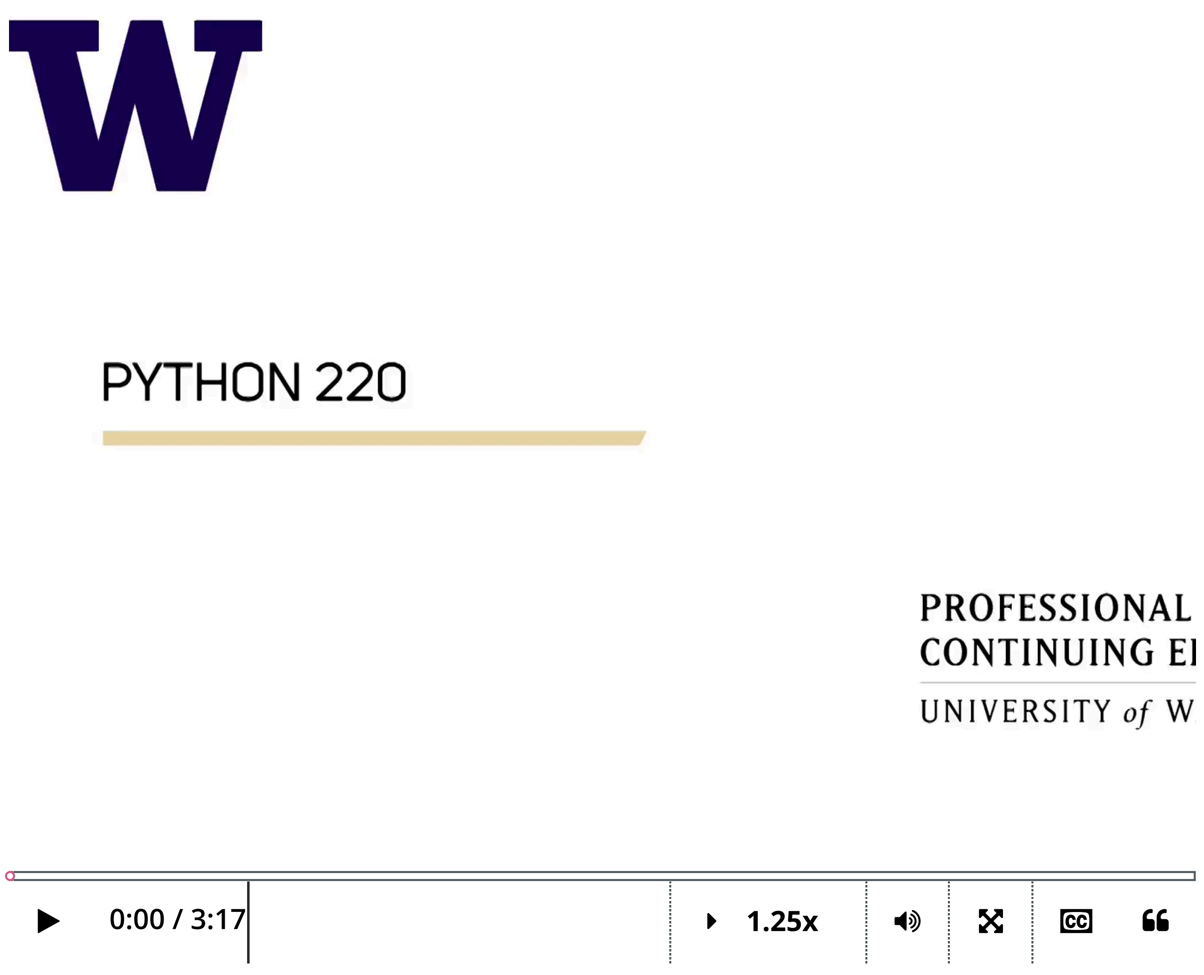
Part 3: Cython

Bookmark this page

Profiling and Performance

Part 3: Cython

Video 1 of 3: Great Circle



Start of transcript. Skip to the end.

[MUSIC PLAYING]

In this segment, I want to set up an exercise for you, the great circle

exercise.

Basically, I've got some code that does this algorithm all coded

up and ready to go.

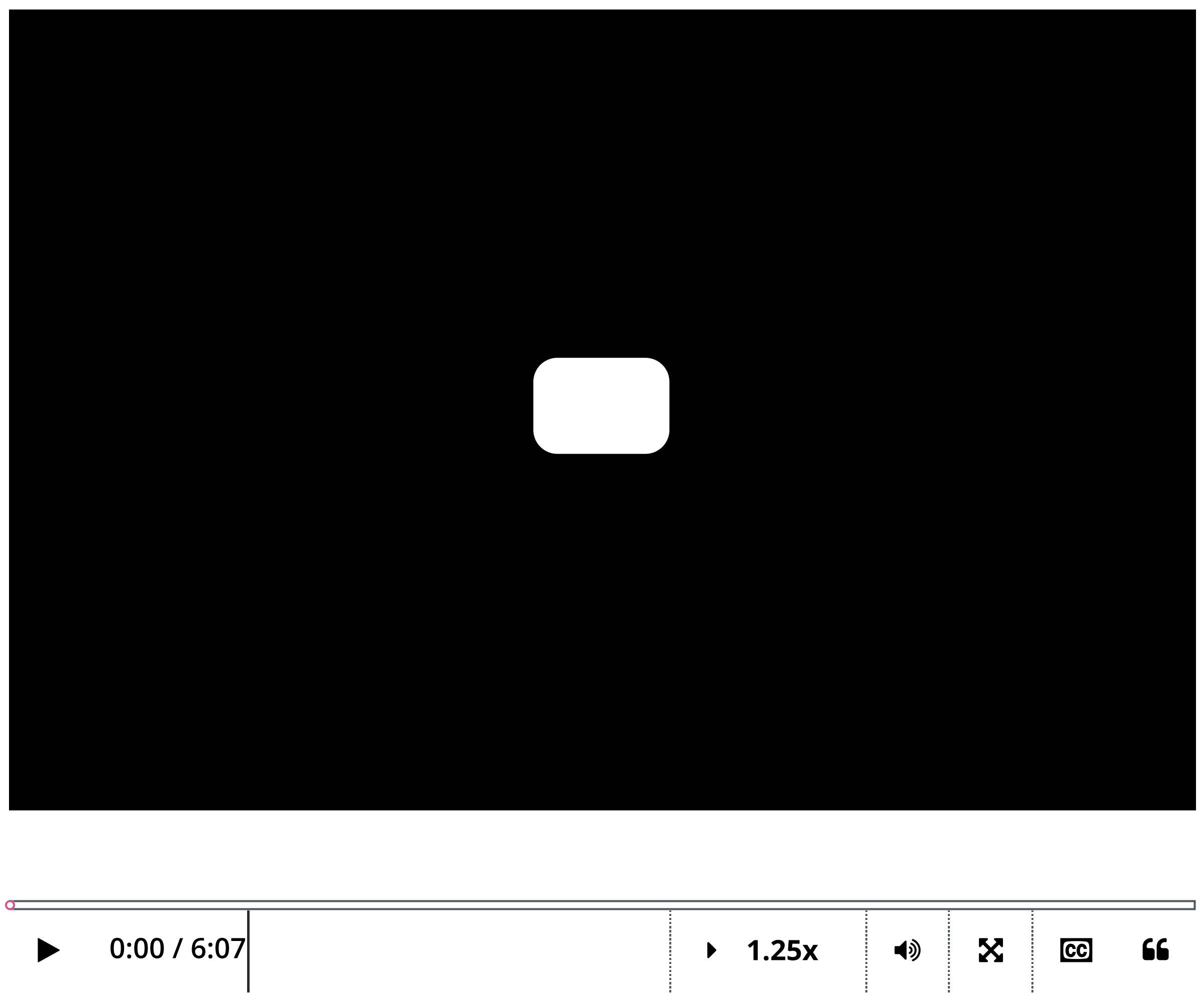
You don't even really have to worry about the math,

as long as you get essentially, I don't know,

Video
Download video file

Transcripts
Download SubRip (.srt) file
Download Text (.txt) file

Video 2 of 3: Cython



Start of transcript. Skip to the end.

[MUSIC PLAYING]

Today we're going to talk about Cython.

It is a way of interacting with C from Python.

It's a superset of the language.

It gives you direct access to the static types

that are available in C. It has tremendous possibilities

for performance in specific cases.

And we'll look at some of those as we go

Video
Download video file

Transcripts
Download SubRip (.srt) file
Download Text (.txt) file

Cython is Python's bridge to C. In allows you to continue to work largely in Python, albeit Python with a few extensions, and yet produce fast, statically compiled libraries to link into your Python code.

Cython provides a keyword, cdef which provides access to C types such as char, int, long, float, double, struct and enum. It also enables type delectations for function or method return values. Take for instance the following function declaration.

```
cdef double great_circle(double lon1, double lat1, double lon2, double lat2):
    cdef double a, b, c, radius, theta, x
```

The great_circle function above is defined specifying double as the type of its return value. Each argument is declared to be of type double, and a series of symbols which will be used later in the function are all defined to be of type double. The cython compiler uses this information to produce a statically compiled library which is called from Python. Since this is no longer standard Python we no longer save it in files with .py extensions, instead by convention we use .pyx.

Cython involves a different workflow, because it is a statically compiled environment. Statically compiled programming environments, including C and Cython, have separate compile, link and run steps. You can't simply type your code into an interpreter and get an immediate result. Statically compiled programming environments also have tooling to support the static build workflow. These tools include Python's distutils used by cython setup files and optionally Gnu Make used to manage the build dependencies of projects large and small.

This is `great_circle_setup_v1.py` which is one of the cython setup files for use with the Great Circle code.

```
from distutils.core import setup
from Cython.Build import cythonize

setup(
    name='Great Circle v1',
    ext_modules=cythonize("great_circle_v1.pyx"),
)
```

The setup file tells Python how to have Cython build a statically compiled library to call from within Python.

```
$ python great_circle_setup_v1.py build_ext --inplace
```

There are two main hurdles in learning to use Cython. The first involves the extensions to the Python language which are for the most part borrowed directly from C. Without familiarity with C it may not be clear what these new language elements mean, or how or why they are used. The second hurdle involves the build process associated with statically compiled languages — i.e., the separate compile, link and run steps which will feel foreign to someone coming strictly from an interpreted language like Python. In both cases programmers familiar with C will have an advantage over those without. As with any new worthwhile programming strategy it will take an investment in time and effort to learn this new tool.

The video links start to pull this information together. Arm yourself with the sample code when you watch Part 2.

https://github.com/rriehle/ProfilingPerformance/tree/master/source/solutions/cython

Video 3 of 3: Cython pt 2

Start of transcript. Skip to the end.

‹ Previous

Next ›

0:00 / 0:00 1.25x

[MUSIC PLAYING]

All right so far, so good.

We've seen a simple use of Cython inside of Jupiter notebook,

now we're going to start using it in more of a real world example.

Great circle is out there.

We've already used it when we were looking at Pypi.

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file