Converting string into datetime

Short and simple. I've got a huge list of date-times like this as strings:

```
Jun 1 2005  1:33PM
Aug 28 1999 12:00AM
```

I'm going to be shoving these back into proper datetime fields in a database so I need to magic them into real datetime objects.

Any help (even if it's just a kick in the right direction) would be appreciated.

Edit: This is going through Django's ORM so I can't use SQL to do the conversion on insert.

python django datetime

edited Jan 21 '09 at 18:09
Ben Blank
27.5k ● 12 ● 86 ● 133

asked Jan 21 '09 at 18:00
Oli
86k ● 39 ● 157 ● 220

12 Answers

```
from datetime import datetime

date_object = datetime.strptime('Jun 1 2005  1:33PM', '%b %d %Y %I:%M%p')
```

Link to the Python documentation for strptime

and a link for the strftime format mask

edited Oct 15 '15 at 18:13
jesterjunk
1,204 ● 8 ● 12

answered Jan 21 '09 at 18:08
Patrick Harrington
13.4k ● 4 ● 14 ● 18

2 why does that return a date_object and not a datetime_object? – Jon Crowell Apr 28 '14 at 19:07

'%b', '%p' may fail in non-English locale. – J.F. Sebastian Apr 29 '14 at 10:55

What is the string doesn't have the time, just "April 25, 2014" – User Apr 30 '14 at 1:56

2 @User You'll have to know ahead of time to exclude that part of the format string, but if you want a `date` instead of a `datetime`, going through `datetime` handles it nicely: `datetime.strptime('Jun 1 2005', '%b %d %Y').date() == date(2005, 6, 1)` – Izkata Nov 11 '14 at 20:02

To parse default unix 'date' command output which has timezone in it e.g. "Sun Oct 4 07:48:48 UTC 2015" one can use datetime.strptime(currentDateStr, "%a %b %d %H:%M:%S %Z %Y"). – gaoithe Oct 7 '15 at 10:28

Check out strptime in the time module. It is the inverse of strftime.

edited Mar 3 '14 at 10:24

answered Jan 21 '09 at 18:07

oluckyman
943 ● 13 ● 22

florin
9,763 ● 4 ● 27 ● 45

93 For the provided examples, the format string would be time.strptime(stamp, '%b %d %Y %I:%M%p'). – Ben Blank Jan 21 '09 at 18:15

14 From what I understand, this answer only outputs time objects, not datetime objects -- which is why the answer would be buried compared to Patrick's answer. – Alexander Bird Sep 7 '10 at 13:08

13 the answer below (by Patrick Harrington) is more correct, because time.strptime only outputs time, not datetime – Anatoly G Jun 19 '11 at 19:56

3 As Alexander said, this return a struct_time, not a datetime. Of course you can convert it to a datetime, but Patrick's answer is more straight forward if you want a datetime object in the end. – Leandro Alves Mar 9 '13 at 15:20

1 @hobbes3 `parse` and `format` . – VINCENT Oct 22 '14 at 12:07

Use the third party dateutil library:

```
from dateutil import parser
dt = parser.parse("Aug 28 1999 12:00AM")
```

It can handle most date formats, including the one you need to parse. It's more convenient than strptime as it can guess the correct format most of the time.

It very useful for writing tests, where readability is more important than performance.

You can install it with:

```
pip install python-dateutil
```

edited Mar 23 at 16:35

Decko
3,214 ● 2 ● 15 ● 27

answered Jan 22 '09 at 18:27

Simon Willison
5,804 ● 5 ● 22 ● 37

24 Be aware that for large data amounts this might not be the most optimal way to approach the problem. Guessing the format every single time may be horribly slow. – Reef Jul 3 '11 at 0:08

7 This is nice but it would be nice to have a solution that is built-in rather than having to go to a third party. – brian buck Oct 12 '11 at 20:33

When I try to parse "32nd jan", it returns me "2032-01-06".. which is incorrect. is there any way to check whether the string is a valid date or not – Kartik Mar 6 '13 at 6:11

2 @Reef: 5 times as slow according to my quick and dirty benchmark. Not *so* horribly slow as I would expect. – Antony Hatchkins Apr 30 '13 at 18:19

Note that the current version of dateutil, version 2.2, depends on the six library, which is a Python 2/3 compatibility library. This allows dateutil 2.2 to work with both Python 2.x and Python 3.x. – Simon Tewsi Nov 28 '13 at 21:11

I have put together a project that can convert some really neat expressions. Check out **timestring**.

Here are some examples below:

`pip install timestring`

```
>>> import timestring
>>> timestring.Range('next week')
<timestring.Range From 03/03/14 00:00:00 to 03/10/14 00:00:00 4496004880>
>>> timestring.Date('monday, aug 15th 2015 at 8:40 pm')
<timestring.Date 2015-08-15 20:40:00 4491909392>
```

edited Dec 28 '14 at 12:59

answered Mar 2 '14 at 14:22

Steve Peak
929 ● 6 ● 14

Wow. Wow. Wow. Wow. This is so easy. I've got a datetime string and I just want to pull out the year. As simple as: `import timestring` `timestring.Date('27 Mar 2014 12:32:29 GMT').year` This lib made it SO EASY! Thank you. – brandonjp Apr 11 '14 at 5:09

Your very welcome. I would love your comments and ideas on improving this package. Let me know, use github issues. Thanks! – Steve Peak Apr 14 '14 at 14:30

@Steve Peak timestring works great! Needed to parse article dates with scrapy and this has been converting them perfectly. – arctelix Oct 22 '14 at 19:58

Hi steve, the module is great. Would be nice to have a weekday string attribute as well. Otherwise not sure if you start from Monday or Sunday – Anake Oct 23 '14 at 10:00

@Anake you can create an issue to request this added at github.com/stevepeak/timestring thanks! – Steve Peak Oct 25 '14 at 22:22

Many timestamps have an implied timezone. To ensure that your code will work in every timezone, you should use UTC internally and attach a timezone each time a foreign object enters the system.

Python 3.2+:

```
>>> datetime.datetime.strptime(
...     "March 5, 2014, 20:13:50", "%B %d, %Y, %H:%M:%S"
... ).replace(tzinfo=datetime.timezone(datetime.timedelta(hours=-3)))
```

edited Jun 5 '15 at 12:34

answered Mar 6 '14 at 11:53

Janus Troelsen

8,643 ● 3 ● 64 ● 110

Why do you keep the ugly and sometimes wrong (`mktime()` during DST transitions) 1st method if you know the 2nd method (`datetime.strptime()`)? If you want to avoid an exception during a leap second (the 2nd method fails) then you could use `calendar.timegm` instead: `(datetime(1970,1,1)+timedelta(seconds=timegm(time.strptime(..)))).replace(tzinfo=timezone(timedelta(-3)))` – J.F. Sebastian Sep 14 '14 at 17:36

I have no good arguments for that. Thanks for the insightful observations. – Janus Troelsen Sep 14 '14 at 17:58

Something that isn't mentioned here and is useful: adding a suffix to the day. I decoupled the suffix logic so you can use it for any number you like, not just dates.

```
import time

def num_suffix(n):
    '''
    Returns the suffix for any given int
    '''
    suf = ('th','st', 'nd', 'rd')
    n = abs(n) # wise guy
    tens = int(str(n)[-2:])
    units = n % 10
    if tens > 10 and tens < 20:
        return suf[0] # teens with 'th'
    elif units <= 3:
        return suf[units]
    else:
        return suf[0] # 'th'

def day_suffix(t):
    '''
    Returns the suffix of the given struct_time day
    '''
    return num_suffix(t.tm_mday)

# Examples
print num_suffix(123)
print num_suffix(3431)
print num_suffix(1234)
print ''
print day_suffix(time.strptime("1 Dec 00", "%d %b %y"))
print day_suffix(time.strptime("2 Nov 01", "%d %b %y"))
print day_suffix(time.strptime("3 Oct 02", "%d %b %y"))
print day_suffix(time.strptime("4 Sep 03", "%d %b %y"))
print day_suffix(time.strptime("13 Nov 90", "%d %b %y"))
print day_suffix(time.strptime("14 Oct 10", "%d %b %y"))
```

edited Oct 14 '11 at 0:28

answered Oct 14 '11 at 0:13

Aram Kocharyan

12.1k ● 7 ● 49 ● 68

You string representation of datetime is

`Jun 1 2005  1:33PM`

which is equals to

`%b %d %Y %I:%M%p`

> %b Month as locale's abbreviated name(Jun)
>
> %d Day of the month as a zero-padded decimal number(1)
>
> %Y Year with century as a decimal number(2015)
>
> %I Hour (12-hour clock) as a zero-padded decimal number(01)
>
> %M Minute as a zero-padded decimal number(33)
>
> %p Locale's equivalent of either AM or PM(PM)

```
>>> dates = []
```

```
>>> dates.append('Jun 1 2005  1:33PM')
>>> dates.append('Aug 28 1999 12:00AM')
>>> from datetime import datetime
>>> for d in dates:
...     date = datetime.strptime(d, '%b %d %Y %I:%M%p')
...     print type(date)
...     print date
...
```

Output

```
<type 'datetime.datetime'>
2005-06-01 13:33:00
<type 'datetime.datetime'>
1999-08-28 00:00:00
```

edited Jan 5 at 17:34

answered Dec 10 '14 at 13:00

Rizwan Mumtaz
1,319 ● 8 ● 21

Here is a solution using Pandas to convert dates formatted as strings into datetime.date objects.

```
import pandas as pd

dates = ['2015-12-25', '2015-12-26']

>>> [d.date() for d in pd.to_datetime(dates)]
[datetime.date(2015, 12, 25), datetime.date(2015, 12, 26)]
```

And here is how to convert the OP's original date-time examples:

```
datetimes = ['Jun 1 2005  1:33PM', 'Aug 28 1999 12:00AM']

>>> pd.to_datetime(datetimes).to_pydatetime().tolist()
[datetime.datetime(2005, 6, 1, 13, 33),
 datetime.datetime(1999, 8, 28, 0, 0)]
```

There are many options for converting from the strings to Pandas Timestamps using `to_datetime`, so check the docs if you need anything special.

Likewise, Timestamps have many properties and methods that can be accessed in addition to `.date`

edited Dec 20 '15 at 3:26

answered Dec 20 '15 at 3:03

Alexander
17.5k ● 6 ● 19 ● 42

```
In [34]: import datetime

In [35]: _now = datetime.datetime.now()

In [36]: _now
Out[36]: datetime.datetime(2016, 1, 19, 9, 47, 0, 432000)

In [37]: print _now
2016-01-19 09:47:00.432000

In [38]: _parsed = datetime.datetime.strptime(str(_now),"%Y-%m-%d %H:%M:%S.%f")

In [39]: _parsed
Out[39]: datetime.datetime(2016, 1, 19, 9, 47, 0, 432000)

In [40]: assert _now == _parsed
```

answered Jan 19 at 7:48

guneysus
1,573 ● 10 ● 19

Django Timezone aware datetime object example.

```
import datetime
from django.utils.timezone import get_current_timezone
tz = get_current_timezone()

format = '%b %d %Y %I:%M%p'
date_object = datetime.datetime.strptime('Jun 1 2005  1:33PM', format)
date_obj = tz.localize(date_object)
```

This conversion is very important for Django and Python when you have `USE_TZ = True`:

```
RuntimeWarning: DateTimeField MyModel.created received a naive datetime (2016-03-04
00:00:00) while time zone support is active.
```

edited Mar 3 at 13:44

answered Nov 20 '14 at 17:58

Tarek Kalaji
305 ● 6 ● 17


Ryu_hayabusa
969 ● 1 ● 7 ● 17

You can use easy_date to make it easy:

```
import date_converter
converted_date = date_converter.string_to_datetime('Jun 1 2005  1:33PM', '%b %d %Y
%I:%M%p')
```

answered Jun 1 '15 at 15:15


Raphael Amoedo
1,143 ● 1 ● 6 ● 17

Create a small utility function like:

```
def date(datestr="", format="%Y-%m-%d"):
    from datetime import datetime
    if not datestr:
        return datetime.today().date()
    return datetime.strptime(datestr, format).date()
```

This is versatile enough:

- If you dont pass any arguments it will return today's date.
- theres a date format as default that you can override.
- You can easily modify it to return a datetime.

answered Feb 4 at 13:43


Mackraken
69 ● 3

protected by casperOne Apr 26 '12 at 12:03

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site.

Would you like to answer one of these unanswered questions instead?