Lab 1: Python a different way

Objectives

- Get acquainted with the PyCharm development environment.
- To re-acquaint yourself with programming in Python

Preliminary Setup

1. Download and install PyCharm Community Edition.
2. Make a folder on your computer where you will work on both projects and labs for this course. We'll call this folder your **workspace**. Each of the projects and labs will be **folders inside the workspace folder**.

Integrated Development Environments

In previous classes where you've used Python, you used a program called Idle to edit and run your programs. However, Python **does not equal** Idle. Python is a *language* in which you can write *programs*, but the tool you use to edit those programs is simply a *text editor*. Any text editor will do. [Note: Similarly, English is not Microsoft Word].

Once your programs are in the right language (in this case Python), they can be *compiled* or *interpreted* by appropriate tools (in this case, the Python interpreter). When you used Idle before, it provided a window for text file editing (where you could type your programs) and a way to run those programs. To run the programs, Idle would call the Python interpreter to do the job. Because editing and running are *integrated* together in Idle, we call Idle an **Integrated Development Environment (IDE)**.

Another IDE, one used by professional developers, is called **PyCharm**. PyCharm is made by the JetBrains company, which also makes other IDEs, including ones designed for Java, C++, Ruby, etc.... In this course, you will use PyCharm instead of Idle. But remember, the language you are using is still the same old Python you are familiar with.

Getting Started with PyCharm

Launch PyCharm. You'll be presented with options to Create a New Project or Open an existing one. Choose the Create New Project option.

Note: before you click Create, make sure the project is setup the way we want:

1. Its location should be inside the workspace folder you created.
2. It should be using a system Python interpreter (**not a virtual environment**).
3. It should be using a version of Python 3 (preferably > 3.7)

4. Then change the location of the project by typing in the Location box.
5. Do not use the New Environment using virtual environment.
6. We can fix this by selecting Existing Interpreter and choosing the Python 3 interpreter we have installed. If there are no Python Interpreters to use, you might need to install one. See python.org

Add some Starter Code

Copy/paste and add **tictactoc.py** to your project folder.

```
boardfile = open("tictactoe_board.txt")
board = boardfile.read()
boardfile.close()

rows = board.split('\n')

print(rows)
```

Have a look around

Now that you have a project and some code, explore to see what you have:

1. Read the code. Can you figure out what each line is doing? If there are functions you don't know, look up the at the documentation official python documentation site.
2. Look at your files and folders. Notice that your project is just a folder (named the same as your project) that contains the .py files you've added to it.

Run the code. The first time you run the code, you need to tell PyCharm what you want to run: choose "Run..." from the "Run" menu, and select **tictactoe.py**. Afterwards, you can re-run by clicking the green play button in the toolbar, or by pressing control-R. Notice that when the program runs, it will probably complain that **tictactoe_board.txt** doesn't exist; so ask PyCharm to make a new file called **tictactoe_board.txt**. Fill it with some text like:

```
XXX
OOX
OOX
```

Now that you've run the code successfully, take a look at the output. Based on what you see in the output, what kind of variable is 'rows' in the code?

Add some more code

For the following, make each of these additions separately, one at a time. After each one, test that your program works (this might involve making new board files to test with). *Do not try to jump in and make all of these changes at once.* Complete the lab without defining functions. Write simple comments for each part.

1. Add to or modify the code I've given you so that it prints an error message if the number of rows is not exactly 3.
2. Make it so that an error message is printed if the number of columns is not exactly 3. (i.e. if any row has fewer or more than 3 characters).
3. Make it so that a message is printed if any of the rows is three-in-a-row (either X or O).
4. Make it so that a message is printed if there is three in a row, horizontally, vertically, or diagonally.

5. For each step above use a different file that can test one at a time.

How to turn in this lab

Before turning in any program in this class, remember this mantra:

Just because it works does not mean it's good.

Your grade will also come from the following aspects of your code:

- Submission
- Accuracy/correctness
- Readability
- Neatness
- Presentation
- Style
- Testing
- Commenting

For all labs, turn in only an **electronic** version.

Please submit the followings after all labs:

- zip file of your project (the project folder, not just the .py file(s) or .java file(s)): zip file name will be your YourFirstNameLastName.zip
- a single pdf file of all your codes (.py or .java), all input files, screenshots of your output including all normal/extreme cases: pdf file name will be YourFirstNameLastName.pdf

Submit

- **the zip and pdf files**
- after Monday Lab session until 8 PM EST
- from the Nexus Lab submission link that will be accessible in Week 2-10

Ask for help if you are having problems!