Python for Engineers

Contents

 No AI use

Time required: 60 minutes

This series of tutorials will dive into two powerful tools for crunching numbers and making cool visualizations: MATLAB and Python.

MATLAB: Imagine a Swiss Army knife for engineers. It's got built-in functions for:

- **Math:** Calculus, linear algebra, statistics, you name it!
- **Signal processing:** Filtering, Fourier transforms, all things audio and signals.
- **Control systems:** Designing robots, drones, anything that moves autonomously.
- **Data visualization:** Creating stunning graphs, plots, and animations.

Pros:

- **Easy to learn:** MATLAB's syntax is similar to math equations, making it intuitive for engineers.
- **Powerful built-in functions:** No need to install extra libraries for basic tasks.
- **Excellent documentation and community:** Tons of resources and support available online.

Cons:

- **Expensive:** MATLAB requires a paid license, which can be a barrier.
- **Not as versatile as Python:** MATLAB is primarily focused on engineering tasks.

Python (with NumPy, SciPy, Matplotlib): Think of Python as a modular toolbox. It's a general-purpose language, but with powerful libraries for scientific computing:

- **NumPy:** Provides multi-dimensional arrays for efficient math operations. Think spreadsheets on steroids!
- **SciPy:** Offers advanced algorithms for statistics, optimization, signal processing, and more.
- **Matplotlib:** Creates publication-quality plots, charts, and graphs with tons of customization options.

Pros:

- **Free and open source:** Anyone can use and contribute to Python and its libraries.
- **Highly versatile:** Python can be used for web development, data science, machine learning, and much more.
- **Large and active community:** Tons of libraries, tutorials, and support available online.

Cons:

- **Steeper learning curve:** Python requires more upfront learning compared to MATLAB's intuitive syntax.
- **Requires installing libraries:** NumPy, SciPy, and Matplotlib need to be installed separately.

Jupyter Notebooks with Google Colab

We will be using Python Jupyter Notebooks with Google Colab.

Think of Google Colab as a **digital laboratory notebook on steroids**. You can write code in various programming languages (Python is a popular choice), execute it line by line, and see the results instantly displayed alongside your code. This **interactive environment** makes it perfect for experimenting, testing hypotheses, and iteratively refining your analysis.

Jupyter notebooks are like **blank canvases** where you can paint your data story. Here's a glimpse of what you can do:

- **Write and execute code:** Write Python code, run it cell by cell, and see the output right below.
- **Visualize data:** Create stunning graphs, charts, and plots using libraries like Matplotlib and Seaborn.
- **Format text and equations:** Add rich text formatting, markdown elements, and even LaTeX equations to make your notebook visually appealing and informative.
- **Share and collaborate:** Easily share your notebooks with others for collaboration or feedback, making it ideal for team projects.

Tutorial 1: Get Started with Google Colab

Google Colab is a free cloud based Jupyter Notebook. You will need a Gmail account to use it.

1. Go to https://colab.research.google.com
2. Login with or create a new Gmail account. You can use your wncc.edu email.

Tutorial 2: Google Colab Learning Journal Notebook

Use this notebook to store code, notes, and your assignments as you go through this project.

Create a Google Colab notebook named: {Your Name} **Python Learning Journal**
For example: **Bill's Python Learning Journal**

Use this notebook to store:

- Assignments, code, comments, and ideas about your understanding as we go.
- Think of your notebook as a Data Science learning journal.
- The quality of your notebook will be part of the grade for that week.

Tutorial 3: MATLAB and Python Math Operations

Here are some simple code examples of Math operations to compare the flavor of MATLAB and Python. Save and run each program in the respective environments.

MATLAB

1. Create a MATLAB file named **MathOperations.m**
2. Type in the following code.

```
disp('---   MATLAB Math Operations   ---')
a = 10;
b = 2;

sum = a + b;
difference = a - b;
product = a * b;
quotient = a / b;
exponent = a^b;
squareRoot = sqrt(a);

% Convert variable a to an integer for integer math with idivide
% idivide must have one integer in the operation
a = int16(a);
integerQuotient = idivide(a, b);
modulo = mod(a, b);


disp(['             Sum: ', num2str(sum)]);
disp(['      Difference: ', num2str(difference)]);
disp(['         Product: ', num2str(product)]);
disp(['        Quotient: ', num2str(quotient)]);
disp(['        Exponent: ', num2str(exponent)]);
disp(['     Square Root: ', num2str(squareRoot)]);
disp(['Integer Quotient: ', num2str(integerQuotient)]);
disp(['          Modulo: ', num2str(modulo)]);
```

Example run:

```
---   MATLAB Math Operations   ---
             Sum: 12
      Difference: 8
         Product: 20
        Quotient: 5
        Exponent: 100
     Square Root: 3.1623
Integer Quotient: 5
          Modulo: 0
```

Python

AI is a useful tool for help your speed up your workflow. It can hinder your learning if you use it to complete assignments. The act of typing in the code help imprint what you are doing into long term memory

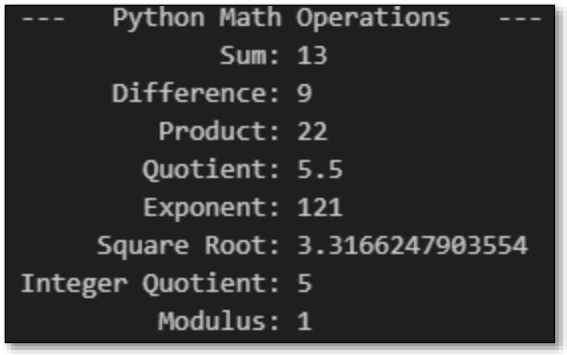
Disable AI

1. In **Google Colab → Tools → Settings**.
2. Un check **Show AI-powered inline completions**.

Type this code into your Colab Notebook.

```
import math
print("---   Python Math Operations   ---")
a = 11
b = 2

sum = a + b
difference = a - b
product = a * b
quotient = a / b

exponent = a**b
square_root = math.sqrt(a)

# Integer division throws away the remainder
integer_quotient = a // b

# modulo is integer division which returns the remainder
modulo = a % b

print(f"             Sum: {sum}")
print(f"      Difference: {difference}")
print(f"         Product: {product}")
print(f"        Quotient: {quotient}")
print(f"        Exponent: {exponent}")
print(f"     Square Root: {square_root}")
print(f"Integer Quotient: {integer_quotient}")
print(f"         Modulus: {modulo}")
```

Example run:



Assignment Submission

- In Google Colab → Click the Share button in the upper right hand side.
    - Change General Access → Anyone with the link → Click Copy link.
- Attach the MATLAB code and screenshots of the command window showing successful execution of the programs.
- Submit in Blackboard.