Python Programming Basics

Testing Working with Dates and Time in Python

Contents

Introduction

- Python has a range of date manipulation functions in datetime library that allow us to work with dates and time.
- Working on dates and time can be tedious when the data come with date values in different format.
- The datetime library of Python, converts a variety of character date formats into Python dates. Once converted to dates, the following functions will return information about dates: `second, minute, hour, month, year`
- Pandas also has a Timestamp function. The inbuilt function offers a nice way to make easy parsing in dates and times.

Base Package Functions

`datetime.strptime()` converts dates entered as strings into numeric dates.

```
datetime.strptime(x, "%Y-%m-%d")
```

x is a string object to be converted

`"%Y-%m-%d"` is the format (in which the date appears within the string) composed of codes such as:

Day	day as a number (01-31)	%d
	abbreviated weekday (Mon)	%a
	full weekday name (Monday)	%A
Month	abbreviated month (Jan)	%b
	full month name (January)	%B
	month as a number (01-12)	%m
Year	2-digit year (16)	%y
	4-digit year (2016)	%Y

datetime library functions

```
# Formatting a date
```

```
from datetime import datetime
x = '5 jan 2010'
ndate = datetime.strftime(datetime.strptime(x, '%d %b %Y'),'%d %b %Y')
ndate
```

```
'05 Jan 2010'
```

In Python we need to specify the format of the input date.

```
type(ndate)
```

```
str
```

```
ndate2 = datetime.strptime(x, '%d %b %Y')
type(ndate2)
```

datetime.strptime() converts x to a datetime object

```
datetime.datetime
```

```
# Using strftime argument to extract parts of date
```

```
datetime.strptime(ndate, '%d %b %Y').strftime('%Y%B')
```

```
'2010January'
```

Format codes can also be used to extract parts of dates using **strftime ()**

datetime library functions

```
#To extract Day of the week.
datetime.strptime(ndate, '%d %b %Y').strftime('%A')
'Tuesday'
```

```
#To extract Month of the Year.
datetime.strptime(ndate, '%d %b %Y').strftime('%B')
'January'
```

Apart from datetime library, pandas also have functions that deal with timestamps.

```
#To extract Quarter no.
import pandas as pd
pd.Timestamp(ndate).quarter
1
```

datetime library functions

```
#To parse character strings into dates.
datetime.strptime("12-01-2015", "%m-%d-%Y").strftime("%Y-%m-%d")
'2015-12-01'
```

```
#To capture the Current date and time.
date = datetime.now()
date
datetime.datetime(2019, 11, 13, 11, 46, 37, 306825)
```

- The letters **Y**, **m**, and **d** correspond to the year, month, and day elements of a date-time.
- **strftime()** is used to specify the format in which one wants the date to be in.

datetime library functions

```
#To extract the hour component from the date object.
date.hour
11

#To extract the minute component from the date object.
date.minute
46
```

```
#To extract the second component from the date object.
date.second
37
```

datetime library functions

```
# Capture current date
```

```python
from datetime import date
today = date.today()
today
```

```
datetime.date(2019, 11, 13)
```

date.today() returns the your system's current date

```
# Using operators with dates
```

```python
d1 = datetime.date(datetime.strptime("20101201",'%Y%m%d'))
d2 = datetime.date(datetime.strptime("10/7/04",'%m/%d/%y'))
d1
d2
```

```
datetime.date(2010, 12, 1)
datetime.date(2004, 10, 7)
```

```python
d1-d2
```

```
datetime.timedelta(2246)
```

Different operators can be used with date objects

Merge Three Different Columns Into a Date in Python

```
# My Dataframe
```

```
d = {'EmpID' : [101,102,103,104,105],
     'year' : [1977,1989,2000,2012,2015],
     'month' : [2,5,10,1,11],
     'day' : [2,3,1,1,5]}

datedf = pd.DataFrame(d)
datedf
```

```
    EmpID  year  month  day
0     101  1977      2    2
1     102  1989      5    3
2     103  2000     10    1
3     104  2012      1    1
4     105  2015     11    5
```

Data: Employee ID (EmpID) and joining date (split into 3 columns: year month & day)

Merge Three Different Columns Into a Date in Python

We are having 3 separate columns as year, month, and day in our dataframe datedf.

```
# Merge 3 columns into one date column
datedf['Date']=pd.to_datetime(datedf[['year','month','day']])
datedf
```

```
   EmpID  year  month  day       Date
0    101  1977      2    2 1977-02-02
1    102  1989      5    3 1989-05-03
2    103  2000     10    1 2000-10-01
3    104  2012      1    1 2012-01-01
4    105  2015     11    5 2015-11-05
```

- ❑ new column date is created using **[]**
- ❑ **to_datetime()** converts **date** column into a date type

Format a Vector With Inconsistent Date Formats

Converting dates entered as strings into numeric dates in Python is a little tricky if the date information is not represented consistently. Let's see how to deal with this kind of situation.

```
dates = ["12aug08","01sep09","7august06","9august2007","20july1999"]
ndates = pd.to_datetime(dates)
```

```
ndates
DatetimeIndex(['2008-08-12', '2009-09-01', '2006-08-07', '2007-08-09',
               '1999-07-20'],
              dtype='datetime64[ns]', freq=None)
```

Note that Pandas function to_datetime() is capable of handling such discrepancies as long as order of the date elements is consistent.

Quick Recap

In this session, we learnt how to deal with dates and time using base package functions in Python & pandas Timestamp, how to merge 3 different columns into one date column and how to format a vector with inconsistent dates. Here is a quick recap:

datetime functions	• **`today(), now(), hour(), minute(), second(), quarter, month, year`**
Pandas Timestamp functions	• **`to_datetime()`**
Date manipulation tasks	• Merge Three Different Columns Into a Date in Python using **`pandas()`**