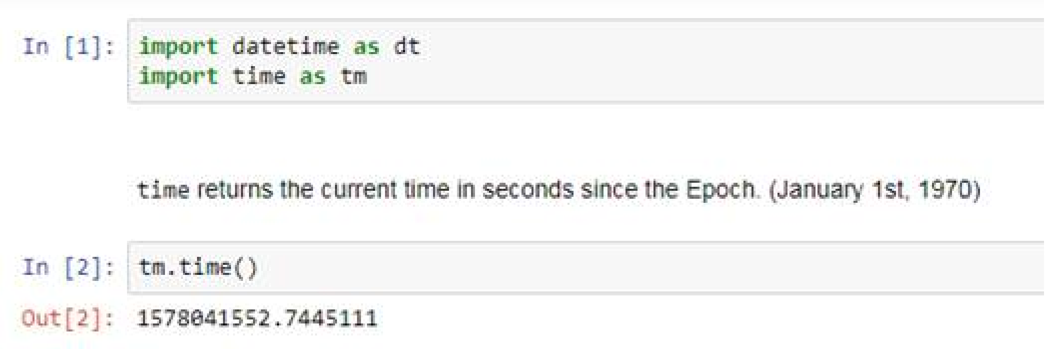
Python Dates and Times

A lot of analysis you do might relate to dates and times.
For instance, finding the average number of sales over a given period, selecting a list of products to data mine if they were purchased in a given period, or trying to find the period with the most activity in online discussion forum systems.

First, you should be aware that date and times can be stored in many different ways.
One of the most common legacy methods for storing the date and time in online transactions systems is based on the offset from the epoch, which is January 1, 1970.
There's a lot of historical cruft around this, but it's not uncommon to see systems storing the date of a transaction in seconds or milliseconds since this date.
So if you see large numbers where you expect to see date and time, you'll need to convert them to make much sense out of the data.

In Python, you can get the current time since the epoch using the time module.



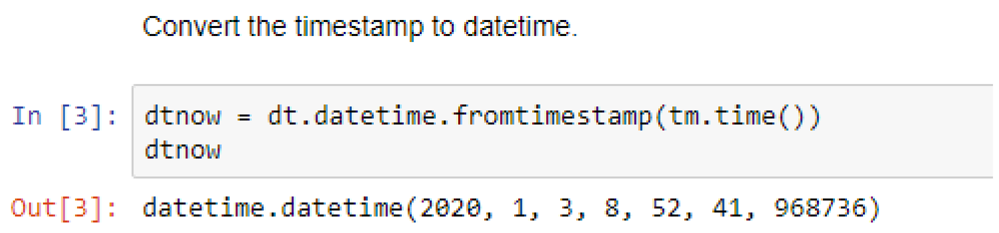
```
In [1]: import datetime as dt
        import time as tm
```

time returns the current time in seconds since the Epoch. (January 1st, 1970)

```
In [2]: tm.time()

Out[2]: 1578041552.7445111
```

You can then create a time stamp using the from time stamp function on the date time object.



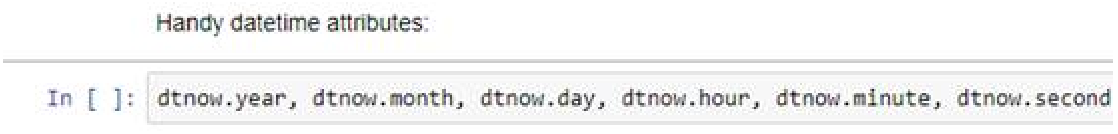
Convert the timestamp to datetime.

```
In [3]: dtnow = dt.datetime.fromtimestamp(tm.time())
        dtnow

Out[3]: datetime.datetime(2020, 1, 3, 8, 52, 41, 968736)
```

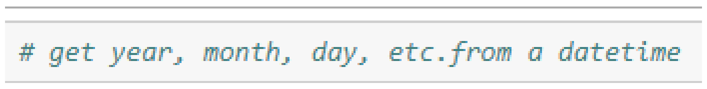
When we print this value out, we see that the year, month, day, and so forth are also printed out.

The date time object has handy attributes to get the representative hour, day, seconds, etc.



Handy datetime attributes:

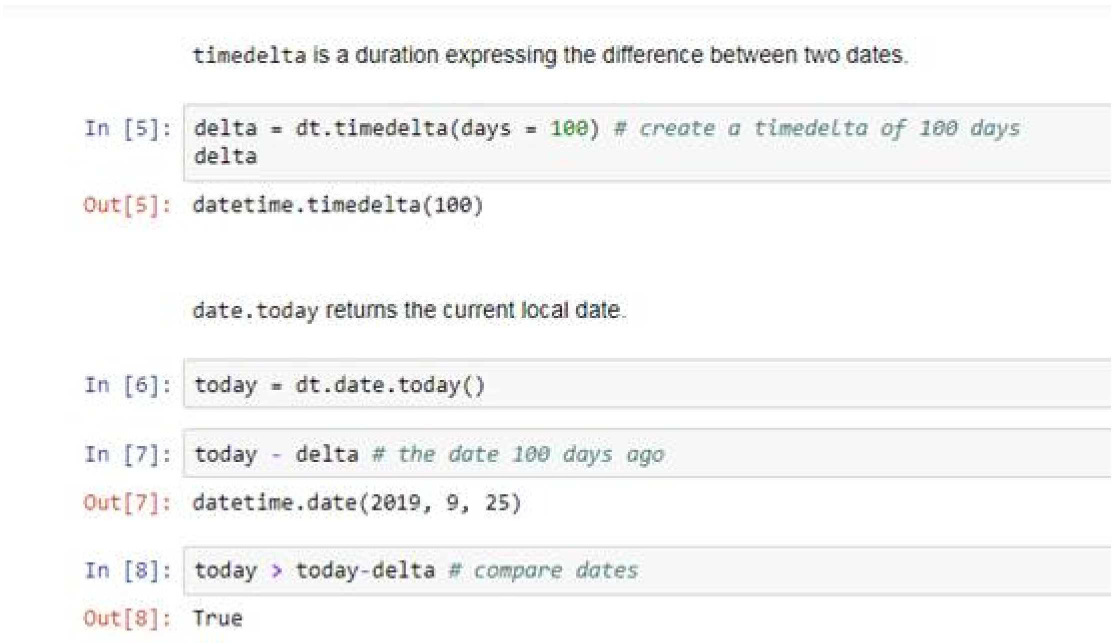
```
In [ ]: dtnow.year, dtnow.month, dtnow.day, dtnow.hour, dtnow.minute, dtnow.second
```



```
# get year, month, day, etc.from a datetime
```

Date time objects allow for simple math using time deltas.
For instance, here, we can create a time delta of 100 days, then do subtraction and comparisons with the date time object.

This is commonly used in data science for creating sliding windows.
For instance, where you might want to look for any five day span of time where sales were highest, and flag that for follow up.



`timedelta` is a duration expressing the difference between two dates.

```
In [5]: delta = dt.timedelta(days = 100) # create a timedelta of 100 days
        delta

Out[5]: datetime.timedelta(100)
```

`date.today` returns the current local date.

```
In [6]: today = dt.date.today()

In [7]: today - delta # the date 100 days ago

Out[7]: datetime.date(2019, 9, 25)

In [8]: today > today-delta # compare dates

Out[8]: True
```

This is just a little glimpse at dates and times in Python.
In week three of this course, we're going to investigate dates and times a bit more using the pandas date time library.