**AP CSP CodeX**

|  |  |  |
| --- | --- | --- |
| **LESSON: Defining Functions** | | **Time: 45 minutes** |
| **Project Goal:** Students will learn and to define and call functions.  **Learning Targets**   * I can define “abstraction” and “function”. * I can identify places in code where a function can be defined. * I can define a function that accomplishes a task. * I can call a function in code. * I can state benefits for using functions. | **Key Concepts**   * A function is a form of procedural abstraction. It hides the details of accomplishing the task. * Abstraction is an integral part of computer science and coding. * Programmers write functions that accomplish a task. * A function can be called multiple times and in any order in a program. * A function must be called before its code is executed. | |
| **Assessment Opportunities**   * Defining Functions Activity Guide * Pixels1\_functions program | **Success Criteria**   * Identify when code can be written as a function * Define a function * Call a function | |
| **AP CSP Framework**  **AAP-3.A** Write statements to call procedures.  **AAP-3.B** Explain how the use of procedural abstraction manages complexity in a program.  **AAP-3.C** Develop procedural abstractions to manage complexity in a program by writing procedures.  **Computational Thinking Practices 3.B** Use abstraction to manage complexity in a program.  **Computational Thinking Practices 3.C** Explain how abstraction manages complexity. | **Materials**   * Defining Functions Activity Guide / Answers * AP CSP CodeX Vocabulary List * AP CSP CodeX Python Code List * Unit 1 Review Links and Test Questions * Solution code for Pixels\_functions | |
| **Teacher Notes**   * This lesson introduces functions before it happens in a Mission or CodeTrek. Functions are a vital concept for AP CSP, and students should have a lot of practice defining and calling functions. This lesson will have students modify the program created in Mission 3 by defining functions. * This lesson will be completed on the computer, using CodeSpace for programming.. * Use the Sandbox in CodeSpace for programming. This lesson is not part of a mission. * Students should do a “Save As” with each program to keep the original code. * The assignment can be distributed digitally. Space is provided for students to take notes while programming. * Students will modify a previous program. The best experience will come from them modifying their own code. However, we want all students to be engaged, so you can give them the original code to modify if needed. * The most recent version of the program can be found in an earlier assignment. If you are giving code to students, use the solution code found there:   + Pixels1\_final (Mission 3) * Follow the slides for instructions and guidance. * Solution code for the final program with functions is provided. * At the end of the lesson, discuss clearing the CodeX before turning it in. You can use the Clearing CodeX slides. * The assignment is best completed digitally. Prepare the assignment for distributing through your LMS. * Another suggestion for assessment is for students to keep a daily journal, or use a reflection form for students to process information they learned and reflect on questions they may still have. * You may consider having students (or the class collectively) keep a chart of errors and the ways to fix them. * You can also add vocabulary to a word wall and keep a document or chart of the Python code learned during each mission. * The teaching guide (below) gives the narration for one way to present the lesson. | | |

**Teaching Guide**

**Warm-up / Functions and Abstraction (10 minutes)**

This short warm-up is to introduce abstraction and functions. Students take guided notes on their activity guide.

💡 **Teaching tip – warm-up**

* Go through slides #1-8
* Include any student discussion about abstraction and functions. Discussion can also wait until the wrap-up when students have experience with these two new terms.

**Modify Mission 3 (20 minutes)**

💻 Students will work in pairs (or individually) at their computers.

💡 **Teaching tip:**

You can go over the instructions with the students, or you can use the slides (slides 9-15).

* Students are asked to modify their code from Mission 3. Students should be able to use their own code. However, if you want to give them code to use, use the solution code for Pixels1\_final (Mission 3).

**Challenge (optional – 10 minutes)**

💻 Students will work in pairs (or individually) at their computers.

💡 **Teaching tip:**

A challenge is not explicitly given in the activity guide or slides. However, if you have more time during the class period, have students create more functions for the pixels and call the functions in the main program for a fabulous light show.

✅ You decide what want you want students to turn in for a grade, and how they turn it in: activity guide and/or pixels1\_functions program. You can look at the code on student computers, or have them submit code. The assignment document can also be turned in.

**Wrap-Up (5 minutes)**

The wrap-up can be very short for this lesson (slides 27-28). This is the class’s chance to review abstraction and functions and make their own meaning with the terms.

Formative Assessment:

* Daily reflection journal
* Class discussion on what they learned about abstraction and functions
* Activity Guide completion
* Defining and calling functions in Mission 3
* Exit ticket

**SUCCESS CRITERIA:**

* Define “abstraction” and “function”
* Correctly define a function
* Understand how and when to call a function
* Define and call one or more functions in a program
* Determine when a function should be used in code