Adventure Lab

Instructor Guide

Overview

This lab is designed to be an intro to computer programming (it will likely be students' first exposure to writing code.) In this lab they will learn about how code can manipulate numbers as in algebra.

Learning Goals

- print() with no input makes an empty line
- input() asks for the player to put in text then press enter, with the prompt text as a parameter
- input() can also be used to pause the text
- Variables can hold values (such as the output of an input() call)
- Functions have multiple lines of code that work for one goal
- Functions can have parameters (like print and input) or not (like the rooms)
- == checks if two things are equal
- Expressions are chunks of code that evaluate to some value
- Conditionals are expressions that can be true or false
- If-elif-else statements can be combined with logic to a series of different results
- (optional) Return stops the function, or getting to the end of the function

Have comments, concerns, or feedback? We love to hear about ways to improve our curriculum! Fill out our Feedback Form or email teknow@andrew.cmu.edu, and we will get back to you as soon as possible.

Personal Growth Goals

- Programming is fun! ("I can make my own games")
- Design Creativity: students will get to design their own games, and be able to answer questions like "how do I want to make my game," "why do I want to make it this way," "how will I implement it," etc.

Skills Required

- Introductory knowledge of print(), input(), functions, loops, and conditionals (all this should be covered by completion of Algepocalypse Lab)

Resources Required

- Computers for either every student or every pair of students
- Python 3 needs to be installed on all the computers
- One mentor per student or per 2 students

Instructor Preparation

1. Make sure all the computers students will use have Python and IDLE installed (check to see that students have a way to save/access files)
2. Load all the programming files onto each computer.

In Depth Description

This lab is very open ended and involves the students creating their own adventures, based off of a sample adventure. They will use basic functions, if-else statements, input(), and print(). To do more interesting things they can use mathematical operations, loops, random numbers, etc. The goal is for this lab is to show the creative aspect of code and get them interested and wanting to learn more so they could add more features to their games.

MOST of the time should be given for them to play with the code and write their own adventure, with the help of their mentors. Either each student can make their own, or students can work together, depending on their preferences. Make sure there is time at the end for people to play each others' games.

NOTE: each phase will make much more sense if you look at the code first.

Have comments, concerns, or feedback? We love to hear about ways to improve our curriculum! Fill out our Feedback Form or email teknow@andrew.cmu.edu, and we will get back to you as soon as possible.

Phase 1: Warmup/Review

We'll be using a lot of print() and input() today, which we learned about last week. Who remembers what print() does? …
Cool, how about input() - what does that do? …
Yeah, nice!
And what is a variable?
(a label for something that holds a programming value, like an integer or string)
Yeah!
All three of these things we'll be using a lot today.

We'll also be learning about functions, which are the key to doing a lot more complicated things with code.

But first, for a warmup, make a python file named 'fave_color.py'.
In this file, write a short program (mine is three lines) that asks you "What is your favorite color?" then lets you type in something ("blue" for example), and prints back "Your favorite color is blue".
Here's an example (write on board an example or run the solution file without showing code).

… (let them write)
(write solution quickly together after they have written it)

Phase 2: Introducing Functions

1. Code with them basicMathGame.py as you explain to them functions
    a. Explain if-else (was only briefly touched on last week)!
    b. Explain functions as a black-box that may take an input, always performs some operation, and then optionally returns something. Use input and print as concrete examples of functions.
    c. Explain how to define your own function, and the associated indentation
    d. Explain that some functions (i.e. print, input) take in some information, but others (i.e. playMathGame) don't.
    e. Explain how to call a function, connect this to print and input (parenthesis)
    f. Describe the difference between definition and execution, (i.e. playMathGame is defined on line 2-10, but the code is not run until after we call it.) Explain that no further code is executed until the function runs to completion. Explain how indentation plays a role.
2. Let students modify basicMathGame.py. Suggested modifications:
    a. Change the name of the function (again if they don't change it everywhere, explain the error)
    b. Change what you do before and after the function "playMathGame" is called in order to understand order of execution.

Have comments, concerns, or feedback? We love to hear about ways to improve our curriculum! Fill out our Feedback Form or email teknow@andrew.cmu.edu, and we will get back to you as soon as possible.

c. OPTIONAL: Change the whole file as desired

Phase 3: Basic Game Outline

1. Show and Run mathGame.py
    a. Explain what each function does
    b. Explain how inputs are used to pause the text
2. Let students modify mathGame.py. Suggested modifications:
    a. Change the answer to the math question
    b. OPTIONAL: Change the whole file as desired

(take a break?)

Phase 4: Complex Game

1. Show and Run adventure.py
    a. Point out how rooms can go back to each other like library and startRoom
        i. The power of functions!
    b. Note the loop in slide()
        i. Simple to code, longer to play..!
    c. Any questions about how this game works? You will make your own very soon.

Make Your Own Adventure!

1. Now, based on these sample adventures, try making your own adventure!
2. There is a template in adventureStart.py

Play Each Others' Games!

1. Make sure to leave 10-15 minutes at the end for people to try each others' games.
2. They can take their file home (or email it to themselves) and keep making it a longer game - encourage that.

Lesson Plan

*(:10) means that this part should be done by the tenth minute of the lesson)

1. Setup (:15)
    a. Divide into groups
    b. Get ready to code (open IDLE all the "adventure" Python files.)
2. Phase 1: Warmup (:30)
3. Phase 2: Introducing Functions (:45)
4. Phase 3: Basic Game Outline (:55)
5. Phase 4: Complex Game (1:05)
6. Make your own game! (until 15 minutes before end)

Have comments, concerns, or feedback? We love to hear about ways to improve our curriculum! Fill out our Feedback Form or email teknow@andrew.cmu.edu, and we will get back to you as soon as possible.

7. Play Each Others' Games (:end)

Have comments, concerns, or feedback? We love to hear about ways to improve our curriculum! Fill out our Feedback Form or email teknow@andrew.cmu.edu, and we will get back to you as soon as possible.