Intro to Time Series in Python

Learning Objectives:

By the end of this lesson, students will be able to:

- Recognize datetime objects in Python
- Apply various formats to datetime objects

Time Series Basics

What is a time series? What is required to be considered time series data?

- A time series is any data in which the rows represent different points in time. *It can be just a date, just a time, or a combination of both date and time.*
- A time series may have rows that represent evenly spaced times or the rows may represent unevenly spaced *times.*
  - If the times are evenly spaced, time series has a frequency. This means there is a fixed distance between each observation/row (e.g. daily, monthly, yearly, etc)

Representing Time Series in Python

- There are 2 primary ways we will represent date/time with python:
  - Python's datetime module with datetime variables.
    - https://docs.python.org/3/library/datetime.html
  - Pandas's Custom Datetime Classes built on-top of Python's datetime objects.
    - https://pandas.pydata.org/docs/user_guide/timeseries.html
- Pandas makes it easy to convert any column of date/time values into a "datetime" datatype, but uses its own classes which work similarly to standard python datetime variables.

`datetime` Objects (Basic Python)

- Python has a `datetime` module that has a `date` class (dates without hours/minutes/seconds), a `time` class (hours/minutes/seconds without a date) and a `datetime` class, which is a combination of both.
- We will use the datetime class as it is the most flexible.

First, we must import the datetime library:

```
import datetime as dt
```

dt has .datetime methods. For example, we can obtain the current date and time with .now()

```
## You can obtain the current date and time with .now()
now = dt.datetime.now()
now
```

```
datetime.datetime(2022, 7, 25, 16, 47, 27, 90245)
```

Note that your output will be different, and should be the current date and time.

Notice the formatting of the output: It contains year, month, day, hour, minute, second, microsecond.

We can extract each of these individual pieces of the date and time as attributes:

- For Example:

- to obtain just the year, use .year
- to obtain just the month, use .month
- etc.

```
## use datetime attribute to obtain just the year
now.year
```

```
2022
```

Datetime Formatting

Displayed

- When datetime variables are *displayed*, we see all of the individual pieces of the date as a list.

```
## display using display function
display(now)
```

```
datetime.datetime(2022, 7, 25, 16, 47, 27, 90245)
```

```
# display by leaving as final line in cell
now
```

```
datetime.datetime(2022, 7, 25, 16, 47, 27, 90245)
```

Printed

- When datetime variables are *printed*, they look very different:

```
print(now)
```

```
2022-07-25 16:47:27.090245
```

- As we can see, this looks a lot closer to how we would expect a date and time to look.
- Datetime objects have a specific date format.
  - The default format for a datetime object is:
    - `{4 digit year}-{2 digit month}-{2 digit day} {hour in 24 hour time}:{minute}:{seconds}.{microseconds}`
  - We can represent this format using a format string: `"%Y-%m-%d %H:%M:%S.%f"` (which we will explain in the next section)
- Datetime objects have a method that controls which format is used for printed/text version of the date, called `.strftime` .

Datetime.strftime

As you may imagine, there are many options for how you choose to format dates and time! Refer to Documentation for strftime for a table of format codes. These codes allow you to specify what type of date formatting should be used.

Some examples of formatting options:

For day of the month:

One option is:

- %d: 0-padded 2-digit day of the month (e.g. 01, 12, 26, etc)

Try it with now and see:

```
# just output the day with a leading 0 if needed
format_a = "%d"
now.strftime(format_a)
```

```
'25'
```

Notice that since we only specified a format for day in format_a, our output only included the day of the month.

For month:

A few options are:

- %B: full month name (e.g. January, February)
- %b: abbreviated month name (e.g. Jan, Feb, etc)
- %m: 0-padded 2 digit month of the year (e.g. 01,06,11,12)

Let's try adding the full month name before our day of the month

```
# Add full month name to day of month
format_b = "%B %d"
now.strftime(format_b)
```

```
'July 25'
```

Try some of the other options for month! Which do you think looks the best?

For the day of the week:

One options for day of the week:

- %a: Weekday as locale's abbreviated name. (e.g. "Sun, Mon")

Let's start with the day of the week, followed by a comma.

- Notice that any additional spacing or characters used when defining the format will be included in the final string.

```
# Add day of the week to month and day
format_c = "%a, %B %d"
now.strftime(format_c)
```

```
'Mon, July 25'
```

Use the documentation to see what other options for day of the week you might want to use!

For the year:

A few options include:

- %Y: 4-digit year (2000, 2022,etc)
- %y: 2-digit 0-padded year without century (eg. 00,22,etc)

Let' say we just want to output the abbreviated month and the 2 digit year

```
# include abbreviated month and 2 digit year
format_d = "%b %y"
now.strftime(format_d)
```

```
'Jul 22'
```

For Time:

- For quick/easy 24-hour military time with minutes and seconds:
  - %T (e.g. 17:04:49)
- For the hour of the day:
  - %H: 0-padded 24-hour time (e.g. 01, 12, 13,23)

    - %I: 0-padded 12-hour time (e.g. 01, 12, 01, 11)
        - %p: AM/PM
- For minutes:
    - %M: 0-padded 2-digit minutes
- For seconds:
    - %S

The code below defines the default datetime format with all of the codes used.

```
## the default datetime format
default ="%Y-%m-%d %H:%M:%S.%f"
print(now)
now.strftime(default)
```

```
2022-07-25 16:47:27.090245
'2022-07-25 16:47:27.090245'
```

Let's try some time formats. Include the date with slashed between and add the 12 hour time with AM or PM. Do not include milliseconds.

```
# Example US Formatted 12-hour date/time with AM/PM
fmt = "%m/%d/%Y %I:%M:%S %p"
now.strftime(fmt)
```

```
'07/25/2022 04:47:27 PM'
```

Let's try just the 24 hour time with no date:

```
## just the time (24 hour time)
now.strftime("%T")
```

```
'16:47:27'
```

Summary

This lesson introduced time series as any data that has a time, date, or time and date component. We also explored the datetime class in basic Python and demonstrated several options for changing the formatting with .strftime(). There are so many options for formatting! You won't remember the codes, but you should know where to find them and how to apply them to achieve what you are looking for in your datetime output.

Key Terms

time series

frequency

Previous

Next