Assignment #1

Due: 23:55pm February 12, 2020

Upload at: https://www.gradescope.com/courses/75501/assignments/316145

Problem 1 (10pts)

In COS 302 you'll typeset your assignments in LaTeX, which is the way that almost all documents are composed in mathematical fields. It's a good thing to learn and LaTeX can produce beautiful documents. The first part of this assignment is going to get you started with it. There are a variety of ways to write and compile LaTeX, from web-based tools like Overleaf, to various GUI tools for Mac, Linux and Windows, to plain text editors like Emacs.

If you aren't an experienced user of LaTeX, with tools already installed on your computer, we **strongly** recommend that you use Overleaf for this course. Overleaf runs in your browser so you don't have to worry about Mac vs Linux vs Windows vs Chromebook. To make this as easy as possible, we'll post links to templates of each homework to get you going. Just use the "copy project" option in the menu to the left. We have posted a screencast to walk you through it.

There are two files in the project. One of them, `cos302.cls` is a formatting ("class") file that you should not have to worry about or edit. The other, `hw1.tex`, is the file you'll actually edit. When you typeset in LaTeX, you're basically just writing markup code. You then compile it and get a nice looking PDF file. Note that this compilation process just produces a PDF; it does not submit it for grading automatically or anything like that. To submit the PDF, you'll need to upload via Gradescope. A screencast is available to walk you through the process of submitting the homework.

To start out, look at the very top of the file where you see:

```
% No 'submit' option for the problems by themselves.
\documentclass{cos302}

% Use the 'submit' option when you submit your solutions.
% \documentclass[submit]{cos302}
```

Lines that start with the "%" symbol are comments. LaTeX commands start with backslashes and arguments are usually in curly braces {...} with options in square brackets [...]. The `\documentclass` command starts the document and declares what kind of thing we're writing. Here the document class is `cos302` because we're using a custom class for assignments; in scientific documents it would be `article` 90% of the time. In any case, here there is one possible option, `submit`. You can turn that on by commenting out the first `\documentclass` and uncommenting the second one that has the `[submit]` option active. Do that and compile the document. What happened? Explain what happened by replacing the commented text below that looks like this:

```
\end{problem}

% UNCOMMENT AND PUT PROBLEM ANSWER HERE.
```

Problem 2 (10pts)

Now, hopefully you discovered in the previous problem that when the document compiled with the `[submit]` option, it put a name, email and the course info in the top left corner, and a "discussants" section in the top right. You won't be surprised to hear that you should change these in the code above (the "preamble" is what this part before the `\begin{document}` is called) to be your actual name and email address. The "discussants" section is a place to list the people that you discussed the problems with (consistent with the collaboration policy, of course).

```
% Put in your full name and email address.
\name{Your Name}
\email{email@princeton.edu}
\discussants{Marie Curie \\ Leonhard Euler}
```

The double backslash is a way to insert a line break if you want one. You really do need to change this on each assignment and use the `[submit]` option, so that your name appears at the top. (You get this problem right by putting your name and email in the document.)

Problem 3 (10pts)

The reason mathematicians, computer scientists, and other technical folks like LaTeX is because it typesets equations so nicely. This problem will give you a little bit of practice with that. Good written communication is important. There are lots of tutorials online for typesetting mathematics with LaTeX, but this is a good starting point. Use the `equation` (or equivalent) environment to typeset the following:

(A) Euler's formula

(B) the Taylor series of the exponential function

(C) the normal equations with matrix $\boldsymbol{A}$, $\boldsymbol{x}$, and $\boldsymbol{b}$.

Don't worry if you don't know what these are yet. Just find their forms on the links above and complete the equations below. Soon you'll be on your way to being a LaTeX typesetting pro.

(A)

$$e^{ix} = ??? \tag{1}$$

(B)

$$e^{x} = ??? \tag{2}$$

(C)

$$??? \tag{3}$$

Problem 4 (18pts)

Now let's use LaTeX to do some problems. Multiply these matrices, if possible. If not possible, say so.

(A)

$$\begin{bmatrix}1 & 2\\4 & 5\\7 & 8\end{bmatrix}\begin{bmatrix}1 & 1 & 0\\0 & 1 & 1\\1 & 0 & 1\end{bmatrix}$$

(B)

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}\begin{bmatrix}1 & 1 & 0\\0 & 1 & 1\\1 & 0 & 1\end{bmatrix}$$

(C)

$$\begin{bmatrix}1 & 1 & 0\\0 & 1 & 1\\1 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$$

(D)

$$\begin{bmatrix}1 & 2 & 1 & 2\\4 & 1 & -1 & -4\end{bmatrix}\begin{bmatrix}0 & 3\\1 & -1\\2 & 1\\5 & 2\end{bmatrix}$$

(E) Did you notice anything interesting about (B) vs (C)? Do they have the same answer?

(A)

$$\begin{bmatrix}1 & 2\\4 & 5\\7 & 8\end{bmatrix}\begin{bmatrix}1 & 1 & 0\\0 & 1 & 1\\1 & 0 & 1\end{bmatrix} = ???$$

(B)

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}\begin{bmatrix}1 & 1 & 0\\0 & 1 & 1\\1 & 0 & 1\end{bmatrix} = ???$$

(C)

$$\begin{bmatrix}1 & 1 & 0\\0 & 1 & 1\\1 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix} = ???$$

(D)

$$\begin{bmatrix}1 & 2 & 1 & 2\\4 & 1 & -1 & -4\end{bmatrix}\begin{bmatrix}0 & 3\\1 & -1\\2 & 1\\5 & 2\end{bmatrix} = ???$$

(E)

Problem 5 (10pts)

The other fantastic browser-based tool we'll use in this course is Google's Colaboratory. Colaboratory is an environment for running Python notebooks that is *free* and runs in the cloud. So that means you don't have to worry at all about setting up a local environment. This is based on an open source project called Jupyter. Colab/Jupyter notebooks are a way to write code in the browser, create visualizations, interact with data, and write documentation all in one place. They are extremely common in the workflows of machine learning practitioners and data scientists, as they make it easy to iterate on things and see the results. Much like LaTeX, it's a good investment to learn how to use Python notebooks. Typically, you would set up Jupyter on your own computer, but in this course we'll take advantage of the cloud and use Colab.

I suggest going and checking out the Welcome Notebook and watching the video there to get an idea as to what Colab is about, along with pointers to resources that you might find interesting for using it to do machine learning. A more comprehensive Overview Notebook is also available and worth taking a look at.

The goal if this problem in the assignment is to get you started using Colab.

(A) Go to colab.research.com and select "New Python 3 Notebook".[a]

(B) Rename the notebook by clicking on the filename in the upper left corner. Give it a name like `COS 302 HW1 - Firstname Lastname.ipynb`.

(C) Make the first cell into a text cell via `Ctrl-m m` or insert a text cell and move it to the top. As mentioned in the video, these text cells can be easily formatted using Markdown. Put the notebook title into that first cell using something like:

```
# COS 302 HW1 - Firstname Lastname
```

Use `Shift-Enter` to execute the cell and render it.

Keep this notebook open for the next problem.

[a] If you know and care about this kind of thing: we're going to use Python 3 exclusively as Python 2 has now entered sunset.

Problem 6 (15pts)

Now that you have a Colab notebook going, it's time to write a little bit of code.

(A) Start out by making a text cell with a level-two header identifying what problem you're working on:

```
## Problem 6
```

This is something you should generally do in the class to make it clear which parts of the notebook correspond to which parts of the assignment.

(B) We're assuming that you've written some code before, but perhaps not Python specifically, so in this problem we'll hit some of the basics of the syntax. (Python is great, but this isn't going to be a comprehensive Python course.) Let's start out in the most basic way with a "Hello world" code cell:

```
print("hello world!")
```

Do a `Shift-Enter` or hit the play button on the left to execute this.

(C) The key thing that can be weird at first with Python is that blocks are delineated by indentation rather than, e.g., curly braces. That means that in Python whitespace matters in a way that it doesn't for languages like C and Java. In a code cell, write an `if/else` statement that looks something like this:

```
course_name = "COS 302"
if course_name == "COS 302":
    print("This is super fun!")
else:
    print("Fun, but I wish I was in COS 302.")
```

Here we use the equals sign to assign a string to the variable `course_name` and then a conditional that compares that variable to a string. The equality comparison is done with a double-equals; the single vs. double equals thing here matters as the single equals does assignment and not comparison. The `if` statement doesn't need to be in parentheses, although this is permitted. The conditional statement ends in a colon and then the next line is indented to indicate what code should run if the statement is true. It does not matter how many spaces there are, but the block all has to have the same indentation.

(D) Create and execute code cells for each of the following and figure out which work and which don't:

```
if course_name == "COS 302":
        print("This is super fun!")
        print("I love Python.")
else:
    print("Fun, but I wish I was in COS 302.")
```

```
if course_name == "COS 302":
  print("This is super fun!")
  print("I love Python.")
else:
print("Fun, but I wish I was in COS 302.")
```

```
if course_name == "COS 302":
  print("This is super fun!")
   print("I love Python.")
else:
  print("Fun, but I wish I was in COS 302.")
```

Problem 7 (15pts)

Outside of the basic language, most interesting functionality in Python happens in packages. Some of these are distributed with Python, some are commonly installed by default in comprehensive distributions like Anaconda, and others need to be installed using tools like `pip`. Part of the reason we're using Colab is so we don't have to worry about all that and you can just `import` things and get to work.

Start a new problem in the notebook using a level-two heading in a text cell, as described previously.

(A) For basic scalar math, the included `math` package will get you going. Figure out how to write a loop that implements a chaotic map defined by $f(n+1) = \ln|f(n)|$, where $f(0)$ is any real number excluding 1 and e. Complete the `TODO`s below in a fresh code cell.

```
import math
num_iters = 10
f = # TODO
for ii in range(num_iters):
  f = # TODO
  print(f)
```

Here we're doing two new syntactual things: using an `import` statement, and using a `for` loop. The `import` is fairly self-explanatory, and the `for` loop has the same colon-and-whitespace structure as the conditional from the previous problem. However, now you're looping with `ii` taking values from `0` to `num_iters-1`. The `range` function here returns a sequence of integers that you can iterate over.

(B) For this course, and for most non-trivial mathematical computations in Python, we'll use NumPy. NumPy lets you do things fast with vectors and matrices of floating point numbers. Let's rewrite this code using NumPy. Create a new code cell and fill in the `TODO`s:

```
import numpy as np
num_iters = 100
f_values = np.empty(num_iters)
f_values[0] = # TODO
for ii in range(1, num_iters):
  f_values[ii] = # TODO
print(f_values)
```

The "`as np`" part of the import just lets you write `np.foo` rather than `numpy.foo` and is pretty conventional. Another differences is that we're pre-allocating the memory for the values by creating a NumPy array (vector) of size `num_iters`; you'll often also see code that does this with `np.zeros`, which allocates a zero array and is about as fast. Finally, because we're operating on pre-allocated memory, we're indexing into the array using square brackets.

(C) One of the things we often want to be able to do is visualize data. Matplotlib is not the only package out there for doings this in Python, but it is powerful and popular, so we'll use it in this course. It also integrates nicely with Colab notebooks, so well in fact that the notebooks have special features to make figures easy to embed. Turn on this integration with:

```
%matplotlib inline
```

Then import the package and display the values you've generated via:

```
import matplotlib.pyplot as plt
plt.plot(f_values, '.')
```

Problem 8 (10pts)

For the last problem, you learn how to turn in the Colab part of the homework assignment. There will be two aspects to this: attaching a PDF to the LaTeX file, and giving us a read-only link to your notebook. We've made a screencast to help you walk through this.

(A) Make sure you have run all of the cells in your notebook so that we can see the output. If we can't see the output in the PDF, you won't receive credit for the problem.

(B) Close the table of contents on the left side so we can see your work clearly.

(C) In *File » Print*, print to a PDF file and save it to your computer.

(D) Insert the PDF into this document. In Overleaf, that means uploading the PDF to the project; if you're editing LaTeX locally then you'll probably need to put it in the same directory as this file. At the end of this `.tex` file, you'll see the following lines:

```
%\includepdf[pages=-]{mynotebook.pdf}

\end{document}
```

You should uncomment the `includepdf` command and change the name of the PDF to the one that you just downloaded. Note that you probably don't want any spaces in the PDF file name. If this worked, after you compile you should see your notebook at the end of this document.

(E) We'd also like to be able to run your code ourselves. As a side effect, you'll get to learn a little bit about version control. For that, we'd like a link to what you consider your final version. Go to the *File* menu in the notebook (not the browser itself) and select *Save and pin revision*. It's helpful to rename the revision so that we know exactly which one you want graded, so open *File » Revision history* and then click the little three vertical dots next to your pinned revision and rename the file to something like "Submitted". Now close the revision history and click the *Share* button in the upper right. Make sure you have *anyone with a link can view* selected, copy the URL and then paste it below, replacing the dummy URL.

My notebook URL:

`https://colab.research.google.com/XXXXXXXXXXXXXXXXXXXXXXX`

You probably realize that you can continue to edit your notebook after the deadline. Don't do this, as we'll be comparing your PDF to the notebook URL.

Problem 9 (2pts)

Approximately how many hours did this assignment take you to complete?

Changelog

- 2 February 2020 – Distributed version.
- 6 December 2019 – Initial version.