Dates in Python

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author

datacamp

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

datacamp

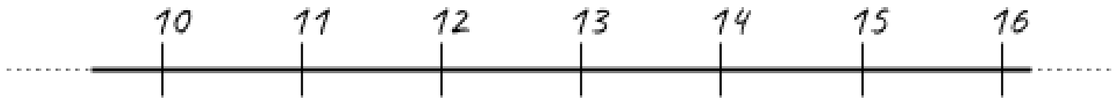
Math with Dates

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author

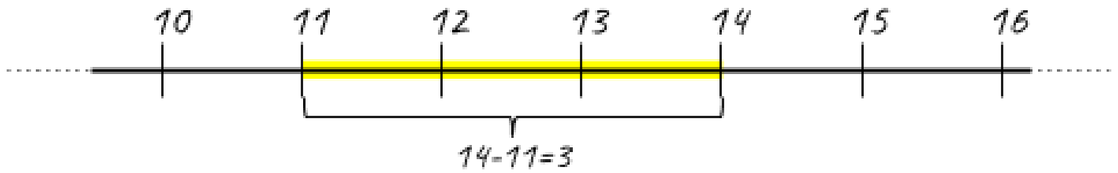
datacamp

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



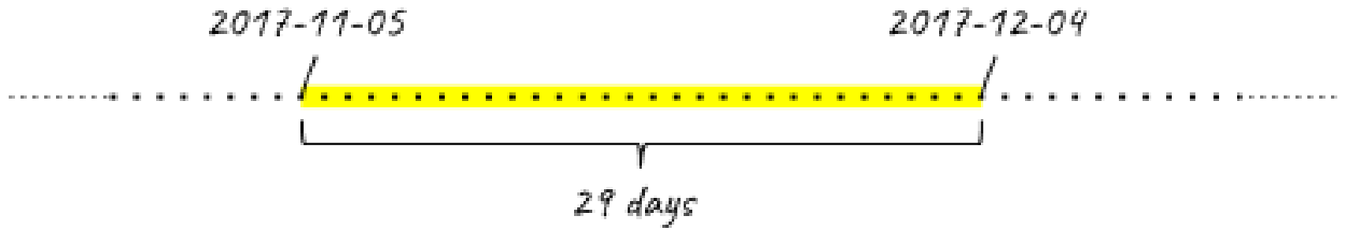

```
# Subtract two numbers
print(b - a)
3

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

Math with dates

Math with dates

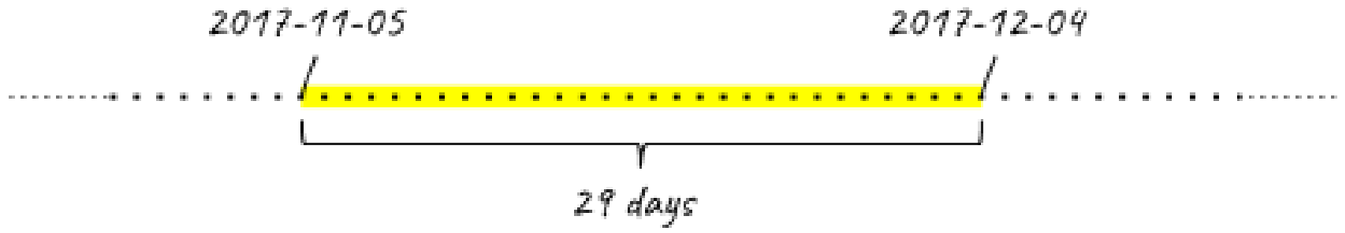

```
# Import date
from datetime import date
# Create our dates
d1 = date(2017, 11, 5)
d2 = date(2017, 12, 4)
l = [d1, d2]
print(min(l))
2017-11-05
```

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

datacamp

Turning dates into strings

WORKING WITH DATES AND TIMES IN PYTHON

Max Shron
Data Scientist and Author

datacamp

ISO 8601 format

```
from datetime import date


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

3. ISO 8601 format
The ISO 8601 format has another nice advantage. To demonstrate, we've created a variable called some_dates and represented two dates here as strings: January 1, 2000, and December 31, 1999. Dates formatted as ISO 8601 strings sort correctly. When we print the sorted version of this list, the earlier day is first, and the later date is second. For example, if we use ISO 8601 dates in filenames, they can be correctly sorted from earliest to latest. If we had month or day first, the strings would not sort in chronological order.

```
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()
```

4. Every other format
If you don't want to put dates in ISO 8601 format, Python has a flexible set of options for representing dates in other ways, using the strftime() method.

Every other format: strftime

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

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

5. Every other format: strftime
strftime() works by letting you pass a "format string" which Python uses to format your date. Let's see an example. We again create an example date of January 5th, 2017. We then call strftime() on d, with the format string of % capital Y. Strftime reads the % capital Y and fills in the year in this string for us. strftime() though is very flexible: we can give it arbitrary strings with % capital Y in them for the format string, and it will stick the year in. For example, we can use the format string of "Year is %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
```

6. Every other format: strftime
Strftime has other placeholders besides %Y: % lowercase m gives the month, and % lowercase d gives the day of the month. Using these, we can represent dates in arbitrary formats for whatever our needs are.

Turning dates into strings

WORKING WITH DATES AND TIMES IN PYTHON

datacamp