**Implementing a Matrix Class**

In this project you will implement a Matrix class in Python. Specifically, you will implement the following methods:

**class** **Matrix**:

**def** **determinant**(self):

*# your code*

**def** **trace**(self):

*# your code*

**def** **inverse**(self):

*# your code*

**def** **transpose**(self):

*# your code*

*# Overloaded operators*

**def** **\_\_add\_\_**(self,other):

*# your code*

**def** **\_\_sub\_\_**(self,other):

*# your code*

**def** **\_\_mul\_\_**(self,other):

*# your code*

When your class is working properly you will be able to manipulate matrices in code as if they were regular numbers (for the most part). For example:

> A = Matrix([

[2,4],

[3,1]

])

> print( A.transpose())

2.0 3.0

4.0 1.0

>

> I = Matrix([

[1,0],

[0,1]

])

>

> print(A\*I)

2.0 4.0

3.0 1.0

>

> print(A \* A.inverse())

1.0 0.0

0.0 1.0

# Project Instructions

Your project workspace will contain several files. On the next page, you will see something like this:

A screenshot of a cell phone

Description automatically generated

We recommend that you open all of these files (except for two noted below) in new tabs so that your browser looks something like this:

A screenshot of a cell phone

Description automatically generated

**You can ignore datagenerator.py and test.py, which you won't need to modify**

1. **matrix.py** - This contains the beginnings of a Matrix class (which you will complete) as well as some helper functions zeroes and identity. **This is the file you will be doing most of your work in.**
2. **matrix\_playground.ipynb** - A notebook that imports your Matrix class and calls the test code. You may find it useful to use this notebook as a place to **use** the matrix math code you will write in matrix.py.
3. **matrix\_cheat\_sheet.ipynb** - A Jupyter notebook with a **glossary**, **explanation of matrix notation** and list of **matrix equations**. Use this as a reference when filling out the methods in the Matrix class!
4. **kalman\_filter\_demo.ipynb** - You don't need to do anything with this notebook but you may find it interesting. Once your matrix class is working properly, the KF implemented here will actually work!

**Other Files** (feel free to ignore).

1. **test.py** - Contains test code which demonstrates the expected functionality of your code.
2. **datagenerator.py** - this just contains some helper code which is used by the Kalman Filter.

## Evaluation

Once you have completed your project, use the [**Project Rubric**](https://review.udacity.com/#!/rubrics/1154/view) to review the project. If you have covered all of the points in the rubric, then you are ready to submit! If you see room for improvement in any category in which you do not meet specifications, keep working!

Your project will be evaluated by a Udacity reviewer according to the same Project Rubric. Your project must "meet specifications" in each category in order for your submission to pass.