Python for Structural Engineers

What, Why, Demos and Learning Paths

Satish Annigeri Ph.D.

Structural Engineer
Former Professor of Civil Engineering
B.V.B. College of Engineering & Technology (KLE Technological University)
Hubballi 580031

February 20, 2026

Objectives of this Presentation

- Should structural engineers learn Python programming?
  - Use commercial black-box applications
  - Use Microsoft® Excel and formulae
- Is Python a good choice for structural engineers?
  - Can I start learning a new programming language?
  - Is it worth the effort?
- What can a structural engineer do with Python?
  - Write customized programs that are easy to debug and maintain
  - Develop full fledged applications
- Are there alternatives to Python?
  - Matlab is similar to Python but is expensive, slow and lacks the ecosystem compared to Python
  - **Julia** is free and open source, fast compared to Python but lacks the ecosystem
- Demonstrate a few applications through working examples

The purpose of this talk is to demonstrate what you can do with Python and help you decide if it is worth learning.

Agenda

- Structural engineers and computing
- What is Python?
- Why Python?
- How to use Python – some demonstrations of use
- Way forward

Structural Engineers and Computer Programming

- Structural engineers extensively use **spreadsheets**
- Structural engineers need to **visualize data**
- Structural engineers **generate documents/reports** based on their calculations
- Programming languages preferred by engineers have changed over time – Fortran, Matlab®, C/C++, Python. In future Julia, Mojo, . . . ?
- Structural engineer's approach to programming is different compared to that by computer science engineers
  - Our applications are predominantly numerical computation and data visualization
  - Rapid prototyping of algorithms, testing to see what works best – exploratory computing
  - One-time (throwaway) scripts that may not serve a long-term purpose but are important for a specific task

Programming languages are a tool, the focus is on problem-solving

What is Python?

- Python is a **general purpose** programming language (unlike Matlab®, Scilab, Julia which are higher level languages targeted at scientific computation and visualization)
- Created by **Guido van Rossum** in 1991 – 35 years old!
- It is free and open source software (**FOSS**) and is governed by the **Python Software Foundation (PSF)**
- Python 3.0 was released in December 2008 and Python 2.7 reached end-of-life in January 2020. Python 2 and Python 3 coexisted for about 12 years.
- Its current version is 3.14.x with version 3.15.0 due in October 2026
- People ask these questions about free and open source software
  - How long will the software live?
  - Who decides the future growth of the software?
- People don't ask these questions about closed source software
  - Will the company make the improvements I want to see in the software?
  - What happens if the company is sold to another and that company loses interest in the software?

Why Python?

- **Easy to learn and use**
  - Interactive - Try before you code
  - Designed to be "**readable**"
  - Well documented – base language and also third-party packages
  - Encouraging and inclusive community – Code of conduct, conferences across different countries: **PyCon**, **SciPy**
- **Supports scientific computing**
  - Exploratory computing – interactive, execute one line at a time
  - Data visualization – plot 2D and 3D graphs, statistical plots
  - Data structures and algorithms for numerical computing
  - Symbolic computing – Accurate integration and differentiation without loss of accuracy inherent in numerical computations
  - Support for statistics and machine learning
- **Batteries included** – large collection of packages (libraries) for a variety of tasks
- Access to databases – RDBMS (MySQL, PostgreSQL, SQLite) and NoSQL (MongoDB)
- Application development for multiple devices – web, desktop, mobile

Python for Scientific Computing

- Scientific computing encompasses numeric and symbolic computing and data visualization
- Applications of scientific computing include
  - Solving mathematical equations arising from mathematical models – linear algebra, ODEs
  - Statistics and probability
  - Machine learning, data science, artificial intelligence
- **NumPy** package provides an efficient **n-dimensioned** array data structure that is at the heart of all numerical algorithms in Python
- **SciPy** provides efficient algorithms for a broad variety of scientific computing tasks including linear algebra, optimization, signal processing, image processing etc.
- **Matplotlib** provides the main graph plotting capability of Python
- **Pandas** or **Polars** provide the DataFrame data structure analogous to worksheets in a spreadsheet program – with named columns and numbered rows

Installing Python and Creating a Virtual Environment

- Install from Python home page
- Install `uv` from Astral `uv` homepage
- You will be required to use **PowerShell** (in Microsoft® Windows) or the **terminal** (in GNU/Linux or macOS)
- You can use `uv` to check the installed Python versions on your PC and install multiple versions of Python if needed or remove installed versions of Python that are not needed
- You can create a Virtual Environment (`venv`) using any version of Python that is currently available

```
C:\> uv -V
C:\> uv help
C:\> uv python list
C:\> uv python install 3.14
C:\> uv python uninstall 3.10
C:\> uv python upgrade 3.11
```

Working with Python

- REPL (**R**ead, **E**valuate and **P**rint **L**oop) is the default way Python is available
- **Spyder IDE** is a Matlab® like GUI for Python
- **VS Code** IDE is a popular **I**ntegrated **D**evelopment **E**nvironment with large number of Extensions to customize it for Python
- Web Notebooks
  - **Jupyter Notebooks** are non-reactive. Change to a variable are not automatically propagated throughout the Notebook
  - **Marimo Notebooks** are reactive. Change to a variable is automatically reflected **throughout** the Notebook
- Web application frameworks – **Flask**, **Flet** and **NiceGUI**
- Mobile app frameworks – **Flet**, **NiceGUI**

In this presentation, I will predominantly make use of Marimo Notebooks

Alternatives to Python

- Free and Open Source Software
  - **Julia** which has a focus on scientific computing, interactive like Python and much faster in execution speed
  - **GNU Octave** a Matlab® clone
  - **Scilab** which is similar to Matlab®
- Closed Source or Proprietary Software
  - **Matlab®**

Choosing Your Approach to Automation

- **User**
  - Write code only for your own use
  - Predominantly use packages developed by others where there are packages available
  - Focus on getting your own task done without worrying how others may carry out the same task
  - Write the code only for your own consumption, with minimal documentation
- **Developer**
  - Develop packages that you want others to use
  - Pay attention to the best way to carry out a task, not just getting the results
  - Write user documentation to help others understand how to use the package
  - Write developer documentation to help others modify the code you have written or collaborate with others
  - Lear how to upload your package to the Python repository **PyPI**

Start as a user and move on to a developer if you feel you have ideas others can benefit from

Topics for Demonstration

1. Numerical calculations and plotting graphs – **NumPy**, **Matplotlib**
2. Reading, modifying and writing Microsoft® Excel and CSV files – **Pandas** or alternately Polars
3. Manipulating PDF files – **PyMuPDF**
4. Analysis and design of RC sections and design of steel members using our own scripts
5. Properties of steel sections – **sectionproperties**
6. Generating PDF documents from Microsoft® Word templates after populating it with calculated data – **docx-mailmerge2**
7. **OpenSeesPy** for linear and nonlinear static and dynamic Finite Element Analysis
8. **Osdag** for design of steel connections, tension members and flexural

Numerical Calculations and Plotting Graphs

- Work with numbers and matrices
- Linear algebra - linear simultaneous equations, eigenvalues and eigenvectors
- Finding roots of nonlinear equations
- Plotting graphs
- Optimization
- Machine learning
- Packages: **NumPy**, **Matplotlib**, SciPy, Scikit-learn
- Demo script: `01_np_plt.py`

Working with Microsoft® Excel and CSV Files

- Read data from Microsoft® Excel and/or CSV files, clean up the data, calculate new columns from existing columns
- Exactly like Microsoft® Excel, but programmatic instead of visual
- Read data output by Staad.Pro/ETABS, sort the data, group data based on values in one or more columns
- Packages: **Pandas**, Polars, or narwhals
- Demo scripts
  - `021_excel_csv.py` – Reading and writing Microsoft® Excel files
  - `022_footing_design.py` – Toy example for reading output from Staad.Pro, proportioning an isolated rectangular footing and writing results to file

Manipulating PDF Files

- Organize PDF files – split, remove, add, reorder pages in a PDF file
- Extract text, tables, images from a PDF file
- Insert text, header, footer to a PDF file
- Packages: **PyMuPDF**, pikepdf
- Demo script: `030_pdf.py`

Analysis and Design of RC Sections

- Use existing packages to design RC sections - rectangular (singly and doubly reinforced), flanged sections, columns
- Develop your own code to do design
- Develop solutions for repeated designs such as footings, retaining walls, underground tanks etc.
- Design requires some amount of interaction and is difficult to make it fully automatic
- Package: **rcdesign**
- Demo scripts
  - `041_rcsec.py` – Design of RC rectangular and flanged sections for bending
  - `042_rcsec.py` – Analysis of RC rectangular and flanged sections for bending and rectangular section under axial compression and uniaxial bending

Properties of Sections and Design of Steel Members

- Calculate geometric and plastic properties of cross sections – rolled steel sections, thin walled hollow steel sections
- Design steel members
- Package: **sectionproperties**
- Demo scripts
  - `051_secprop.py` – Properties of cross sections
  - `052_steel_design.py` – Design of beams

Generating PDF Documents from Microsoft® Word Templates

- Structural engineers have to generate design documents, which may be slight changes to previous documents, if not exact replicas. But there is no scope for errors in these documents which may be proof checked by third parties
- Manually copying and pasting values from your design program into your Microsoft® Word report is tedious and error prone
- You can write a template file in Microsoft® Word, leaving space holders for values that you can fill later automatically from your design program
- Take the example of the design report for a steel column including splice design and baseplate, steel beam design including stiffeners and splices, retaining wall etc.
- You can also prepare templates in HTML or LaTeX format
- **Packages: Jinja2**, **docx-mailmerge2**, **weasyprint**
- **External programs required: LibreOffice**, **TinyTeX**
  - `061_docxgen.py` – Generate documents from Microsoft® Word files

OpenSeesPy

- **OpenSees** (**Open S**ystem for **E**arthquake **E**ngineering **S**imulation) is a software framework for simulating the seismic response of structural and geotechnical systems, developed at PEER (Pacific Earthquake Engineering Research Center) of the University of California, Berkeley
- It can perform linear and nonlinear static and dynamic analysis
- It offers a number of finite elements and material models to model 1D, 2D and 3D skeletal and 2D and 3D finite element models.
- It offers Fiber Section elements, spring elements, damping elements to carry out plastic analysis of frame structures and simplified soil structure interaction by representing soil as Winkler springs
- It also has support for fluid-structure interaction and thermal analysis for modelling structures under fire.
- Modeling and analysis are usually done via scripts written either in **Tcl** or **Python** scripting languages

OpenSeesPy

- **OpenSeesPy** is the Python interface to OpenSees, which is based on Tcl, a not so well known programming language
- Python is gaining popularity compared to Tcl because many engineers already know Python
- Python is easier to learn and can also plot graphs
- There are GUIs available but are closed source commercial products

Osdag – Open Steel Design and Graphics

- **Osdag** is being developed by a team under the guidance of Prof. Siddhartha Ghosh under the Government of India supported NMEICT funding at the Department of Civil Engineering, IIT Bombay
- It provides an interactive GUI and is a free and open source software written in Python
- It can carry out the following designs as per IS 800:2007
  - Connections
    - Shear connections: Fin plate, Cleat angle, End plate and Seated angle
    - Moment connections: Cover plate bolted, Cover plate welded, End plate
    - Base plate
  - Tension member: bolted to end gusset, welded to end gusset
  - Compression member: Strut in truss
  - Flexural member: Simply supported beam, Cantilever beam
- It generates detailed design reports in PDF format

Some Useful Applications for the Structural Design Office

- **BentoPDF** is a web application for manipulating PDF files, converting images and Microsoft® Office documents to PDF etc.
  - BentoPDF can be setup as a server and accessed by everyone connected to the server, say within the LAN of an office
- **Paperless ngx** is a web application for document management
  - Organize and index documents and full-text search of your repository
  - Supports PDF documents, images, plain text files, Microsoft® Office documents (Word, Excel, PowerPoint, and LibreOffice equivalents)1 and more
  - OCR and searchable PDF documents from scanned images
  - Automatic tag generation using AI
- **FreeCAD** is a desktop application for 3D modeling with support for BIM and Finite Element Analysis

Develop Your Own Applications

- **Packaging code for distribution on PyPI**
  - Project management: Astral uv
  - Source code version management: GitHub
  - Source code testing: pytest
  - Source code documentation: MkDocs
- **Web Applications Development Frameworks**
  - **Marimo** Notebooks – For individual use or a small office, not full fledged web application
  - **Streamlit** – For simple framework to develop in-house applications
  - **Flask** – For full fledged custom web applications
- **Desktop Application Development Frameworks**
  - **PySide**/**PyQT** – For full-fledged desktop applications
  - **NiceGUI** and **Flet** – For web apps, desktop apps or mobile apps, from single source code

Not Covered in this Talk

- Many structural engineering applications offer a Python API so that you can interact with those applications through your Python scripts
  - Microsoft® Excel
  - Staad.Pro
  - SAP 2000
- Using database to store, organize, and access large amounts of data
- Building web and desktop applications
- Developing packages for distribution to others
- Implementing Machine Learning or using LLMs for structural engineering applications

Python Learning Path

Basic

- Variables
- Boolean operations
- Operators
- Control flow
- Loops and iterables
- Built-in data structures
- Functions
- Mutable vs immutable
- Common methods
- File I/O

Intermediate

- OOP
- Comprehensions
- Lambda functions
- Map, Filter
- Collections
- `*args`, `**kwargs`
- Inheritance
- Magic/dunder methods
- Package management
- Modules
- Virtual environments

Expert

- Decorators
- Generators
- Context managers
- Metaclasses
- Async I/O, Parallelism
- Testing
- Package development
- Cython/Numba

Scientific Python Learning Path

Numerical Computations

- NumPy - Linear algebra
- Matplotlib - Data vis
- Pandas - Data cleaning
- SciPy - Linear algebra

Machine Learning

- Scikit-Learn

Symbolic Computing

- SymPy - CAS

Python Development Tools

Code Editors

- Spyder
- Jupyter Notebooks

IDEs

- VS Code
- SublimeText 3

Package and Virtual Environment Management

- uv
- pip with venv
- conda or mamba
- poetry

Source Code Documentation

- Sphinx with MyST for Markdown
- MkDocs

Source Code Management

- git

Lexers

- ruff
- ty

Resources

- Python home page, Python documentation, Python Package Index
- NumPy, SciPy, Matplotlib, Pandas
- **OpenSees**, **OpenSeesPy** finite element analysis for static and dynamic analysis
- **Marimo** reactive notebooks, Jupyter Lab traditional notebooks
- **Osdag** steel connection design
- **freeCodeCamp** learning resources for free, for several programming languages, including Python

The End