Chris Piech, Mehran Sahami and Julie Zelenski Section #2

Code in Place April 25, 2021

Preparing for Section 2

This is going to be the first week of coding *Python* for your students. What a beautiful thing :-).

## **Two simple problems, with some teaching**

[Here](https://edstem.org/us/courses/10000/lessons/13333/) is the section handout the students will see, with solutions! There are two problems -- both of which are quite short. Interestingly, the second problem will use a concept they have not seen previously: comparing variables with ==. It is up to you to teach this idea.

## **Learning Objectives**

Students are learning about control flow and the very basics of coding. Specifically:

1. Variable types
2. The difference between assigning variables and using variables
3. How to “cast” between types
4. Using == in if statements

## **What can I assume students already know?**

1. **Intro to Variables (Monday Lecture) [**[**slides**](https://drive.google.com/file/d/1DO0tc4eKN83xwvcEZPLBdZ4AykXc2nB3/view)**]**

Uses the analogy that a variable is like a piece of **luggage**, and the name is a tag.

1. **Expressions (Wednesday Lecture) [**[**slides**](https://drive.google.com/file/d/1q35uOxhQvMjeqnTuxJGQ9vlUVe4GzRO3/view?usp=sharing)**]**

How to do computations with variables. Covers random numbers.

They **do not** know how to use if statements or while loops in Python: they will see them for the first time in your section! On Friday, they will get the content in lecture. As always, assume that students have not read the section handout, and that they have very little understanding of the material from Wednesday’s lecture.

## **Getting ready for section**

Make sure to send an email reminder about the section to your students 24 hours before your section. Make an announcement on Ed and tick the ‘Send Email’ checkbox!

Make sure to practice the problems early!

## 

## **Recommended Section Plan**

### ***Social Time [5 minutes]***

While students are joining the meeting, take a few moments to catch up. In five minutes you probably can’t hear from each person one by one. A few options are:

1. Have everyone answer the same question in chat. If you could have Karel know a fifth command, what would it be?

2. Take 5 minutes to tell your section a story. Could be related to programming, could be just something about you.

### ***Brief conceptual review [2-3 minutes]***

This week’s section focuses on variables. Take a moment to state the things that they should know about variables. If you make slides for conceptual review, resist the urge to spend too much time “lecturing” mode. Rather, focus on priming students for the problems to follow.

### ***Teaching Mars Weight [5-10 minutes]***

You can choose how to structure this time. One good option is to present the problem, have students spend 2 mins in breakouts and then you solve the question, explaining as you go. After you answer questions that came up.

***Teaching 8-Ball [20 minutes]***

1. Explain the problem (1 min).
2. Review random numbers (1 - 2 mins).
3. Teach them if statements and the == condition. Tell them that they will learn more conditions on Friday. Show a contrasting case between “=” and “==” (5 mins)
4. Set up milestone #1 together by asking the user for a question and generating a random number.
5. Split students into 5 groups and have them each add one **if** statement. Let them be creative with what they print.
   1. **Warning:** Don’t use elif at first or random.choice. Just a normal if statement repeated should be fine. Unless someone asks you about those, wait until you have finished the first milestone.
6. Milestone #2: with any remaining time, add in the while loop. Recall that this will be their first time seeing a while loop in python. Start by just putting your code from milestone #1 in a while True loop. Then if you have extra time consider making it stop when the user enters the empty string. Try not to use break or return to solve the fence-post problem. If you want, introduce them to elif.

## **Teaching Tips**

* Post to your “Ed” group early and often!
* You can pseudocode in whatever way makes sense to you as long as the students are the ones coming up with the steps! One format we recommend is writing your pseudocode as comments in the code.
* Avoid using terms like:
  + “This is simple/easy.”
  + “You should already know how to do this.”
* Let students make mistakes and roll with them! You can turn these bugs into learning moments by going back and saying, “What might have gone wrong?” or  
  “Where do you think the bug might be?”
* Make sure to run the code frequently to see if things are working as expected. This instills good testing habits for students.
* Value different solutions - encourage students to think about other ways to solve the problem if you finish one solution early, and discuss the potential pros and cons of each!
* When planning out programs, Chris and Mehran emphasize thinking about the pre- and post-conditions for programs. If you would like to talk about these terms while pseudocoding, you can ask your students what the pre- and post-conditions of the overall program would be for these problems (or for particular milestones within the problem).

## **Code in Place conventions to keep in mind**

* All code must be written inside a **main()** function that gets called under:  
   **if \_\_name\_\_ == “\_\_main\_\_”:**   
  We want to avoid having any code (besides constants and imports) outside functions.
* We don’t explain what the if statement above means until later in the course. If students ask about it, just mention that it’s necessary for the computer to run the program, but they don’t need to worry about it for now!
* In Code in Place, we make a clear distinction about “functionality” and “style.” Style has to do specifically with code readability and reusability. In the class, there are conventions that we emphasize and that we do not want you to deviate from:
  + We follow Python’s [PEP 8](https://www.python.org/dev/peps/pep-0008/) style. This means that variables and function names use “snake case” (all lowercase letters with underscores for spaces). Constants (which will be taught later) will be in all capital letters. **Please do not use camel case (e.g. myVar) or any other type of naming conventions.** If you’re not familiar with Python or PEP 8, take some time to skim the online guidelines.

Your function and variable names should be **descriptive but concise**. Avoid names like **var1**, **var2**, **foo()**, **bar()**, and **function\_that\_has\_very\_long\_name()**