(http://csmatters.org) 2 - 14

0b10 - 0b1110

Functions: Scope and Abstraction

Unit 2. Developing Programming Tools

Revision Date: Sep 18, 2019
Duration: 2 50-minute sessions

Lesson Summary

Summary

In the first portion of this lesson, students continue their inquiry into the properties of functions, with a focus on communication to and from other functions. In the lab portion of the lesson, students develop three Python modules using both Runestone Interactive and their Python IDE. Students use their own functions to perform calculations and draw a variety of polygons and a circle using turtle graphics.

Outcomes

- Students will understand the process of communication with functions.
- Students will be able to create and call functions.
- Students will be able to use both "fruitful functions" (sometimes just called *functions*) and "unfruitful functions" (which do not return a value - these are usually called *procedures*).
- Students will understand how functions simplify a project's creation.

Overview

Session 1

1. Warmup (10 min)
2. Exploring Functions (40 min)

Session 2

1. Function Labs (45 min)
2. Wrap-up (5 min)

Learning Objectives

CSP Objectives

- *EU CRD-1 - Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.*
    - LO CRD-1.C - Demonstrate effective interpersonal skills during collaboration.
- *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.C - Identify input(s) to a program.
- *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.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.
    - LO AAP-3.B - Explain how the use of procedural abstraction manages complexity in a program.
    - LO AAP-3.C - Develop procedural abstractions to manage complexity in a program by writing procedures.

Math Common Core Practice:

- MP4: Model with mathematics.
- MP5: Use appropriate tools strategically.

Common Core ELA:

- RST 12.2 - Determine central ideas and conclusions in the text
- 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
- RST 12.10 - Read and comprehend science/technical texts

NGSS Practices:

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

Key Concepts

- Students will understand the process of communication with functions.
- Students will be able to create and call functions.
- Students will be able to use both fruitful and unfruitful functions.

- Students will understand how functions simplify a project's creation.
- Students will work collaboratively in pairs to enhance performance and communication

Essential Questions

- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How are algorithms implemented and executed on computers and computational devices?
- How are programs developed to help people, organizations or society solve problems?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?

Teacher Resources

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

In the Lesson Resources folder:

- Slope Y-Intercept Project (to be developed)

Other:

- Python for Everybody (http://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf)
- Runestone Interactive: How to Think Like a Computer Scientist (http://interactivepython.org/runestone/static/thinkcspy/index.html)
- Turtle graphics API documentation for Python 3.4.1: https://docs.python.org/3.4/library/turtle.html (https://docs.python.org/3.4/library/turtle.html)
- Circle Lab (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01.html (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01.html))
- Triangle Lab (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01a.html (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01a.html))

Lesson Plan

Session 1

Getting Started (10 min)

- Display *Python for Everybody* Chapter 4 (Functions) - Section 4.11.
- Students record in their journals four reasons to use functions and discuss with elbow partners to identify their two most important reasons.
- Ask students to identify two or three people they know who have the same name.

- Explain to their elbow partners who the people are and how they tell them apart when talking about them.
- Explain that we are going to working with functions today and that sometimes functions use the same names for different variables.
- Before writing programs with functions, we want to address the possibility of functions that use the same names.

Exploring Functions (40 min)

Teaching Note:

Use of a format such as Google forms is sugggested for collecting student responses to the questions for each activity. It is important to keep this portion of the class moving so students have enough time for the labs. Have students work with partners during their program development so they will have someone to share their progress with. Use a timer and have students briefly share their progress roughly every 10 minutes. This will not only help them understand an iterative development process, but also gives students practice with a collaborative development style.

Debugging

- Display Section 4.12 of *Python for Everybody*.
- Explain: when progam developers write their own functions, they often run into problems (bugs) they need to fix. These debugging suggestions should help the students to prevent or correct problems.
- As students investigate and work with functions today, they should an eye out for function design problems and strategies used to prevent or solve them. We will add the rules we develop to this list.

Variables and Parameters

- Students should open the Runestone Interactive Functions page (http://interactivepython.org/runestone/static/thinkcspy/Functions/toctree.html (http://interactivepython.org/runestone/static/thinkcspy/Functions/toctree.html)).
- Have students read "Variables and parameters are local" and run the Codelens to completion.
- Ask:
    - What caused the error that occurred when the last line of the code was executed?
    - What happens to local variables when a function finishes?
    - Are parameter's values that are assigned inside a function also accessible outside the function?

Global Variables (Don't Use Them!)

- Run ActiveCode 9.
- What is the value of the variable `power` inside the function?
- At what point was the variable `power` given that value?
- This form of communication is risky, since programmers may not realize that a variable used in one code segment is also used inside a function elsewhere in the program.
- What is the recommended way to communicate values to a variable?

- Emphasize that avoiding global variables supports collaboration and reuse of code, since different authors can avoid interfering with each other's work.

Variable Scope

- Complete the "Check for Understanding" after CodeLens 5, answering questions func-10, func-11, and func-12.
- Share "Check for Understanding" results either with elbow partners or the class as a whole.

Accumulating Values

- Go to the section entitled "The accumulator pattern."
- Do ActiveCode 10 and CodeLens 6.
- After reviewing the execution in the CodeLens, answer the "Check for Understanding" questions func-13 through func-14.
- See if the students can add any rules for creating and using functions to the list from *Python for Informatics*.

Functional Decomposition

- Functions support division of a progam into smaller modular parts.
- Functions that generalize a process promote the advantage of code reuse.
- Go to the section entitled "Functions can call other functions".
- Do CodeLens 7 and ActiveCode 12.
- Ask:
  - What function is started first when the code in ActiveCode 12 is executed?
  - What function is finished first when the code is executed?

Session 2

Function Labs (45 min)

- Review concepts from Session 1. Reinforce pair programming dynamics of navigator /driver, changing roles and communication goals to resolve any conflicts, and allow everyone to participate.
- Encourage students to refer to the API to better understand how functions work. (Remind them API (Applicaiton Program Interface) explains the functions available and how to use them without the need to know the internal strucutre of the function.
- Functional abstraction make programs more readable and managable.
- Preview the three lab projects that students will complete and help students to get started with the Slope Y-Intercept project.
  - Discuss the iterative development process after watching the video "Design as an Iterative Process"
  - http://www.youtube.com/watch?v=9NxWW4poljU (http://www.youtube.com/watch?v=9NxWW4poljU)
- Complete the Slope Y-Intercept Project and the Drawing a Circle Labs.
  - The Slope Y-Intercept Project is described in the document of the same name.
  - The document includes descriptions of two functions to complete and contains a small sample of test data.

-
  - The Slope Y-Intercept Project is in the Lesson Resources folder along with the starter Python file.
- Continue in Runestone to the Lab entitled Drawing a Circle (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01.html (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01.html)).
  - In this second lab, students develop a function that uses a turtle to draw a circle.
  - After finishing the lab, students should copy their code to a Python module in PyCharm.
  - Students should create three functions named `drawSquare` , `drawTriangle` , and `drawOctagon` , along with main module code that calls each function.
- After completing the Drawing a Circle lab, students should follow the link below to the Lessons from a Triangle lab.
  - http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01a.html (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01a.html)
- Students should create a second Python module with the code from the starting point for the `drawPolygon` function code and finish implementing the `drawPolygon` function.
- Students should complete the Function Labs by developing the `drawCircle` function in Finally a Circle (the second half of the Lessons from a Triangle lab).
  - Additional components of the lab can be used as extensions for more able students.

Wrap-Up (5 minutes)

- After submitting the labs, students present their labs with elbow partners and groups.
- Students reflect on the following prompts about writing programs with functions.
- What problems did they encounter? (with concepts or partner dynamics)
- How did they overcome the problems? How was the API useful?
- How do functions simplify the task of developing a program?
- How can effective collaboration enhance performance?

Options for Differentiated Instruction

Iterative development works by developing, then sharing, programs at many points during the development process. Students should work in pairs as they create their programs and share the work through various completion stages.

Students who are completing projects quickly should be introduced to the Python turtle API at (https://docs.python.org/3.4/library/turtle.html). Students can investigate and implement such methods as `fill` , `speed` and others as described by the API documentation.

Evidence of Learning

Formative Assessment

Check for understanding by assessing student performance on the Runestone Interactive questions. Students should first try to resolve any difficulties with their partners and groups.

Students should be able to make suggestions for creating and using functions.

Identify and address any areas discovered that students have been unable to come to a consensus understanding.

Have students reflect on the pair programming process as prompted in the lesson.

Summative Assessment

Create functions that receive parameters, perform calculations using those parameters, and return a value.

Write a function to return the slope and y-intercept of a function of the line through two points.

Write functions to create a variety of polygons and a circle.

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