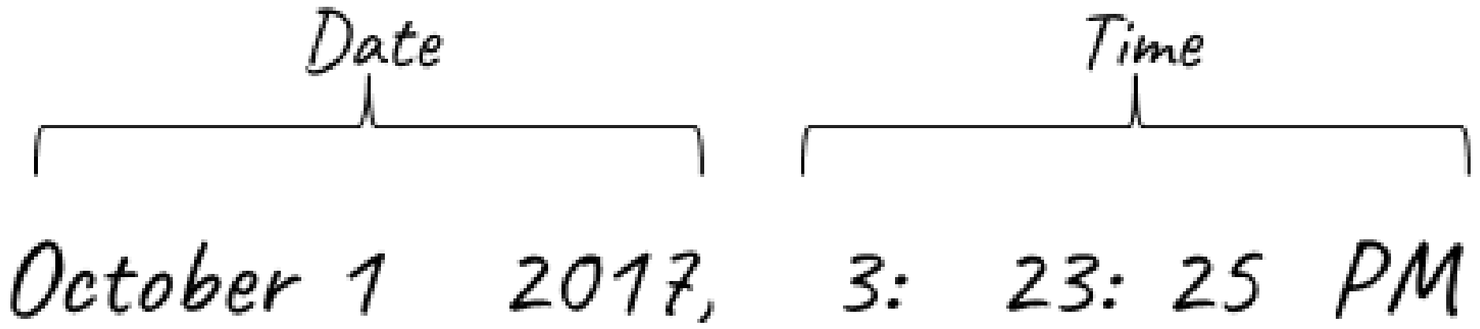
# Adding time to the mix

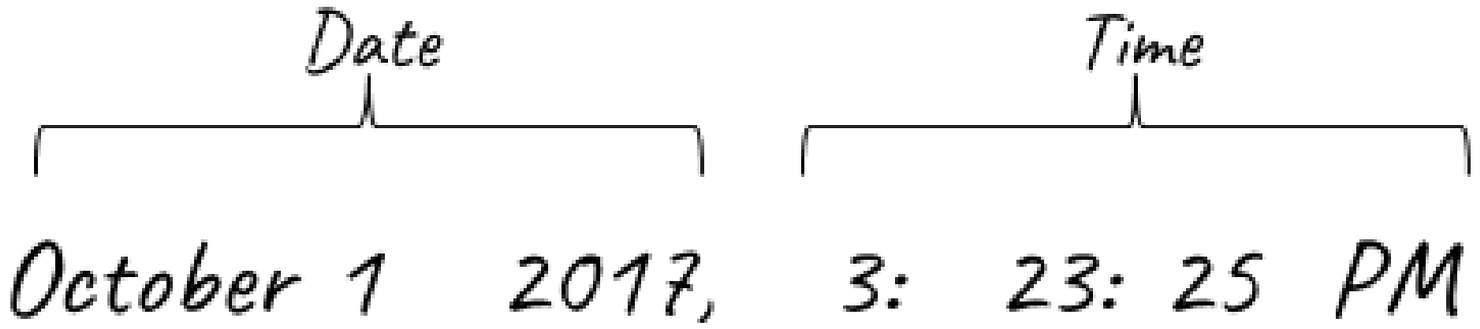
WORKING WITH DATES AND TIMES IN PYTHON

**Max Shron**
Data Scientist and Author

# Dates and Times

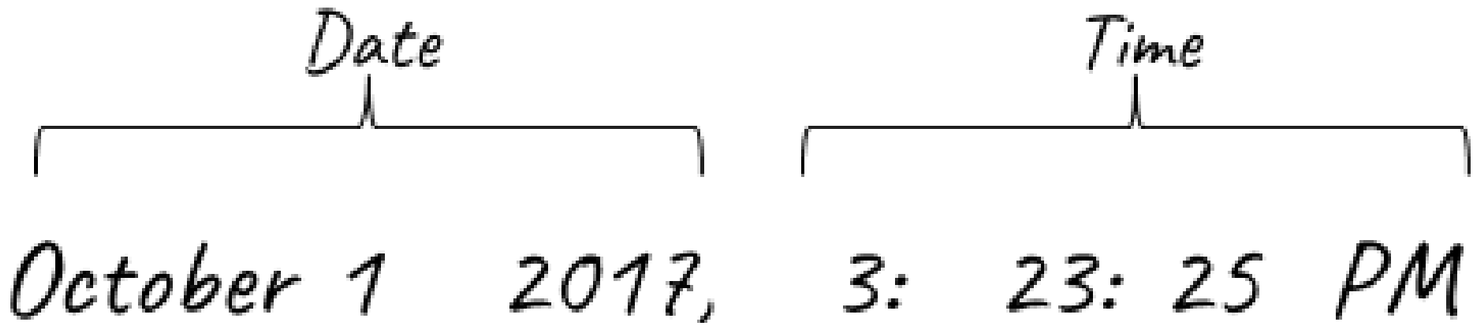


# Dates and Times




```
# Import datetime
from datetime import datetime
```

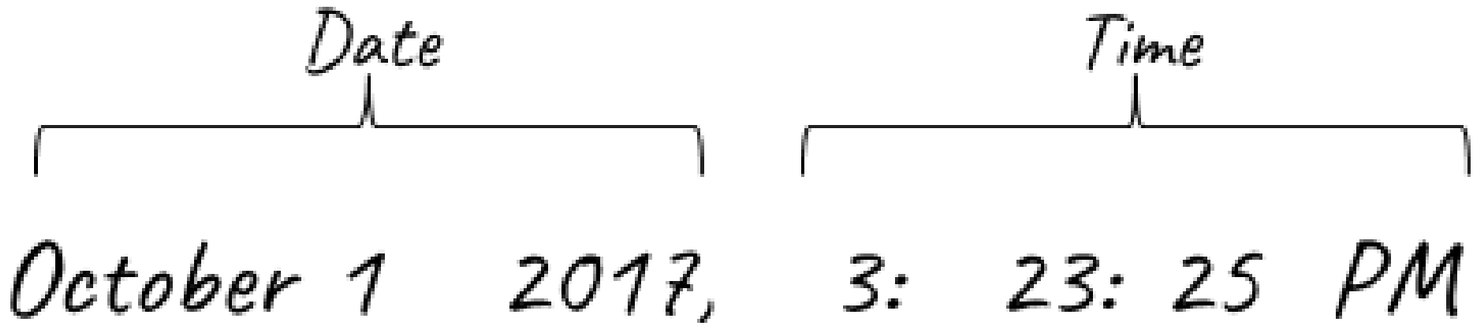
# Dates and Times




```
# Import datetime
from datetime import datetime

dt = datetime(
```

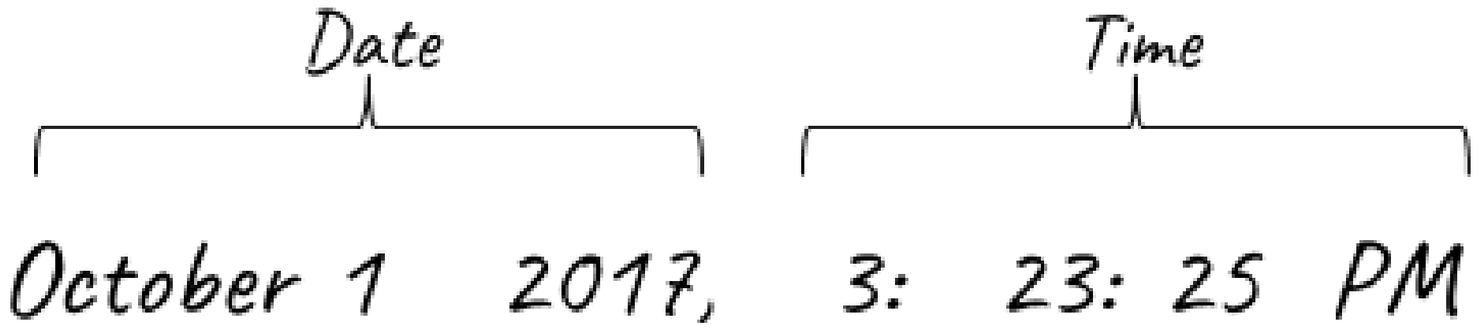
# Dates and Times




```
# Import datetime
from datetime import datetime

dt = datetime(2017, 10, 1
```

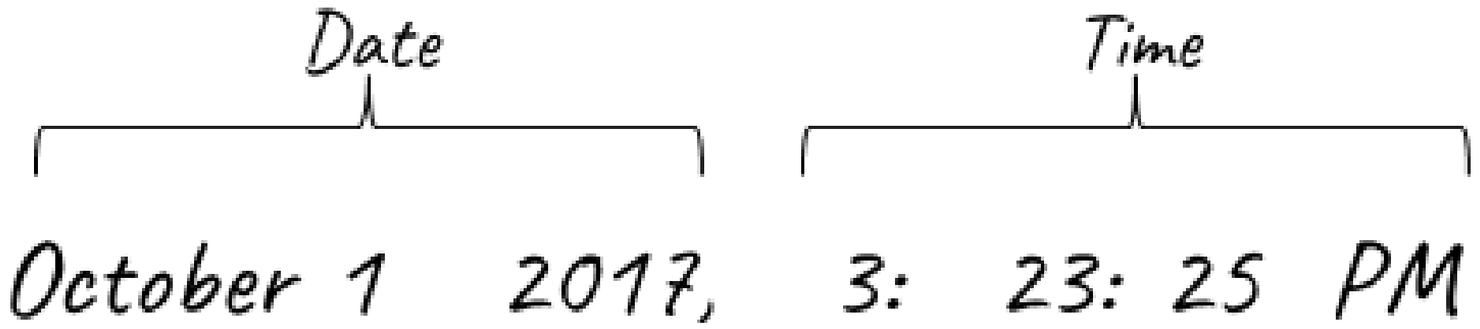
# Dates and Times




```
# Import datetime
from datetime import datetime

dt = datetime(2017, 10, 1, 15
```

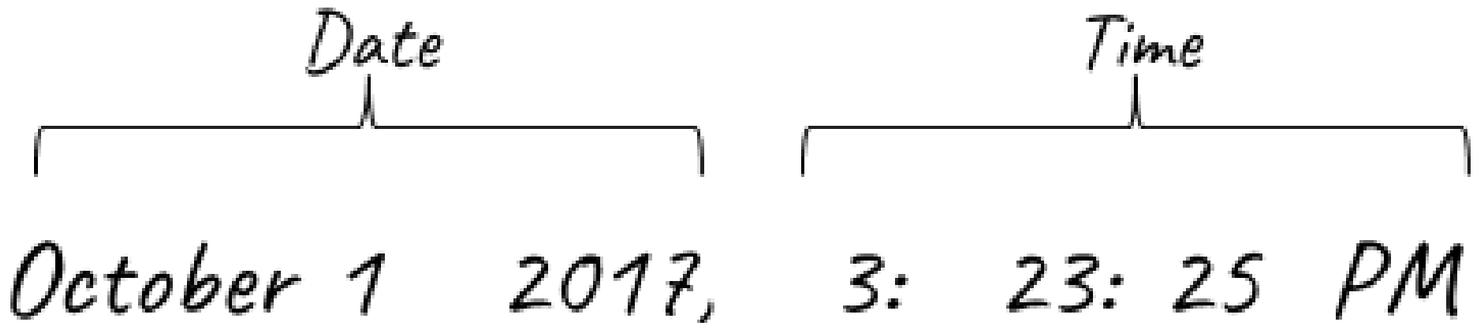
# Dates and Times




```
# Import datetime
from datetime import datetime

dt = datetime(2017, 10, 1, 15, 23,
```

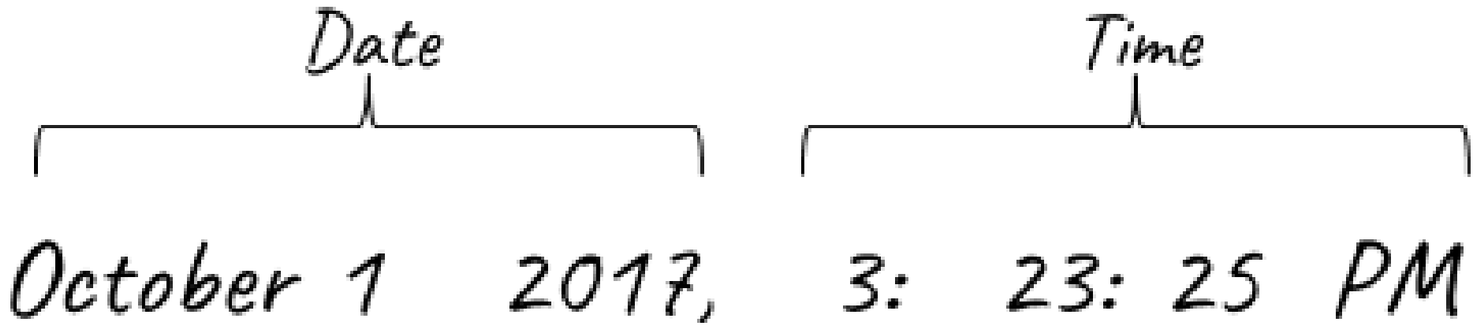
# Dates and Times




```
# Import datetime
from datetime import datetime

dt = datetime(2017, 10, 1, 15, 23, 25)
```

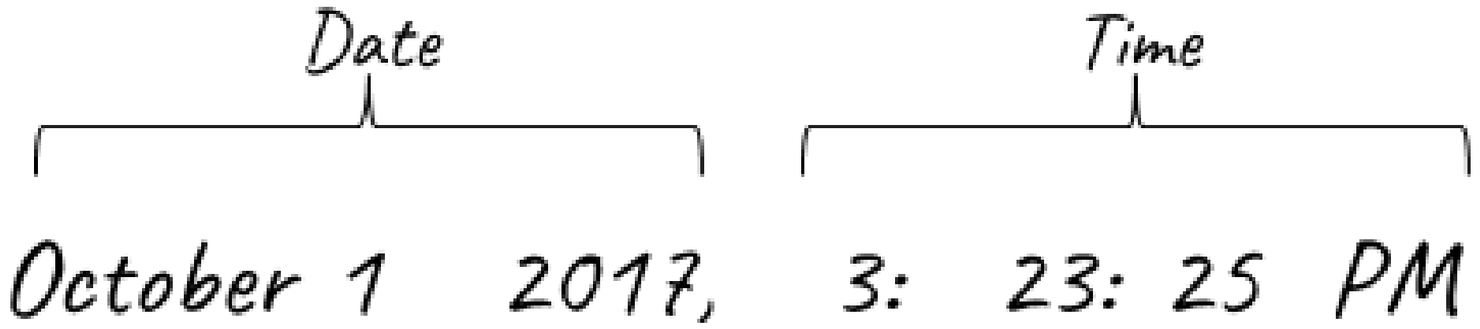
# Dates and Times




```
# Import datetime
from datetime import datetime

dt = datetime(2017, 10, 1, 15, 23, 25, 500000)
```

# Dates and Times




```
# Import datetime
from datetime import datetime

dt = datetime(year=2017, month=10, day=1,
              hour=15, minute=23, second=25,
              microsecond=500000)
```

# Replacing parts of a datetime

```
print(dt)
```

```
2017-10-01 15:23:25.500000
```

```
dt_hr = dt.replace(minute=0, second=0, microsecond=0)
print(dt_hr)
```

```
2017-10-01 15:00:00
```

# Capital Bikeshare


*Capital Bikeshare Station Installed at the Lincoln Memorial* by *Euan Fisk*, licensed *CC B 2.0*

# Adding time to the mix

WORKING WITH DATES AND TIMES IN PYTHON

# Printing and parsing datetimes

WORKING WITH DATES AND TIMES IN PYTHON

**Max Shron**
Data Scientist and Author

datacamp

# Printing datetimes

```
# Create datetime
dt = datetime(2017, 12, 30, 15, 19, 13)
print(dt.strftime("%Y-%m-%d"))
```

```
2017-12-30
```

```
print(dt.strftime("%Y-%m-%d %H:%M:%S"))
```

```
2017-12-30 15:19:13
```

# Printing datetimes

```
print(dt.strftime("%H:%M:%S on %d/%m/%Y"))
```

```
15:19:13 on 2017/12/30
```

# ISO 8601 Format

```
# ISO 8601 format
print(dt.isoformat())
```

```
2017-12-30T15:19:13
```

4. ISO 8601 Format
Finally, we can use the isoformat() method, just like with dates, to get a standards-compliant way of writing down a datetime. The officially correct way of writing a datetime is the year, month, day, then a capital T, then the time in 24 hour time, followed by the minute and second. When in doubt, this is a good format to use.

# Parsing datetimes with strptime

```
# Import datetime
from datetime import datetime
```

# Parsing datetimes with strptime

```
# Import datetime
from datetime import datetime


dt = datetime.strptime(
```

6. Parsing datetimes with strptime
The method we're going to use is called strptime(), which is short for string parse time. strptime() takes two arguments: the first argument is the string to turn into a datetime, and the second argument is the format string that says how to do it.

# Parsing datetimes with strptime

```
# Import datetime
from datetime import datetime


dt = datetime.strptime("12/30/2017 15:19:13"
```

# Parsing datetimes with strptime

```
# Import datetime
from datetime import datetime

dt = datetime.strptime("12/30/2017  15:19:13",
                       "%m/%d/%Y %H:%M:%S")
```

8. Parsing datetimes with strptime
Then we pass the format string, which as mentioned before uses the same format codes we used with strftime(). In this case, first the month, then the day, then the year, all separated by slashes, then a space, and then the hour, minutes, and seconds separated by colons. You usually need to figure this out once per data set.

# Parsing datetimes with strptime

```
# What did we make?
print(type(dt))
```

```
<class 'datetime.datetime'>
```

```
# Print out datetime object
print(dt)
```

```
2017-12-30 15:19:13
```

# Parsing datetimes with strptime

```
# Import datetime
from datetime import datetime


# Incorrect format string
dt = datetime.strptime("2017-12-30 15:19:13", "%Y-%m-%d")
```

```
ValueError: unconverted data remains:  15:19:13
```

10. Parsing datetimes with strptime
We need an exact match to do a string conversion. For example, if we leave out how to parse the time, Python will throw an error. And similarly, if there is an errant comma or other symbols, strptime() will not be happy.

# Parsing datetimes with Pandas

```
# A timestamp
ts = 1514665153.0
# Convert to datetime and print
print(datetime.fromtimestamp(ts))
```

```
2017-12-30 15:19:13
```

11. Parsing datetimes with Pandas
Finally, there is another kind of datetime you will sometimes encounter: the Unix timestamp. Many computers store datetime information behind the scenes as the number of seconds since January 1, 1970. This date is largely considered the birth of modern-style computers. To read a Unix timestamp, use the datetime.fromtimestamp() method. Python will read your timestamp and return a datetime.

# Printing and parsing datetimes

WORKING WITH DATES AND TIMES IN PYTHON

# Working with durations

WORKING WITH DATES AND TIMES IN PYTHON

**Max Shron**
Data Scientist and Author

datacamp

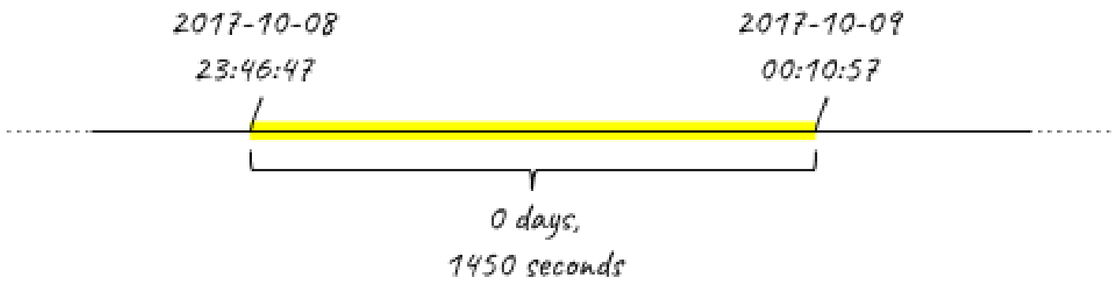
# Working with durations

# Working with durations


```
# Create example datetimes
start = datetime(2017, 10, 8, 23, 46, 47)
end = datetime(2017, 10, 9, 0, 10, 57)
```

```
# Subtract datetimes to create a timedelta
duration = end - start
```

# Working with durations




```
# Subtract datetimes to create a timedelta
print(duration.total_seconds())
```

```
1450.0
```

# Creating timedeltas

```
# Import timedelta
from datetime import timedelta
```

```
# Create a timedelta
delta1 = timedelta(seconds=1)
```

# Creating timedeltas

```
print(start)
```

```
2017-10-08 23:46:47
```

```
# One second later
print(start + delta1)
```

```
2017-10-08 23:46:48
```

# Creating timedeltas

```
# Create a one day and one second timedelta
delta2 = timedelta(days=1, seconds=1)
```

```
print(start)
```

```
2017-10-08 23:46:47
```

```
# One day and one second later
print(start + delta2)
```

```
2017-10-09 23:46:48
```

7. Creating timedeltas
We also create a timedelta, delta2, which is one day and one second in duration. Now when we add it to start, we get a new datetime which is the next day and one second later.
Timedeltas can be created with any number of weeks, days, minutes, hours, seconds, or microseconds, and can be as small as a microsecond or as large as 2.7 million years.

# Negative timedeltas

```
# Create a negative timedelta of one week
delta3 = timedelta(weeks=-1)
```

```
print(start)
```

```
2017-10-08 23:46:47
```

```
# One week earlier
print(start + delta3)
```

```
2017-10-01 23:46:47
```

# Negative timedeltas

```
# Same, but we'll subtract this time
delta4 = timedelta(weeks=1)
```

```
print(start)
```

```
2017-10-08 23:46:47
```

```
# One week earlier
print(start - delta4)
```

```
2017-10-01 23:46:47
```

# Working with durations

WORKING WITH DATES AND TIMES IN PYTHON