Try a Nybble of Python

Teacher Handbook

Information for Class Registration

Make the requirements and recommendations for class clear in the class description including how homework will be submitted. Provide the course requirements enough in advance for the students to get their book and set up their computer. Make sure the students successfully install Python 3.7 or higher on their computers *before* the first day of class. I have a link to the video to get help downloading Python here. (Do not install Python for them. In fact, I recommend not touching student computers ever.) Also, remind students to wear their glasses or contact lenses if they need them. Their vision will have to adjust to the book, their laptop screen, the large screen, and the whiteboard.

Requirements I recommend for class:

- Laptop running Windows Operating System
- Internet Access
- Email address (if a minor, one with parental oversight)
- Download of Python 3.7 or higher
- Book: 2nd Edition of **Try a Nybble of Python**
- Prerequisite: Successful completion of High School Algebra 1
- Prerequisite: Adequate typing skills to keep up with the class

[Note: If your students submit files to you via email, have them save their Python files as text files with the .txt extension before submitting them as attachments over email. Python files are executable files. These files may get blocked or set off a security risk warning to the receiver of the email. If you decide to run the code, open the text file in IDLE and save it as a Python file first.]

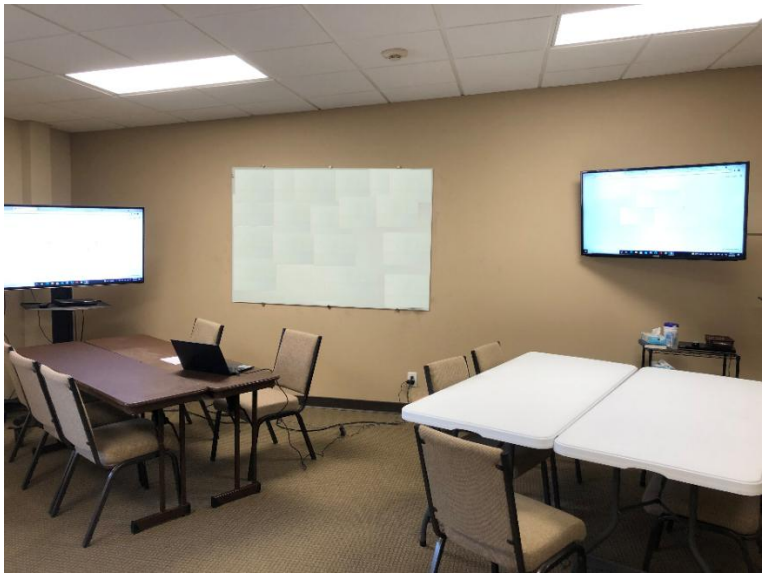
Preparing for Class Before Arriving

Before arriving on site, rehearse the material before teaching it. Read the text and type in all the code you are going to cover for the upcoming class. In the book, code following a disk icon 🖫 is meant to be saved as a Python file. Go ahead and save those files in a folder you create for your files for class. How to do this is covered in Chapter 1.

Preparing the Classroom

Arrive at your classroom at least 20 minutes early as setting up takes a little time. For the workstation(s), arrange tables such that five to six students can sit together in front of a large screen. Remember, you will need a seat, too.

Whether by cable, projector, or wireless, connect to the classroom screen. If you need to set the classroom up for more students, then you may need additional screens or equipment (e.g. an HDMI splitter for additional screens).



Example of classroom set up for 10 students.

To cut down on your typing during class, open the Python files you rehearsed and saved before class. Arrange the file windows in the sequence to be presented. Also, a whiteboard is handy to list the concepts being learned for the lesson, illustrations, expectations, and homework.

During Class

When the students arrive, make sure no food or drink is allowed on the tables. If you are nice, bring a power strip with a surge protector. (Inevitably, a student will need to charge his computer during class.) Warn students that they plug in and/or access local internet services at their own risk.

After the students are set up, discuss the concepts to be covered and explain any example(s). For code that follows the shell icon , live code along with them. Chapter 1 provides information about coding in the IDLE Shell. If the code is in a saved Python file, then display the Python file window on the large screen. (You will need to open and run your Python files through IDLE. Merely double-clicking on them in your folder will execute them in a running window which often disappears quickly.) Make sure they create a separate folder for their class Python files… they should not use the same folder that holds the Python download. They will need to be reminded in each class to save files in the proper folder.

Save your speaking for when students are not typing in the code. They will not be able to simultaneously concentrate on what you say while typing in code correctly. When you speak, make sure you speak loudly enough to be heard. If there is more than one workstation set up, position yourself such that you can face and address all the students clearly.

The Compute sections in the book are designed to be worked by the students in class. Let the students have plenty of time to solve them. The answers are at the end of each chapter.

Encourage students to work together when errors occur. You do not need to be the only one helping to debug student code. Debugging in community is a great skill to learn!

Have fun! Be a little silly and encourage tinkering with the code. The fun lessons will be remembered better than boring lessons.

Finishing Class

Finish class early enough such that the students have time to carefully pack up their computers, cords, etc. Be clear about the assignments required by the next class. Have students help you rearrange the classroom back to what it was before you arranged it for class.

Chapter Practice Solutions

There is a link to the Chapter Practice Solutions here. Bear in mind that in programming, like math, there is often more than one method to achieve the correct answer.

Decisions to be Made

Typically, my class meets once a week for sixteen weeks. We proceed at a chapter a week working through an entire chapter in class. The Chapter Practice sections are assigned for homework. Grades are determined by class participation and completion of Chapter Practice projects.

From time to time, if time allows, consider throwing in a relevant computer science topic for a Socratic class discussion. (A question is posed to the students. The instructor speaks less and moderates the conversation by asking steering questions.) I have moderated a variety of discussions on moral conundrums or current events involving artificial intelligence (AI). For example, my classes have accessed the situational Moral Machine publicly available from the MIT website and discussed the trolley problem. Isaac Asimov's *three laws for robotics* make for an interesting class discussion and on occasion I have assigned Asimov's short story *Evidence* as an outside reading. Or, administering a Turing Test to a free, publicly available AI never fails to spawn a funny conversation.

Other activities I have included are base conversions (converting decimal numbers back and forth to binary and binary to hexadecimal numbers) and physically wiring "and" and "or" logic gates and LEDs for students to physically see the results of Boolean logic.

Other instructors might opt to include tests, lessons on the history of computing, a demonstration of a Raspberry Pi project using Python code, etc.

Teach a few classes before determining how many extra topics to include, if at all. Consider the amount of time you have in a class and the number of times you meet, but do not skimp on the programming experience as that is the core goal of the class. The presentation of programming topics and project experiences offered by **Try a Nybble of Python** provide a solid foundation to beginning programming.