Python falls down hard there
- Pendulum handles common date, time, and time zone computations in an easier and safer way