Dates in Python

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author

Course overview

- Chapter 1: Dates and Calendars
- Chapter 2: Combining Dates and Times
- Chapter 3: Time zones and Daylight Saving
- Chapter 4: Dates and Times in Pandas

Dates in Python

string

"Hello"

array

number

42

date

Why do we need a date class in Python?

```
two_hurricanes = ["10/7/2016", "6/21/2017"]
```

How would you:

- Figure out how many days had elapsed?
- Check that they were in order from earliest to latest?
- Know which day of the week each was?
- Filter out hurricanes which happened between certain dates?

Creating date objects

```
# Import date
from datetime import date

# Create dates
two_hurricanes_dates = [date(2016, 10, 7), date(2017, 6, 21)]
```

Attributes of a date

```
# Import date
from datetime import date

# Create dates
two_hurricanes_dates = [date(2016, 10, 7), date(2017, 6, 21)]
print(two_hurricanes_dates[0].year)
print(two_hurricanes_dates[0].month)
print(two_hurricanes_dates[0].day)
```

```
2016
10
7
```

Finding the weekday of a date

```
print(two_hurricanes_dates[0].weekday())
```

```
4
```

- Weekdays in Python
  - 0 = Monday
  - 1 = Tuesday
  - 2 = Wednesday
  - ...
  - 6 = Sunday

Dates in Python

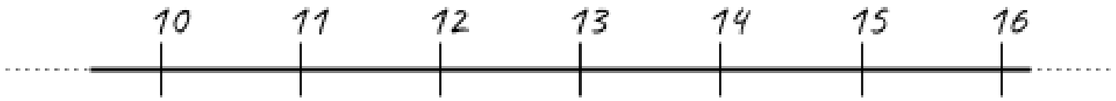
WORKING WITH DATES AND TIMES IN PYTHON

Math with Dates

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author

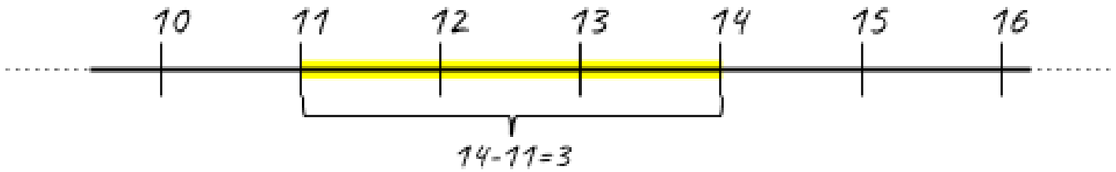
Math with dates




```
# Example numbers
a = 11
b = 14
l = [a, b]

# Find the least least in the list
print(min(l))
11
```

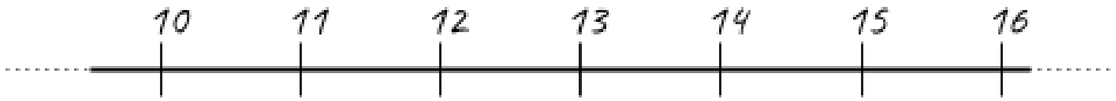
Math with dates



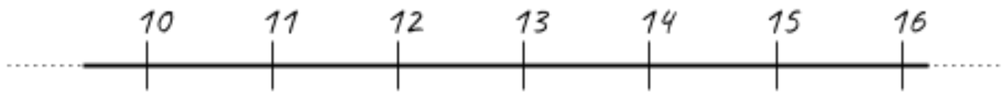

```
# Subtract two numbers
print(b - a)
3

# Add 3 to a
print(a + 3)
14
```

Math with dates



Math with dates

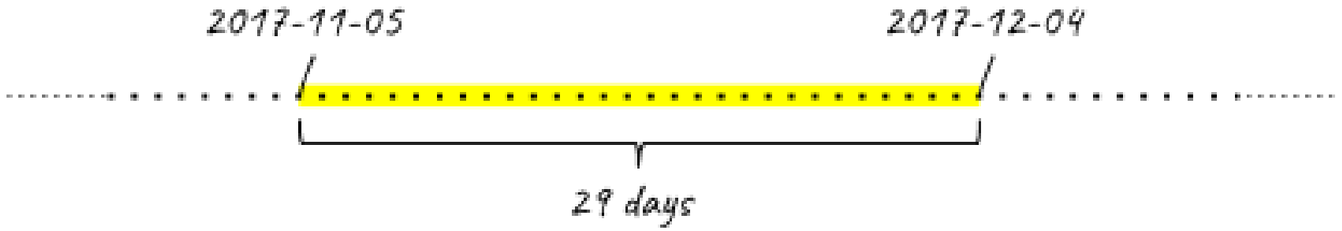



```
# Import date
from datetime import date

# Create our dates
d1 = date(2017, 11, 5)
d2 = date(2017, 12, 4)
l = [d1, d2]

print(min(l))
2017-11-05
```

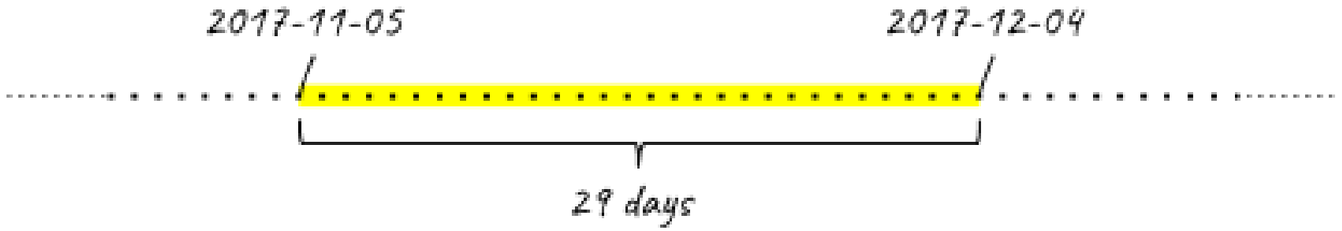
Math with dates




```
# Subtract two dates
delta = d2 - d1

print(delta.days)
```
```
29
```

Math with dates




```
# Import timedelta
from datetime import timedelta

# Create a 29 day timedelta
td = timedelta(days=29)
print(d1 + td)
2017-12-04
```

Incrementing variables with +=

```
# Initialize x to be zero
x = 0

# Increment x
x = x + 1

print(x)
1
```

```
# Initialize x to be zero
x = 0

# Increment x
x += 1

print(x)
1
```

Let's Practice!

WORKING WITH DATES AND TIMES IN PYTHON

Turning dates into strings

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author

ISO 8601 format

```
from datetime import date

# Example date
d = date(2017, 11, 5)
# ISO format: YYYY-MM-DD
print(d)
```

```
2017-11-05
```

```
# Express the date in ISO 8601 format and put it in a list
print( [d.isoformat()] )
```

```
['2017-11-05']
```

ISO 8601 format

```
# A few dates that computers once had trouble with
some_dates = ['2000-01-01', '1999-12-31']

# Print them in order
print(sorted(some_dates))
```

```
['1999-12-31', '2000-01-01']
```

Every other format

```
d.strftime()
```

Every other format: strftime

```
# Example date
d = date(2017, 1, 5)

print(d.strftime("%Y"))
```

```
2017
```

```
# Format string with more text in it
print(d.strftime("Year is %Y"))
```

```
Year is 2017
```

Every other format: strftime

```
# Format: YYYY/MM/DD
print(d.strftime("%Y/%m/%d"))
```

```
2017/01/05
```

Turning dates into strings

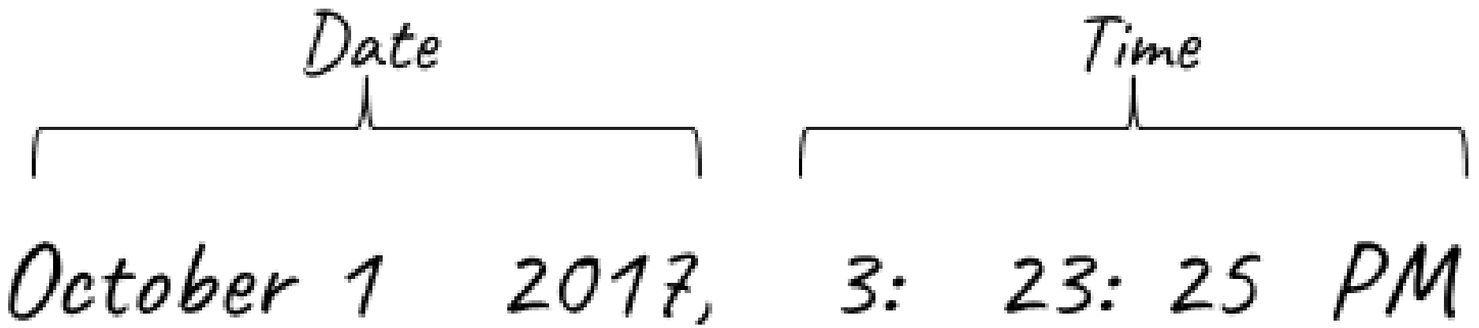
WORKING WITH DATES AND TIMES IN PYTHON

Adding time to the mix

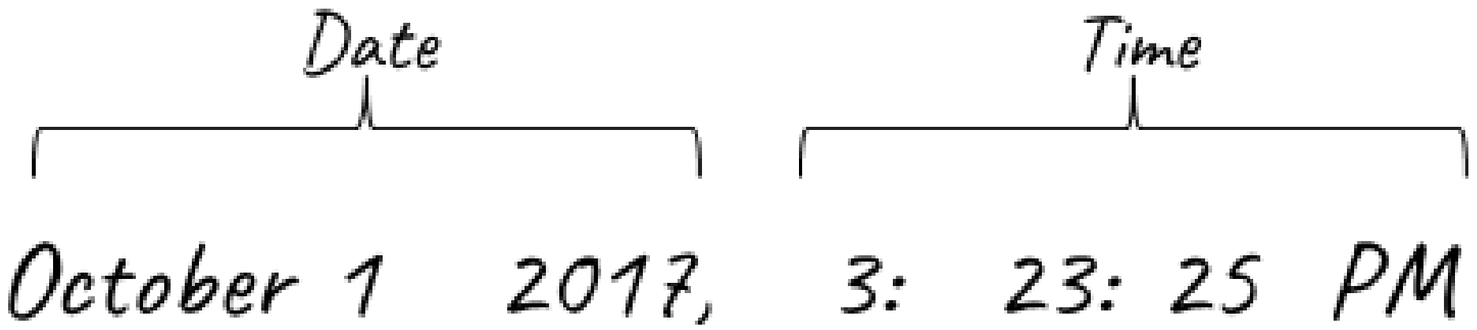
WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author

Dates and Times

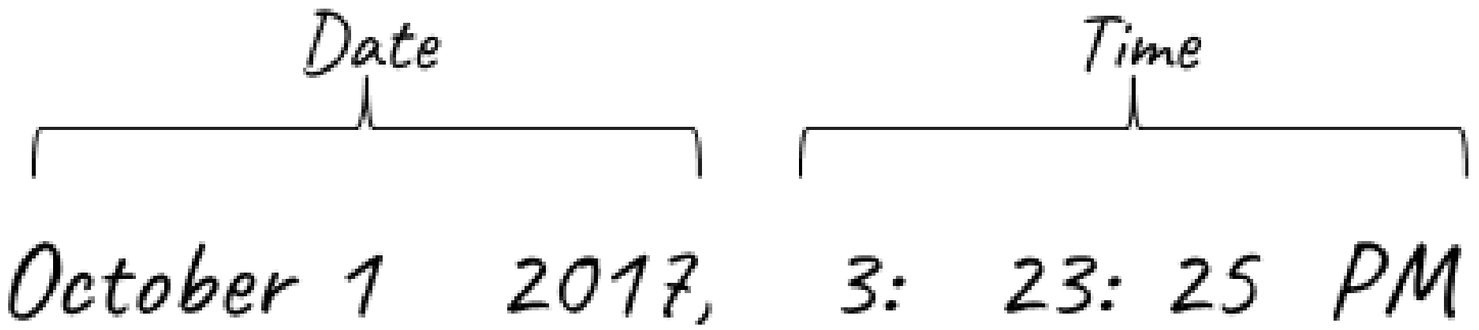


Dates and Times




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

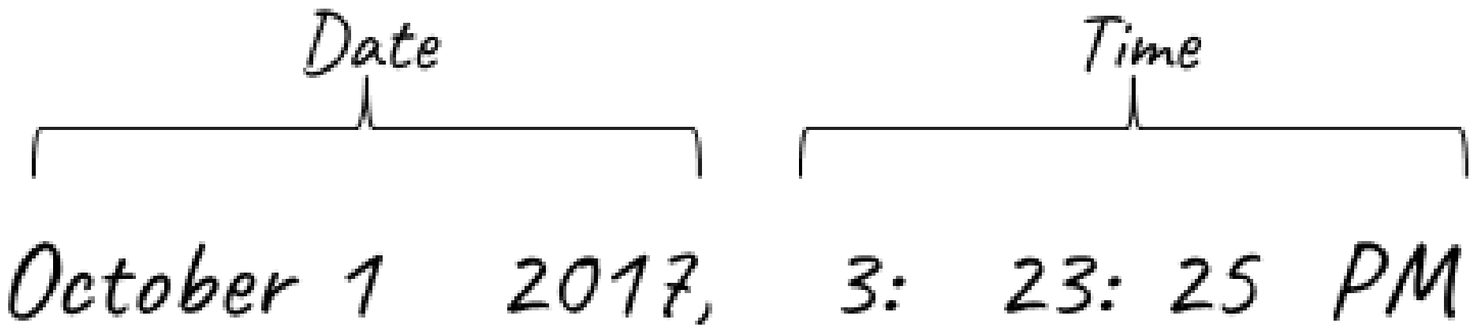
Dates and Times




```
# Import datetime
from datetime import datetime


dt = datetime(
```

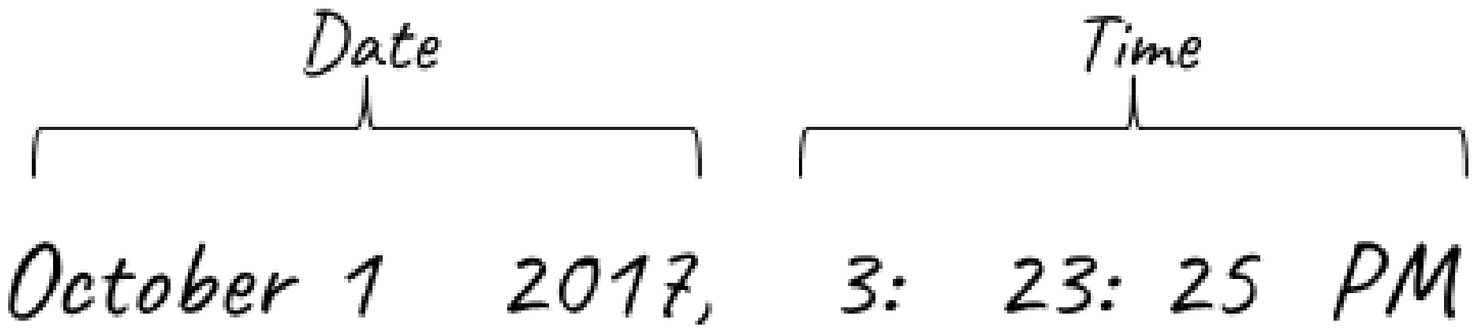
Dates and Times




```
# Import datetime
from datetime import datetime


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

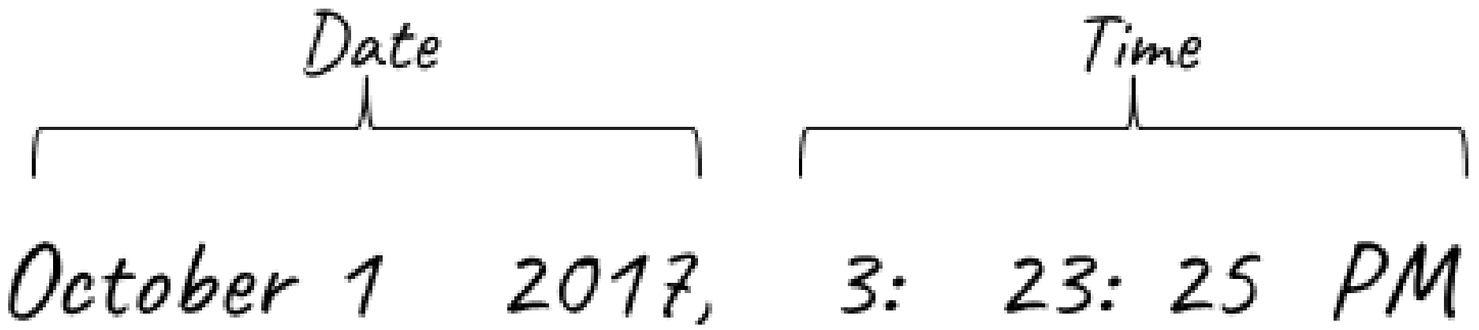
Dates and Times




```
# Import datetime
from datetime import datetime

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

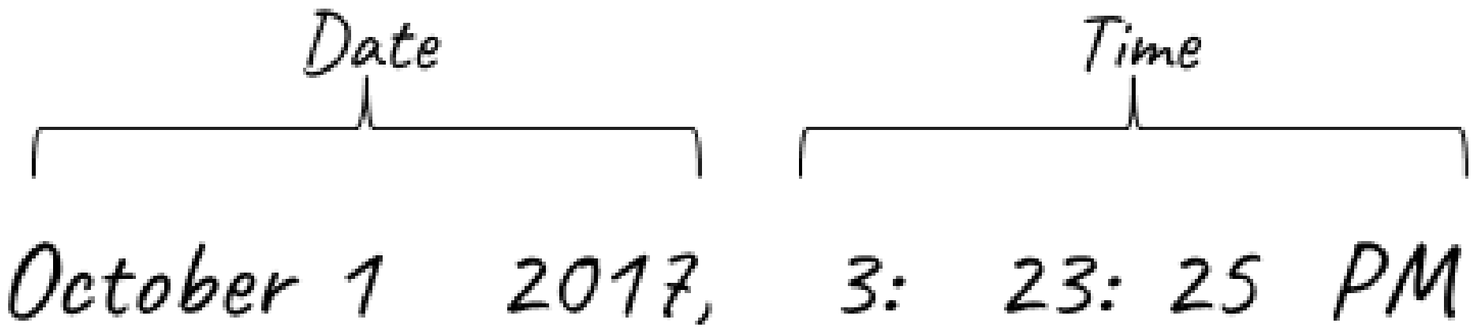
Dates and Times




```
# Import datetime
from datetime import datetime


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

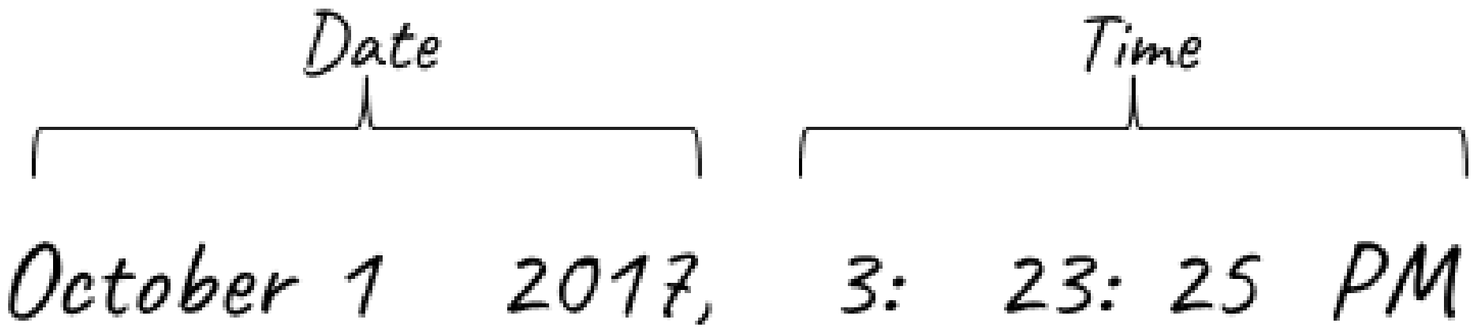
Dates and Times




```
# Import datetime
from datetime import datetime


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

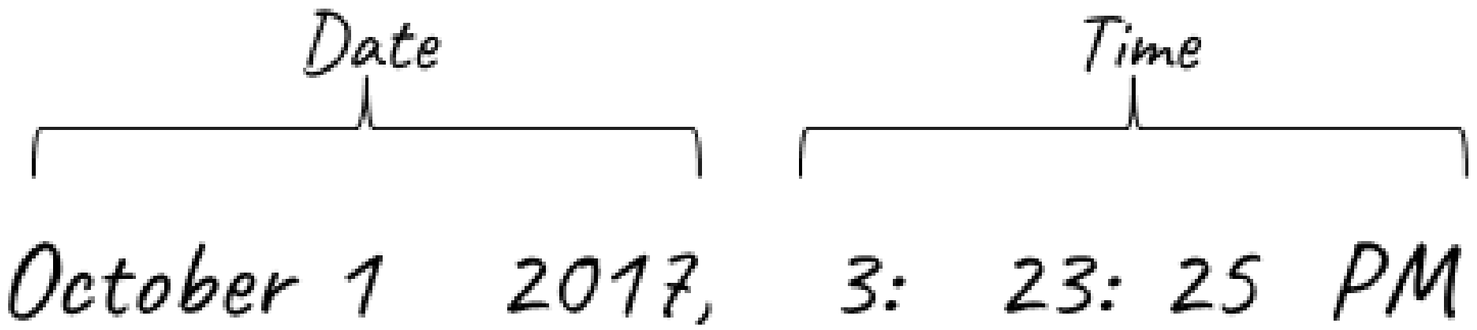
Dates and Times




```
# 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

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
```

Parsing datetimes with strptime

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

Parsing datetimes with strptime

```
# Import datetime
from datetime import datetime

dt = datetime.strptime(
```

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")
```

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
```

Parsing datetimes with Pandas

```
# A timestamp
ts = 1514665153.0

# Convert to datetime and print
print(datetime.fromtimestamp(ts))
```

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

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

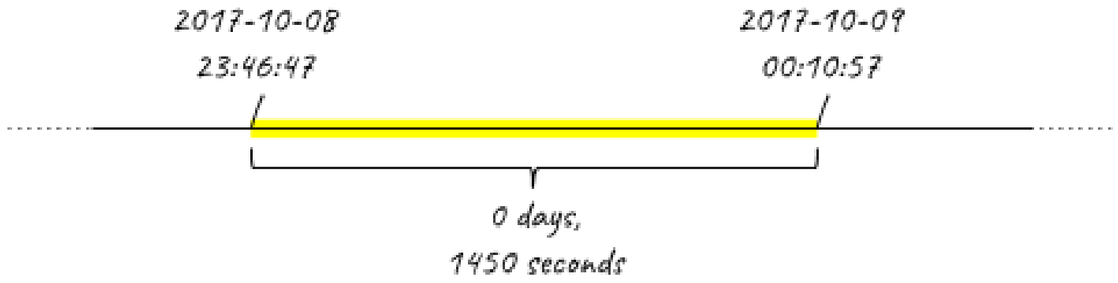
Working with durations

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
```

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
```

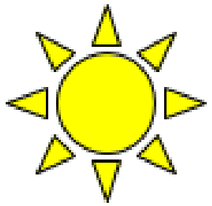
Working with durations

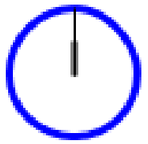
WORKING WITH DATES AND TIMES IN PYTHON

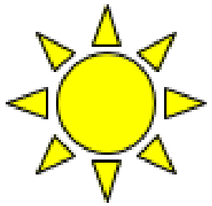
UTC offsets

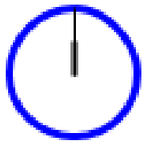
WORKING WITH DATES AND TIMES IN PYTHON

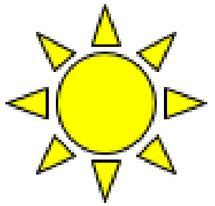
Max Shron
Data Scientist and Author

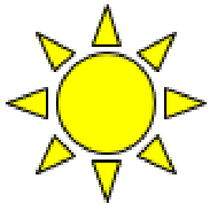


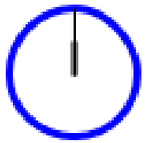


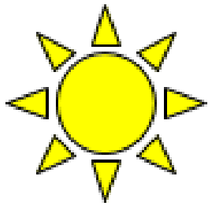


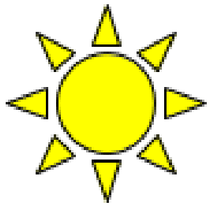


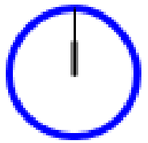


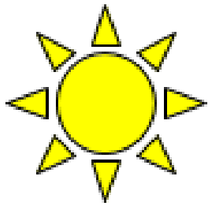




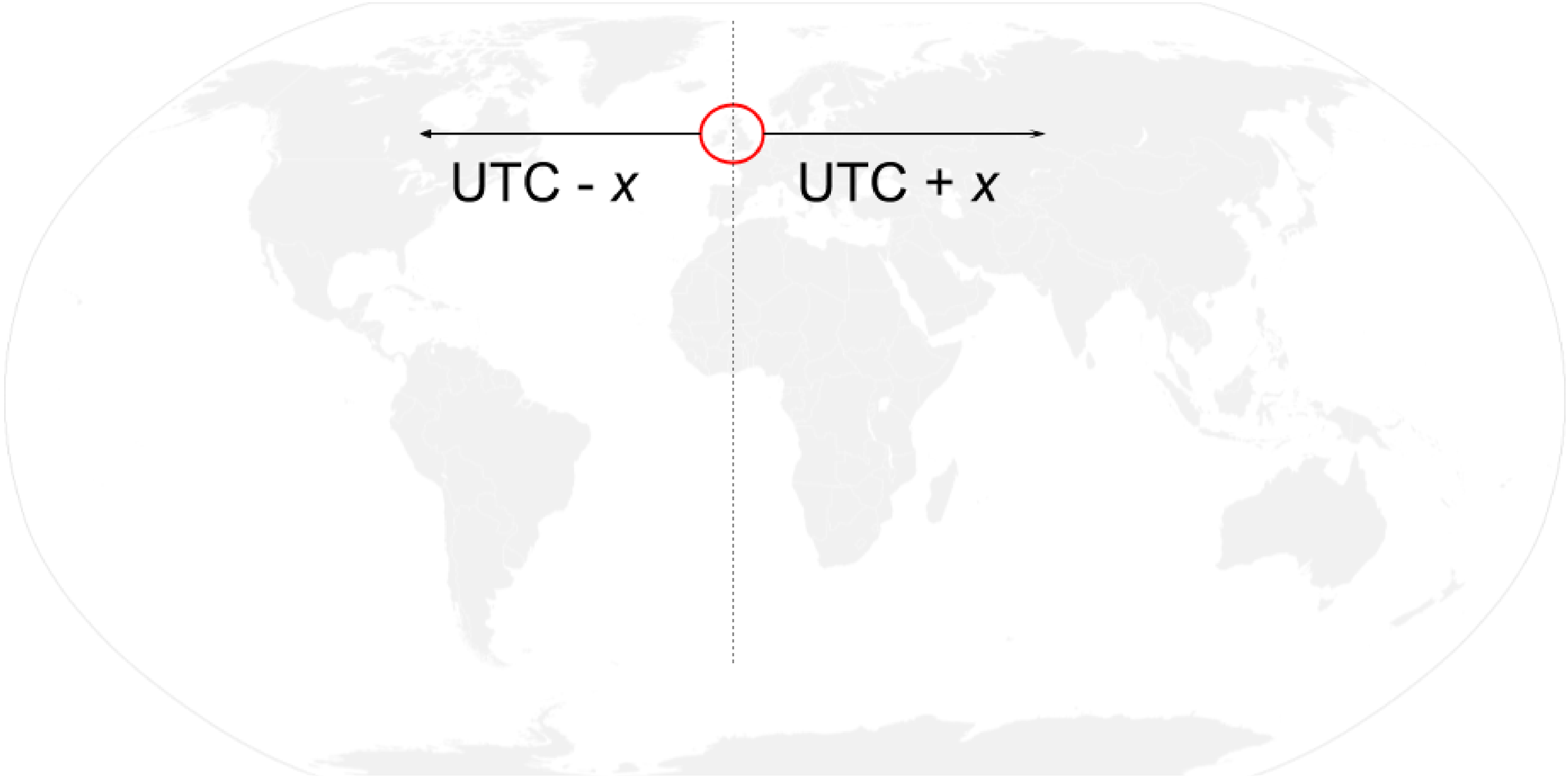









CANADA
SEATTLE
PORTLAND
PACIFIC
3 PM
MOUNTAIN
4 PM
SALT LAKE CITY
DENVER
SAN FRANCISCO
LAS VEGAS
LOS ANGELES
PHOENIX
EL PASO
MINNEAPOLIS
CHICAGO
CENTRAL
5 PM
OKLAHOMA CITY
DALLAS
MEMPHIS
DETROIT
BOSTON
NEW YORK
WASHINGTON
LOUISVILLE
ATLANTA
EASTERN
6 PM
PACIFIC OCEAN
ATLANTIC OCEAN


UTC - x
UTC + x

UTC

```
# Import relevant classes
from datetime import datetime, timedelta, timezone
```

UTC

```
# Import relevant classes
from datetime import datetime, timedelta, timezone

# US Eastern Standard time zone
ET = timezone(timedelta(hours=-5))

# Timezone-aware datetime
dt = datetime(2017, 12, 30, 15, 9, 3, tzinfo = ET)
```

```
print(dt)
```

```
'2017-12-30 15:09:03-05:00'
```

UTC

```
# India Standard time zone
IST = timezone(timedelta(hours=5, minutes=30))

# Convert to IST
print(dt.astimezone(IST))
```

```
'2017-12-31 01:39:03+05:30'
```

Adjusting timezone vs changing tzinfo

```
print(dt)
```

```
'2017-12-30 15:09:03-05:00'
```

```
print(dt.replace(tzinfo=timezone.utc))
```

```
'2017-12-30 15:09:03+00:00'
```

```
# Change original to match UTC
print(dt.astimezone(timezone.utc))
```

```
'2017-12-30 20:09:03+00:00'
```

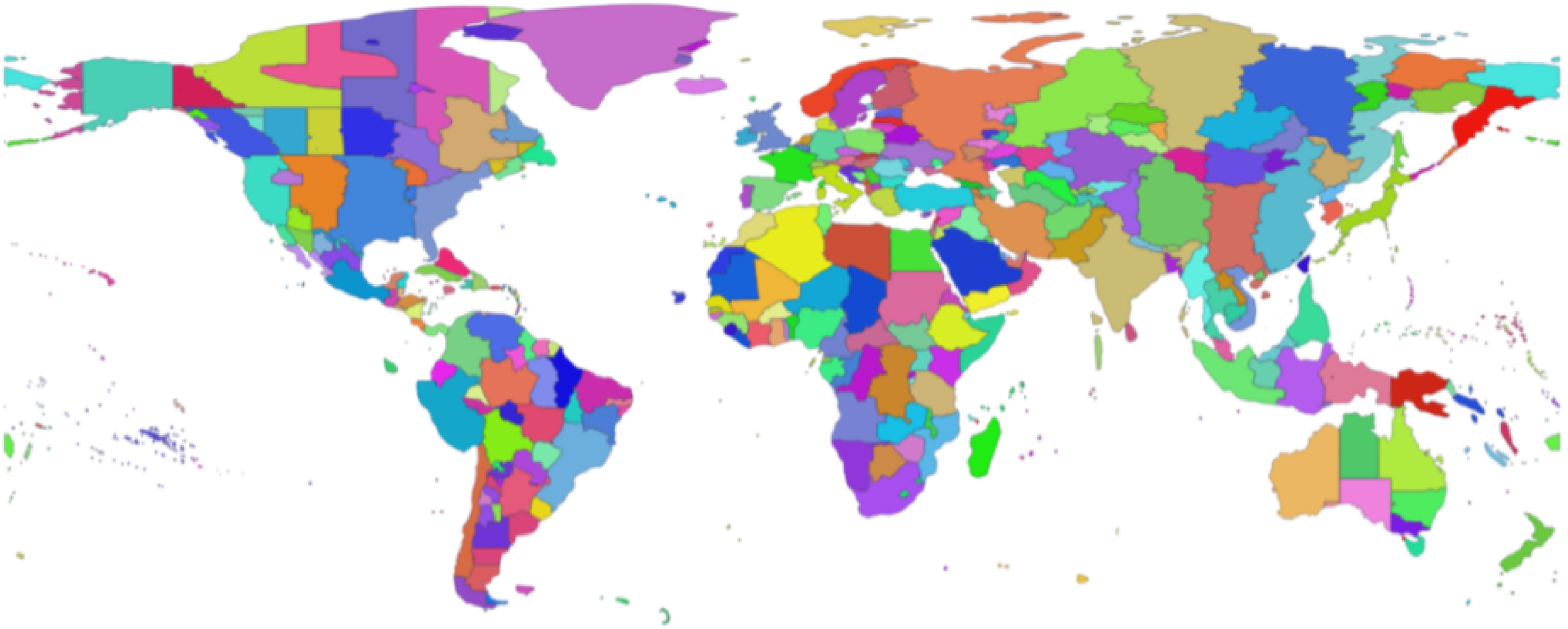
UTC Offsets

WORKING WITH DATES AND TIMES IN PYTHON

Time zone database

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author



Time zone database

```
# Imports
from datetime import datetime
from dateutil import tz
```

tz database

Time zone database

```
# Imports
from datetime import datetime
from dateutil import tz

# Eastern time
et = tz.gettz('America/New_York')
```

tz database

- Format: 'Continent/City'

Time zone database

```
# Imports
from datetime import datetime
from dateutil import tz

# Eastern time
et = tz.gettz('America/New_York')
```

tz database

- Format: 'Continent/City'
- Examples:
  - 'America/New_York'
  - 'America/Mexico_City'
  - 'Europe/London'
  - 'Africa/Accra'

Time zone database

```
# Last ride
last = datetime(2017, 12, 30, 15, 9, 3, tzinfo=et)
```

```
print(last)
```

```
'2017-12-30 15:09:03-05:00'
```

Time zone database

```
# Last ride
last = datetime(2017, 12, 30, 15, 9, 3, tzinfo=et)
print(last)
```

```
'2017-12-30 15:09:03-05:00'
```

```
# First ride
first = datetime(2017, 10, 1, 15, 23, 25, tzinfo=et)
print(first)
```

```
'2017-10-01 15:23:25-04:00'
```

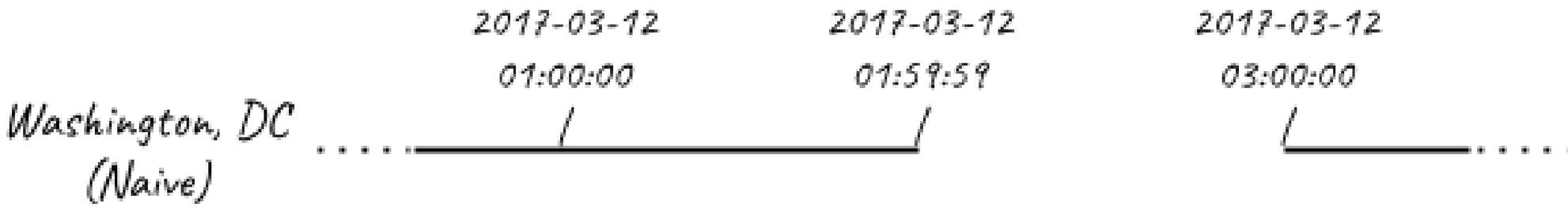
Time zone database

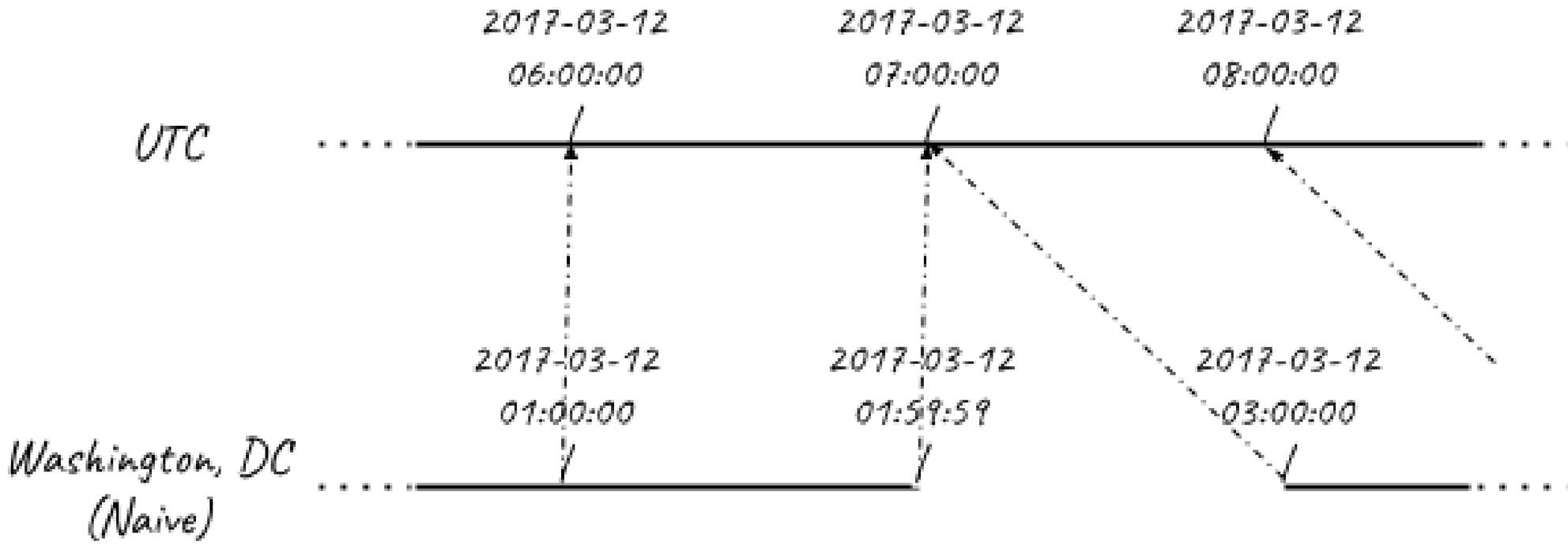
WORKING WITH DATES AND TIMES IN PYTHON

Starting Daylight Saving Time

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author


2017-03-12
01:00:00
2017-03-12
01:59:59
2017-03-12
03:00:00
Washington, DC
(Naive)


2017-03-12
06:00:00
2017-03-12
07:00:00
2017-03-12
08:00:00
UTC
2017-03-12
01:00:00
2017-03-12
01:59:59
2017-03-12
03:00:00
Washington, DC
(Naive)

Start of Daylight Saving Time

```
spring_ahead_159am = datetime(2017, 3, 12, 1, 59, 59)
spring_ahead_159am.isoformat()
```

```
'2017-03-12T01:59:59'
```

```
spring_ahead_3am = datetime(2017, 3, 12, 3, 0, 0)
spring_ahead_3am.isoformat()
```

```
'2017-03-12T03:00:00'
```

```
(spring_ahead_3am - spring_ahead_159am).total_seconds()
```

```
3601
```

Start of Daylight Saving Time

```
from datetime import timezone, timedelta

EST = timezone(timedelta(hours=-5))
EDT = timezone(timedelta(hours=-4))
```

Start of Daylight Saving Time

```
spring_ahead_159am = spring_ahead_159am.replace(tzinfo = EST)
spring_ahead_159am.isoformat()
```

```
'2017-03-12T01:59:59-05:00'
```

```
spring_ahead_3am = spring_ahead_159am.replace(tzinfo = EDT)
spring_ahead_3am.isoformat()
```

```
'2017-03-12T03:00:00-04:00'
```

```
(spring_ahead_3am - spring_ahead_159am).seconds
```

```
1
```

Start of Daylight Saving Time

Using `dateutil`

```
# Import tz
from dateutil import tz

# Create eastern timezone
eastern = tz.gettz('America/New_York')

# 2017-03-12 01:59:59 in Eastern Time (EST)
spring_ahead_159am = datetime(2017, 3, 12, 1, 59, 59,
                              tzinfo = eastern)

# 2017-03-12 03:00:00 in Eastern Time (EDT)
spring_ahead_3am = datetime(2017, 3, 12, 3, 0, 0,
                            tzinfo = eastern)
```

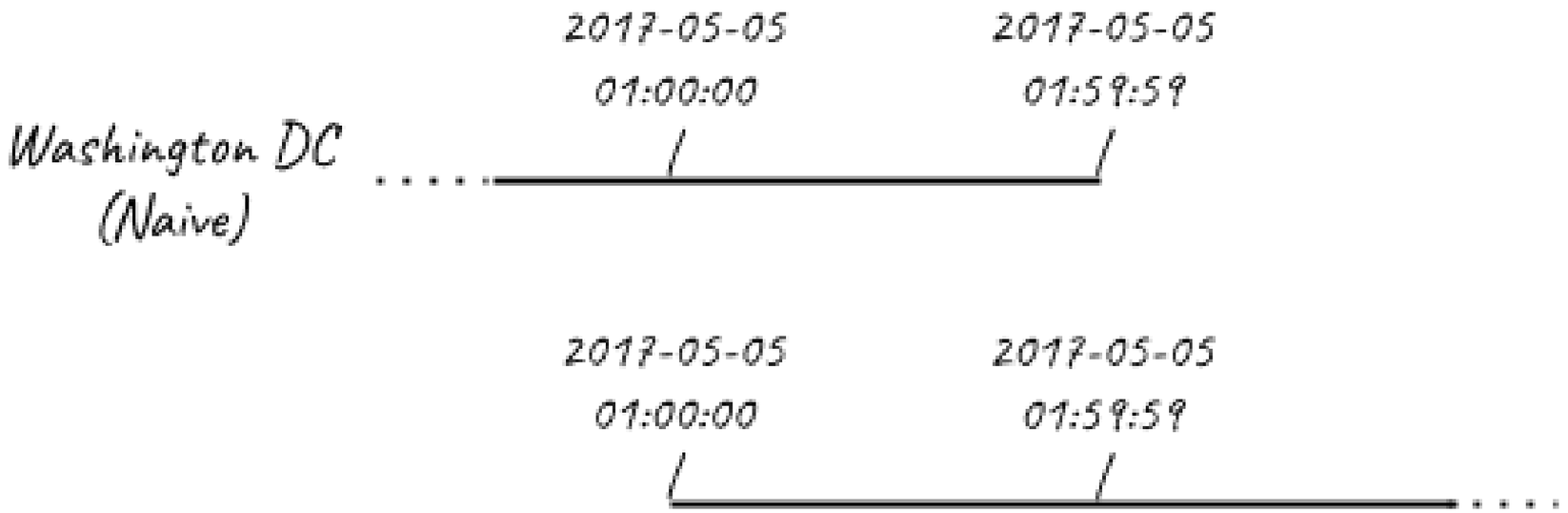
Daylight Saving

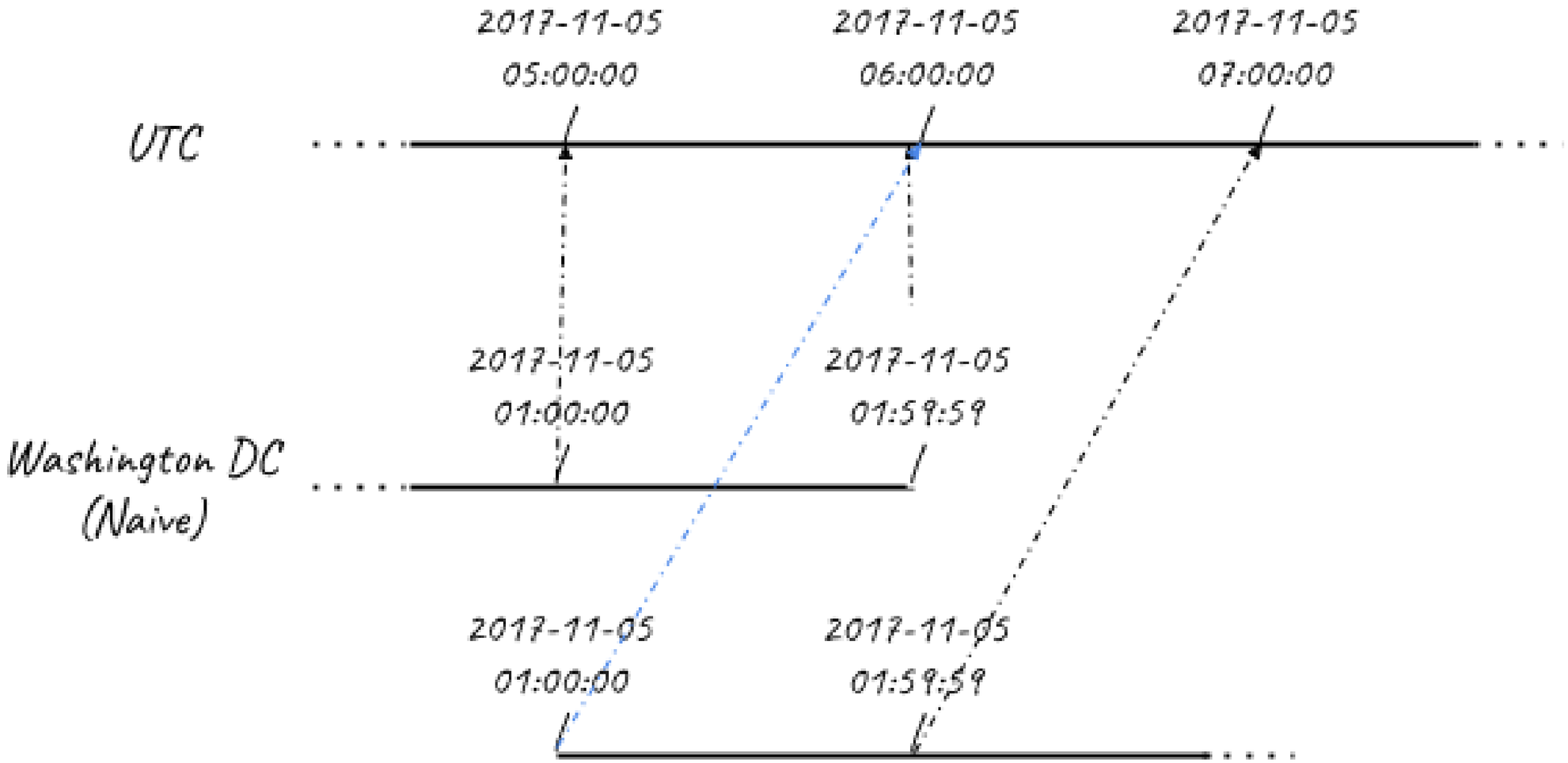
WORKING WITH DATES AND TIMES IN PYTHON

Ending Daylight Saving Time

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author


2017-05-05
01:00:00
2017-05-05
01:59:59
Washington DC
(Naive)
2017-05-05
01:00:00
2017-05-05
01:59:59


2017-11-05
05:00:00
2017-11-05
06:00:00
2017-11-05
07:00:00
UTC
2017-11-05
01:00:00
2017-11-05
01:59:59
Washington DC
(Naive)
2017-11-05
01:00:00
2017-11-05
01:59:59

Ending Daylight Saving Time

```
eastern = tz.gettz('US/Eastern')
# 2017-11-05 01:00:00
first_1am = datetime(2017, 11, 5, 1, 0, 0,
                     tzinfo = eastern)
tz.datetime_ambiguous(first_1am)
```

```
True
```

```
# 2017-11-05 01:00:00 again
second_1am = datetime(2017, 11, 5, 1, 0, 0,
                      tzinfo = eastern)
second_1am = tz.enfold(second_1am)
```

Ending Daylight Saving Time

```
(first_1am - second_1am).total_seconds()
```

```
0.0
```

```
first_1am = first_1am.astimezone(tz.UTC)
second_1am = second_1am.astimezone(tz.UTC)
(first_1am - second_1am).total_seconds()
```

```
3600.0
```

Ending Daylight Saving Time

WORKING WITH DATES AND TIMES IN PYTHON

Reading date and time data in Pandas

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author

A simple Pandas example

```
# Load Pandas
import pandas as pd

# Import W20529's rides in Q4 2017
rides = pd.read_csv('capital-onebike.csv')
```

A simple Pandas example

```
# See our data
print(rides.head(3))
```

```
            Start date             End date                    Start station  \
0 2017-10-01 15:23:25 2017-10-01 15:26:26              Glebe Rd & 11th St N
1 2017-10-01 15:42:57 2017-10-01 17:49:59  George Mason Dr & Wilson Blvd
2 2017-10-02 06:37:10 2017-10-02 06:42:53  George Mason Dr & Wilson Blvd

                          End station Bike number Member type
0         George Mason Dr & Wilson Blvd      W20529      Member
1         George Mason Dr & Wilson Blvd      W20529      Casual
2  Ballston Metro / N Stuart & 9th St N      W20529      Member
```

A simple Pandas example

```
rides['Start date']
```

```
0     2017-10-01 15:23:25
1     2017-10-01 15:42:57
...
Name: Start date, Length: 290, dtype: object
```

```
rides.iloc[2]
```

```
Start date                          2017-10-02 06:37:10
End date                            2017-10-02 06:42:53
...
Name: 1, dtype: object
```

Loading datetimes with parse_dates

```
# Import W20529's rides in Q4 2017
rides = pd.read_csv('capital-onebike.csv',
                    parse_dates = ['Start date', 'End date'])

# Or:
rides['Start date'] = pd.to_datetime(rides['Start date'],
                                     format = "%Y-%m-%d %H:%M:%S")
```

Loading datetimes with parse_dates

```
# Select Start date for row 2
rides['Start date'].iloc[2]
```

```
Timestamp('2017-10-02 06:37:10')
```

Timezone-aware arithmetic

```
# Create a duration column
rides['Duration'] = rides['End date'] - rides['Start date']

# Print the first 5 rows
print(rides['Duration'].head(5))
```

```
0   00:03:01
1   02:07:02
2   00:05:43
3   00:21:18
4   00:21:17
Name: Duration, dtype: timedelta64[ns]
```

Loading datetimes with parse_dates

```
rides['Duration']\
  .dt.total_seconds()\
  .head(5)
```

```
0     181.0
1    7622.0
2     343.0
3    1278.0
4    1277.0
Name: Duration, dtype: float64
```

Reading date and time data in Pandas

WORKING WITH DATES AND TIMES IN PYTHON

Summarizing datetime data in Pandas

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author

Summarizing data in Pandas

```
# Average time out of the dock
rides['Duration'].mean()
```

```
Timedelta('0 days 00:19:38.931034')
```

```
# Total time out of the dock
rides['Duration'].sum()
```

```
Timedelta('3 days 22:58:10')
```

Summarizing data in Pandas

```
# Percent of time out of the dock
rides['Duration'].sum() / timedelta(days=91)
```

```
0.043484177859177786
```

Summarizing data in Pandas

```
# Count how many time the bike started at each station
rides['Member type'].value_counts()
```

```
Member    236
Casual     54
Name: Member type, dtype: int64
```

```
# Percent of rides by member
rides['Member type'].value_counts() / len(rides)
```

```
Member    0.813793
Casual    0.186207
Name: Member type, dtype: float64
```

Summarizing datetime in Pandas

```
# Add duration (in seconds) column
rides['Duration seconds'] = rides['Duration'].dt.total_seconds()

# Average duration per member type
rides.groupby('Member type')['Duration seconds'].mean()
```

```
Member type
Casual    1994.666667
Member     992.279661
Name: Duration seconds, dtype: float64
```

Summarizing datetime in Pandas

```
# Average duration by month
rides.resample('M', on = 'Start date')['Duration seconds'].mean()
```

```
Start date
2017-10-31    1886.453704
2017-11-30     854.174757
2017-12-31     635.101266
Freq: M, Name: Duration seconds, dtype: float64
```

Summarizing datetime in Pandas

```
# Size per group
rides.groupby('Member type').size()
```

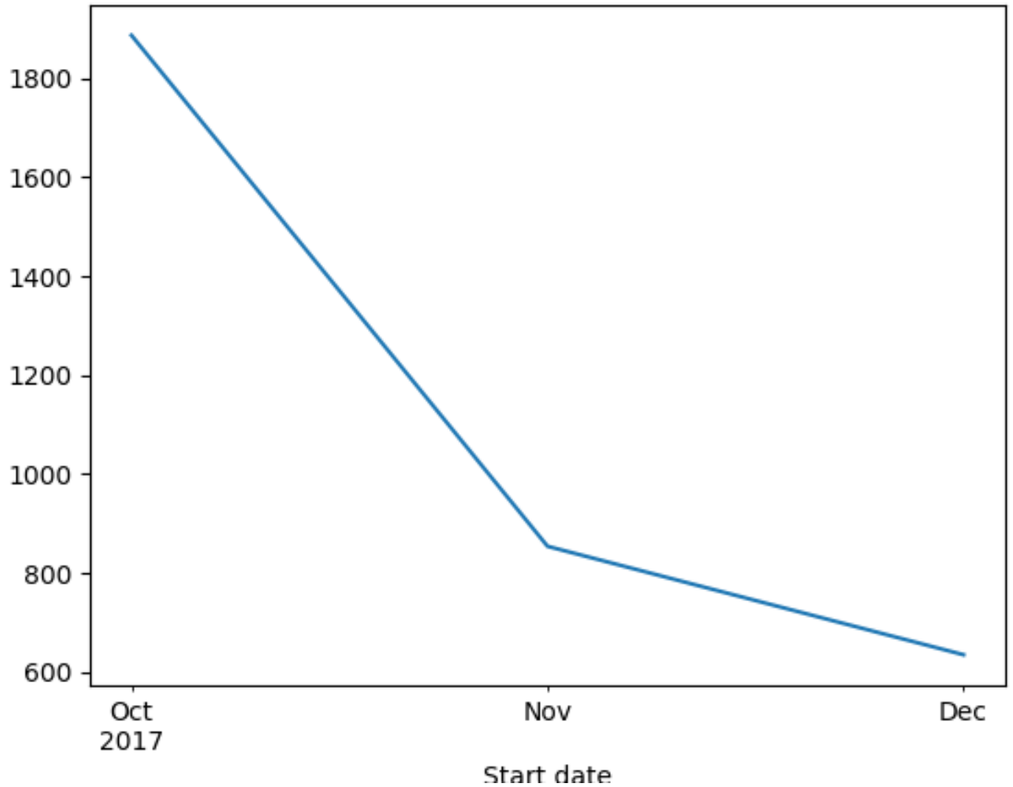
```
Member type
Casual      54
Member     236
dtype: int64
```

```
# First ride per group
rides.groupby('Member type').first()
```

```
                Duration     ...
Member type                  ...
Casual      02:07:02         ...
Member      00:03:01         ...
```

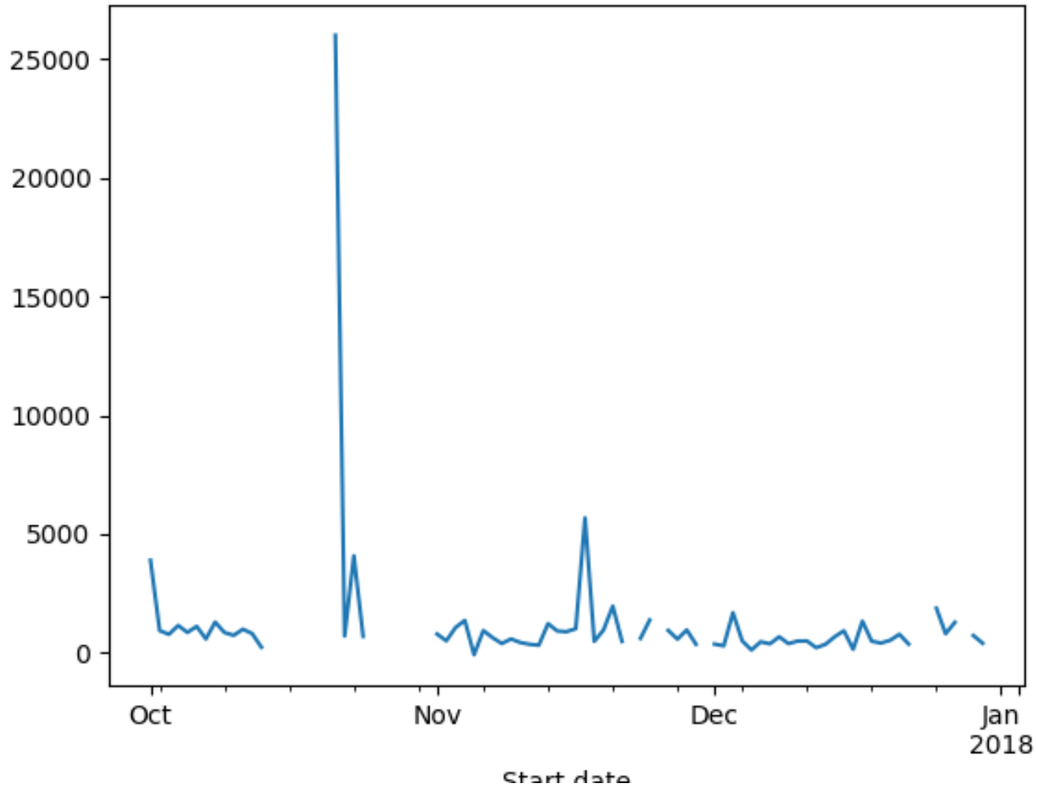
Summarizing datetime in Pandas

```
rides\
  .resample('M', on = 'Start date')\
  ['Duration seconds']\
  .mean()\
  .plot()
```



Summarizing datetime in Pandas

```
rides\
  .resample('D', on = 'Start date')\
  ['Duration seconds']\
  .mean()\
  .plot()
```



Summarizing datetime data in Pandas

WORKING WITH DATES AND TIMES IN PYTHON

Additional datetime methods in Pandas

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist & Author

Timezones in Pandas

```
rides['Duration'].dt.total_seconds().min()
```

```
-3346.0
```

Timezones in Pandas

```
rides['Start date'].head(3)
```

```
0   2017-10-01 15:23:25
1   2017-10-01 15:42:57
2   2017-10-02 06:37:10
Name: Start date, dtype: datetime64[ns]
```

```
rides['Start date'].head(3)\
  .dt.tz_localize('America/New_York')
```

```
0   2017-10-01 15:23:25-04:00
1   2017-10-01 15:42:57-04:00
2   2017-10-02 06:37:10-04:00
Name: Start date, dtype: datetime64[ns, America/New_York]
```

Timezones in Pandas

```
# Try to set a timezone...
rides['Start date'] = rides['Start date']\
  .dt.tz_localize('America/New_York')
```

```
AmbiguousTimeError: Cannot infer dst time from '2017-11-05 01:56:50',
try using the 'ambiguous' argument
```

```
# Handle ambiguous datetimes
rides['Start date'] = rides['Start date']\
  .dt.tz_localize('America/New_York', ambiguous='NaT')

rides['End date'] = rides['End date']\
  .dt.tz_localize('America/New_York', ambiguous='NaT')
```

Timezones in Pandas

```
# Re-calculate duration, ignoring bad row
rides['Duration'] = rides['Start date'] - rides['End date']

# Find the minimum again
rides['Duration'].dt.total_seconds().min()
```

```
116.0
```

Timezones in Pandas

```
# Look at problematic row
rides.iloc[129]
```

```
Duration                          NaT
Start date                        NaT
End date                          NaT
Start station          6th & H St NE
End station            3rd & M St NE
Bike number                   W20529
Member type                   Member
Name: 129, dtype: object
```

Other datetime operations in Pandas

```
# Year of first three rows
rides['Start date']\
  .head(3)\
  .dt.year
```

```
0    2017
1    2017
2    2017
Name: Start date, dtype: int64
```

```
# See weekdays for first three rides
rides['Start date']\
  .head(3)\
  .dt.weekday_name
```

```
0    Sunday
1    Sunday
2    Monday
Name: Start date, dtype: object
```

Other parts of Pandas

```
# Shift the indexes forward one, padding with NaT
rides['End date'].shift(1).head(3)
```

```
0                          NaT
1   2017-10-01 15:26:26-04:00
2   2017-10-01 17:49:59-04:00
Name: End date, dtype: datetime64[ns, America/New_York]
```

Additional datetime methods in Pandas

WORKING WITH DATES AND TIMES IN PYTHON

Wrap-up

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author

Recap: Dates and Calendars

- The `date()` class takes a year, month, and day as arguments
- A `date` object has accessors like `.year`, and also methods like `.weekday()`
- `date` objects can be compared like numbers, using `min()`, `max()`, and `sort()`
- You can subtract one `date` from another to get a `timedelta`
- To turn `date` objects into strings, use the `.isoformat()` or `.strftime()` methods

Recap: Combining Dates and Times

- The `datetime()` class takes all the arguments of `date()`, plus an hour, minute, second, and microsecond
- All of the additional arguments are optional; otherwise, they're set to zero by default
- You can replace any value in a `datetime` with the `.replace()` method
- Convert a `timedelta` into an integer with its `.total_seconds()` method
- Turn strings into dates with `.strptime()` and dates into strings with `.strftime()`

Recap: Timezones and Daylight Saving

- A `datetime` is "timezone aware" when it has its `tzinfo` set. Otherwise it is "timezone naive"
- Setting a timezone tells a `datetime` how to align itself to UTC, the universal time standard
- Use the `.replace()` method to change the timezone of a `datetime` , leaving the date and time the same
- Use the `.astimezone()` method to shift the date and time to match the new timezone
- `dateutil.tz` provides a comprehensive, updated timezone database

Recap: Easy and Powerful Timestamps in Pandas

- When reading a csv, set the `parse_dates` argument to be the list of columns which should be parsed as datetimes
- If setting `parse_dates` doesn't work, use the `pd.to_datetime()` function
- Grouping rows with `.groupby()` lets you calculate aggregates per group. For example, `.first()`, `.min()` or `.mean()`
- `.resample()` groups rows on the basis of a `datetime` column, by year, month, day, and so on
- Use `.tz_localize()` to set a timezone, keeping the date and time the same
- Use `.tz_convert()` to change the date and time to match a new timezone

Congratulations!

WORKING WITH DATES AND TIMES IN PYTHON