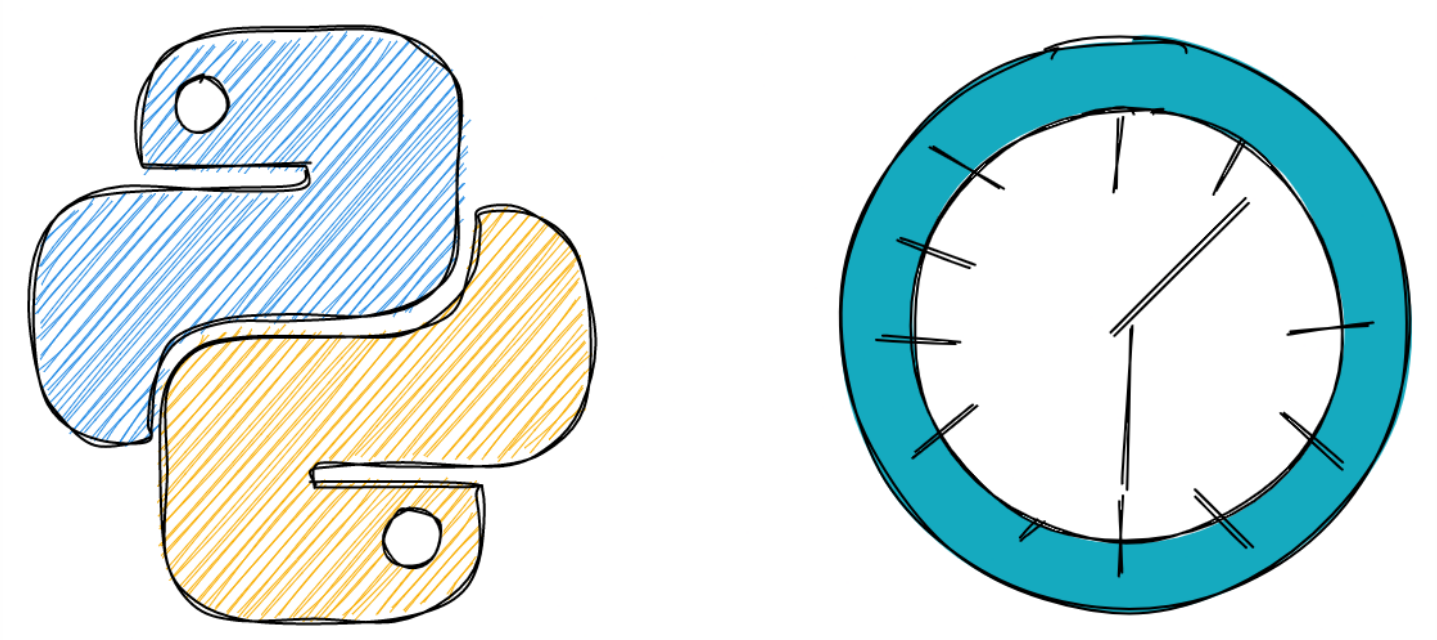
Efficient Python Tricks and Tools for Data Scientists - By Khuyen Tran

Time Series

GitHub View on GitHub Book View Book

This section shows some tools to work with datetime and time series.



datefinder: Automatically Find Dates and Time in a Python String

```
$ pip install datefinder
```

If you want to automatically find date and time with different formats in a Python string, try datefinder.

```
from datefinder import find_dates

text = """We have one meeting on May 17th,
2021 at 9:00am and another meeting on 5/18/2021
at 10:00. I hope you can attend one of the
meetings."""

matches = find_dates(text)

for match in matches:
    print("Date and time:", match)
    print("Only day:", match.day)
```

```
Date and time: 2021-05-17 09:00:00
Only day: 17
Date and time: 2021-05-18 10:00:00
Only day: 18
```

Link to datefinder.

Fastai's add_datepart: Add Relevant DateTime Features in One Line of Code

```
$ pip install fastai
```

When working with time series, other features such as year, month, week, day of the week, day of the year, whether it is the end of the year or not, can be really helpful to predict future events. Is there a way that you can get all of those features in one line of code?

Fastai's add_datepart method allows you to do exactly that.

```
import pandas as pd
from fastai.tabular.core import add_datepart
from datetime import datetime

df = pd.DataFrame(
    {
        "date": [
            datetime(2020, 2, 5),
            datetime(2020, 2, 6),
```

```
            datetime(2020, 2, 7),
            datetime(2020, 2, 8),
        ],
        "val": [1, 2, 3, 4],
    }
)
df
```

	date	val
0	2020-02-05	1
1	2020-02-06	2
2	2020-02-07	3
3	2020-02-08	4

```
df = add_datepart(df, "date")
df.columns
```

```
Index(['val', 'Year', 'Month', 'Week', 'Day',
'Dayofweek', 'Dayofyear',
       'Is_month_end', 'Is_month_start',
'Is_quarter_end', 'Is_quarter_start',
       'Is_year_end', 'Is_year_start',
'Elapsed'],
      dtype='object')
```

Link to Fastai's methods to work with tabular data

Maya: Convert the string to datetime automatically

```
$ pip install maya
```

If you want to convert a string type to a datetime type, the common way is to use strptime(date_string, format). But it is quite inconvenient to specify the structure of your datetime string, such as ' %Y-%m-%d %H:%M:%S'.

There is a tool that helps you convert the string to datetime automatically called maya. You just need to parse the string, and maya will figure out the structure of your string.

```
import maya

# Automatically parse datetime string
string = "2016-12-16 18:23:45.423992+00:00"
maya.parse(string).datetime()
```

```
datetime.datetime(2016, 12, 16, 18, 23, 45,
423992, tzinfo=<UTC>)
```

Better yet, if you want to convert the string to a different time zone (for example, CST), you can parse that into maya's datetime function.

```
maya.parse(string).datetime(to_timezone="US/Central")
```

```
datetime.datetime(2016, 12, 16, 12, 23, 45,
423992, tzinfo=<DstTzInfo 'US/Central' CST-1 day,
18:00:00 STD>)
```

Link to maya.

traces: A Python Library for Unevenly-Spaced Time Series Analysis

```
$ pip install traces
```

If you are working with unevenly-spaced time series, try traces. traces allows you to get the values of the datetimes not specified in your time series based on the values of other datetimes.

For example, while logging our working hours for each date, we forgot to log the working hours for some dates.

```
# Log working hours for each date
import traces
from datetime import datetime

working_hours = traces.TimeSeries()
working_hours[datetime(2021, 9, 10)] = 10
working_hours[datetime(2021, 9, 12)] = 5
working_hours[datetime(2021, 9, 13)] = 6
working_hours[datetime(2021, 9, 16)] = 2
```

We can get the working hours of dates we forgot to log using traces.

```
# Get value on 2021/09/11
working_hours[datetime(2021, 9, 11)]
```

```
10
```

```
# Get value on 2021/09/14
working_hours[datetime(2021, 9, 14)]
```

```
6
```

We can also get the distribution of our working hours from `2021-9-10` to `2021-9-16` using `distribution`:

```
distribution = working_hours.distribution(
    start=datetime(2021, 9, 10),
    end=datetime(2021, 9, 16)
)
distribution
```

```
Histogram({5: 0.16666666666666666, 6: 0.5, 10:
0.3333333333333333})
```

From the output above, it seems like we work 6 hours per day 50% of the time.

Get the median working hours:

```
distribution.median()
```

```
6.0
```

Get the mean working hours:

```
distribution.mean()
```

```
7.166666666666666
```

Link to traces

Extract holiday from date column

```
$ pip install holidays
```

You have a date column and you think the holidays might affect the target of your data. Is there an easy way to extract the holidays from the date? That is when holidays package comes in handy.

Holidays package provides a dictionary of holidays for different countries. The code below is to confirm whether 2020-07-04 is a US holiday and extract the name of the holiday.

```
from datetime import date
import holidays

us_holidays = holidays.UnitedStates()

"2014-07-04" in us_holidays
```

```
True
```

The great thing about this package is that you can write the date in whatever way you want and the package is still able to detect which date you are talking about.

```
us_holidays.get("2014-7-4")
```

```
'Independence Day'
```

```
us_holidays.get("2014/7/4")
```

```
'Independence Day'
```

Link to holidays.

Workalendar: Handle Working-Day Computation in Python

```
$ pip install workalendar
```

If you want to handle calendars, holidays, and working-day-related computations, use Workalendar. Workalendar supports nearly 100 countries over the world.

```
from datetime import date
from workalendar.usa import UnitedStates
from workalendar.asia import Japan
```

```
# Get all holidays in the US

US_cal = UnitedStates()
US_cal.holidays(2022)
```

```
[(datetime.date(2021, 12, 31), 'New year
(Observed)'),
 (datetime.date(2022, 1, 1), 'New year'),
 (datetime.date(2022, 1, 17), 'Birthday of Martin
Luther King, Jr.'),
 (datetime.date(2022, 2, 21), "Washington's
Birthday"),
 (datetime.date(2022, 5, 30), 'Memorial Day'),
 (datetime.date(2022, 7, 4), 'Independence Day'),
 (datetime.date(2022, 9, 5), 'Labor Day'),
 (datetime.date(2022, 10, 10), 'Columbus Day'),
 (datetime.date(2022, 11, 11), 'Veterans Day'),
 (datetime.date(2022, 11, 24), 'Thanksgiving
Day'),
 (datetime.date(2022, 12, 25), 'Christmas Day'),
 (datetime.date(2022, 12, 26), 'Christmas Day
(Observed)')]
```

```
US_cal.is_working_day(date(2022, 1, 22)) #
Saturday
```

```
False
```

```
US_cal.is_working_day(date(2021, 12, 24)) #
Thanksgiving Day
```

```
False
```

```
# Calculate working days between 2022/1/19 and
2022/5/15
US_cal.get_working_days_delta(date(2022, 1, 19),
date(2022, 5, 15))
```

```
81
```

```
# Get holidays in Japan
JA_cal = Japan()
JA_cal.holidays(2022)
```

```
[(datetime.date(2022, 1, 1), 'New year'),
 (datetime.date(2022, 1, 10), 'Coming of Age
Day'),
 (datetime.date(2022, 2, 11), 'Foundation Day'),
 (datetime.date(2022, 2, 23), "The Emperor's
Birthday"),
 (datetime.date(2022, 3, 21), 'Vernal Equinox
Day'),
```

```
 (datetime.date(2022, 4, 29), 'Showa Day'),
 (datetime.date(2022, 5, 3), 'Constitution
Memorial Day'),
 (datetime.date(2022, 5, 4), 'Greenery Day'),
 (datetime.date(2022, 5, 5), "Children's Day"),
 (datetime.date(2022, 7, 18), 'Marine Day'),
 (datetime.date(2022, 8, 11), 'Mountain Day'),
 (datetime.date(2022, 9, 19), 'Respect-for-the-
Aged Day'),
 (datetime.date(2022, 9, 23), 'Autumnal Equinox
Day'),
 (datetime.date(2022, 10, 10), 'Sports Day'),
 (datetime.date(2022, 11, 3), 'Culture Day'),
 (datetime.date(2022, 11, 23), 'Labour
Thanksgiving Day')]
```

Link to Workalendar.