

Lesson 2: “Welcome to Python”

OVERVIEW

1. Icebreaker: How to Make an Algorithm Sandwich
2. Introduction to Python and Dr. Python
3. Type in and run a simple program
4. Make changes to the simple program

CLASSROOM PREPARATION & MATERIALS

- Props for Algorithm Sandwich icebreaker: bread, peanut butter and a knife.
- Code of conduct from first class
- Whiteboard and markers

TRAINER TECHNICAL PREPARATION

- Read the Dr. Python help files to familiarize yourself with the interface and the first 3 pages of this article: www.pygame.org/docs/tut/intro/intro.html
- Type in the sample program and run it to make sure your program works and that you know how to fix any simple errors before class. Make sure that the graphics files are in the same folder that you save your “Don’t Mess with Texas” programs in.
- Test all technical equipment. All computer must be able to run Dr. Python.
- Check that students have access to the image “trash.png”
- If the students have problems they can’t fix and the program will not run, check the error messages that come up when you try to run the program. They will lead you to the exact line number that is not readable. Retype the lines that are giving trouble and if all else fails, retype the whole script. Be careful with blank space and indents. Their programs should look exactly like the sample one on the overhead with tabs on the right lines and colons where they need to be.
- All the code and graphics will be in the “Don’t Mess with Texas” folder. Each students needs to put the graphics into the same place as their “Don’t Mess with Texas” program is stored.

KEY CONCEPTS

- Python is a programming language, like Spanish is a language. It has very specific rules that you must follow, just like when you are learning to speak Spanish, you have to put the nouns and verbs in a specific order to make the sentence understandable.
- The Dr. Python interface is a development environment, just like Scratch, that we will use this semester to program our game in Python. It is essentially a text editor (like Notepad) with built-in programming tools, such as automatic syntax checking (similar to Grammar checking in Word). However, unlike Grammar checking in Word, the syntax checker won’t give you suggestions on how to fix a mistake. It will only tell you what line is wrong!

SUGGESTED AGENDA (90 minutes)

- 10 minutes: Icebreaker: How to Make an Algorithm Sandwich
- 15 minutes: Intro to Python
- 15 minutes: Intro to Dr. Python
- 5 minutes: Break
- 10 minutes: Discuss code sheet for DMWT-Lesson2
- 30 minutes: Enter script and experiment with making changes
- 5 minutes: Closing discussion

ACTIVITIES:

Activity #1: How to Make an Algorithm Sandwich

Time: 10 minutes Materials: bread, knife, peanut butter

Programming is telling a computer what to do. A computer will do EXACTLY what you tell it, so you must give extremely clear instructions and be careful not to leave out steps in the process. Defining how you want the computer to do something is called an algorithm. Let's create an algorithm for how to make a peanut butter sandwich:

- Have the group write a list of steps (i.e. an algorithm) on the board for how to make a PB sandwich. Assume you are given a jar of peanut butter, a loaf of bread, and a knife on a table.
- Have a volunteer (as the "computer") follow the instructions EXACTLY as they are written. Instruct the volunteer to do literally everything in the instructions and absolutely nothing that is not on the list. This should be fairly comical! For example, if the instructions haven't told her to open the jar of peanut butter before getting peanut butter on the knife, the "computer" could just stab at the jar. Have the class edit their algorithm until the "computer" is able to successfully make a sandwich.
- Ask students to explain the phrase: "Most people think computers are really smart, when quite the opposite is true. Humans have to 'dumb themselves down' in order to speak a language the computer can understand."

Activity #2: Introduction to Python

Time: 15 minutes

Talk briefly about how this semester will focus on programming in Python. Python is simply one of the languages that can tell a computer what to do (just like English is simply one of the languages that people can use to communicate with each other).

To use Python, you have to know its syntax.

- Definition of syntax: the rules governing the formation of statements in a language (including spoken languages like English and programming languages like Python).

- These syntax rules govern any language like English, Spanish or German. If you don't follow the syntax rules exactly right and get the nouns and the verbs in the wrong order, other people might not know what you are saying. This is the same thing with programming

Example of statements formed using the syntax of the language:

- In English: Computer, display the word "Hello" on the screen.
- In Bad English (i.e. bad syntax): Computer, display word "Hello" on screen.
- In Python: print "Hello"

The Power of Python: Python is used in all kinds of companies around the world. IT girls are learning a programming language that can be used to build any kind of software.

Examples of companies using Python:

- You Tube
- BitTorrent
- NASA to design space shuttle missions
- AstraZeneca to discover new drugs
- Nokia Series 60 phones run Python
- Industrial Light and Magic (visual effects company for Star Wars)
"Python plays a key role in our production pipeline. Without it a project the size of Star Wars: Episode II would have been very difficult to pull off."
- Google
"Python has been an important part of Google since the beginning, and remains so as the system grows and evolves. Today dozens of Google engineers use Python, and we're looking for more people with skills in this language." said Peter Norvig, Director of Search Quality

Activity #3: Introduction to Dr. Python

Time: 15 minutes

Using the projector, open Dr. Python and run the script for DMWT-Lesson2. Ask students about the similarities between what they see in the game window and what they saw last week when using Scratch. It may be helpful to open Scratch and run the Trash pick-up game script again to refresh the students' memories.

Have each girl open Dr. Python on her computer discuss the various parts of the screen.

Dr. Python is a Development Environment just like Scratch. It is essentially a text editor (like Notepad) with built-in programming tools, such as automatic syntax checking (similar to Grammar checking in Word).

1. **To start a new program, click on File and New.** This will create a document called Untitled 1. Students should start on line 1, and type: Hello, Dr. Python.

2. **Save the document by clicking on File, Save As.** In the pop-up window, click on the House icon. Then click on the Desktop folder. Each student should have a folder on the desktop that is labeled DMWT. Open the folder and save the document as Dr. Python Intro. At this time, an error message should appear, telling the students that there is invalid syntax in line 1. We will discuss proper syntax later in the lesson.
3. **To check for syntax errors and typos, click on the “Check Syntax” button.** It has two small green arrows and is on the toolbar above the area where you wrote your code. Read the syntax error box that comes up. It will point you to a line number where you messed up and usually tell you how to fix it. As we write our programs we will *Check Syntax after entering every 3-4 lines of code!*
4. **To run a program, click the Run button (the blue arrow next to the Check Syntax.)** If there are errors running the program, a box will pop up and will tell you which line number has the problem and what you need to fix.
5. **The column on the left, “Source Browser” gives you an outline of your program.** This is automatically populated with the programs used to execute the code you have written. Don't worry about this window for now; it's only helpful for very large files.

Let's try our language statements from the Introduction to Python:

- English Version

Delete the code “Hello Dr. Python.”

On Line 1, type in: **Computer, display the word “Hello” on the screen.** Try running it. You should get a syntax error because we're not using the Python syntax; we're using the English syntax. Again, this is like Grammar Check in Word. A prompt box at the bottom of the screen should also appear telling you what file you ran and what the problem was. What line does it tell you the syntax error is on?

- Python Version

Type in: **print “Hello”** Try running it. What happens? Hello should appear in the bottom box. Congratulations! You ran your first Python program! *So, what exactly happened?* The computer understands the command “print.” Anytime you want the computer to write/display a word, tell it to print. However, the computer can only print what it knows and recognizes—by putting quotation marks around “Hello,” you are telling the computer that “Hello” is a word. Once the computer understands “Hello” is a word, it can print the word. Notice, “Hello” is highlighted in Green, since it is recognized.

Have students minimize Dr. Python and open Scratch. What are the similarities and differences between the two programs?

~5 Minute BREAK~

Activity #4: Discuss Python script for "Don't Mess with Texas" game

Time: 10 minutes

Handout a copy of the script for DMWT-Lesson2 for each student. Briefly discuss how the code is structured and the use of comments lines.

Activity #5: Enter script into Dr. Python and check syntax

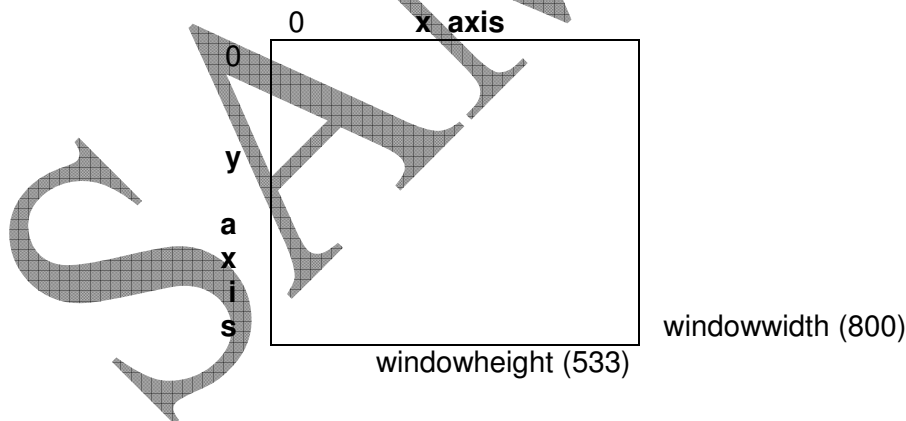
Time: 30 minutes

The class will now write the beginning version of the "Don't Mess with Texas" game. The end-product will be the "trash bounce" that was shown earlier in class.

1. Begin writing the code: Instructor should enter code line-by-line (shown on the projector) and discuss each line. What does the line of code do? What are some of the keywords? Discuss the definition and use of import, init() and flip()

2. Add a comment for each line of code: After entering each line of code, write a brief comment. Explain to students the procedure for adding a comment (using the # symbol) and the reason for writing comments (so you can remember what the code does, and so others can understand your work.) Have students write comments in their own words to accompany their code. Some students may struggle with putting things in their own terms; let them know they can always go back later to enter their comments. Instruct students not to take too much time writing comments.

When commenting on how to create the window size, draw this graphic and briefly explain the coordinate system in programming Python. In this lesson, we define the windowwidth as 800 and the windowheight as 533.



3. Emphasize the importance of indentation: Indentation creates a logical organization that helps the computer execute the code. Every time there is an → in the code handout, it indicates an indentation on that line of code. 2→ means two indents, 3→ means three indents. Students DO NOT write the arrow and the number (3→) into the actual code; it is merely a prompt to indent.

4. Have students save their games in the directory: students must save their programs before they can check syntax or run the program. This version of DMWT should be called DMWT-Lesson 2. They should save it in the same DMWT folder on the desktop that they used earlier for the Dr. Python Intro. code.

5. Check Syntax: If errors occur, students check the lines referenced in the error messages.

6. Run the program: press Run button. Even if there are no syntax errors, there may still be problems in the code which will cause the program not to run correctly. Have students go through the code meticulously, checking for misspellings, indentation mistakes, punctuation errors, etc. Have students work in pairs to check each others code if problems persist.

Activity #6: Explore the DMWT folder and experiment with making changes

Time: As allows

Have the students minimize the Dr. Python interface. Then have them open the DMWT folder on the desktop. The folder should contain 4 folders: fonts, trash-images, wildlife-images and sounds. There will be some python programs including dmwtmedia.py There should also be an image: “background.png” Have students click on the image to view it. Discuss the size of the image (800 x 533) and how it matches exactly the size of the window that we created in the DMWT-Lesson2 script. If students would like to find another image and use it in the code they must follow a few basic steps:

1. Find an image that is compatible with the window size (or else change the image size or window size.) The image must be a .gif, .png or .jpg file in order to work with Dr. Python.
2. Save the image in the DMWT folder. Once it has been saved, they should see it in the folder along with background.png. They should not delete background.png
3. Find the line that loads the background image, and replace “background.png” with the title of the new image. They must write the image name in quotations and include it in the parentheses already in the code.

Other possible changes to the DMWT code:

- the speed of the trash
- the size of the window
- any numbers in the code

After making every change, students should check syntax, save and run the code.

Activity #7: Closing Discussion

Time: 5 minutes

- What happened to their programs when they made changes?
- Did errors occur?
- What changes were they able to make?
- What are the differences and similarities between Scratch and Dr. Python?

SUPPLEMENT MATERIALS

- Access to Scratch and the Trash Bounce project from Lesson 1
- DMWT folder on each student desktop
- Access to Google Images or other graphics resources for finding images
- DMWT-Lesson2 code sheets

REFERENCES

- Python/Pygame Introduction: www.pygame.org/docs/tut/intro/intro.html
- Python Quotes: <http://python.org/Quotes.html>
- Python Success Stories: <http://www.python.org/about/success/>
- Python and Nokia phones: <http://www.forum.nokia.com/python>

SAMPLE