We will do each of the following steps together in class. In Labs and Homework, named files and folders indicated using a blue Courier New font: **somefile.py**. Your deliverables for this Lab are those files requested in the following steps **[L8]** and **[L14]**; upload each by submitting within Canvas to the L1 tool.

***Download starting Python code to your local PC***

**[L1]** Download the file **lab\_1\_starting.zip** from Canvas, linked within the **Lab 1 Assignment** tool. Put it somewhere on your computer where you can save and modify files; your desktop is fine. Unzip to obtain the folder **lab\_1\_starting** with its variety of **.py** (Python source) files.

***Verify Anaconda/Miniconda Python 3.7 installation***

**[L2]** Although different Python 3.7 installation packages exist, we will use the Anaconda distribution of Python, found online as: <https://docs.anaconda.com/anaconda/>

Installing Anaconda provides you with software to create and run Python 3.7 scripts. It also allows you tools to manage many different Python packages, installing and updating them so that other code can use their package contents. An example is **conda**, the Anaconda package manager for installing and managing their access by other Python code. You can run **conda** from the command-line (**Command Prompt** on Windows or **Terminal** on Mac), which we will demonstrate in **[L3]** below.

Along with providing **conda**, the full Anaconda installation includes hundreds of open-source Python packages, many of which are essential in programming for data science. However, this "full" installation is ~3GB in size and can be slow to download. A compact alternative is to install Miniconda, which is ~400MB since it omits installing non-core Python packages. If you choose Miniconda, you can later install or update any missing or incompatible packages using **conda**.

Full Anaconda is installed in all UST PCs in campus labs and classrooms; access it by logging on with your UST credentials. (Do NOT try to install Anaconda on classroom or lab computers! Only on your own PCs as described next.) You may install Anaconda or Miniconda on your own PC or laptop running Windows 10, MacOS X, or Linux. In class, you may use either the classroom PCs or else bring your own laptop with Anaconda/Miniconda installed. We will discuss how to do these installations on Windows 10 in class. Posted help videos on our Canvas site will repeat and elaborate the sequence of steps for this process as well as those on different platforms including MacOS X, as well as describe how to "uninstall" Anaconda on each platform.

Documentation for such installations is can be accessed through here: <http://docs.anaconda.com/anaconda/install/>

***Run a command-line Python interpreter***

**[L3]** The following will verify the correct installation of Miniconda with Python 3.7.

Start a command line (**Anaconda Prompt** in Windows though the standard **Command Prompt** also works; **Utilities/Terminal** in Mac). Change your working folder ("current directory") to the folder you just downloaded and unzipped: use the **cd** command. (The command prompt in Windows always shows the path of your current folder; this may be abbreviated on Mac.)

Type **conda list** to see all packages that are currently installed in the active environment.

Next, update the app by typing **conda update conda**. If it's up to date, this will be reported; otherwise, **conda** waits for you to accept the proposed updates to one or more Python packages. Accept by typing **y** or by just pressing **ENTER**.

Now type **python** to run the Python interpreter (on Mac, try **python3**). You should see the standard Python prompt **>>>** and a message about the version of Python of the interpreter, which should be 3.7. If this doesn't work ("Command not found" or a different non-3.7 version), you'll need to configure your PC to recognize the location of **python.exe**. Details in class...

Within the interpreter's **>>>** prompt, try entering some simple expression (**1 + 1**) and hit **ENTER** to evaluate it. The Python interpreter then reads the input, evaluates the entered expression, prints the result, and prints another **>>>** prompt, waiting for more   
  
  
input. Thus, the interpreter is an example of a **REPL** (Read-Evaluate-Print Loop) environment. However, this doesn't work well in Python for multiple line input expressions, as the next step will demonstrate.

**[L4]** Within the downloaded and unzipped folder, find and open the Python source file **pickteams.py** in some **text** editor (NOT MS Word). Suggestion: use the free text editor Notepad++ on Windows, since Microsoft's Notepad text editor doesn't work well with Python source files. Select the entire file's contents (the **Python source code**), then copy and paste after the **>>>** interpreter prompt, and finally hit **ENTER**. What happens? **Moral**: don't paste multiple-line Python source into the standard Python interpreter.   
  
Now quit out of the interpreter: type **quit()** or **Control-Z** (Windows) or **Control-D** (Mac). You should then return to the command line prompt.

***Use the ActiveCode browser interpreter within our HTT book***

**[L5]** Start your browser and navigate to Chapter 1 Section 13 in our book (abbreviated HTT1.13), starting here: <https://runestone.academy/runestone/static/thinkcspy/index.html> Tip: bookmark this URL of the book's **Table of Contents** within your browser. Paste the contents of the **pickteams.py** file from **[L4]** into the ActiveCode area **scratch\_01** (book's Python interpreter text area) at the end of HTT1.13. Run the program and observe the results. Don't worry about understanding the code: it includes Python elements that we'll cover later.

**[L6]** Run your pasted code in the browser using the book's CodeLens (interactive execution environment), which visually shows how the Python interpreter executes code. Details in class...

***Modify the* pickteams.py *script and submit it to Lab 1 Assignment on Canvas***

**[L7]** Now edit the program source code **pickteams.py** using a text editor (Notepad++): replace the **"a"** string in the **unchosen** list with your own name (**"Eric Level"** for example). Save and rerun as you did in **[L6]**. This Python program (script) generates and prints a set of random teams of two using the strings given the source code, including your own name. When you submit it, I'll extract each submitted name to create a version with names of every student in the class. I'll use this in the future to generate occasional random teams for in-class Lab work. We'll also use this code in the next Lab.

**[L8]** Each student should submit their final version of **pickteams.py** to the **Lab 1 Assignment** link on Canvas. Also add a submission comment that includes your name in the submission. This is the first of two .py submissions for this Lab.

***Work within PyCharm, do 2nd submission to Canvas, and play with turtle graphics***

**[L9]** If you are working on a classroom PC where (hopefully) the PyCharm Community Edition is already installed, you can skip to the next step. Otherwise, we'll discuss in class how to download and install it on your own laptop or PC. Help videos will also be available... soon.

**[L10]** Start the PyCharm Community Edition IDE (Integrated Development Environment) on your local PC or laptop. It should present you with a Welcome to PyCharm dialog. (If you don't see this, you might see the last project opened. In this case, select   
File->Close Project first.). In this welcome dialog, select the Configure->Settings popup command (Configure->Preferences on Mac). Together, we'll show how to set the default Python interpreter that PyCharm uses to run your source code within a PyCharm Project (a folder containing .**py** source and other files).

**[L11]** Return to the welcome dialog, which allows you to Create a New Project, or Open, or Check Out from Version Control. Select Open, then navigate to your **lab\_1\_starting** folder (select the folder and **NOT** anything inside it) and click on the Open button to create a new project from the existing Python sources. It should then configure your project to use the Python interpreter you set in **L10**.

**[L12]** You should see the **lab\_1\_starting** folder within the left Project area within PyCharm. Open it to view your previously-edited **pickteams.py** and other project files listed within it. Together, we will practice opening, editing, and running these Python scripts (programs) within PyCharm. A posted help video on Canvas also reviews basic skills for working with PyCharm.

**[L13]** At end of class, add comments (lines starting with **#**, ignored by the interpreter) at the top of your previously edited **pickteams.py** file. In your comments, give your name, favorite food, and favorite local restaurant. Be sure to run your script again to check that it's correct. Then close your project (File->Close) and exit PyCharm by closing the welcome dialog (alternatively on Mac via PyCharm->Exit PyCharm).

**[L14]** Find the **lab\_1\_starting** folder and submit your modified **pickteams.py** within it to the **Lab 1 Assignment** link on Canvas; this is a resubmission of the same file as earlier. Also upload in the same submission any one of the other **.py** files (the book's HTT2 code examples) found within the same project folder.

Canvas allows you multiple submissions for any assignment, though you can't delete earlier submissions. However, **I will only grade the files in your final submission** for any Lab or Homework assignment. Thus, you can resubmit updates to a previously-submitted file, as well as submit additional files - but **your final submission must include ALL the requested files for the entire assignment**. Even those you submitted previously: Canvas only allows me to download the final submission files for all students.

**[L15]** (**optional**) From the command line, enter **python -m turtledemo**. Try running the different examples via the Examples menu. This illustrates Python turtle graphics: more on this next week...