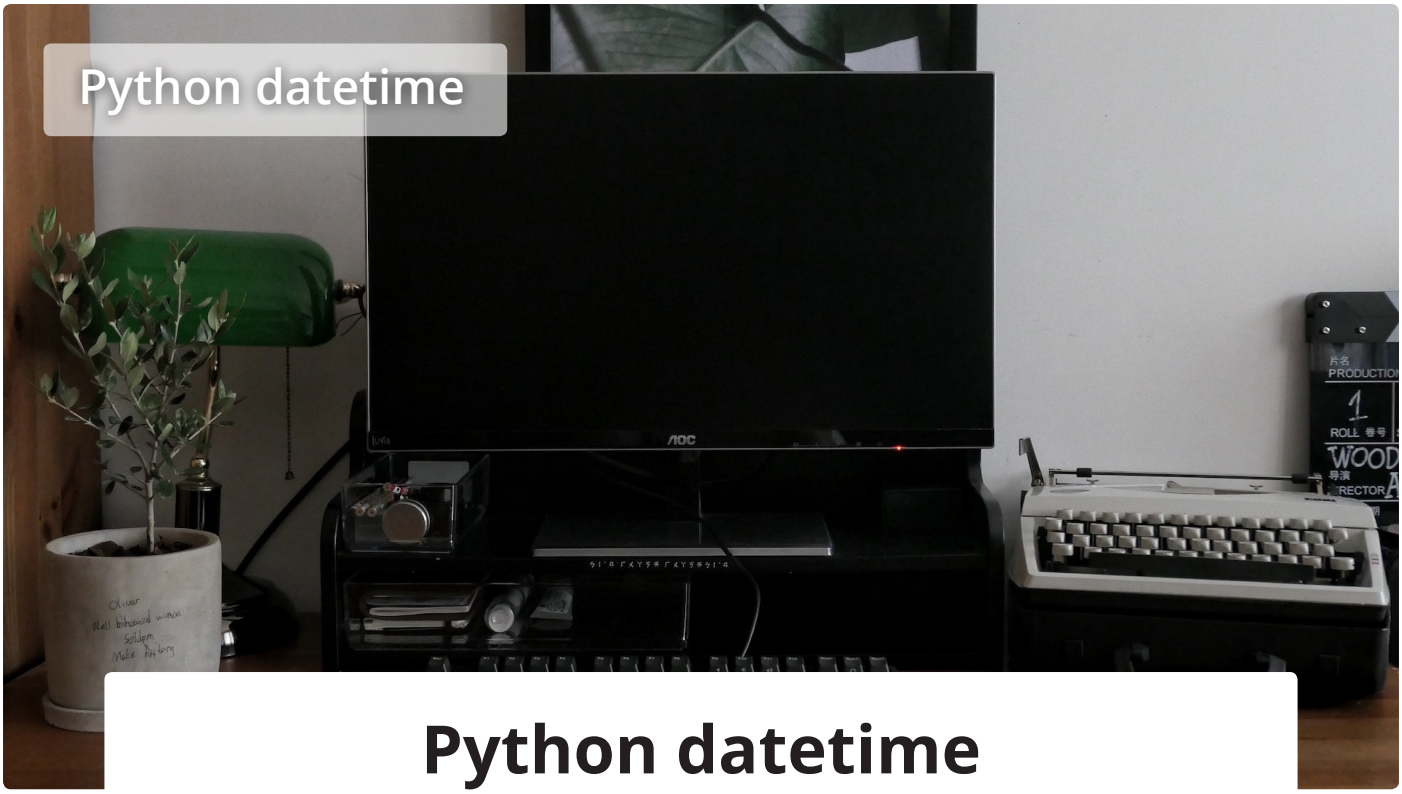

Python datetime

Hey there! Today we bring you frequently used datetime syntax, as well as best practices and examples!

Frequently used syntax

This syntax below can be used in the `strftime` and `strptime` methods, like so:

```
import datetime
now = datetime.datetime.now()
print(now.strftime('%d-%m-%Y'))  # 13-03-2018
print(now.strftime('%A, %B %d, %Y')  # Tuesday, March 13, 2018

user_date = input('Enter the current date as %Y-%m-%d: ')
print(datetime.datetime.strptime(user_date, "%Y-%m-%d"))
```

Code	Meaning	Example
%A	Day of the week as text	Monday
%d	Day of the month from 01 to 31	17
%B	Month name as text	October
%m	Month of the year from 01 to 12	7
%Y	Year with century	2016
%H	Hour from 00 to 23	15
%I	Hour from 00 to 12	7
%p	Equivalent of AM or PM in the current language	AM
%M	Month from 01 to 12	10
%S	Seconds from 00 to 59	54
%x	Date representation in the current language	06/10/15
%X	Time representation in the current language	19:54:22

Examples

Here's a few code examples from things I've personally written in the past. Use them as a reference or just to have a read over!

```
import datetime

user_date = input('Enter the current date as %Y-%m-%d: ')
print(datetime.datetime.strptime(user_date, "%Y-%m-%d"))
```

Here's an example of both using the input function in a dictionary, to get the values for each field, and also getting the current time.

There can be issues with debugging if you do shorthands like these, so it can sometimes be easier to just create a variable for each value (as in the example directly below this next one).

```
import datetime

users = []

new_user = {
  ‘name’: input(‘Enter your name: ‘)
  ‘location’: input(‘Enter your location: ‘)
  ‘registered’: datetime.datetime.now(datetime.timezone.utc)
}

users.append(new_user)
```

Here’s an example with arguably better (and more readable) code. Creating a variable for each means that debugging can be a bit easier—you can just set a breakpoint on the line that is going wrong to easily identify what’s happening.

```
import datetime

users = []

name = input(‘Enter your name: ‘)
location = input(‘Enter your location: ‘)
registered = datetime.datetime.now(datetime.timezone.utc)


new_user = {
  ‘name’: name
  ‘location’: location
  ‘registered’: registered
}

users.append(new_user)
```

Frequently when creating a new object you may want to store when it was created. This is common with users, for example. Below is a way to do this. Notice that some database engines let you populate a field in the database automatically with the current date when a new row is created, so sometimes you won’t need to do this in Python.

```
import datetime

class User:
  def __init__(self, username, password):
    self.username = username
    self.password = password
    self.registered = datetime.datetime.now(datetime.timezone.utc)
```

Below we mention a best practice which is using timestamps. Here's how you can turn a datetime object into a timestamp and back again:

```
import datetime

now = datetime.datetime.now(datetime.timezone.utc)
current_timestamp = now.timestamp()

now_from_timestamp = datetime.utcfromtimestamp(current_timestamp)
```

Best practices

The single most important best practice is, as mentioned in the course, to store all your dates and times in a database as UTC. That means that each datetime will have some timezone information associated with is. That timezone must be UTC (which has an "offset" of 0 hours).

The offset is always relative to UTC, so naturally a UTC datetime has an offset of 0. A CET timezone has a +01:00 offset, which means it is an hour ahead of UTC.

If you store all your timezones as UTC, then you can display them to your user as their local timezone (this is quite simple, just ask them where they live or get that information from their IP address).

Something else we do quite frequently is store dates and times as timestamps. A timestamp is the number of seconds since 1st January 1970 at midnight in UTC. Thus normally you'll calculate a timestamp and store that in your database, then turn it back to a human-readable format using the last example above.

Happy coding!