(http://csmatters.org) 2 - 7

0b10 - 0b111

Creating and Assigning Variables

Unit 2. Developing Programming Tools

Revision Date: Jul 23, 2019
Duration: 1 50-minute session

Lesson Summary

Summary

Students will learn to manipulate variables and value assignments through an activity in which they must become the variable. By the end of the lesson, they will have identified variables as memory locations. They will also assign, copy, and destroy values in order to perform a swap algorithm and visualize Python's manipulation of variables and values in memory.

Outcomes

- Students will learn about variable manipulation and assignment.
- Students will understand variables representing memory locations.
- Students will be able to use variables in Python.
- Students will be introduced to assignment on the exam reference sheet.

Overview

1. Getting Started (5 min) - Journal
2. Introduction of Content (25 min) - Visualizing Variables
3. Independent Activity (15 min) - "Swap to the Top"
4. Wrap Up (5 min)

Learning Objectives

CSP Objectives

- *EU CRD-2 - Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.*
    - LO CRD-2.B - Explain how a program or code segment functions.
    - LO CRD-2.C - Identify input(s) to a program.

- LO CRD-2.E - Develop a program using a development process.
- LO CRD-2.G - Describe the purpose of a code segment or program by writing documentation.

- *EU AAP-1 - To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.*
    - LO AAP-1.A - Represent a value with a variable.
    - LO AAP-1.B - Determine the value of a variable as a result of an assignment.
- *EU AAP-2 - The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.*
    - LO AAP-2.A - Express an algorithm that uses sequencing without using a programming language.
    - LO AAP-2.B - Represent a step-by-step algorithmic process using sequential code statements.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP3: Construct viable arguments and critique the reasoning of others.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.
- MP8: Look for and express regularity in repeated reasoning.

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- WHST 12.1 - Write arguments on discipline specific content
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)
- 5. Using mathematics and computational thinking

Key Concepts

- Variables are memory locations that we have named, so we can put things in those locations.
- Variables contain values, and can only contain one value at a time.
- When assigning one variable from another, a copy is made rather than a transfer.
- If a variable is being assigned after having been initialized to a previous value, that previous value is lost and no longer accessible unless that value has been copied to another variable space.

- Programs implement algorithms. In order to accomplish a task: create a plan (algorithm), translate it into code and identify what kinds of information will be stored and maniuplated (variables)

Essential Questions

- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How are programs developed to help people, organizations or society solve problems?
- How do computer programs implement algorithms?

Teacher Resources

Student computer usage for this lesson is: **none**

For the Students:

- Cups
- Notecards
- String and paper to create large name labels (could also use nametags)
- Worksheet: "Swap to the Top.docx" (in the Lesson Resources folder)
- Homework

Optional:

- Differentiated one-pager (see "Options for Differentiated Instruction" in lesson).

For the Teacher:

- Write the following code on the board (after students have a chance to brainstorm their own solutions):

```
team1 = "Miami Heat"
team2 = "Washingon Wizards"
temp = team1
team1 = team2
team2 = temp
```

Lesson Plan

Getting Started (5 min)

Journal

What is a variable? What are some things in your life that change often?

Introduction of Content (25 min)

Introduction of Variables [5 min]

See slides in Lesson Resources Folder for a guided introduction.

Guided Activity: Visualizing Variables [15 min]

Materials

Disposable cups, index cards, names for variables on strings (to hang around students' necks)

Setup

1. We're about to talk about sports teams and their ranking in terms of variables. When we place teams on a ranking board, Team 1 is always at the top, followed by Team 2, Team 3, etc. Write a ranked set of 3 to 5 (real or invented) teams on the board. As they play one another, this ranking may change and someone new becomes Team 1.
2. Ask for a student volunteer. Give this volunteer a cup: they are now representing a new variable! In order for a computer to know which variable is which, they need names . Variable names can be any sequence (beginning with a letter or underscore, and including no spaces), but usually we use words that describe what is inside the variable. (For example, if I have a variable for "student's age," I'm not going to name it `tree` , because the word `tree` has no relevance to your age.)
3. Label this variable `team1` because we're talking about sports teams. Put the name around the student's neck to indicate they have become the variable team1. This student has now become `team1` , and should not respond to any other names! *Note:* A variable name will *never* have quotes around it - that would indicate that it is a string (i.e., a type of *value* rather than a variable's *name)*. Variables can be identified (loosely) by words or letters that are not in quotes and are not keywords (for, if, else, etc.).
4. Next, we will *assign* a value to the variable. This means we will store that value in the location `team1` . Because we have never used this variable before, we are *initializing* it - putting something in this box (memory location) for the first time. On the board, write `team1 = "Miami Heat"`
5. **CFU: What is the type of "Miami Heat"?**
   ("Miami Heat" is a string. We know this because it has quotes around it.)
6. The equals sign is what allows us to make this assignment. We are not saying that `team1` is the same as the Washington Wizards; it is only where were are currently keeping that team. Write "Miami Heat" on a notecard and put it into the cup.
7. Next, we need to *initialize* our second variable and *assign* a value to it. Follow the same process as `team1` and "Miami Heat", this time using `team2` and "Washington Wizards".
8. **CFU: What is the difference between a value and a variable?**
   (Values can be stored in variables. The content of a variable can change to different values, but its name will always be the same because the name is just an identifier of a location in memory.)
9. We've now *initialized* our two variables, but as it turns out… last night the two teams played, and the order was upset, putting the Wizards ahead! We have to model a *swap* in our code to reflect this change and fix our scoreboard.
10. **Think-Pair-Share:** A value is only safe (and not lost to the world of cyberspace) if it is in a variable. A variable can only hold one value at a time. How can we swap values between

`team1` and `team2` ?
(Allow students 3 minutes to brainstorm.)

11. Call on students and have them direct `team1` and `team2` to swap their values. **Key point:** when we access a variable and put its value in the place of another variable, that value is being copied. By doing so, however, any previous value in the variable is lost**.**
12. Ideally, a student will realize they must create another variable and do so, walking each other through the completion of the swap. If not, you should guide them towards this discovery.
13. **CFU (if necessary): Cold call a student to create a third variable,** `temp` **.**
(Make sure to call up a third member of the class to become the variable `temp` .)
14. If students have not at this point finished the swap, continue to walk them through the swap of a value to the `temp` variable, so the variables do not overwrite one another.
15. *Initialize* a variable `temp` (named so because we will not be using it very long). Write `temp = team1` on the board. Notice that in this assignment, it looks like we are setting a variable equal to another variable. Instead, we are setting the variable `temp` equal to the *value* inside the variable `team1` . This means that the value is copied to a second location.
16. Call a new volunteer to the floor. Give them a cup and name them `temp` .
17. Write a second notecard labeled "Miami Heat" and put it in the `temp` cup.
18. Now that the "Miami Heat" value has been saved somewhere, we can swap "Washington Wizards" into `team1` . Write `team1 = team2` on the board.
19. Go to `team2` and copy the text from the notecard onto a new one. Move to `team1` and replace the previous notecard with this new one.
20. A variable can only store one value at the time, which means that we've lost whatever was in `team1` . Rip up the notecard or throw it in the trash.
21. Complete the final assignment of `team2 = temp` by following the same process.

Synopsis

Look at the code written on the board and ask a student to walk us through each step.

CFU: Why did we create the variable `temp` ?

(Because variables overwrite the values of one another, and if we were to just set `team2 = team1` we would lose one of the values.)

Students should notice:

- Variables are memory locations that we have named, so we can put things in those locations.
- Variables contain values, and can only contain one value at a time.
- When assigning one variable from another, a copy is made rather than a transfer.
- If a variable is being assigned after having been initialized to a previous value, that previous value is lost and no longer accessible.

Discussion [5 min]

In reality, Python actually has a "shortcut syntax" that allows us to make this swap in one step. It looks like this:

```
a,b = b,a
```

If we wanted to swap the *values* in `team1` with `team2` , we would simply have to write:

```
team1,team2  = team2, team1
```

Here, Python is doing exactly what we were doing. It is internally creating a variable (which has no name but serves the same purpose as our `temp` variable), using it as a place holder, and then completing the swap.

To assign a value to a variable in Python we use the equal sign such as age = 15. On the exam produced by the College Board an arrow is used to indicate assignment.

Using the syntax on the College Board's exam reference sheet, an arrow "←" is used so the same assignment is written age ← 25.

Think-Pair-Share

Why would the makers of Python build in this function? What other uses does it have?

Independent Activity (15 min)

A version of this worksheet can be found in the Lesson Resources folder, titled "Swap to the Top".

Give students a list of games that have been played by the teams on the board, and the resultant new ranking. Have them create a piece of code that will reorganize the teams into the correct ranking. (They can assume that the variables `team1` ... `team13` already exist and have been initialized to the Friday ranking.) Give them the option of doing so through the use of manipulatives, or on their own paper.

Wrap Up (5 min)

Discuss: All computer programs can be broken down into smaller, simpler steps. By developing components, testing to be sure they are correct and combining them you can create complex, correct programs. Did students get a feeling for systematic development, and how to trace a program one step at a time to verify correctness? Distribute Exit Ticket in Lesson Resources Folder which asks students to figure out the values of variables hello and goodbye after the code in each exercises been executed.

Journal

What is the difference between a variable in a math class and in a computer science class? What is the difference between a float and an integer? Why would you use one instead of the other?

Homework

Continue working on "Swap to the Top" worksheet.

Options for Differentiated Instruction

Create a 3 column 'one-pager'. In the left column, create a copy of the code from the activity (a swap algorithm) including `team1` **,** `team2` **,** and `temp` that performs the complete swap. In the middle column, write steps that occur in the code (Step 1: Initialize variable and assign the value).

In the third column, the students should draw a visual of what that looks like in terms of disposable cups and index cards. (They can just write the variable name on the cup, rather than drawing in a person as well.)

Evidence of Learning

Formative Assessment

Checks for Understanding are embedded in the lesson. They are also shared below.

What is the type of “Miami Heat”?

(“Miami Heat" is a string. We know this because it has quotes around it .)

What is the difference between a value and a variable?

(Values can be stored in variables. The content of a variable can change to different values, but its name will always be the same because the name is just an identifier of a location in memory.)

Think-Pair-Share: A value is only safe (and not lost to the world of cyberspace) if it is in a variable. A variable can only hold one value at a time. How can we swap values between *team1* and *team2*?

(various answers)

Why did we create the variable `temp` ?

(Because variables overwrite the values of one another, and if we were to just set `team2 = team1,` we would lose one of the values.)

Summative Assessment

Task: Create a piece of code that will rearrange the ranking of the teams in order to reflect the outcome of previous games. (Independent Practice)

(http://www.umbc.edu/) (http://www.umd.edu/)

(http://www.nsf.gov/)

Authored by: CS Matters in Maryland
Website: csmatters.org (http://csmatters.org)
Email: csmattersinmaryland@gmail.com (mailto:csmattersinmaryland@gmail.com)