Lab 01 - Getting Started

Submission: `lab01.ipynb`, pushed to your Git repo on `master`.
Points: 10
Due: Friday, January 18, 11:59pm

Objectives

- Installing Anaconda for your OS
- Setting up your Git repository
- Setting up your system for Jupyter Notebook (and optionally, an IDE)
- Pushing your first Python notebook

Introduction

A major change is underway in the data science community. There is a battle going on! R vs. Python! Which do we choose? I monitor the trends, and have been doing so for the last decade. Over the past 5 years, there has been a steady uptick in Python users in the data science community. Now, in my opinion, it has overtaken the R userbase as a preferred language for data science. Therefore, for the first time in this course, I will be requiring students to use Python instead of R.

This is an exciting change! First, every student in CSCI 349 must have Python experience already from prior courses. Therefore, unlike R, I can be sure that students will rapidly come up to speed with the language itself. Instead, I can have students focus on learning specific packages that are used by the data science community in Python.

Let's get started!

Active Learning

First, let's make sure every student has a complete understanding of how this course will work. How will you learn the tools in this course? **Active Learning**:

> **Active Learning** – a form of learning in which teaching strives to involve students in the learning process far more directly than other methods. My other primary course I teach, CSCI 205, should have shown you that I am a *strong* advocate of active, hands-on learning. The science is clear - you learn far less when you sit and listen to me lecture. You learn far more by actively doing things to learn topics!

Active Learning requires you to be a active student, playing an active role what you learn, and how you learn it. Moreover, as an advanced elective, with respect to the tools in this course, more responsibility will be on YOU to learn them. What does this mean?

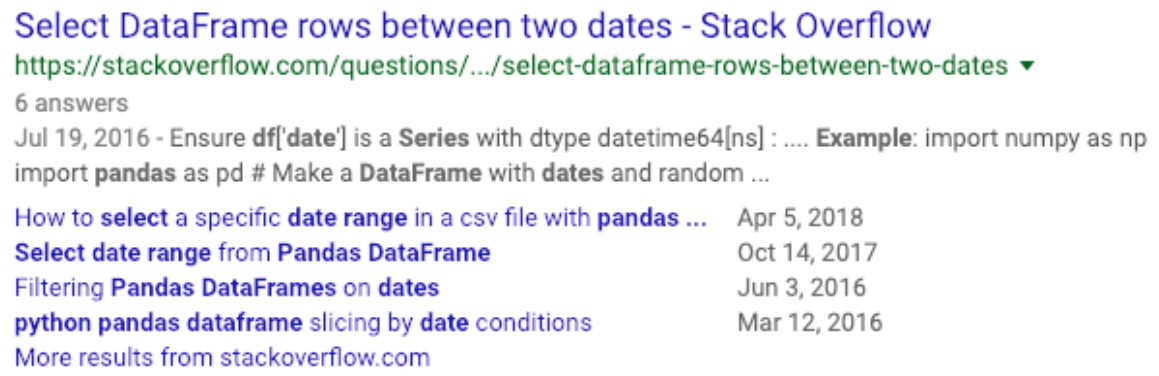
Google, Stack Overflow and Slack will be your closest friends

Learning all of the Python skills and packages used in this course will be *entirely* on your shoulders. As junior and senior CS students, you must be comfortable using Google to search for answers.

For example, over the next few weeks you will soon learn about *data frames*. Data frames are the most common data structure used to store data for analysis and mining purposes. (Think of a data frame as a spreadsheet, arranged in rows and columns.) Data frames are a core part of the `pandas` package in Python. I may give you

some resources (tutorials, exercises, etc.) to get you going on teaching yourselves the basics of the packages we will be using. However, you will have many specific challenges you need to solve on your own.

For example, you may have to select some data from your data frame by a date range. I'm not going to show you how to do this. Become comfortable with using Google to search phrases such as "`pandas dataframe select by date range example`". When I enter this (as of early Jan 2019), I get the following as my top hit:



And indeed, those links contain lots of examples and discussions on how to make this happen! You will come across many, many examples of code to help you solve problems you come across, ranging from installation challenges, system challenges, and most often, just simple coding challenges where you simply need an example to get you past it. And, most of the time, your top hits will be from Stack Overflow. The point is this - **you will have no lack of example code to pull from throughout this entire course**! The Internet is wonderful! And, I expect you to use it to your advantage.

When is Googling cheating?

That is actually a very good question. The work that you submit *must be your own work*. However, I also like to model advanced courses to resemble real world scenarios. Many of you will be headed to the work force, and you will face software design and development challenges. You will jump on the internet and copy over code to help you solve those challenges. There is no reason to "reinvent the wheel" over and over! That is a waste of your time.

However, sometimes the best way to learn is indeed to reimplement the wheel on your own. There is value to doing so, particularly in the classroom setting. Thus, there will be times, you will need to clearly implement code that you know is readily available on the Internet. What do you do? How do you handle it? I'm fine with you reading through other example code online for ideas, but make the work your own. AND, I stress this very strongly:

> You MUST reference all resources in every notebook file you submit that contains code that is not your own. Even reference code that gave you inspiration on how to solve a problem.

Here is your self-check – could you develop a solution to the code you referenced on your own? Do you have a solid understanding of the code that gave you the inspiration to solve the problem? Could you answer basic questions that required you to write similar code in the future? Yes? Then, you're good. Just reference it. However, if you are blindly copying, without absorbing any material in your brain, this is a dangerous place to be, especially for this course, where labs and homework will become increasingly complex. **Don't be ignorant and blindly copy the work of someone else! It will harm you, and will place you at risk of outright plagiarism and having to go before the Board of Review, risking your failure in this course.**

Installing Anaconda Distribution

In general, most of what you will need for the semester are all wrapped up and contained in **Anaconda Distribution**. Therefore, though all of the different tools and libraries that you heard about in your first lecture may sound overwhelming, you will soon understand why Anaconda is so popular in the data science community. It contains nearly everything you need, including Python, choices for IDEs, and hundreds of packages! And, they have worked hard to make the integration simple.

Preparing yourself

Before you begin, take the time to browse through https://docs.anaconda.com/anaconda/ and understand what you are doing. Focus only on **Anaconda Distribution**. Pay close attention to the installation directions for your operating system, and then read through the **User guide**. (You might consider watching a YouTube video or two by searching for "*getting started with Anaconda*" or "*Anaconda tutorial*" to help you understand a bit more about this powerful platform.)

Once you are feeling comfortable that you have some sense of what Anaconda is all about, go ahead and download and install **Anaconda Distribution** for your operating system. The link is provided in the documentation.

What is the difference between **conda** and **Anaconda**? Conda is a general package manager system. It is not necessarily specific to Python, or R. However, it has generally gained the most popularity from Python and data science. There have been so many open source packages developed by the community, often dealing with very specific problems, it was clear a better system needed to be put in place to manage packages. More importantly, we also needed very self-contained, specific **environments** that would prevent one Python installation from containing every single package you ever downloaded! Environments are very important in the Python ecosphere, and conda helps manage that quite well.

Anaconda is a specific distribution of conda, bundled with thousands of files including python, R, libraries, IDEs, and all the packages you will need. It provides a GUI interface for managing your installation by executing conda commands for you.

Getting started with Anaconda

Once you have Anaconda installed, follow through each of these sections in the user guide at https://docs.anaconda.com/anaconda/user-guide/ .

- **Getting started with Anaconda**
  - They will have you run an IDE called Spyder and enter a simple "Hello Anaconda" program. This is one of many IDEs available. You can run it to get through this first exercise, but keep in mind that there are alternative choices for IDEs, discussed later.
  - You will also see how to launch Jupyter Notebook. Take the time to create a new notebook file (which always have a file extension of .ipynb.) These files will always contain your solutions for labs and homeworks. Name it whatever you want. You should get used to creating your own notebooks for to keep track of your own learning. In time, you should have notebook files that server as good reference material for yourself.
- **Anaconda Navigator**
  - This section will introduce you to environments
  - Environments allow you to set up distinct Python installations with a specific set of packages installed for that environment.
  - You seriously do NOT want to underestimate the importance of understanding environments in conda. You can create environments to test and learn, then remove them when you are done.

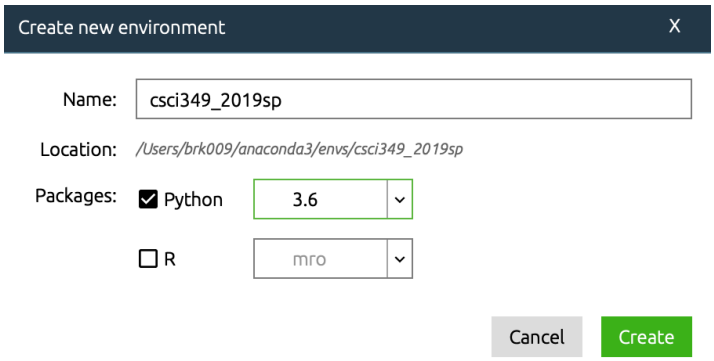
  - **MOST Anaconda experts will strongly advise that you never work in the base (root) environment. You will create a new environment for this course below.**
  - **You can manage the packages that are installed for each environment. Again, don't mess with the base environment.**
- **Conda**
  - Browse through the conda documentation, just so you can understand what conda does as you move forward.
  - You mostly want to pay attention to this document - https://conda.io/docs/user-guide/getting-started.html
  - You will not learn everything about conda. Just get a sense of the main commands, how to execute commands, and keep a reference handy for it.
- **Frequently Asked Questions**
  - Again, just browse through the page to get a sense of how to manage your environment, your packages in each environment, etc.

Anaconda Navigator will be installed as part of the Anaconda distribution. This is your "go to" for managing your entire distribution, and is particularly useful for beginners. I have rarely need the command line to configure or execute something with conda. Though sometimes I use it just to verify that my environment is active (`$ conda env list`) Most of the time, I am able to work right from Anaconda Navigator. However, it is useful to understand how to use the `conda` command.

Set up an environment for CSCI 349

For this next step, do the following:

1) Run Anaconda Navigator
2) Create a new environment called `csci349_2019sp`. SELECT PYTHON 3.6! Do not use 3.7.



3) Be sure your new environment is active. Then, install the following packages in your new environment

```
numpy
scipy
pandas
statsmodels
matplotlib
scikit-learn
jupyter
jupyter_kernel_gateway
```

There will be many additional dependencies that are installed when you install these packages (if they weren't installed already.) Some of the above will be installed as dependencies.

NOTE: You should expect that even more packages will be installed as the semester progresses.

4) GO back to the home screen in Navigator, and click on the **Install** button for **Jupyter notebook**

NOTE – if you decide to use Spyder or VS Code as your IDEs, you can install them now. It's easy to remove them later from your system if you decide you do not want to use them.

Cheat Sheets

I'm generally a fan of cheat sheets. I like to print them and keep them handy on my desk, since I sometimes just need a quick reminder of some method or syntax for dealing with pandas data frames or matplotlib that I'm forgetting. Use them however you see fit.

- Anaconda Distribution - https://docs.anaconda.com/_downloads/Anaconda_Starter_Guide_CheatSheet.pdf
- Conda - https://conda.io/docs/_downloads/conda-cheatsheet.pdf
- DataCamp (which is a site I am also a fan of) has provided many cheat sheets to help you recall most common tasks that you'll need to do with our different packages. They have LOTS of Python cheat sheets for most of the packages we'll be using, but pay close attention, since many of these listed here are also for R, which you won't need. Keep a reference to these, and print them out as you need them - https://www.datacamp.com/community/data-science-cheatsheets

Git

As you likely guessed before coming into this course, you will be using a Git repository to manage all of your coursework for this class, including submitting your own files.

TODO:

- Go to https://gitlab.bucknell.edu. Log into your account. You should already have an account as you must have taken CSCI 205 and other CS courses that use Git to be enrolled in this course.
- Create a new project named **`csci349_2019sp`**. Be sure the visibility of your project is Private. And initialize your repo with a README markdown file.
- Click on **Settings**, then **Members**. Add the instructor as a member
  Brian King (@brk009)
  Set role as Developer
  Click Add to project
- Assume that all work submitted for grading and verification must be submitted to the **`master`** branch unless otherwise noted. I will only pull your master branch!

Don't clone your repo locally yet. That will be done soon.

IDE

I will *not* dictate what IDE you should use for this course. There are *many* options, and you can complete this course with any of them. (In fact, you are free to use IDLE and the command line, though I would advise against it.) I recommend that you choose one that has direct support for data science workflows (any of the options I discuss

below will work.) Additionally, choosing an environment that directly supports Git, markdown, and notebook files will make your life more enjoyable

NOTE: Whatever path you choose, make sure you configure your IDE to use your conda environment! Usually it's simple to configure. You just need to make sure you do it!

I recommend evaluating one or more of the following:

- **Pycharm**
  - If I was asked to make a #1 recommendation, this would be it. I did my entire sabbatical research during '17-'18 using Pycharm. It has direct support for everything we will be doing. Technically you would not need any other editor. Additionally, if you are familiar with any other IDE by Jetbrains, you will feel at home with this one.
  - https://www.jetbrains.com/pycharm/
  - It has integrated support for conda environments you create! But, you need to be sure to set this properly in your project settings. Google how to configure this in your IDE.
  - I believe that you can register as a student and get a license for the full professional version, but this will need to be confirmed.
- **Atom**
  - https://atom.io/
  - Atom has a rich set of packages driven by an active open source development community. It includes excellent support most things the computer scientist would need to do, through a vast package repository. I wish every CS student used Atom.
  - I recommend the following Atom packages to get you started:
    - git-plus
    - Hydrogen - https://nteract.gitbooks.io/hydrogen/docs/Usage/GettingStarted.html
    - hydrogen-launcher
    - data-explorer
  - Unfortunately, as of this writing, there is no direct support for .ipynb files. Atom does have a jupyter-notebook package, but it's not being maintained, and I failed in getting this to install properly. However, I did not spend a lot of time on it, and if a student gets this going, PLEASE post on Slack how you did it!
  - It's strength is the Hydrogen package. It's really nice for Python development!
  - To run atom from your conda environment, you need to start your conda environment from Navigator, open a terminal from Anaconda, and then start atom from the command line to be sure your environment is being used.
  - Though there is not direct support for notebook files, you can use Hydrogen to export notebook files. Pay close attention to https://nteract.gitbooks.io/hydrogen/docs/Usage/NotebookFiles.html to see how easy it is to automatically generate your notebook files. This is a pretty good option to consider, as long as you remember to generate your latest notebook file before you submit your homework! (Remember, I will not check .py files, only your notebook files!) However, be sure to open the resulting notebook file in Jupyter and run every cell to generate the output you specify!
  - Hydrogen is wonderful, but some may prefer Pycharm due to being more robost for handling everything you will need to do without additional packages.
  - Remember, take care regarding how you run atom. If you simply double click the system icon, you will not start atom with access to the proper conda env. Start the terminal from Anaconda Navigator, and then run atom from the terminal prompt.

- **Jupyter Notebook**
  - Technically, you can do the vast majority of what you need right within a Jupyter notebook browser window. For those who do not want to learn yet another IDE, this is a viable alternative.

- A nice aspect of this is that you start Jupyter Notebook right from Anaconda Navigator, so your environment will be set properly.

There are a others worth mentioning, each of which would be valid, but I did not try these out:

- **Rodeo** – This still seems to have a nice community of developers working on it. The aim was to be a clone of RStudio, which is the best IDE for R developers.
- **Spyder** – This seems to have a pretty good following, and is has direct support for installation right from Anaconda Navigator. Additionally, I believe that it has direct support for editing Python notebook files through a plugin.
- **VS Code** – You can also install this directly from Anaconda Navigator. This seems to get more attention from the R community, but is increasingly getting support from Python. It is Microsoft's answer to an Atom type of open source editor. (Microsoft has become a bigger player in data science with its acquisition of the R platform.) It's open source, and seems pretty good. But, when I compared this to Atom, I was convinced that Atom had a stronger user base with a wider range of plugins, at least at the time of ths write-up.
- **Jupyter Lab** – This is only in alpha stage at the time of this writing, but seems to be the way the Jupyter community is headed. I did not try it out.

Slack

We will use slack for course communication. Do NOT send e-mails. You have likely used Slack in a previous course, so time will not be wasted here explaining how to use Slack. It's quite intuitive.

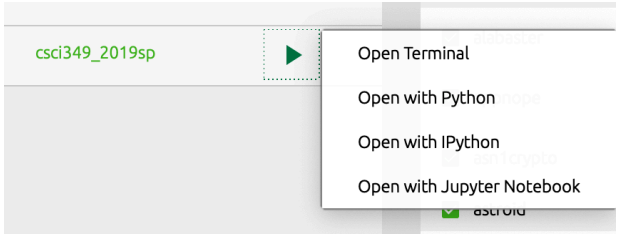
Do the following to sign up for our Slack workspace:

- Go to Moodle and use the signup link, located in the top section
- Use your Bucknell e-mail address to sign in to Slack
- Use your full name for your display name and full name entries. NO NICKNAMES, PLEASE
- You are strongly recommended to download the Slack app for your mobile device as well, but certainly not required.

Set up your project for the course

You need to set up a project you will use for the course, and be sure the project is connected to your Git repo locally and remotely.

1) Determine the directory on your own computer that you want to keep your files for the course, including python code, notebook files, data files, etc.
2) Start Anaconda Navigator. Set the active conda environment to the one created above for the course.
3) Open a terminal from Navigator:



4) Execute the following at the prompt:

$ python –version
$ which python

```
(csci349_2019sp) bash-3.2$ python --version
Python 3.6.8 :: Anaconda, Inc.
(csci349_2019sp) bash-3.2$ which python
/Users/brk009/anaconda3/envs/csci349_2019sp/bin/python
```

Note the version you are using. It should be a 3.6 variant of Python. This is the Python installation for your conda environment for this course. So far, so good!

5) Now, change your directory to where you want to store your workspace files. Then, from that directory, use `git` to `clone` your Gitlab project you created for the course.

6) Create two new directories: `hw` and `labs`. Check out all your files you have so far. You should see something like the following (though you will not have a `.idea` folder unless you already set up PyCharm!):

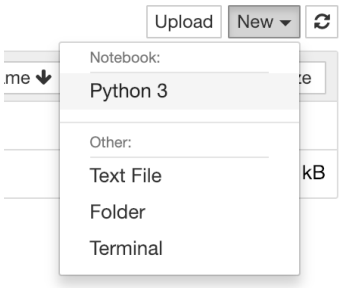
```
(csci349_2019sp) bash-3.2$ ls -la
total 56
drwx------   6 brk009  staff  4096 Jan 13 12:32 .
drwx------  14 brk009  staff  4096 Jan 13 12:18 ..
drwx------   8 brk009  staff  4096 Jan 13 12:42 .git
drwx------   3 brk009  staff  4096 Jan 13 12:44 .idea
-rw-------   1 brk009  staff    18 Jan 13 12:18 README.md
drwx------   2 brk009  staff  4096 Jan 13 12:28 hw
drwx------   2 brk009  staff  4096 Jan 13 12:44 labs
```

7) Execute Jupyter Notebook from this directory by typing

```
$ jupyter-notebook
```

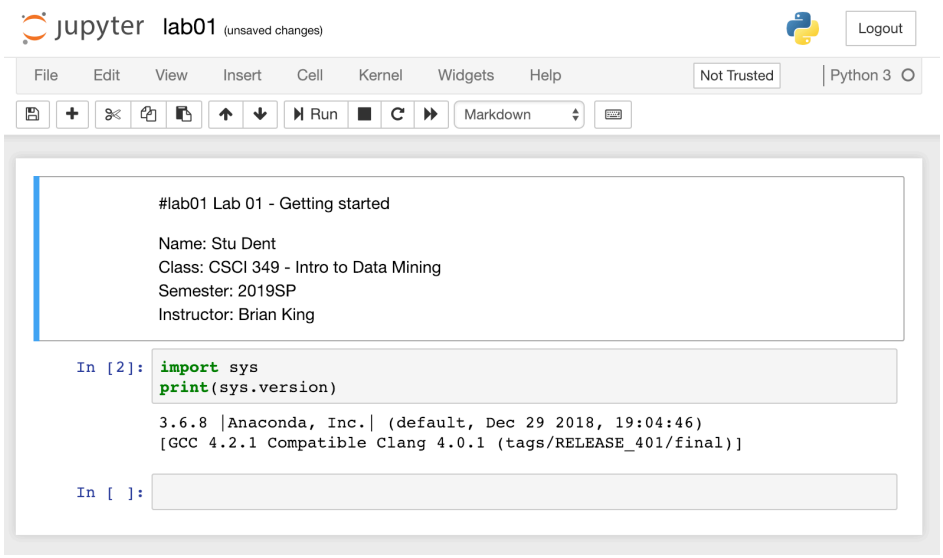
8) A browser window will appear. Check this out…

9) Change to the labs folder, and create a new Python 3 notebook file.



You can change the name from `Untitled` to `lab01`.

10) Create two new cells – a markdown cell, and a code cell. This is to confirm that you are indeed running the correct version of Python:



Replace your name accordingly. **NOTE – be sure to download a markdown cheat sheet. You'll be writing a LOT of markdown!**

When you run the code, your Python version should match the version you observed at the prompt above.

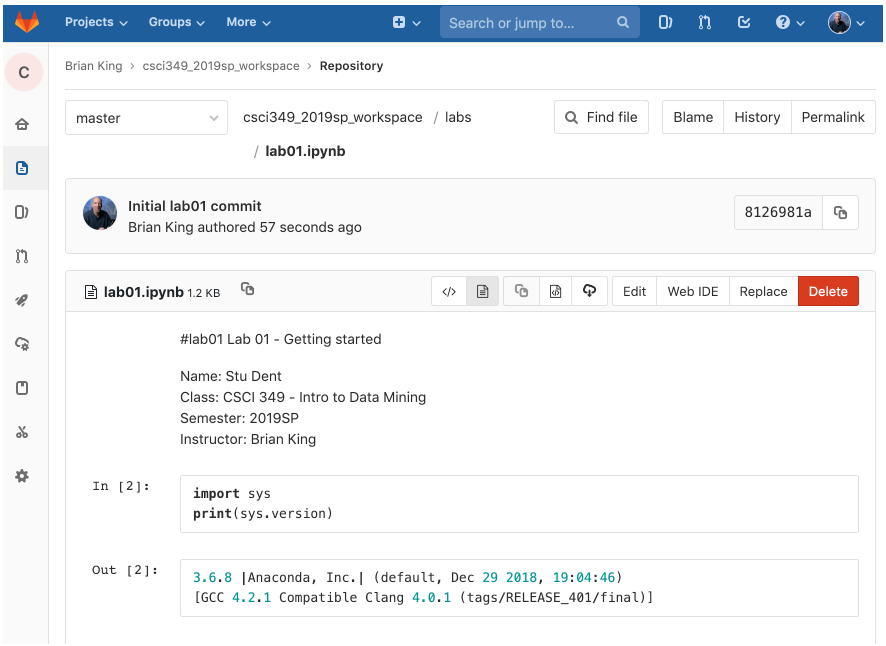
11) Go back to your terminal prompt. Commit **`lab01.ipynb`** to your local repo, then push your lab01 notebook file to your remote repository.

OK… you're almost there. Now, you need to connect this Git local repo to whatever development environment you are deciding to use! Pycharm? Atom? Spyder? Or just stick with Jupyter Notebook? I don't care what you choose. And, HOW YOU DO THIS IS UP TO YOU TO FIGURE OUT! Whatever you choose, it is imperative that you make sure that you are connected to this folder, AND that your Python interpreter in your IDE is the binary located down the your `anaconda3/envs/csci349_2019sp` folder! At this point, you likely want to use Google to search "configure conda environment with Pycharm", or atom, or Spyder, or whatever you are deciding to use.

> Alternatively, if you decide to stick with Jupyter Notebook for your main IDE, your life will be pretty simple, just not quite as fancy. You won't have any integrated Git management, but that's easy to manage from the command line.

If you made it here, woah. Take a break and pat yourself on the back. Well, wait. Before you continue sure you can do the following.

Open your browser, go to your Gitlab repo, and verify that you can view your notebook, already formatted, on Gitlab!



OK. NOW you can pat yourself on the back. You have accomplished much! Take a break!

Your First Exercise – Completing `lab01`

By now, you should be getting a sense that you'll be doing some advanced Python coding. You should recognize the urgent need to review your Python skills! Additionally, you should begin to start learning the `numpy` and `pandas` packages. Since `pandas` is built on `numpy`, the `numpy` package is a great place to start. This lab will get you going by having you answer some basic Python and `numpy` exercises.

> **Exercises in every lab and assignment will be numbered.**
>
> **Please copy the text of the question as markdown prior to giving your answer.**
>
> **[M] - Questions asking for a written answer. Place the answer in a markdown cell as normal text.**
>
> **[P] - Question asking for Python code. Your cell must be a code cell.**
>
> **Finally, you must push your resulting notebook file that has every cell's output generated.**

Before you begin, you should take the time to browse through this page – The `numpy` quickstart tutorial: https://docs.scipy.org/doc/numpy/user/quickstart.html It is dense! Don't try to read through it, just skim to get a sense of the content there.

Here are your exercises.

0) First, fill out your README.md file with some basic biographical info. Your main Gitlab page must state who you are, the course info, your skills, and a sentence or two about what you are hoping to achieve after graduation.

Now, go to your lab01 notebook file, and answer the following…

1) [M] Did you read the syllabus? All of it? Do you agree to abide by the cheating rules? Write a sentence clearly indicating your commitment to not cheating.
2) [M] What are you hoping to get out of this course?

Basic Python exercises to get you going…

3) [P] Print the Python version (available in `sys` package)
4) [P] Create a Python list of 10000 random integers in the range 1 to 100, using the `random` package. Name the list `x_list`.
5) [P] What is the minimum value of `x_list`? What is the `max` value? What is the `mode`?
6) [P] Write a function to take a list of numbers as a parameter, and return the average of the list. Then, use your function to report the average value of `x_list`, printed as a float with 2 places of precision.
7) [P] Create a list called `x_hist` that represents a *histogram*, i.e. a distribution of the numerical data, of `x_list`. Each entry in `x_hist` should contain the range of data in widths of 10. So, `x_hist[0]` represents the frequency of numbers between 1-10, `x_hist[1]` is the frequency of numbers between 11-20, and so on. Be sure to print `x_hist` at the end.

The rest of these exercises are designed to get you started with `numpy`. They are very basic. More will be completed with your next lab.

8) [M] What is `numpy`? What are its strengths? Does it have any weaknesses?
9) [P] Import the `numpy` package as `np` and print the `numpy` version
10) [M] What is the primary object type in `numpy`? Can it store data of different types? Discuss.
11) [M] Discuss the types of data available in `numpy`. You need not list every type. Just generalize, and discuss how the type system is different than the built-in types `int` and `float` in Python.
12) [P] Create a `numpy` array from `x_list`. Reassign it as `x_list`. Show the contents.
13) [P] Redo the previous exercise, but set the data type of each value to `'float32'`
14) [P] Create a length 10 integer array filled with zeros
15) [P] Create a float array of 3 rows and 4 columns, all initialized to one.
16) [P] Create an array of 20 values, evenly spaced between 1 and 3 (HINT – look at `np.linspace`)
17) [P] Set the `numpy` random seed to the value 12345, then create a 10 x 5 array of random integers on the interval [10, 20)

For some of you, this lab will be quite simple. You are comfortable with using Google to figure out what you need. You may already have used numpy. Great! Please be willing to offer some assistance to those struggling. For others, this will be a challenge. I strongly encourage you to work together with a partner to help you get your system set up and configured.

Keeping notes

You are going to come across a LOT of knowledge as you work through numerous exercises. I can't possibly give you work to test every facet of these rich libraries. So, as you read and discover, keep your notes for everything you are learning as Python notebook files! Just keep these notebook files out of lab and hw folders.

Deliverables

Commit and push `lab01.ipynb`. Be sure you have every cell run, and output generated. Verify that your file is pushed properly on Gitlab.