About This Course

In this lesson, we will discuss who should take this course and what to expect from it.

WE'LL COVER THE FOLLOWING

Who should take this course?

This course is designed for scientists and engineers who aren't necessarily computer scientists but want to add scientific programming to their area of expertise.

This course uses Python, as it makes scientific programming tasks quick to use and easy to learn.

Even though we will be refreshing some common Python concepts in the initial chapter, this course assumes that you have a good understanding of Python.

If you are new to Python and want to learn it from scratch, take this free course by Educative.

What to expect from this course

After refreshing the basics of Python, our next step is to learn NumPy, which forms the basis of scientific computation in Python.

NumPy gives us the advantage of working with multi-dimensional arrays and performs vectorized operations without writing loops. We will use these multi-dimensional arrays to solve problems in linear algebra. We then move on to the visualization of 2-D and 3-D data.

Later, we will learn symbolic computation and use it to solve problems in calculus. Before moving on to random variables, we will take a deep dive into common scientific algorithms in order to equip ourselves with some commonly used tools in scientific computation.

In the applications section, you will apply all the learned skills to some new scientific concepts.

Essential packages covered in this course

The following essential packages will be covered throughout the course:

- **NumPy:** Used for creating and operating on multi-dimensional arrays.
- **Matplotlib:** Used for plotting and visualizing data.
- **SymPy:** Used for symbols, algebra, calculus, and solving equations.
- **SciPy:** Used for numeric integration, interpolation, curve fitting, optimization, and Fourier Transforms.

Before we dive into scientific computations in Python, we’ll revisit a few basic Python concepts in the next chapter.