Python strptime() and timezones?

I have a CSV dumpfile from a Blackberry IPD backup, created using IPDDump. The date/time strings in here look something like this (where `EST` is an Australian time-zone):

```
Tue Jun 22 07:46:22 EST 2010
```

I need to be able to parse this date in Python. At first, I tried to use the `strptime()` function from datettime.

```
>>> datetime.datetime.strptime('Tue Jun 22 12:10:20 2010 EST', '%a %b %d %H:%M:%S %Y
%Z')
```

However, for some reason, the `datetime` object that comes back doesn't seem to have any `tzinfo` associated with it.

I did read on this page that apparently `datetime.strptime` silently discards `tzinfo`, however, I checked the documentation, and I can't find anything to that effect documented here.

I have been able to get the date parsed using a third-party Python library, dateutil, however I'm still curious as to how I was using the in-built `strptime()` incorrectly? Is there any way to get `strptime()` to play nicely with timezones?

python datetime timezone

edited Jul 18 '14 at 8:24
user212218

asked Jul 22 '10 at 2:42
victorhooi
4,406 ● 11 ● 54 ● 88

1 Can't you just... convert all dates to GMT? – Robus Jul 22 '10 at 2:48

1 @Robus: Hmm, I was hoping to do that - but I was assuming that strftime/datetime could somehow do that? Either way, I need to store/parse the fact that the datetimes are in the EST timezone, or whatever timezone they happen to me. The script needs to be able to parse generic datetimes with timezone info (e.g. ETC could be any other timezone). – victorhooi Jul 22 '10 at 3:00

3 EST is also a US timezone abbreviation. (Similarly BST is both a UK and a Brazilian timezone abbrev.) Such abbreviations are just inherently ambiguous. Use offsets relative to UTC/GMT instead. (If you need to support abbreviations, you need to make the mapping locale-dependent and that's a messy rat-hole.) – Donal Fellows Jul 22 '10 at 8:14

EST timezone abbreviation is ambiguous. See also: Parsing date/time string with timezone abbreviated name in Python? – J.F. Sebastian Aug 21 '14 at 10:42

3 Answers

The `datetime` module documentation says:

> Return a datetime corresponding to date_string, parsed according to format. This is equivalent to `datetime(*(time.strptime(date_string, format)[0:6]))`.

See that `[0:6]` ? That gets you `(year, month, day, hour, minute, second)`. Nothing else. No mention of timezones.

Interestingly, [Win XP SP2, Python 2.6, 2.7] passing your example to `time.strptime` doesn't work but if you strip off the " %Z" and the " EST" it does work. Also using "UTC" or "GMT" instead of "EST" works. "PST" and "MEZ" don't work. Puzzling.

edited Dec 20 '14 at 5:49
bignose
8,951 ● 4 ● 34 ● 56

answered Jul 22 '10 at 8:08
John Machin
48.9k ● 3 ● 63 ● 126

2 Related Python bug: %Z in strptime doesn't match EST and others – J.F. Sebastian Sep 29 '14 at 12:11

2 Related Python bug: %Z in strptime doesn't match EST and others – J.F. Sebastian Sep 29 '14 at 12:11

I recommend using python-dateutil. Its parser has been able to parse every date format I've thrown at it so far.

```
>>> from dateutil import parser
>>> parser.parse("Tue Jun 22 07:46:22 EST 2010")
datetime.datetime(2010, 6, 22, 7, 46, 22, tzinfo=tzlocal())
>>> parser.parse("Fri, 11 Nov 2011 03:18:09 -0400")
datetime.datetime(2011, 11, 11, 3, 18, 9, tzinfo=tzoffset(None, -14400))
>>> parser.parse("Sun")
datetime.datetime(2011, 12, 18, 0, 0)
>>> parser.parse("10-11-08")
datetime.datetime(2008, 10, 11, 0, 0)
```

and so on. No dealing with `strptime()` format nonsense... just throw a date at it and it Does The Right Thing.

Update: Oops. I missed in your original question that you mentioned that you used `dateutil`, sorry about that. But I hope this answer is still useful to other people who stumble across this question when they have date parsing questions and see the utility of that module.

answered Dec 15 '11 at 18:52

Joe Shaw
9,546 ●10 ●46 ●71

1 A million and one upvotes for this incredible class. Thank you for sharing. – Nick Woodhams Jun 11 '12 at 4:03

+1 this answer has proven really useful! Thanks :-) – nemesisdesign Nov 18 '12 at 19:43

Given that so many people tend to use python-dateutil, I'd like to point us one limitation of that lib. `>>> parser.parse("Thu, 25 Sep 2003 10:49:41,123 -0300") Traceback (most recent call last): File "<stdin>", line 1, in <module> File "/Users/wanghq/awscli/lib/python2.7/site-packages/dateutil/parser.py", line 748, in parse return DEFAULTPARSER.parse(timestr, **kwargs) File "/Users/wanghq/awscli/lib/python2.7/site-packages/dateutil/parser.py", line 310, in parse res, skipped_tokens = self._parse(timestr, **kwargs) TypeError: 'NoneType' object is not iterable` – wanghq Apr 29 '14 at 0:16

@wanghq you need to replace the last comma with period. Then `parser.parse("Thu, 25 Sep 2003 10:49:41.123 -0300") returns: datetime.datetime(2003, 9, 25, 10, 49, 41, 123000, tzinfo=tzoffset(None, -10800))` – flyingfoxlee Aug 1 '14 at 8:37

4 @flyingfoxlee, yes, I understand that. I just want to tell people the limitation of python-dateutil. It does magic things, but sometimes fails to do that. So "just throw a date at it and it Does The Right Thing." is not 100% true. – wanghq Aug 1 '14 at 23:38

Your time string is similar to the time format in rfc 2822 (date format in email, http headers). You could parse it using only stdlib:

```
>>> from email.utils import parsedate_tz
>>> parsedate_tz('Tue Jun 22 07:46:22 EST 2010')
(2010, 6, 22, 7, 46, 22, 0, 1, -1, -18000)
```

See solutions that yield timezone-aware datetime objects for various Python versions: parsing date with timezone from an email.

In this format, `EST` is semantically equivalent to `-0500`. Though, in general, a timezone abbreviation is not enough, to identify a timezone uniquely.

answered Dec 4 '15 at 10:27

J.F. Sebastian
163k ●34 ●289 ●436