(http://csmatters.org) 2 - 5

0b10 - 0b101

Reading Python Code and Debugging

Unit 2. Developing Programming Tools

Revision Date: Sep 19, 2019
Duration: 2 50-minute sessions

Lesson Summary

Summary

Students learn simple Python programs and their structure, as well as the basics of debugging.

Outcomes

- Students will identify the parts of a Python program
- Students will describe the history of the computer bug.
- Students will look at debugging techniques and classifying errors.
- Students will compare different kinds of errors
- Students will understand the use and importance of commenting code

Overview

Session 1:

1. Getting Started (5 min) Discussion: What is a program?
2. Guided Activity (40 min) Exploration
3. Wrap Up (5 min)
4. Homework

Session 2:

1. Getting Started (20 min)
    1. Homework Review [5 min]
    2. Runestone Tutorial [15 min]
2. Activity (20 min)
3. Wrap Up (10 min) Journal
4. Homework

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.A - Describe the purpose of a computing innovation.
  - 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.D - dentify output(s) produced by 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.
  - LO CRD-2.J - Identify inputs and corresponding expected outputs or behaviors that can be used to check the correctness of an algorithm or program.
- *EU DAT-2 - Programs can be used to process data, which allows users to discover information and create new knowledge.*
  - LO DAT-2.E - Explain how programs can be used to gain insight and knowledge from 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.
- *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.G - Express an algorithm that uses selection without using a programming language.
  - LO AAP-2.H - For selection: a. Write conditional statements. b. Determine the result of conditional statements.
  - LO AAP-2.J - Express an algorithm that uses iteration without using a programming language.
  - LO AAP-2.K - For iteration: a. Write iteration statements. b. Determine the result or side-effect of iteration statements.
- *EU AAP-3 - Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.*
  - LO AAP-3.A - For procedure calls: a. Write statements to call procedures. b. Determine the result or effect of a procedure call.

Math Common Core Practice:

- MP2: Reason abstractly and quantitatively.
- MP6: Attend to precision.

- MP7: Look for and make use of structure.

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

NGSS Practices:

- 5. Using mathematics and computational thinking

Key Concepts

Programs can be developed to solve problems (to help people, organizations or society), for creative expression, to satisfy personal curiosity or to create new knowledge.

Additional outcomes beyond the original purpose of a program are possible.

Essential Questions

- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

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

For the Students:

- "Is Eliza Human?"
  - Use Python to build a chatbot.
  - printing, user input, variables, if statements, while loops
  - https://groklearning.com/csedweek/ (https://groklearning.com/csedweek/)

CodingBat practice

Useful once functions have been taught

http://codingbat.com/python (http://codingbat.com/python)

Development environments for python

Spyder Python Development Environment:

https://github.com/spyder-ide/spyder (https://github.com/spyder-ide/spyder)

Python IdlePycharm Python Development Environment using Python 3.4.1

Runestone: Interactive Python:

http://interactivepython.org/runestone/static/thinkcspy/index.html
(http://interactivepython.org/runestone/static/thinkcspy/index.html)

Lesson Plan

Session 1

Getting Started (5 min)

Journal Prompt: What is a bug in a computer program?

- See if anybody knows the history, this will be covered in Runestone General Introduction What is Debugging? lesson that follows.

Guided Activity (40 min)

Runestone Tutorial [30 min]

- Jigsaw: Divide students up into 5 groups. Have each group prepare a creative presentation on one of the topics below with visuals or an active component (rap, song, dramatic reading, etc.) to convey the message of that section. Groups have 15 minutes to prepare a 2 minute presentation.
- What is Debugging? (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/WhatisDebugging.html)
- Syntax errors (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/Syntaxerrors.html)
- Runtime Errors (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/RuntimeErrors.html)
- Semantic Errors (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/SemanticErrors.html)
- Experimental Debugging (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/ExperimentalDebugging.html)
- Its best to minimize time spent debugging. Programming design goals
    1. Plan how will you test to know if a section of code is correct- test cases
    2. Provide output that can be added to let you see what is going on in your program.
    3. Design code to make reading it and tracing its execution easy.
- Assign each group to investigate and share their thoughts about these goals.
- **Explore a simple Python program together. [10 minutes]**
    - Teacher demonstrates the results of running eliza.py using Pycharm or Idle
    - Hand out the code for eliza.py on paper to students, demonstrate on the front screen have them mark up the code as indicated on the student handout together as a class discussing the vocabulary below (all of these concepts will be taught in Unit 2, this is just a preview). (See the teacher guide for answer key.)
    - Vocabulary: (use the Runestone Glossaries if needed)
        - comment
        - function (or procedure)
        - parameter
        - input
        - output
        - loop
        - condition
        - variable
        - constant

- Have students mark up the code at the bottom of the page using the same key on the student handout
- Explain to students that comments are multi-purpose: 1) Are a form of program documentation to be read by people 2) Does not affect how a program runs 3) Used to acknowledge code segments that were developed collaborativel or by another source. Should include the origin or original author's name.

Wrap Up (5 min)

Journal: Use formative assessment Questions 1 - 3:

1. How do you make an output statement in Python?
2. How do you make an input statement?
3. What is a user prompt and what is its purpose?

Homework

Set up your IDE and Python at home (if you can) or use an online Python IDE such as https://repl.it/languages/Python3 (https://repl.it/languages/Python3) or ideone.com . Type up a simple "Hello World" program and get it to run. Bring in evidence that it works or write a few sentences about the issues you are having in trying to install it or write about your experience using an online IDE, or write out the code for Hello World without looking at any notes and report on how easy or hard it was to remember the details. (Be careful not to make any student feel awkward for lack of a home computer they are allowed to install softare on)

Session 2

Getting Started (20 min)

Homework Review [5 min]

Collect handwritten notes or evidence that Python 3 is set up at home or the student is able to use https://repl.it/languages/Python3 (https://repl.it/languages/Python3) or ideone.com. As students work on tutorial, address specific issues with students who had trouble with install.

Runestone Tutorial [15 min]

Complete an introductory tutorial:

Runestone Ch. 1 http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/toctree.html (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/toctree.html)

- Formal and Natural Languages (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/FormalandNaturalLanguages.html)
- A Typical First Program (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/ATypicalFirstProgram.html)
- Comments (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/Comments.html)
- Glossary (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/Glossary.html)

Activity (20 min)

Develop the beginnings of a chat bot where the computer and user introduce themselves to each other. The computer asks a question, the user provides a response and the computer responds back again, including the user input within the response (see https://groklearning.com/csedweek/

(https://groklearning.com/csedweek/) for ideas). Extend: Give your chatbot a personality like a friend, grandfather, therapist, or child.

Wrap Up (10 min)

Journal [10 min]

First discuss: How can additional desired outcomes happen independently of the original purpose of a program? (examples: a program like chatbot could help someone learn a new language, provide entertainment for a shut-in. Computer games can increase reflexes, logic skills, provide motivation for someone in physical therapy)

Answer Formative assessment Questions # 4 - 6:

4) Name several different input devices.

5) How do we comment a Python program? Why do we use comments?

6) In your journal, make a tree diagram, name the three types of errors, and give examples of each. Which type of error do you think is the hardest to detect and why?

Homework

Write code to introduce yourself. Display your name. Greet and ask for three interests. Display the three interests and give a reply like "That's interesting!" Print your code. Next day: Have students introduce one another by “running the code" of a classmate. If you cannot get PyCharm to install, use http://ideone.com/ (http://ideone.com/) or repl.it or create this program by hand on paper.

Options for Differentiated Instruction

Suggested strategies:

- Use red, yellow, green sticky notes at computer for a quick check of understanding.
- Provide guide questions for reading during Runestone interactive tutorial.
- Provide teacher-annotated text excerpts or have students annotate their own excerpts that follow the Runestone interactive.

Evidence of Learning

Formative Assessment

In your journal, make a tree diagram, name the three types of errors, and give examples of each. Which type of error do you think is the hardest to detect and why?

Summative Assessment

Find the errors in an algorithm or in Python statements covered to date

Define what debugging is and give examples of the 3 categories of bugs (syntax, logic(semantic) and runtime)

Describe the differences between programming and debugging.

 (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)