When.py Documentation

Release 0.2.0

Andy Dirnberger

July 01, 2012

CONTENTS

When.py provides user-friendly functions to help perform common date and time actions.

CHAPTER

ONE

USAGE

Friendly Dates and Times

when.**all_timezones**()
Get a list of all time zones.

This is a wrapper for pytz.all_timezones. New in version 0.1.0.

when.**all_timezones_set**()
Get a set of all time zones.

This is a wrapper for pytz.all_timezones_set. New in version 0.1.0.

when.**common_timezones**()
Get a list of common time zones.

This is a wrapper for pytz.common_timezones. New in version 0.1.0.

when.**common_timezones_set**()
Get a set of common time zones.

This is a wrapper for pytz.common_timezones_set. New in version 0.1.0.

when.**future**(*years=0*, *months=0*, *weeks=0*, *days=0*, *hours=0*, *minutes=0*, *seconds=0*, *milliseconds=0*, *microseconds=0*, *utc=False*)
Get a datetime in the future.

future() accepts the all of the parameters of datetime.timedelta, plus includes the parameters years and months. years and months will add their respective units of time to the datetime.

By default future() will return the datetime in the system's local time. If the utc parameter is set to True or set_utc() has been called, the datetime will be based on UTC instead. New in version 0.1.0.

when.**now**(*utc=False*)
Get a datetime representing the current date and time.

By default now() will return the datetime in the system's local time. If the utc parameter is set to True or set_utc() has been called, the datetime will be based on UTC instead. New in version 0.1.0.

when.**past**(*years=0*, *months=0*, *weeks=0*, *days=0*, *hours=0*, *minutes=0*, *seconds=0*, *milliseconds=0*, *microseconds=0*, *utc=False*)
Get a datetime in the past.

past() accepts the all of the parameters of datetime.timedelta, plus includes the parameters years and months. years and months will add their respective units of time to the datetime.

By default past() will return the datetime in the system's local time. If the utc parameter is set to True or set_utc() has been called, the datetime will be based on UTC instead. New in version 0.1.0.

when.**set_utc**()
Set all datetimes to UTC.

The utc parameter of other methods will be ignored, with the global setting taking precedence.

This can be reset by calling unset_utc(). New in version 0.1.0.

when.**shift**(*value*, *from_tz=None*, *to_tz=None*, *utc=False*)
Convert a datetime from one time zone to another.

value will be converted from its time zone (when it is time zone aware) or the time zone specified by from_tz (when it is time zone naive) to the time zone specified by to_tz. These values can either be strings containing the name of the time zone (see pytz.all_timezones for a list of all supported values) or a datetime.tzinfo object.

If no value is provided for either from_tz (when value is time zone naive) or to_tz, the current system time zone will be used. If the utc parameter is set to True or set_utc() has been called, however, UTC will be used instead. Changed in version 0.2.0: Added support for value as a time zone aware datetime

when.**timezone**()
Get the name of the current system time zone. New in version 0.1.0.

when.**timezone_object**(*tz_name=None*)
Get the current system time zone.

This returns a datetime.tzinfo instance. New in version 0.1.0.

when.**today**()
Get a date representing the current date. New in version 0.1.0.

when.**tomorrow**()
Get a date representing tomorrow's date. New in version 0.1.0.

when.**unset_utc**()
Set all datetimes to system time.

The utc parameter of other methods will be used.

This can be changed by calling set_utc(). New in version 0.1.0.

when.**yesterday**()
Get a date representing yesterday's date. New in version 0.1.0.

CHAPTER

TWO

A NOTE ABOUT `future` AND `past`

When changing a datetime from one month (or year) to another, it is often the case that the new month will have fewer days than the original, resulting in an invalid date. When this happens, the days will be adjusted into the future. This is consistent with implementations found elsewhere.

```
>>> when.today()
datetime.date(2012, 2, 29)
>>>
>>> when.future(years=1)
datetime.datetime(2013, 3, 1, 19, 0, 23, 76878)

>>> when.today()
datetime.date(2012, 3, 31)
>>>
>>> when.past(months=1)
datetime.datetime(2012, 3, 2, 19, 7, 36, 317653)
```

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX