Open in Colab

# Lab 1: Intro to Python. APPM 4600

Note: this lab is a bit awkward because it meets before we've had our first lecture!

There are three parts to this:

1. Python installations

- you may wish to do some of this in advance

2. Some basic coding
3. Deliverables: combining parts 1 and 2

- Upload this to Canvas

Read about the lab grading system

Please skim all instructions first before starting to work

## Learning objectives

- Gain familiarity with Python. If you're new to it, realize that it "doesn't bite"
- Learn how to use Python in different environments.
  - Colab is easy, but can you debug effectively in Colab?
- Decide if your linear algebra skills (from the prereq class APPM 3310) need improvement
- Learn how to submit code to Canvas
  - ... and avoid screenshots!

## Part 1: Python installations

**Short-version**: get at least 2 different "types" of Python working, 1 local to your computer and 1 cloud-based

**Details**

There are many ways to use Python, and it can be used on your own computer or on a cloud server. The goal of this part of the lab is to set you up for future computing in the class.

## Get Python to run **locally** on your computer

- You may need to install Python first; there are many ways you can do this, but a popular and suggested method is via the Anaconda distribution
- To run Python locally, do one of the following methods (do at least one, but we suggest trying several):
    - From the command line or powershell, run a Python script, like `python myscript.py`
    - Run IPython from the command line
        - This opens up a REPL environment much like Matlab, and lets you interactively code. This is nice for plots
        - Fun fact: this was developed by Fernando Pérez) when he was a grad student at CU Boulder in the early 2000s
    - Run a Jupyter notebook, which lets you run `.ipynb` files. This is the modern extension of iPython
    - Run a JupyterLab session. This is an extension of Jupyter notebook that gives the features of an IDE. **recommended**
    - Use an IDE like VSCode, PyCharm, Spyder
        - As of 2025, VSCode dominates the market, and using VSCode efficiently is a skill employers value, so we highly **recommend** it. It can edit pure Python file ( `.py` ) as well as jupyter notebooks ( `.ipynb` )

## Get Python to run on a **cloud** server

- Computer Science majors have access to the CS cloud servers
- Use Google's colab service which is a variant of jupyterlab. **recommended** due to simplicity
    - There are instructions on the web about how to link this to a google drive account
- Use CU's research computing (CURC)
    - You may need to make an account; this will be linked to your usual CU identikey. It requires 2FA
    - You can either run code remotely via the command line, or...
    - login to ondemand.rc.colorado.edu., go into "Interactive Apps", and choose either "Jupyter session" or "VS Code-server"

Also, please read our classes AI Policy

# Part 2: basic coding

**Short-version:** write your own code to multiply matrices and compare it to Numpy code

**Details**

- NumPy is the base scientific computation package for Python and we will use it extensively. It is the building block for other packages like SciPy (Scientific Python)
- So make sure it is installed on your system!
  - Pro tip #1: if you want max performance, select a good BLAS when you install it, e.g., MKL if possible. *Not too important for us, only for nerds who want the very best performance*
  - Pro tip #2: in Python, when installing packages, do it with virtual environent, either with the package manager `conda` (which comes with Anaconda), or with `pip` and `venv`. This way, if you have some packages that conflict or get into isues with dependencies, it only affects your current "environment", so it won't mess up your operating system and it won't affect other projects (say, for a different class or different assignment)
  - (Optional) We will be using other packages besides NumPy this semester. Some other recommended packages to install are
    - Matplotlib
    - SciPy
    - Jupyter
- They have great documentation, e.g., NumPy: the absolute basics for beginners
  - See NumPy for MATLAB users if you know Matlab well
- Confusing, NumPy has objects called "matrices" which you should **not** use. Use their "arrays" instead; see "'array' or 'matrix'? Which should I use?"
- Write a function that takes as input two matrices and then multiplies them together
- Create two small matrices, $A$ and $B$, and compare the output of your code to the output of Numpy's code (in Numpy, you can multiply arrays by just using the @ symbol, like `C = A @ B`). Make sure your code is correct
  - Pro tip: pass in rectangular matrices (but make sure the inner dimensions match) in order to find bugs
- Bonus: compare the speed of your code to Numpys

In [ ]:
```
import numpy as np
rng = np.random.default_rng(123456)
A = rng.integers(10, size=(2, 2))
B = rng.integers(10, size=(2, 2))
C = A @ B
print(C)

def my_matmul(A, B):
  m, n = A.shape
  n2,k = B.shape
  if n != n2:
    raise ValueError("Incompatible shapes for matrix multiplication!")

  C = np.zeros((m,k))

  for i in range(m):
    for j in range(k):
      C[i, j] = np.sum(A[i, :] * B[:, j])

  return C

# and now check your output vs numpy's output
print( my_matmul( A, B ) )
```

```
[[ 0 54]
 [24 27]]
[[ 0. 54.]
 [24. 27.]]
```

### Optional

Compare the speed of your code to Numpy

## Part 3: deliverables

In Python, write code that:

1. prints the version of Python (e.g., Python 3.5)
2. prints the version of Numpy
3. prints the output of your matrix multiply code form part 2, using any two small matrices as input

Export the output to a PDF and upload that to Canvas (*not* Gradescope):

- no screenshots!
- please follow our HW/lab submission guidelines

In [26]:
```
from sys import version
print("Python version: ", version)
print("Numpy version: ", np.__version__)

my_matmul( np.array([[1,2,3],[4,5,6]]), np.array([[1,2],[3,4],[5,6]]) )
```

```
Python version:  3.9.6 (default, Dec  2 2025, 07:27:58)
[Clang 17.0.0 (clang-1700.6.3.2)]
Numpy version:  2.0.2
```

Out[26]:
```
array([[22., 28.],
       [49., 64.]])
```