# Lab 01 - Getting Started

**Submission:** `lab01.ipynb`, pushed to your Git repo on `master`.
**Points**: 10
**Due**: Friday, January 17, before class

## Objectives

- Installing Anaconda on your laptop
- Setting up your local and remote Git repository
- Setting up 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 over the best language and tools to use for data science. On one side of the ring is R, and the other side is Python! (And, sitting on the sidelines saying, "Hey, what about me? I'm special too!" is Matlab.) Which do we choose? I regularly 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. In Spring 2019, I made substantial changes to the course to move all course material, labs, and assignments to from R to Python. Therefore, you will be using Python!

> If you are feeling rusty with your Python skills, then NOW is the time to brush up on your skillset. You will be doing a lot of advanced Python coding throughout the entire semester.

This is an exciting change! First, every student in CSCI 349 should have Python experience from prior courses. Therefore, unlike R, I can be sure that students will rapidly come up to speed with the language itself. Therefore, I can have students focus on learning packages that are used by the data science community in Python, not to mention focus on learning about many algorithms for mining data!

Let's get started!

## Active Learning

First, let's make sure every student has a complete understanding of how this course will work. Like CSCI 205, I employ a substantial emphasis on ***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 the material.

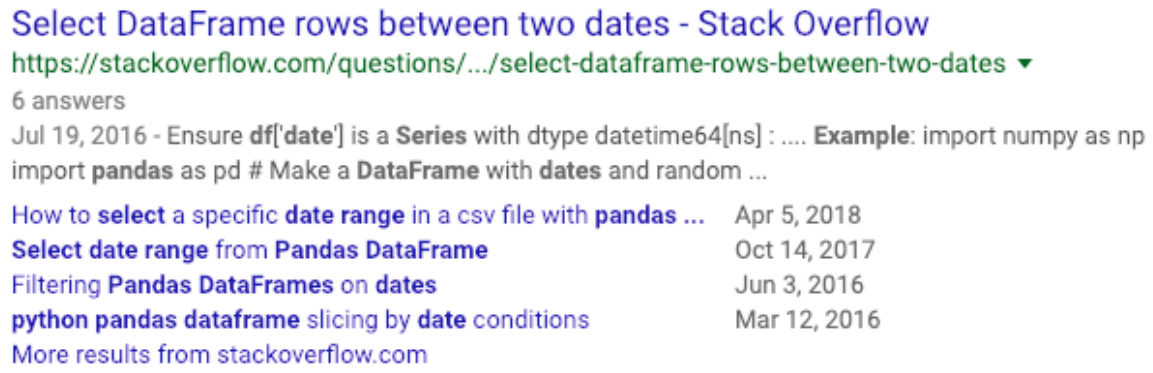
Active Learning requires you to be an *active* student, playing an active role what you learn, and how you learn it. Moreover, as an advanced elective, with respect to the tools in this course, more responsibility will be on YOU to learn them. What does this mean?

**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 installation and the integration simple.

## Preparing yourself

Before you begin the installation, take a bit of time to browse through https://docs.anaconda.com/anaconda/ and understand what you are doing. It's always better to have a plan than to blindly start an installation. **Focus only on Anaconda Distribution**. Pay close attention to the installation directions for your operating system, and then read through the **User guide**. (If you are more visual, 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, and be sure to locate a video that is within the past year at the latest.)

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. Download the Python 3 version. The link is provided in the documentation.

*NOTE: I've only verified the installation process for MacOS. However, no students had any problems in the past that they were not able to quickly resolve on their own with the help of Google. Also, on MacOS and Linux, I would recommend installing using the command line installer rather than the graphical installer, but that's my own preference. The graphical installer should be fine.*

*NOTE for Mac 10.15 (Catalina) users: As of MacOS 10.15 (Catalina), the default terminal shell is now* `zsh`*, not* `bash`*. But, if you upgraded from a previous version, it's possible you are still on* `bash`*. Check which shell you are running, and if you are now using* `zsh`*, then be sure to open your terminal run the command* **`conda init zsh`** *(This is all explained in the instructions you were to read above!)*

---

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 gained the most popularity for 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.

---

At this point, you should have Anaconda Distribution installed! You should verify your installation according to the instructions:

1) Open a terminal window

2) Execute the command `conda list` to see a list of installed packages.
3) Enter the command `conda update -n base -c defaults conda` to update the conda `base` environment.
4) Enter the command `python`. This should run a python interactive shell. To exit the shell, enter the command `quit()`.
5) Finally, you can open the Anaconda Navigator by running the command `anaconda-navigator` at the command prompt.

Proceed to the next section.

## 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/ . In particular, pay close attention to these sections:

- **Getting started with Anaconda**
  - These instructions will have you run an IDE called Spyder and enter a simple "Hello Anaconda" program. This is one of *many* IDEs available. You can skip this and run an example program in Jupyter Notebook instead. There are alternative choices for IDEs, discussed later. Moreover, you have the option to configure Jupyter or JupyterLab to be a pretty impressive editor, and thus you really don't need any IDE.
  - Take the time to create a new notebook file (which always has a file extension of `.ipynb`.) Name it whatever you want. You should get used to creating your own notebooks for to keep track of your own learning as well. In time, you should have notebook files that serve as good reference material for yourself and your future.
- **Anaconda Navigator**
  - This material will introduce you to **environments**. Environments allow you to set up distinct Python installations with a specific set of packages.
  - 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. However, never delete the `base` environment!
  - **MOST Anaconda experts will strongly advise that you never work in the `base` (root) environment. We will adhere to this. Instead, you will create a new environment to do all of your work for this course (instructions given in next section!)**
  - **You can manage the packages that are installed for each environment separately. 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.
  - (I mostly use `conda` from the command line, rather than going through the anaconda navigator interface. It's easier, and more powerful. I recommend you train yourself to do this, rather than use anaconda-navigator to configure your environment.)
  - You mostly want to pay attention to this document - https://conda.io/docs/user-guide/getting-started.html
  - Remember – feel free to create temporary environments, activate them, play around, and then delete them when you are done.
  - You will not learn everything about `conda`. Just get a sense of the main commands, how to execute commands.
  - Search for a conda cheatsheet or reference. Print it. Keep it on hand.
- **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.

**TL;DR RE: Anaconda Navigator** will be installed as part of the Anaconda distribution. This is the "go to" for the new user. It can be used to manage your entire distribution, and is particularly useful for beginners. However, I have found that I usually just jump to the command line and use the `conda` command to manage my environment. You should too!

## Set up an environment for CSCI 349

We're going to set up the conda environment for this course. Open a terminal window, then enter the following steps:

1) `conda create --name csci349 python=3.7`
2) It might take a minute or so. A minimal list of packages will be downloaded and configured. Now, activate your new environment:

   ```
   conda activate csci349
   ```

3) Use the command line to install the following packages in your new environment:

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

   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.

Wasn't that easy? Yes. Yes it was.

## Cheat Sheets

There are many, many packages we will be using throughout the semester. You will never remember all of the functions we'll be using. Thus, 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/anaconda/user-guide/cheatsheet/
- Conda - https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/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. **All work for this course will be submitted on Git**!

---

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_2020sp`**. Be sure the visibility of your project is **Private**. And select the option to 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

As I've told you, 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. (Heck, I suppose you could use IDLE… and I'd ask, "Why? Why would you subject yourself to such misery?") I've had students do all of their work within a Jupyter notebook (or the more recent JupyterLab) quite successfully. Some of the IDEs are quite good. 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 `csci349` environment! You should set up your terminal to automatically execute the command

```
$ conda activate csci349
```

every time you start 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 IntelliJ or 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! You need to be sure to set this properly in your project settings.
  - You should be able to register with Jetbrains as a student and get a license for the full professional version (as well as all of the other Jetbrains tools!) 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. Silly, but true. 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!
    - Its strength is the Hydrogen package. It's really nice for Python development. However, you might need to do some tweaking of your installation to get it up and running. I actually had to install an earlier version from the command line that is known to work:

      ```
      apm install hydrogen@2.8.0
      ```

      Try to install it through the plugin interface in atom first. If it doesn't work, then install it via command line so you can install the older version.
    - NOTE: Always be sure to run Atom from the command line. Why? So you can be sure you are running from the correct conda environment first! Activate your conda environment, then run atom!
    - Though there is no 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. I only check your notebook files!) However, be sure to open the resulting notebook file in Jupyter and run every cell to generate the output you specify!
- **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.

- **JupyterLab**
    - Even better than Jupyter Notebook is JupyterLab. The user community is increasingly pushing users to adopt this version. And, once you start playing with the plugins available, this is a hard platform to beat. There are even plugins for managing git right from JupyterLab.

Atom + Hydrogen is wonderful, but some may prefer Pycharm due to being more robust for handling *everything* you will need to do without additional packages. It's a professional grade tool for the data scientist, and one that is used in practice.

And, there are many others worth mentioning, each of which would be valid, but I did not try any of these:

- **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. (And, Microsoft has become a bigger player in data science with its acquisition of the R platform. Something to pay attention to.) It's open source, and seems pretty good.

But, when I compared this to Atom, at least my quick impression was that Atom had a stronger user base with a wider range of plugins, at least at the time of this write-up. I suspect this will change very soon, and may be one you consider. If you try it out and end up using it, please let me know about your experience.

## 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
- I recommend downloading the Slack app for your mobile device as well, but certainly not required.

## Set up your local Git workspace for the course

Let's get your local workspace on your laptop set up for the semester. We'll make sure your workspace is connected to your Git repo locally and remotely. For now, we're going to just use Jupyter to get things going.

1) Determine the location on your own computer that you want to use to keep your local Git repo located. This is where you will keep your Python code, labs, assignments, and data files you'll be using for the course. (Common locations are your home directory, your ~/Documents folder, etc. It's up to you.)

   Now, open a terminal, change your current directory to the location. Then, from that directory, use `git` to `clone` your Gitlab project you created for the course.

   For example:
   ```
   $ cd ~
   $ git clone git@gitlab.bucknell.edu:userid/csci349_2020sp.git
   ```

   (Of course, use YOUR userid!)

   This step will create a new folder in your current working directory called `csci349_2020sp`. This will represent your entire local git repo. This is the only time you need to do this for the remainder of the semester.

2) Open the terminal. Change your current working directory to the directory you plan to use for the course. Then, activate your conda environment to the one created above for the course.

   ```
   $ conda activate csci349
   ```

3) Execute the following at the prompt:

   ```
   $ python --version
   ```

   On my computer, as of this writing, I have the following response:

   **`Python 3.7.6`**

   As long as you are running 3.7, you are good!

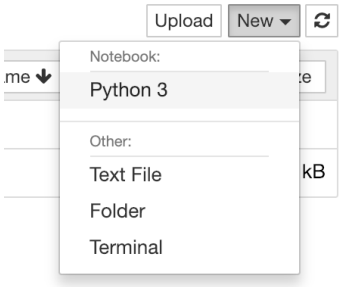
4) Create three new directories: `hw`, `labs`, and `data`.

5) Execute Jupyter Notebook from this directory:

```
$ jupyter-notebook
```

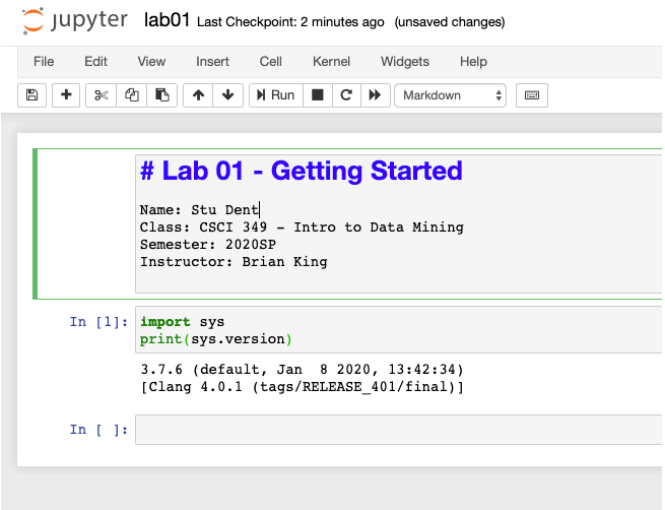
6) A browser window will appear. Check this out…


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




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

8) 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. Run the




Replace your name accordingly. **(If you are not savvy with markdown yet, work on this skill. 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.

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

---

Still not sold on Jupyter? Still considering using a full blown IDE? Pycharm? Atom? 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 configure your IDE to be connected to the correct conda environment! The Python interpreter in your IDE is the binary located down the your `anaconda3/envs/csci349` folder (unless you named your environment a different name!) 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.

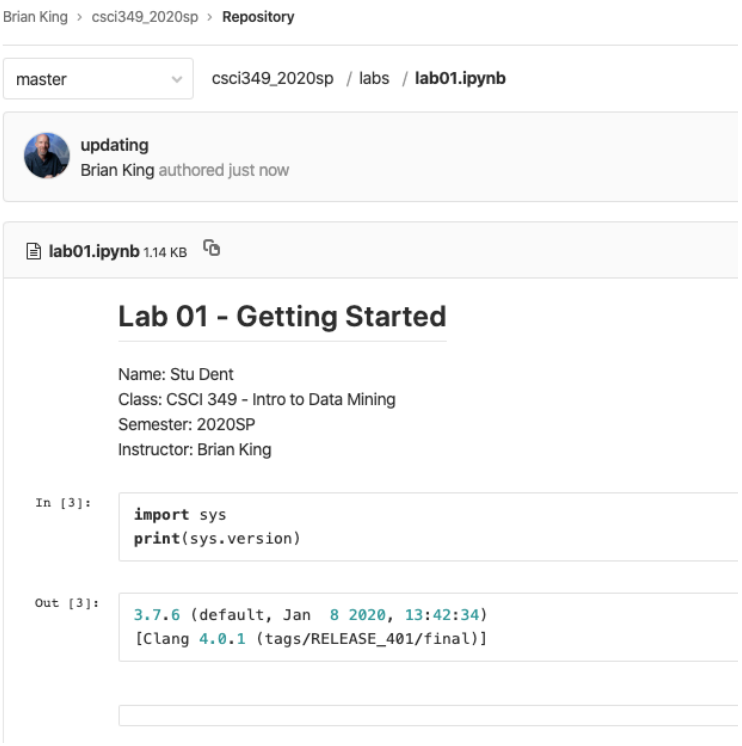
Alternatively, if you decide to stick with Jupyter Notebook for your main IDE, your life will be pretty simple, just not quite as fancy. You won't have any integrated Git management, but that's easy to manage from the command line.

OR… put on your TODO list to learn JupyterLab! It'll be worth your time.

---

If you made it here, woah. Take a break. Pat yourself on the back. Get yourself a… milk and cookies. Well, wait! Before you continue and celebrate, complete the following step:

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




OK. NOW celebrate. Important stuff, you have accomplished. Break, you may take.

## 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 a markdown cell prior to giving your answer. Then, use the following key to determine how to format 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. If the question is asking for specific output, the cell should clearly show the answer to the output of the 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.

1. 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…

2. [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.
3. [M] What are you hoping to get out of this course?

Basic Python exercises to get you going…

4. [P] Print the Python version (available in `sys` package)
5. [P] Create a Python list of 10000 random integers in the range 1 to 100, using the `random` package. Name the list `x_list`.
6. [P] What is the minimum value of `x_list`? What is the `max` value? What is the `mode`?

OK – Let's just make sure you are formatting your notebook correctly. For each question, you should have a markdown cell repeating the question, and a Python code cell showing the code that answers the question. The actual output is then shown below the code when the cell is executed in order (assuming the previous questions were answered. Your result should be something like the following:

```
#### 6. What is the minimum value of x_list? What is the max value? What is the mode?
```

```
from statistics import mode

print("The minimum value of x_list is: {}.".format(min(x_list)))
print("The maximum value of x_list is: {}.".format(max(x_list)))
print("The mode of x_list is: {}.".format(mode(x_list)))
```

```
The minimum value of x_list is: 1.
The maximum value of x_list is: 100.
The mode of x_list is: 29.
```

Finish up the rest of these questions:

7. [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.

8. [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.

9. [M] What is `numpy`? What are its strengths? Does it have any weaknesses?
10. [P] Import the `numpy` package as `np` and print the `numpy` version
11. [M] What is the primary object type in `numpy`? Can it store data of different types? Discuss.

12. [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.
13. [P] Create a `numpy` array from `x_list`. Reassign it as `x_list`. Show the contents.
14. [P] Redo the previous exercise, but set the data type of each value to `'float32'`
15. [P] Create a length 10 integer array filled with zeros
16. [P] Create a float array of 3 rows and 4 columns, all initialized to one.
17. [P] Create an array of 20 values, evenly spaced between 1 and 3 (HINT – look at `np.linspace`)
18. [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.

A WORD OF CAUTION: This is an advanced elective in Computer Science. You are going to be busy at times, but you will learn an immense amount of skills that you will be able to utilize for your future.

At this stage, we have expectations that you are comfortable with using the terminal window, using the command line, and have a solid foundation of Python. If this lab is challenging for you, and you don't believe you have the time to invest into the material to bring yourself up to speed with these basic skills, if you aren't comfortable with supplementing your own learning by diving into Google, Stack Overflow for countless other sites with great tutorials to help you, then I encourage you to consider dropping the course within this first week. Alternatively, find a dedicated partner who is willing to work together with you, who has the time to invest in helping you come up to speed. Working together with a partner can be a great experience for both people.

## 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 your 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.**