(http://csmatters.org) 2 - 6

0b10 - 0b110

Types and Evaluation

Unit 2. Developing Programming Tools

Revision Date: Feb 06, 2020
Duration: 1 50-minute session

Lesson Summary

Pre-lesson Preparation

Create a table of values of all types and print a copy of them on the cardstock for each group. Cut the papers up into individual cards, so each value is on its own card. Place each group's cards in a plastic bag.

Summary

Students are introduced to basic programming vocabulary, including integers, floats, strings, values, and expressions. They will work through a set of guided notes and slides, and, then, be released to explore Python through an independent (or paired) exercise.

Outcomes

- Students will learn that values are fundamental things that the program manipulates.
- Students will program four different types of values: floats, ints, strings, and booleans (briefly).
- Students will understand that in many programming languages, integers are represented by a fixed number of bits, which limits the range of integer values and mathematical operations on those values. This limitation can result in overflow or other errors but that other programming languages provide an abstraction through which the size of representable integers is limited only by the size of the computer's memory; this is the case for the language defined in the exam reference sheet.
- Students will understand that in programming languages, the fixed number of bits used to represent real numbers limits the range and mathematical operations on these values; this limitation can result in round-off and other errors. Some real numbers are represented as approximations in computer storage.
- Students will evaluate expressions the way they are evaluated by Python and show that they result in a value.
- Students will understand how numeric expressions are formed and evaluated.
- Students will understand that some statements are executed by Python and may not result in a value.

Overview

1. Getting Started (5 min)
2. Introduction of Content (10 min)
3. Guided Activity: Type Sort (15 min)
4. Independent Activity (15 min)
5. 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.F - Design a program and its user interface.
    - LO CRD-2.G - Describe the purpose of a code segment or program by writing documentation.
    - LO CRD-2.I - For errors in an algorithm or program: a. Identify the error. b. Correct the error.
- *EU DAT-1 - The way a computer represents data internally is different from the way the data is interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.*
    - LO DAT-1.A - Explain how data can be represented using bits.
    - LO DAT-1.B - Explain the consequences of using bits to represent data.
- *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.
    - LO AAP-1.D - For data abstraction: a. Develop data abstraction using lists to store multiple elements. b. Explain how the use of data abstraction manages complexity in program code.
- *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.
    - LO AAP-2.C - Evaluate expressions that use arithmetic operators.
    - LO AAP-2.D - Evaluate expressions that manipulate strings.
    - LO AAP-2.E - For relationships between two variables, expressions, or values: a. Write expressions using relational operators. b. Evaluate expressions that use relational operators.
    - LO AAP-2.L - Compare multiple algorithms to determine if they yield the same side effect or result.
    - LO AAP-2.M - For algorithms: a. Create algorithms. b. Combine and modify existing algorithms.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- 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 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.8 - Evaluate the hypotheses, data, analysis, and conclusions in a science or technical text
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products

NGSS Practices:

- 2. Developing and using models
- 5. Using mathematics and computational thinking

Key Concepts

- Values are fundamental things that the program manipulates.
- So far, we have seen four different types of values: floats, ints, strings, and booleans (briefly).
- Expressions are **evaluated** by Python and result in a value.
- Statements are **executed** by Python and may not result in a value.
- Python is a high-level language that provides abstractions that make it easy to create powerful programs.
- High-level languages are translated by the computer into a lower-level language that the computer can process.
- The number of bits used by a programming language has a big effect on how values are stored and evaluated.

Essential Questions

- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?
- What is the difference between an expression and a statement?
- What are the different data types?

- How does the representation of a data type affect the way a program works?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- "Values and Types" slides
- "Exploration Question" document

Team Shake App (https://itunes.apple.com/us/app/team-shake/id390812953?mt=8)

Runestone (http://interactivepython.org/runestone/static/thinkcspy/SimplePythonData/simpledata.html#values-and-data-types): Values and Expressions

Lesson Plan

Getting Started (5 min)

- Write a reflection on the homework from last class to write code to introduce yourself. The program should:
  1. Display your name.
  2. Greet and ask for three interests.
  3. Display the three interests
  4. Give a reply like "That's interesting!"

  Did you run into any errors?

  Do you feel that you can easily write this kind of code?

  Share results with an elbow partner.

Teacher Note*:* Ideally, students are paired with people they don't work with as frequently, in order to promote classroom culture.

Introduction of Content (10 min)

Say: Python is a high-level programming language. It provides many tools that make it easy to build complex programs with little effort. Within Python, it is relatively easy to store and retrieve information and to evaluate simple and complex expressions. Evaluating an expression will produce a single value. Python can evaluate expressions that include variables, constants, operators, and functions.

Numeric values can be constants, retrieved from numeric variables or calculated based on expressions that combine values or variables with arithmetic operators.

In order for the computer to execute program code, the code must be translated (compiled) into a lower-level language that a particular computer can understand and process. A variety of different variable types are some of the abstractions available in a high-level programming language, and the language must implement storage, retrieval, and manipulation of values stored in each type. You do not need to understand how the implementation works in order to use variables in your program.

Refer to the PowerPoint called Values and Types in the Lesson Resources folder.

- **Value**:
  - Everything on a computer reduces to numbers
    - Letters are represented by numbers (ASCII codes)
    - Pixel colors are represented using three numbers (red, green, blue)
  - Python tells these numbers apart by the use of **types**
- **Type:**
  - A set of **values** and the operations performed on them
    - Examples of operations: +, -, /, *
    - The meaning of these depends on the **type**!
    - The Exam Reference Sheet uses these same symbols for arithmetic operators.
- **Integers** (ints):
  - Values: ... -3, -2, -1,0,1,2,3,4

        - **Integer** literals have no commas (1, 2, 10, 1034453)
    - In many programming languages, integers are represented by a fixed number of bits, which limits the range of integer values and mathematical operations on those values. This limitation can result in overflow or other errors.
    - The Exam Reference Sheet defines integers that are limited only by the size of the computer's memory.
    - Operations: +, -, * (multiply), /, ** (to the power of), % (MOD operator)
    - The Exam Reference Sheet uses a MOD b, which evaluates to the remainder when a is divided by b, instead of a % b as in Python

- **Floating point** (float):
    - Values: (approximations of) real numbers
        - In Python, a number with a `'.'` is a **float** (ie. 2.0)
        - Without a decimal, a number is an **int** (ie. 2)
    - In programming languages, the fixed number of bits used to represent real numbers limits the range and mathematical operations on these values; this limitation can result in round-off and other errors. Some real numbers are represented as approximations in computer storage.
    - Operations: +, -, * (multiply), /, ** (to the power of)
    - Exponential notation is useful!

- **Boolean** (bool) :
    - Values: True, False
        - Boolean literals are just `True` and `False` (they *must* be capitalized!)
    - Operations: not, and, or
        - not b: True if b is False and False if b is True
        - b and c: True if both b and c are True; False otherwise
        - b or c: True if b is True or c is True; False otherwise

- **Range of values (int and float):**
    - In many programming languages, integers and floats are represented by a fixed number of bits, which limits the range of values and mathematical operations on those values.
    - In some languages such as Python, the range of values is only limited by the size of the computer memory and not the number of bits used to represent the values
    - In programming languages where the range of values is limited by a fixed number of bits, overflow and round-off errors can occur.

Check for Understanding: Have students write an integer on their paper (check with their elbow partner that it is a number). Ask students to turn that integer into a float with the same numerical value. Talk about going the opposite direction (float to int).

(Example answers: int: 3 float: 3.0 or 3.00 or ...)

- **String** (str):
    - Values: Any sequence of characters within double or single quotes
    - Operations: + (referred to as *catenation* or *concatenation*)
    - Concatenation can only apply to strings
        - `"hello" + "world" = "helloworld"`
        - `"hello" + " world" = "hello world"`
        - `"hello" + 2` produces an error
- **Boolean** (bool) :
    - Values: True, False
        - Boolean literals are just `True` and `False` (they *must* be capitalized!)
    - Operations: not, and, or
        - not b: True if b is False and False if b is True
        - b and c: True if both b and c are True; False otherwise
        - b or c: True if b is True or c is True; False otherwise
    - **Check for Understanding:** *Hold up two objects for the students to see, then ask the following questions:*
        - I am holding object A **and** object B. True or False?
        - *Drop one of the objects.* I am holding object A **and** object B. True or False?
        - I am holding object A **or** object B. True or False?
    - Often comes from comparing **int** or **float** values
        - Order comparison: `i< j  i<=j  i>=j  i>j`
        - Equality, inequality: `i == j  i!= j` (`'='` means something else!)

Guided Activity: Type Sort (15 min)

1. Group students (see differentiation for grouping options).

2. Ask teams of students to group the values by types.

Teacher Note: See Differentiation for two variations of this activity.

Individual Activity (15 min)

Students are to complete the Lesson 2.9 Order of Operations in Runestone (https://runestone.academy/runestone/books/published/thinkcspy/SimplePythonData/OrderofOperations.html (https://runestone.academy/runestone/books/published/thinkcspy/SimplePythonData/OrderofOperations.html))

Students will complete the Exploration Questions worksheet which is found in the Lesson Resources folder and complete one of the two exercises below.

- Version 1:
  - Hand out exploration worksheet and allow students to work individually on their computer in PyCharm IDE or terminal/command.
  - *Make sure students are filling in their expected value before determining the calculated value. They need only fill in the "reason for calculated value" if their two values do not match originally.*
- Version 2:
  - If a student needs a little more practice before moving on, send them to the Runestone Interactive to complete the section on values and data types before releasing them to the exploration: http://interactivepython.org/runestone/static/thinkcspy/SimplePythonData/ValuesandDataTypes.html (http://interactivepython.org/runestone/static/thinkcspy/SimplePythonData/ValuesandDataTypes.html)

Wrap Up (5 min)

- Exit ticket: Give the value of the Python expression. Give the name of the operators used.

```
2 + 5
5 * 2
5 ** 2
5 / 2
5 % 2
5 + 2 * 4
5 * 2 - 3
```

- Which of the following Python expressions give an error in Python? Why?

```
"pay attention to details'
' " what's for lunch?", my partner asked'
"What's for lunch?", my partner asked"
```

Options for Differentiated Instruction

Ideas for grouping students:

- Team Shake App (https://itunes.apple.com/us/app/team-shake/id390812953?mt=8) A simple app that allows you to put in your students, decide the number of "teams", and randomly groups students. It will allow you to balance teams based on skill or gender, and will allow you to share teams via facebook or email.
- Require students to go find other people in the room and give them a high-five. They may not high-five the person next to them.
- Group by birthday month (make adjustments to the groups as needed)
- Number off
- Ask students to count up the letters in their first name and determine if they have an odd number of letters or an even number of letters. Find three or four others who also have an even or odd number of letters in their name and form a group.
  - Variation if you have talked about other number systems: Find the number of letters mod the number of groups you need.

Group Activity Variations:

1. Give the students a large piece of colored paper and a glue stick, and allow them to glue the types together for later use in the classroom.
2. Create small, individual sets of cards, and have them glue the cards in to their journals by type.
3. Create multiple sets of values. Hand all groups the first set of values and allow them to organize the values by type. For the remaining sets, have groups compete against one another to see who can sort the values the fastest.
4. Print out Bingo cards of values ahead of time and call for: an integer less than 10 and greater than 5, a String with 3 letters, etc. http://www.teach-nology.com/web_tools/materials/bingo/5/ (http://www.teach-nology.com/web_tools/materials/bingo/5/)

Evidence of Learning

Formative Assessment

Checks for understanding are incorporated throughout the lesson.

Summative Assessment

Exit ticket questions are incorporated into lesson.

(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)

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 United States License (http://creativecommons.org/licenses/by-sa/3.0/us/) by University of Maryland, Baltimore County (http://umbc.edu) and University of Maryland, College Park (http://umd.edu).