pytz_deprecation_shim

Release <unknown>

Paul Ganssle

Jun 11, 2020

CONTENTS

pytz has served the Python community well for many years, but it is no longer the best option for providing time zones. pytz has a non-standard interface that is very easy to misuse; this interface was necessary when pytz was created, because datetime had no way to represent ambiguous datetimes, but this was solved in in Python 3.6, which added a fold attribute to datetimes in PEP 495. With the addition of the zoneinfo module in Python 3.9 (PEP 615), there has never been a better time to migrate away from pytz.

However, since pytz time zones are used very differently from a standard tzinfo, and many libraries have built pytz zones into their standard time zone interface (and thus may have users relying on the existence of the localize and normalize methods); this library provides shim classes that are compatible with both PEP 495 and pytz's interface, to make it easier for libraries to deprecate pytz.

CHAPTER

ONE

USAGE

This library is intended for *temporary usage only*, and should allow you to drop your dependency on `pytz` while also giving your users notice that eventually you will remove support for the `pytz`-specific interface.

Within your own code, use `pytz_deprecation_shim.timezone` shims as if they were `zoneinfo` or `dateutil.tz` zones — do not use `localize` or `normalize`:

```
>>> import pytz_deprecation_shim as pds
>>> from datetime import datetime, timedelta
>>> LA = pds.timezone("America/Los_Angeles")

>>> dt = datetime(2020, 10, 31, 12, tzinfo=LA)
>>> print(dt)
2020-10-31 12:00:00-07:00

>>> dt.tzname()
'PDT'
```

Datetime addition will work like normal Python datetime arithmetic, even across a daylight saving time transition:

```
>>> dt_add = dt + timedelta(days=1)

>>> print(dt_add)
2020-11-01 12:00:00-08:00

>>> dt_add.tzname()
'PST'
```

However, if you have exposed a time zone to end users who are using `localize` and/or `normalize` or any other `pytz`-specific features (or if you've failed to convert some of your own code all the way), those users will see a warning (rather than an exception) when they use those features:

```
>>> dt = LA.localize(datetime(2020, 10, 31, 12))
.../pytz_deprecation_shim/_impl.py:81: PytzUsageWarning: The localize
method is no longer necessary, as this time zone supports the fold
attribute(PEP 495). For more details on migrating to a PEP 495-compliant
implementation, see <TBD>
warnings.warn(

 >>> print(dt)
2020-10-31 12:00:00-07:00
>>> dt.tzname()
'PDT'

>>> dt_add = LA.normalize(dt + timedelta(days=1))
```

(continues on next page)

(continued from previous page)

```
.../pytz_deprecation_shim/_impl.py:131: PytzUsageWarning: The normalize
method is no longer necessary, as this time zone supports the fold
attribute(PEP 495). For more details on migrating to a PEP 495-compliant
implementation, see <TBD>
warnings.warn(

>>> print(dt_add)
2020-11-01 12:00:00-08:00
>>> dt_add.tzname()
'PST'
```

For IANA time zones, calling `str()` on the shim zones (and indeed on `pytz` and `zoneinfo` zones as well) returns the IANA key, so end users who would like to actively migrate to a `zoneinfo` (or `backports.zoneinfo`) can do so:

```
>>> from zoneinfo import ZoneInfo
>>> LA = pds.timezone("America/Los_Angeles")
>>> LA_zi = ZoneInfo(str(LA))
>>> print(LA_zi)
zoneinfo.ZoneInfo(key='America/Los_Angeles')
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search