A LEARNING CONTEXTS AND TASKS

This appendix provides additional detail on the tasks associated with each study.

A.1 Pretest

The optional pretest, which can be run as timed or untimed, is designed to assess the student's level of knowledge of the subject matter.

Figure 26: First 5 questions from 10-question multiple-choice pretest

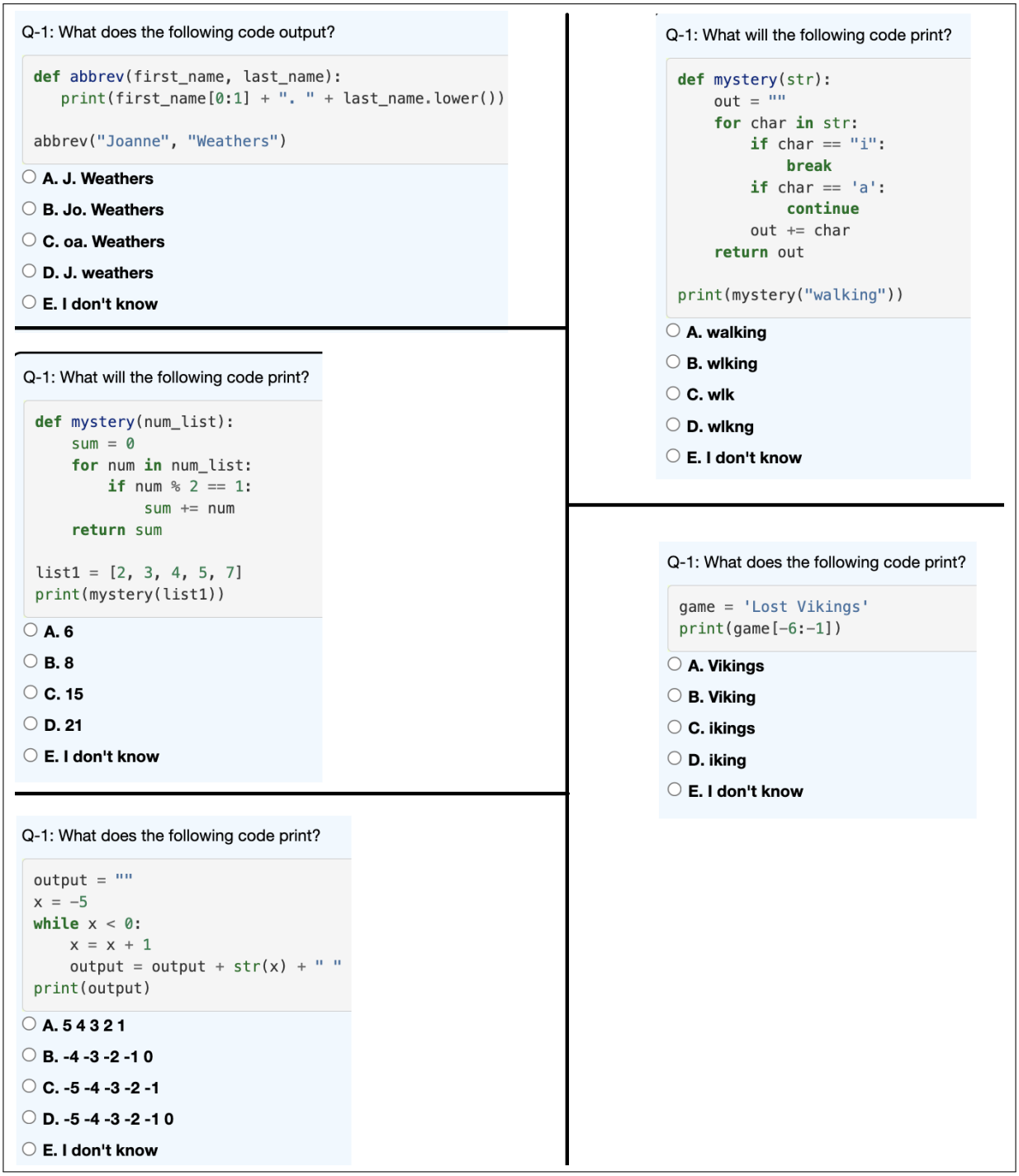

Figure 27: Last 6 questions from 10-question multiple-choice pretest

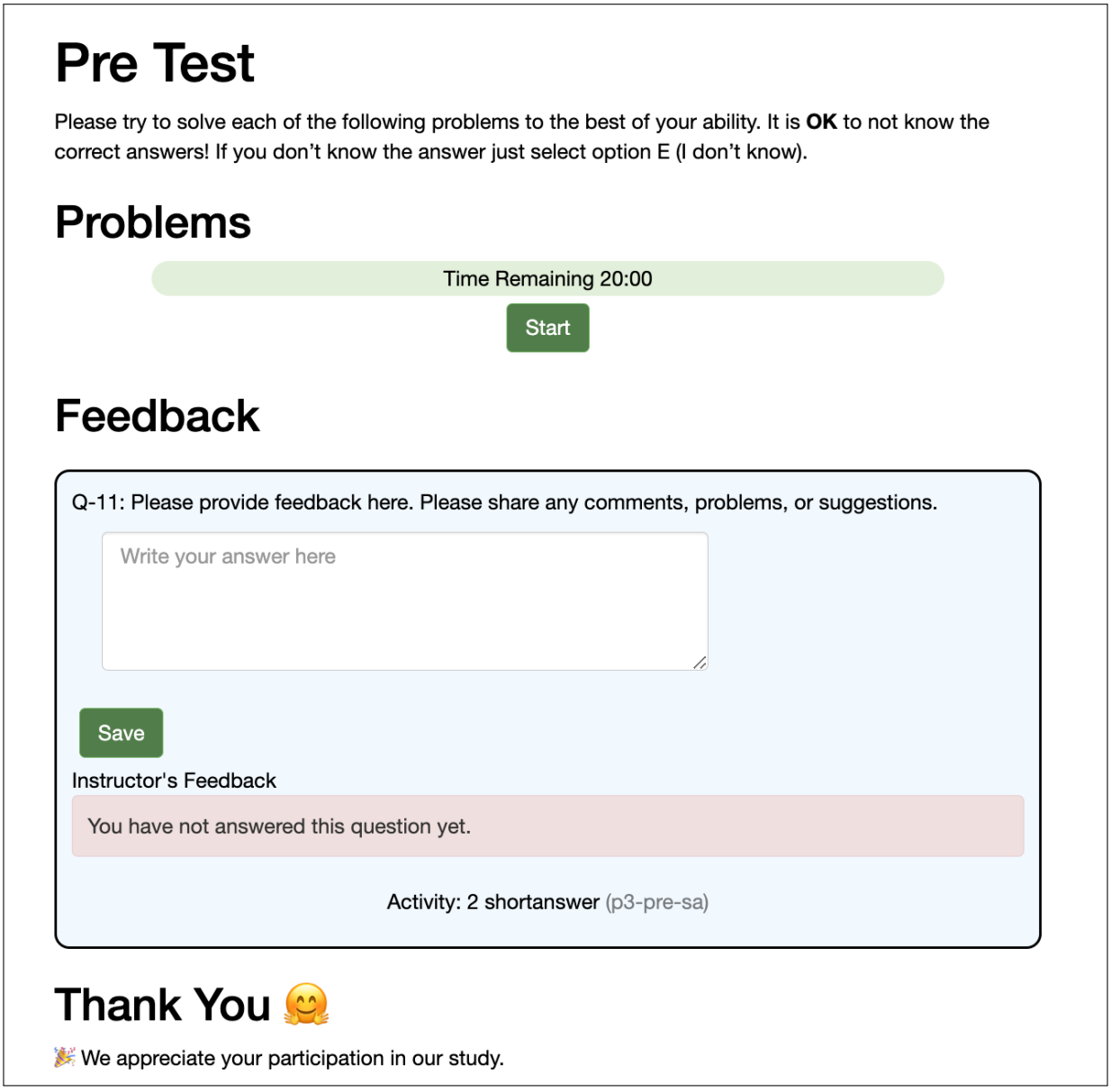

Figure 28: Pretest final feedback question

A.2 Introduction to Problem Types

All of the studies included a primer on how to use the interface, which also served to introduce the problem types and help students familiarise themselves with Parson's problems.

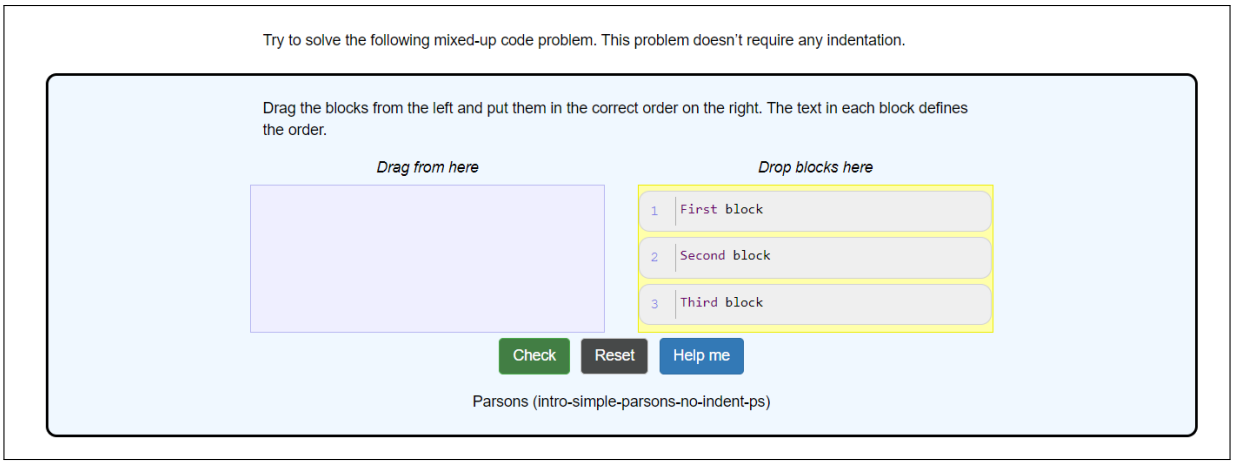

Figure 29: Introduction to problem types - Students familiarise themselves with the drag and drop interface for Parsons problems

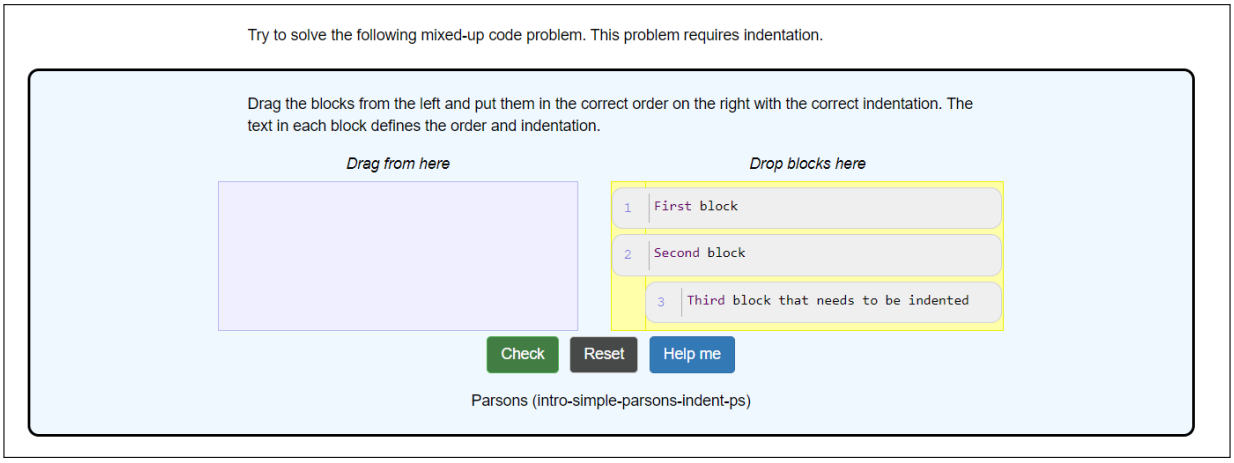

Figure 30: Introduction to problem types - Students are introduced to indentation in Parsons problems

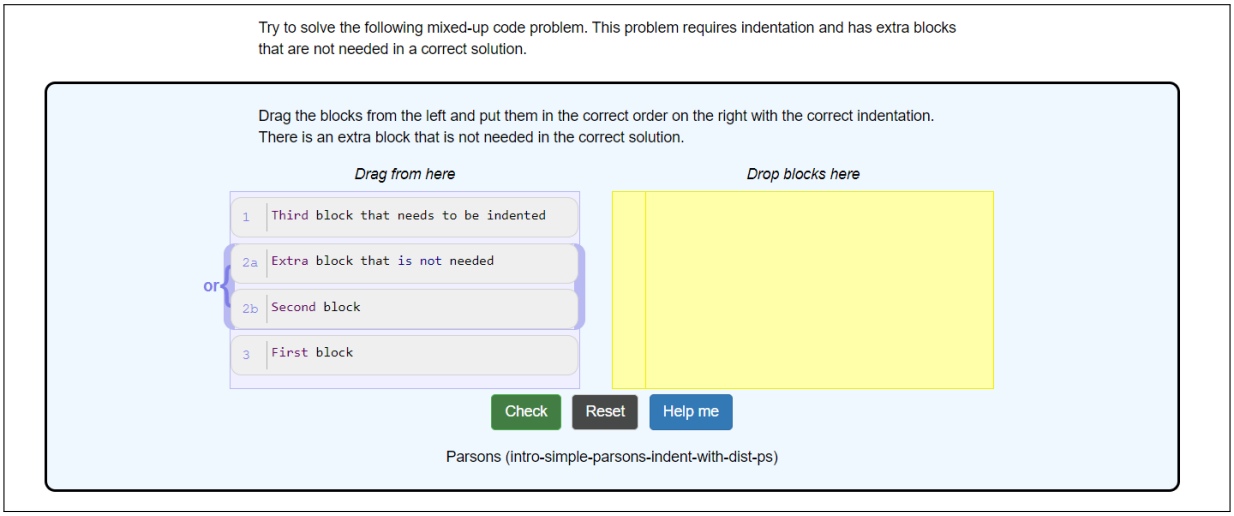

Figure 31: Introduction to problem types - Students practice parsons problems with distractor blocks i.e. blocks which are not needed in a correct solution

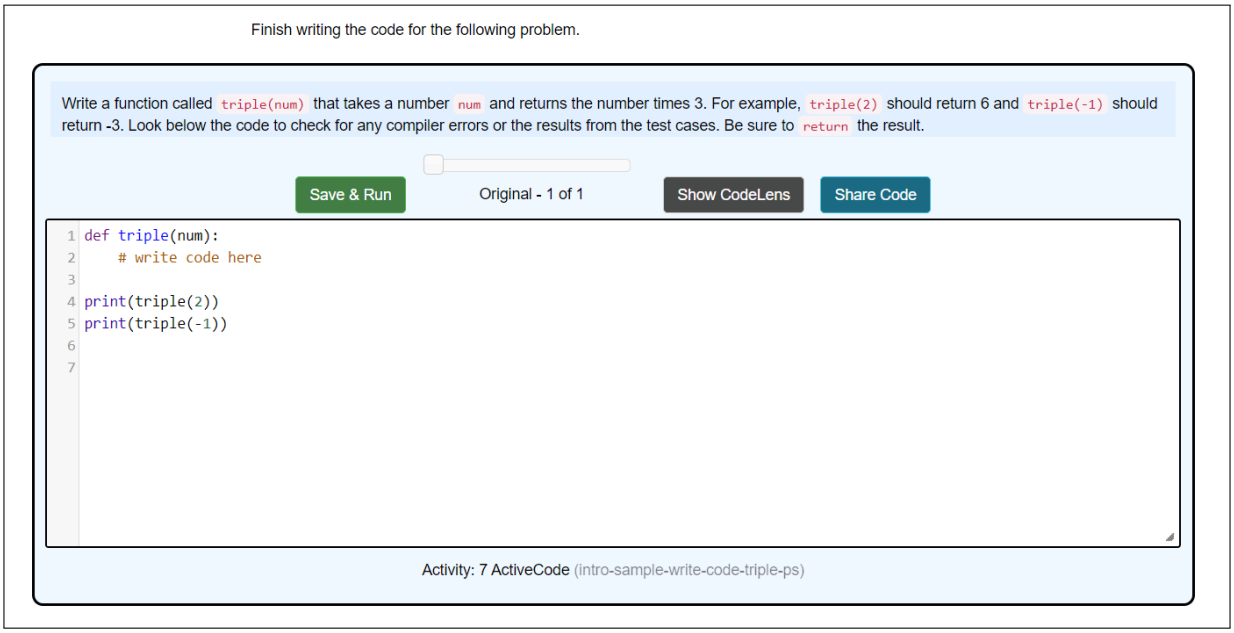

Figure 32: Introduction to problem types - Students practice writing code with unit tests

A.3 *python-swap*

Students are tasked with swapping the value of one variable with the value of another variable.

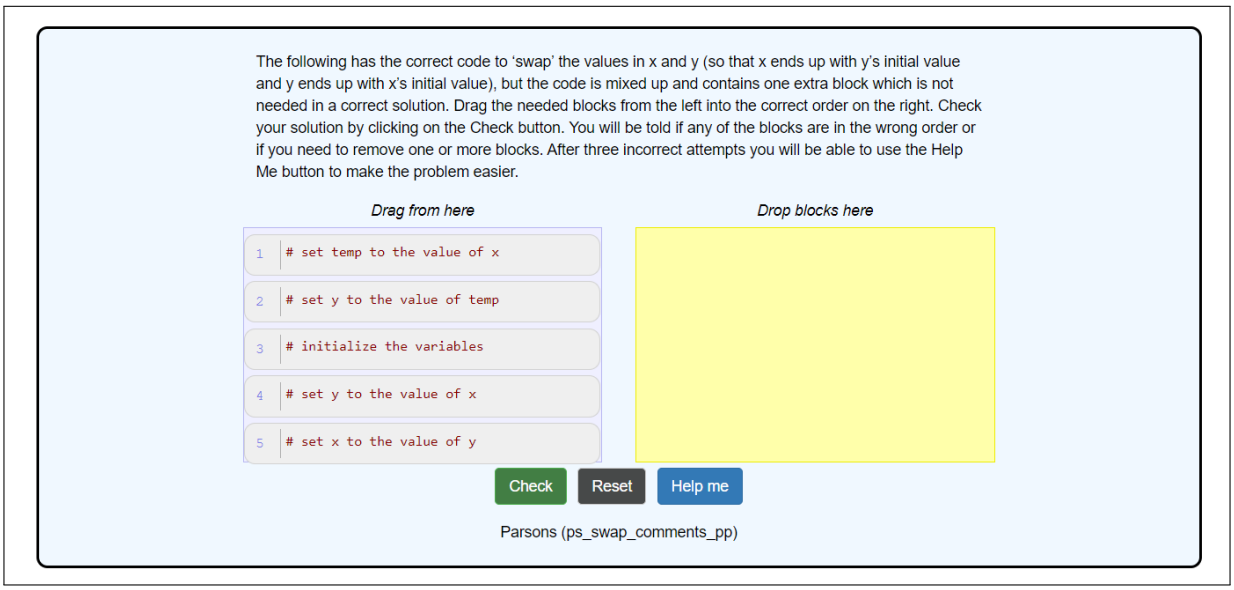

Figure 33: *python-swap* - Students reorganise comment blocks which contain the logic of the algorithm for swapping the values of two variables

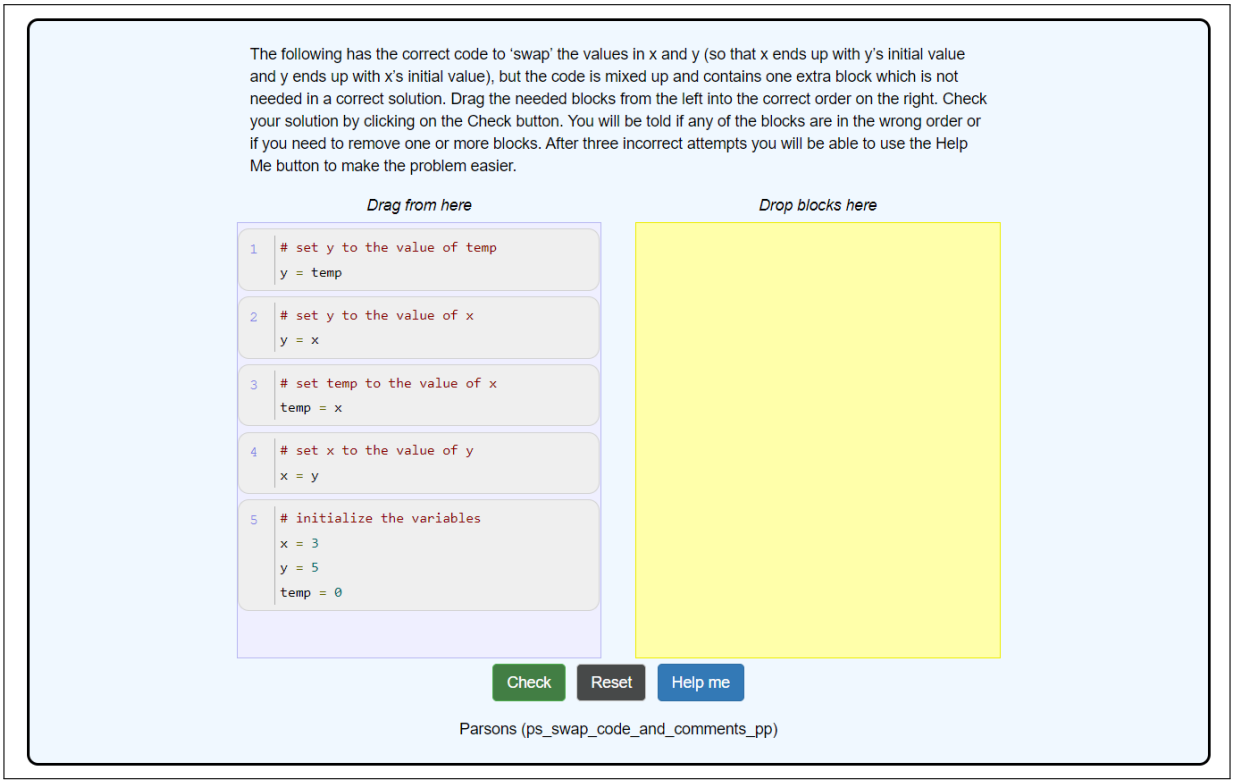

Figure 34: *python-swap* - Students practice combining the code with comments blocks to teach the logic of the process

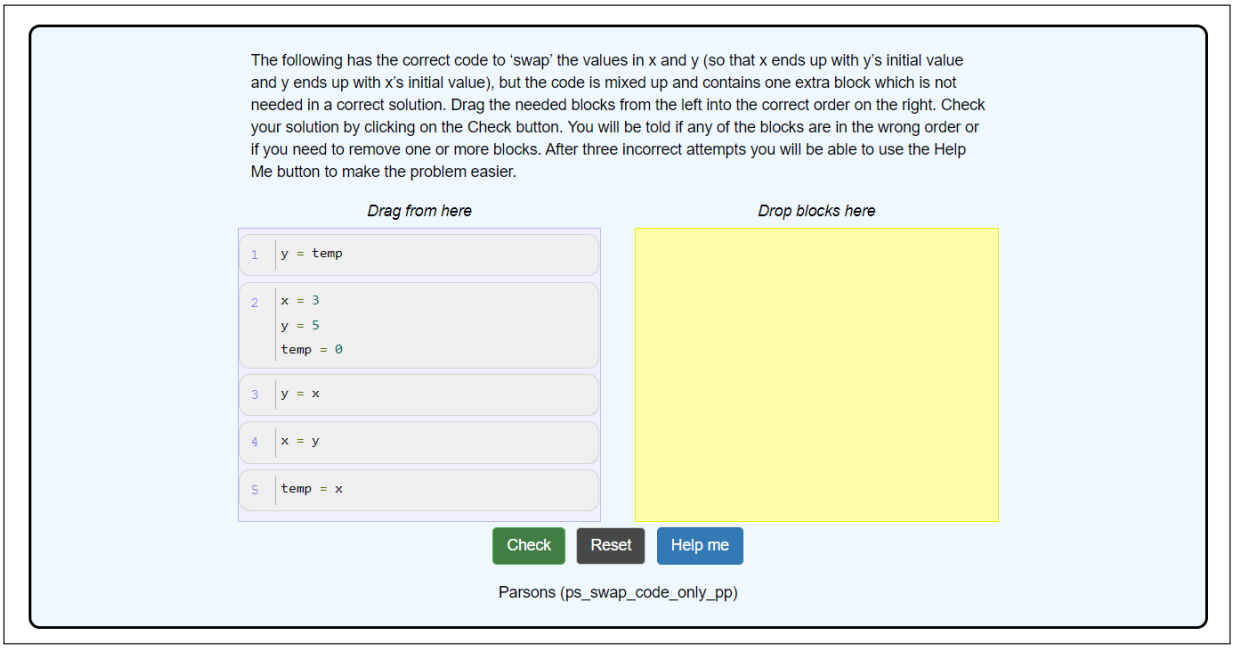

Figure 35: *python-swap* - Students are tasked with a Parsons problem to swap the values of two variables

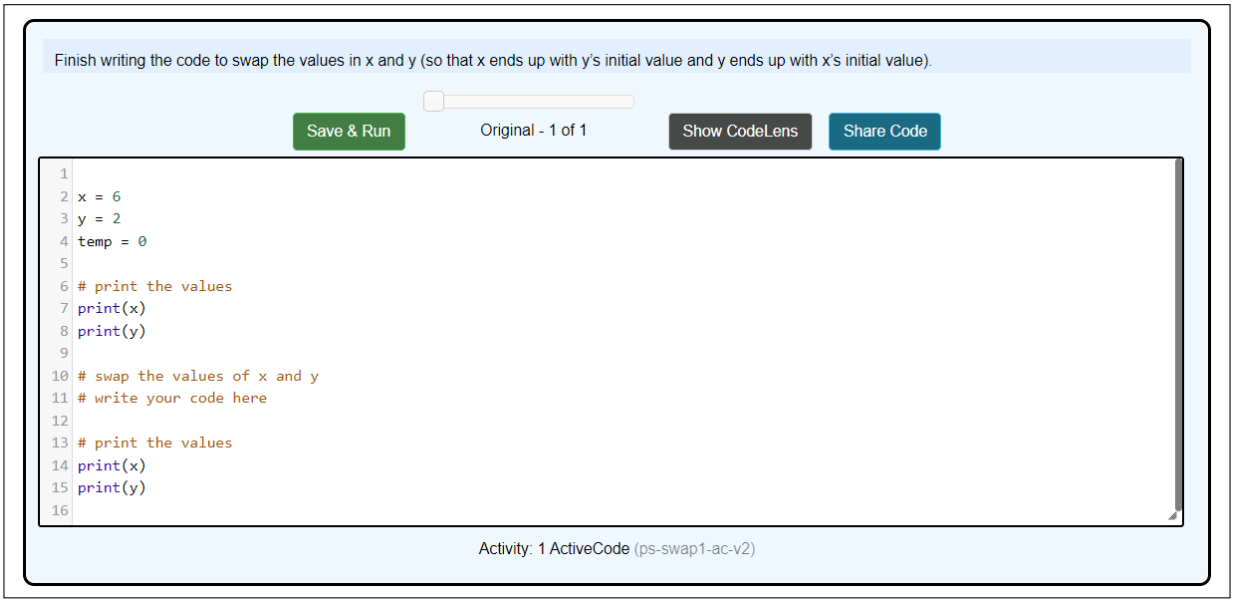

Figure 36: *python-swap* - After completing the Parson's problem exercise, students proceed to write a solution using code

Figure 37: *python-swap* - A similar code-writing task with different variable names for near transfer

A.4 *p3pt*

Students are tasked with defining functions that solve defined problems in Python.

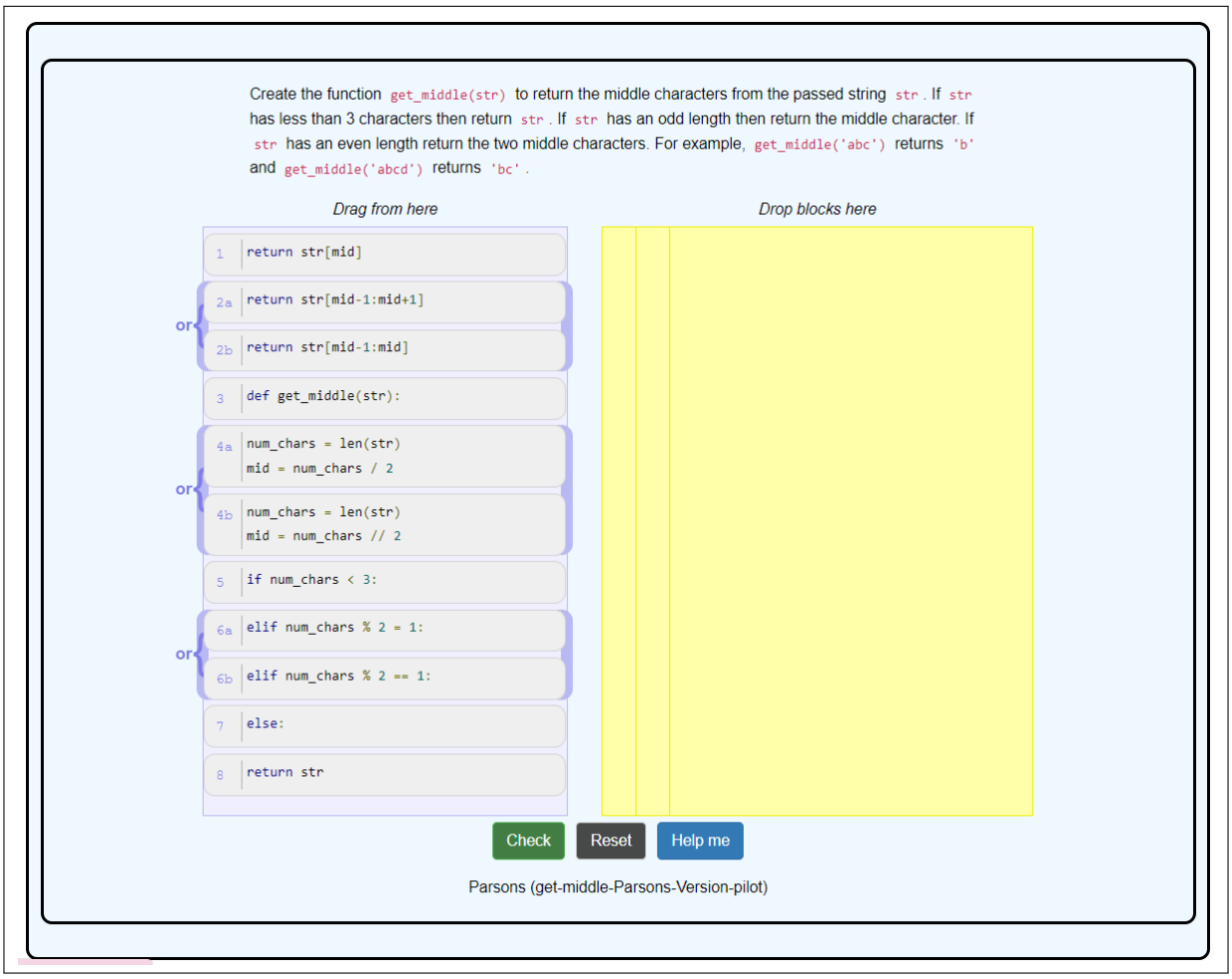

Figure 38: *p3pt* - Defining a function which extracts the middle characters from a string (Parsons)

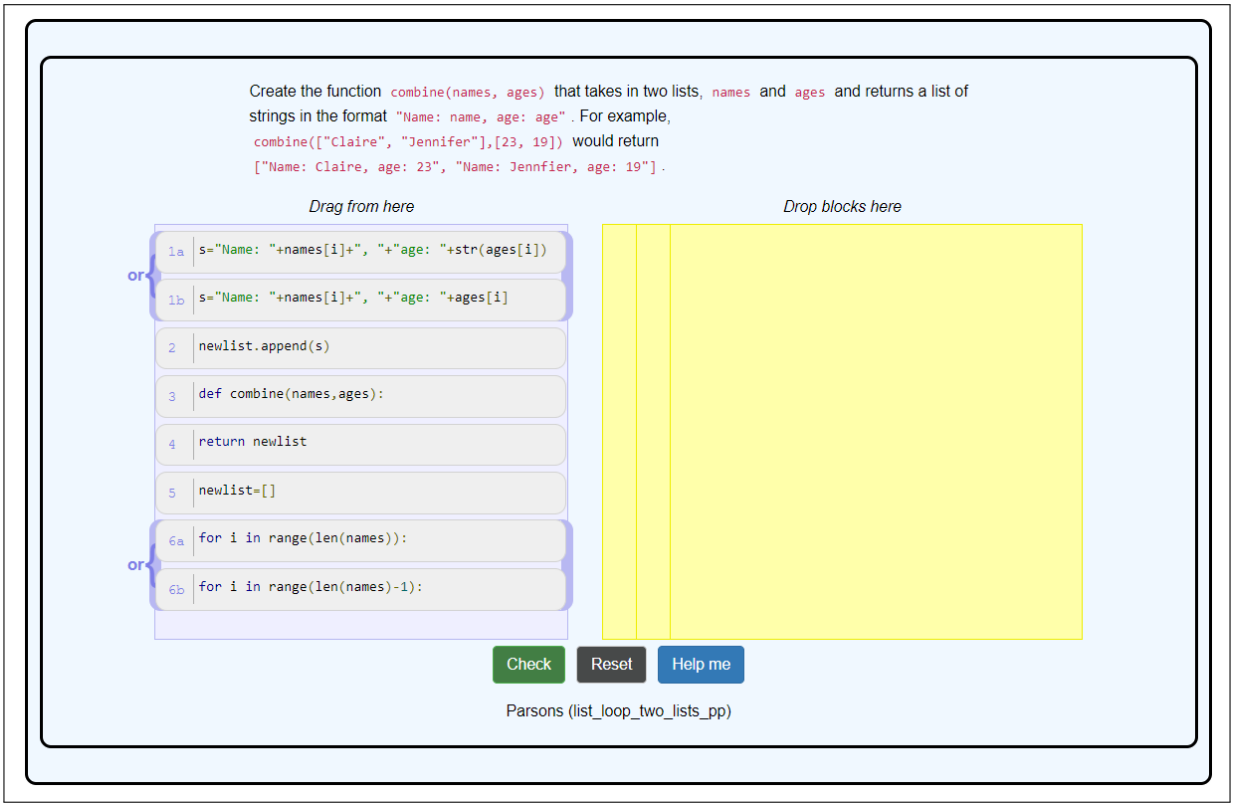

Figure 39: *p3pt* - Defining a function that concatenates two lists according to requirements (Parsons)

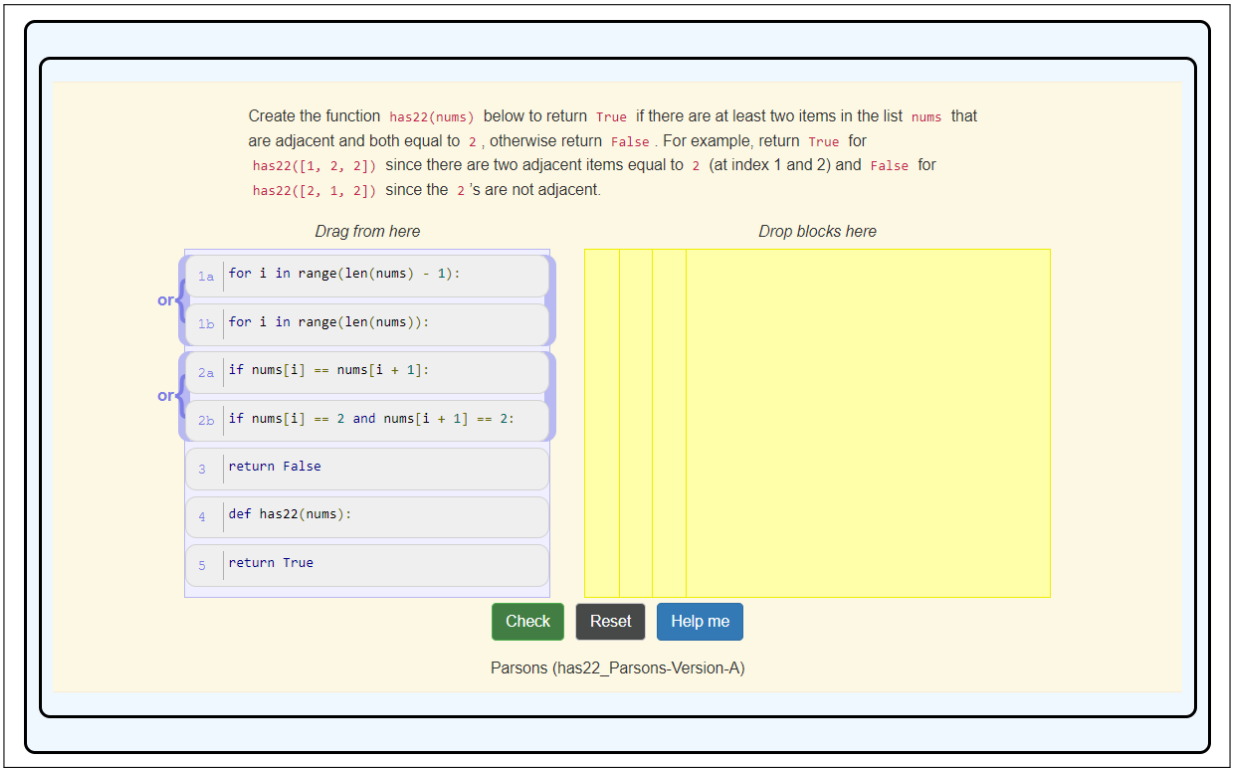

Figure 40: *p3pt* - Defining a function to search a list for adjacent values of two (Parsons)

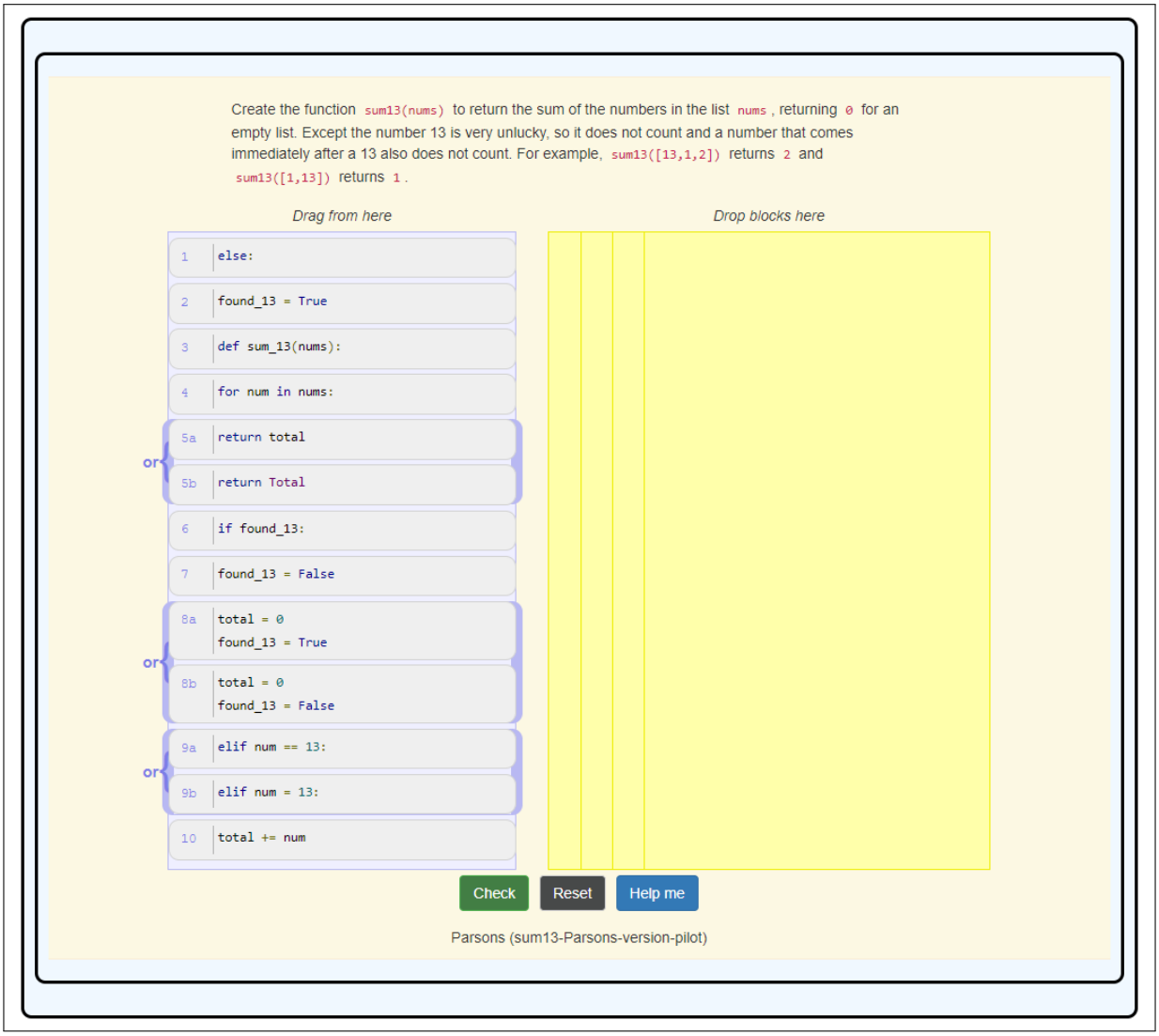

Figure 41: *p3pt* - Defining a function to sum a list of numbers excluding any element that follows the value of 13 (Parsons)

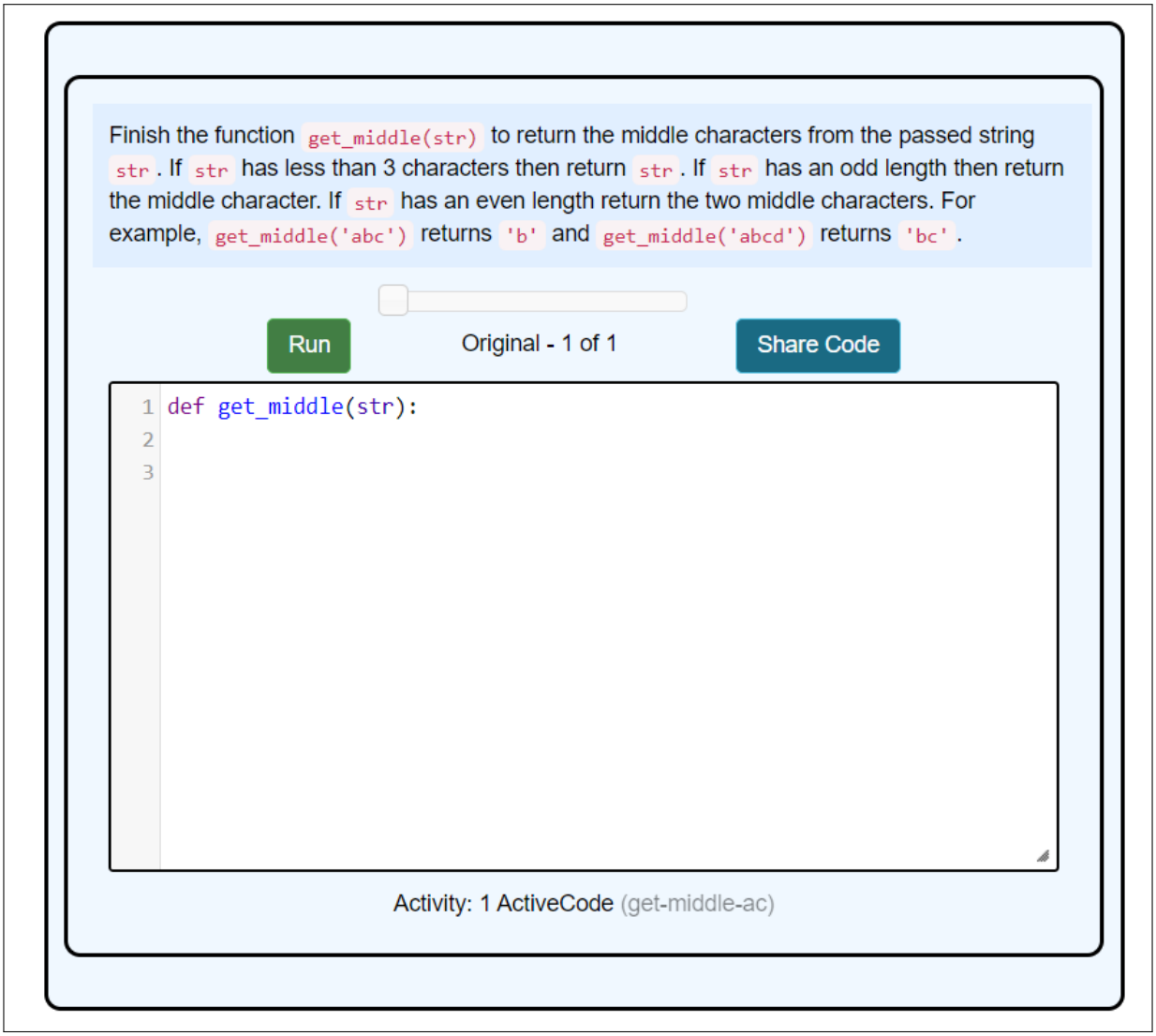

Figure 42: *p3pt* - Defining a function which extracts the middle characters from a string (code writing)

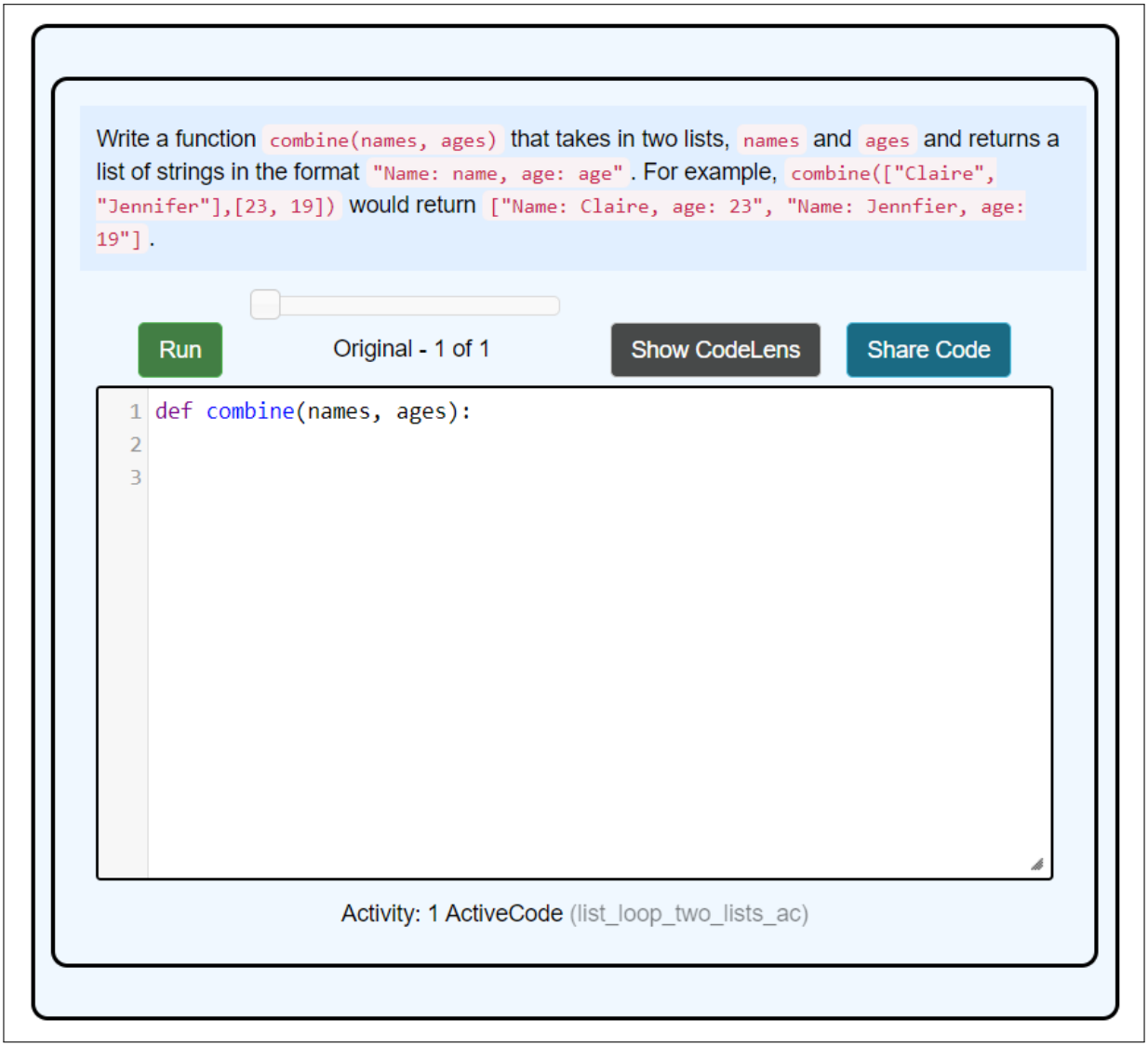

Figure 43: *p3pt* - Defining a function that combines two lists according to requirements (code writing)

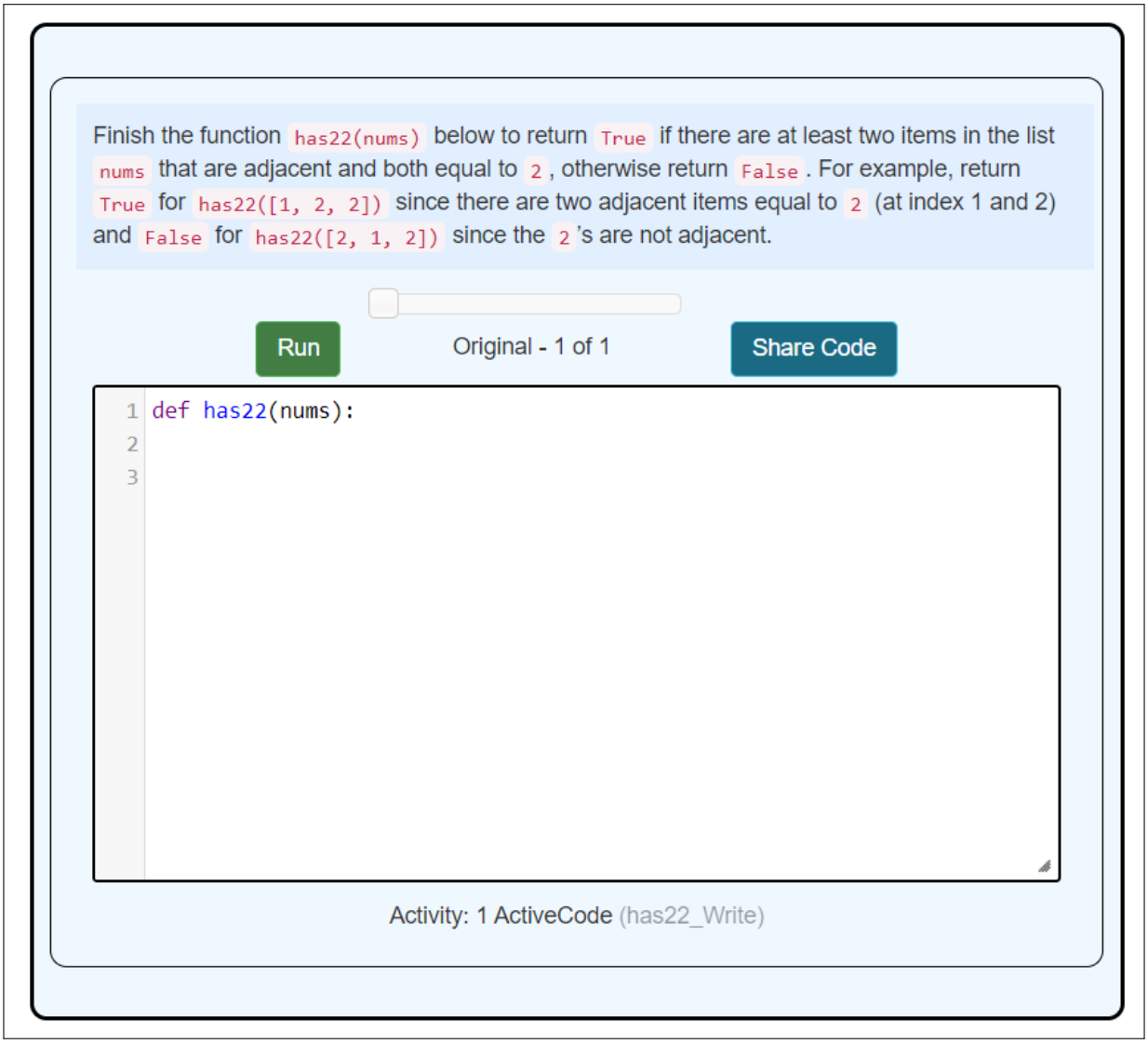

Figure 44: *p3pt* - Defining a function to search a list for adjacent values of two (code writing)

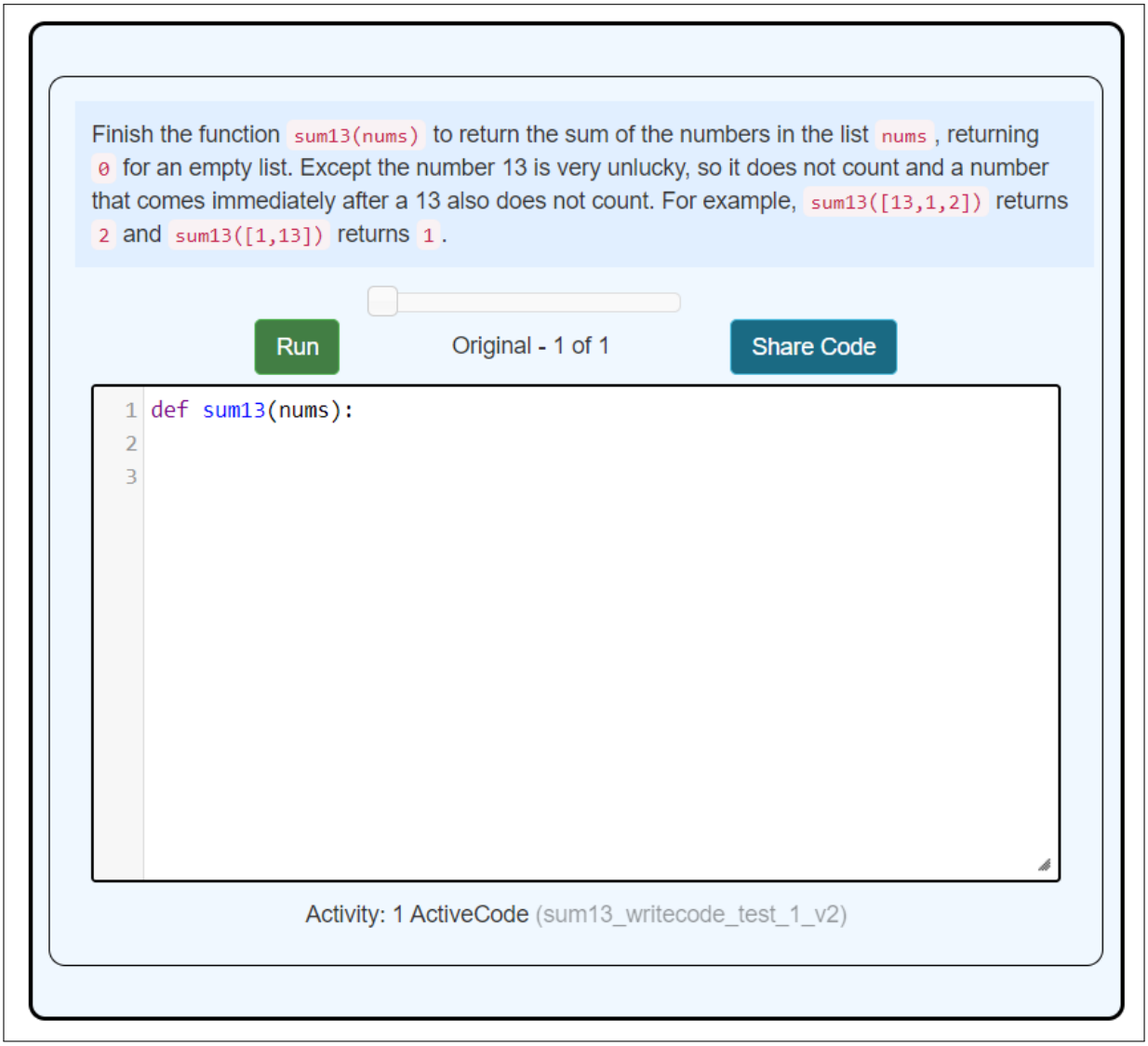

Figure 45: *p3pt* - Defining a function to sum a list of numbers excluding any element that follows the value of 13 (code writing)

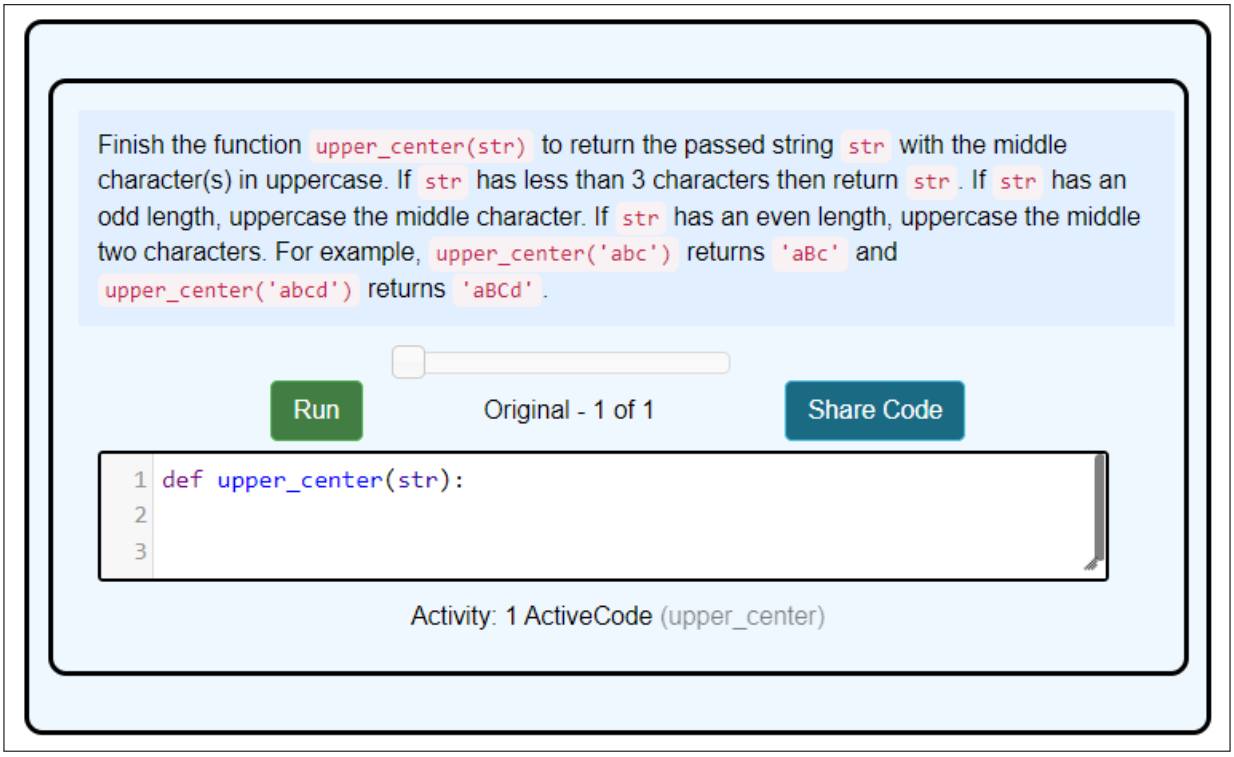

Figure 46: *p3pt* - Posttest based on manipulating and returning a substring

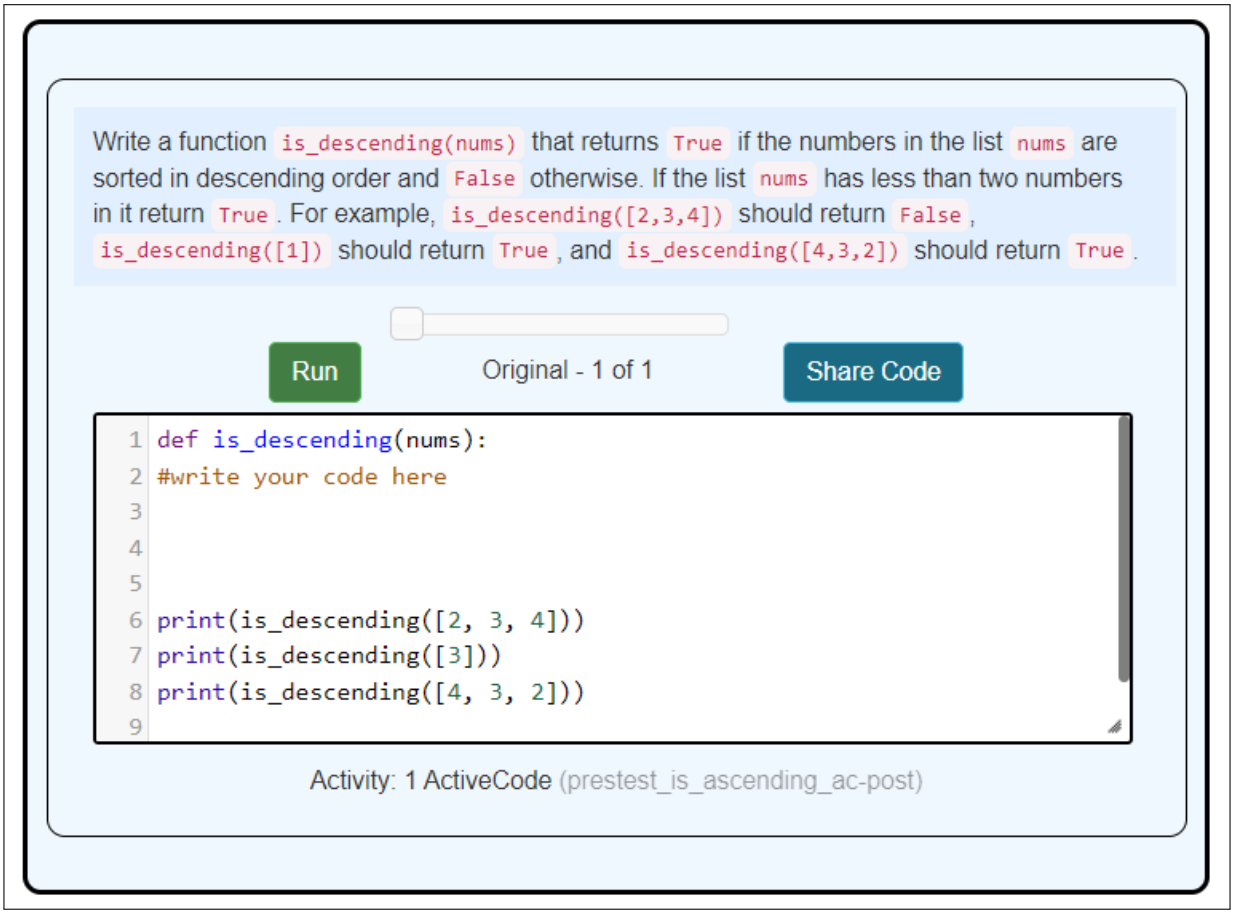

Figure 47: *p3pt* - Posttest based on determining whether a list is in descending order

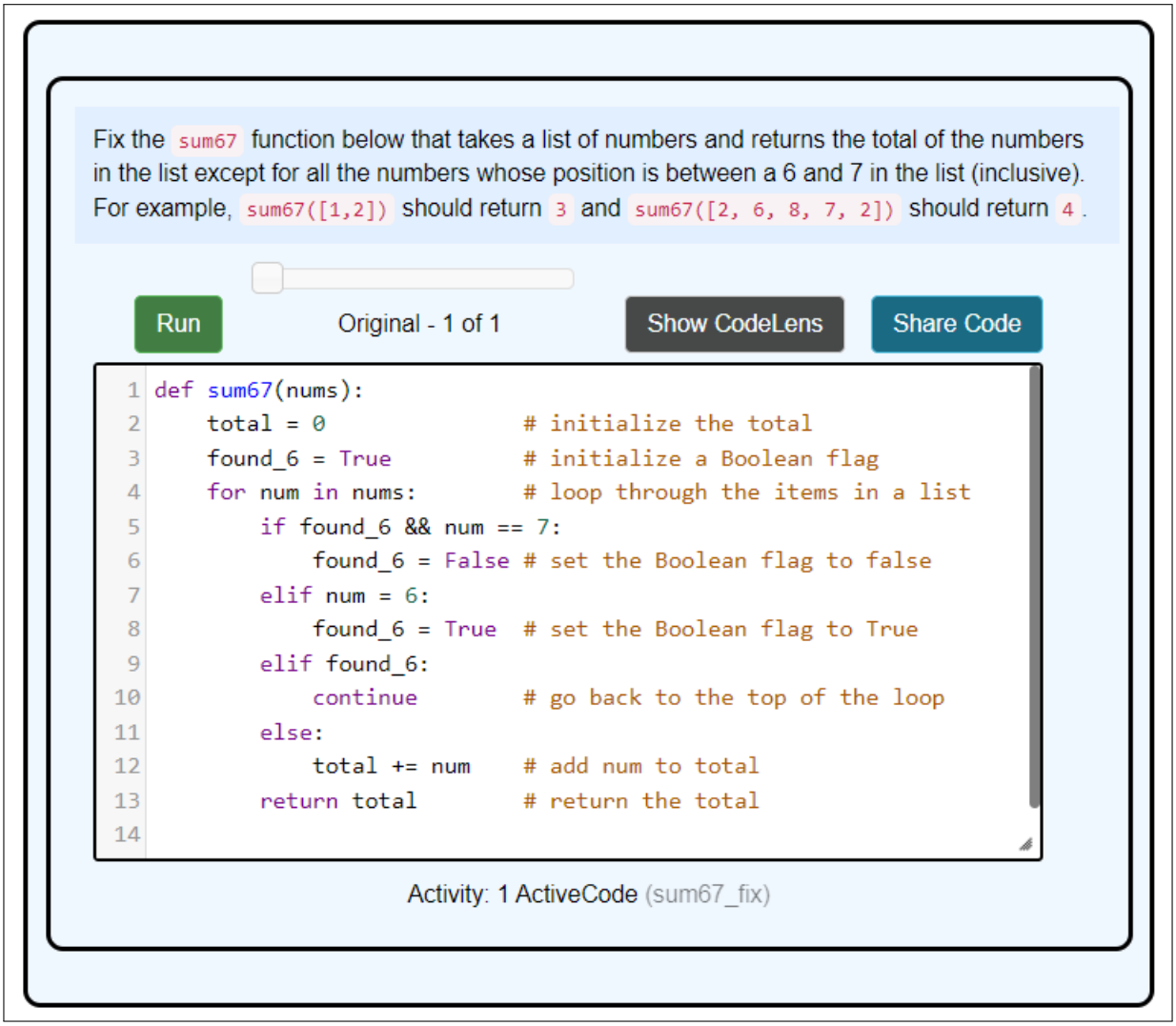

Figure 48: *p3pt* - Posttest based on conditional summation

Figure 49: *p3pt* - Posttest based on combining two lists

A.5 Introduction to Classes

The *classexp* and *classtog* studies involved object-orientated concepts and programming constructs. Brief instruction on how to define classes are provided as part of these respective studies.

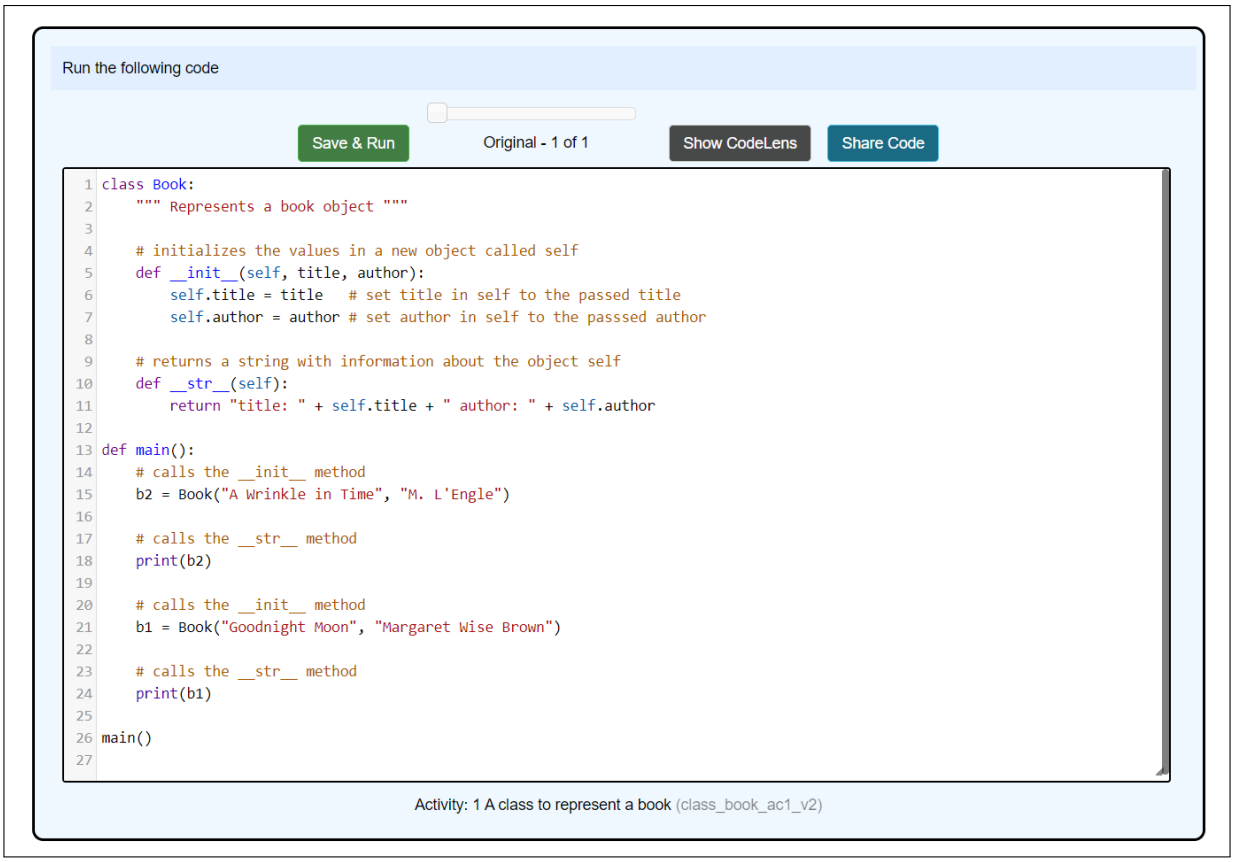

Figure 50: Introduction to classes - Instructions on how to create a class

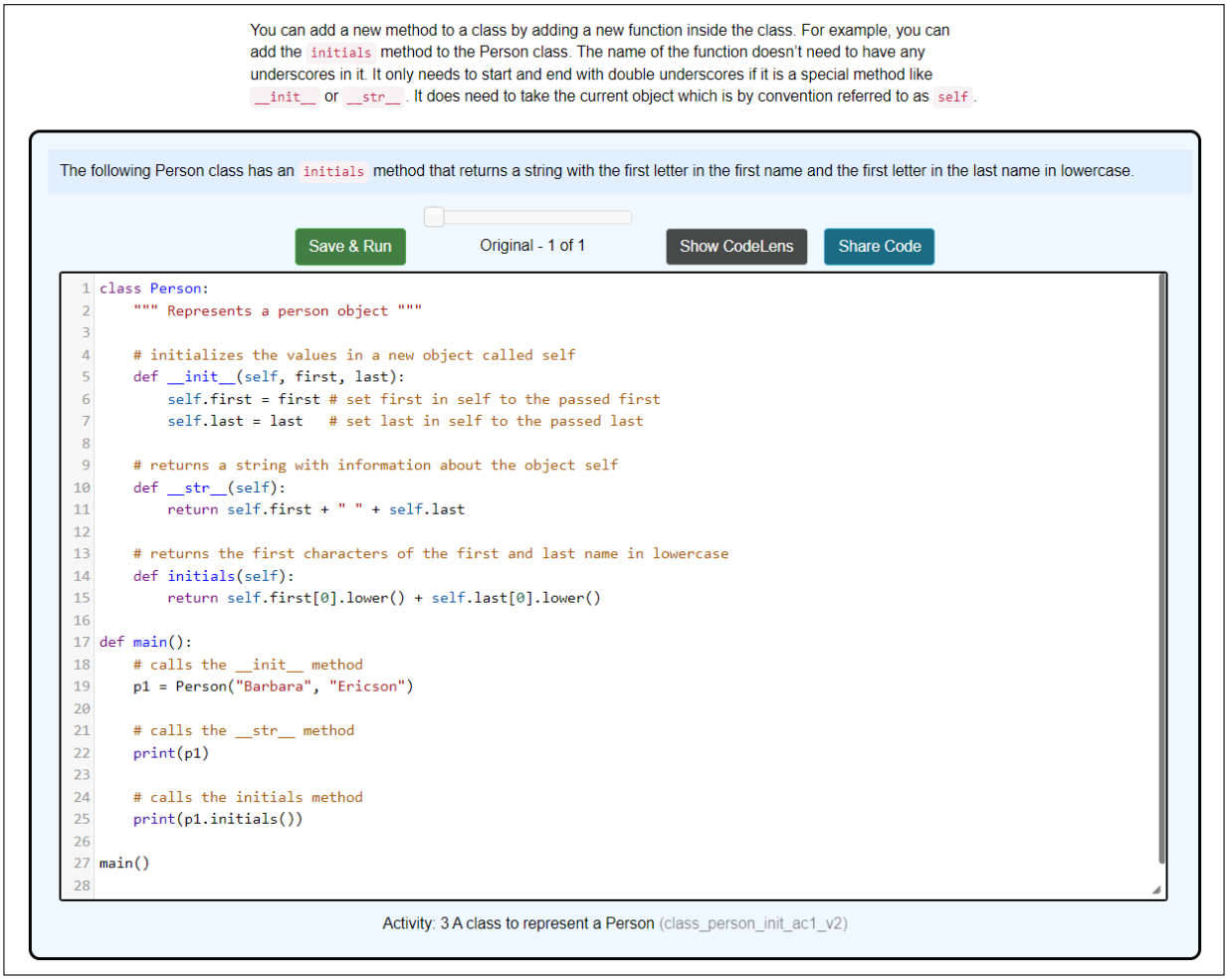

Figure 51: Introduction to classes - worked example

A.6 *class-exp*

Students are tasked with creating classes. One arrangement of the learning materials uses distractors while the other does not.

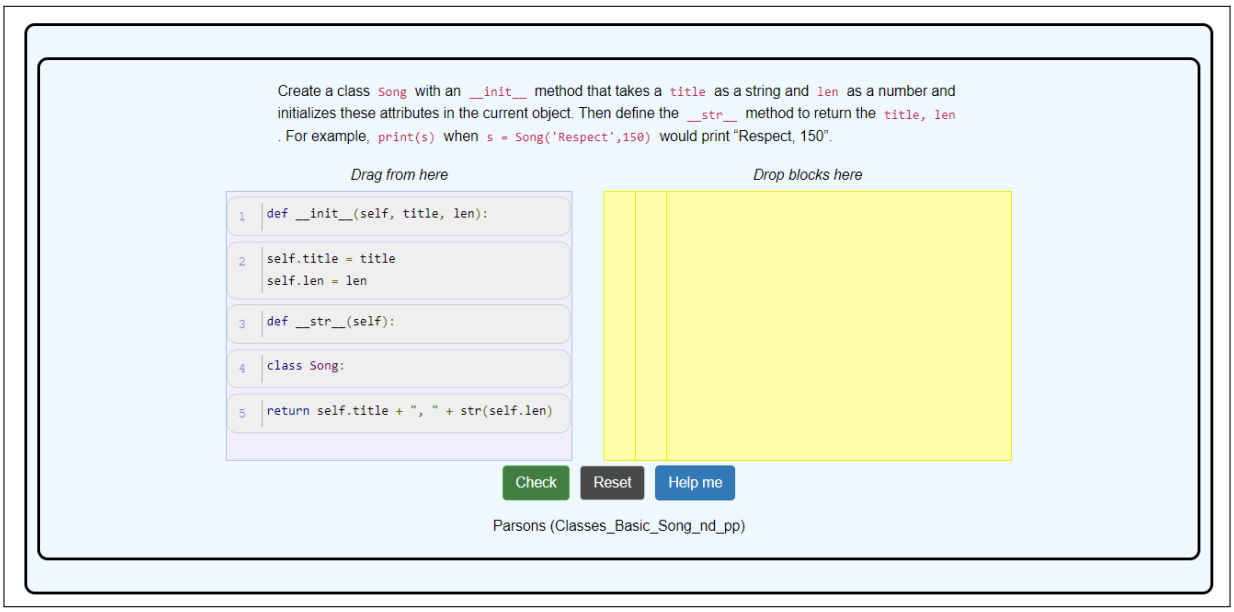

Figure 52: *class-exp* - Define a song class with name and duration attributes

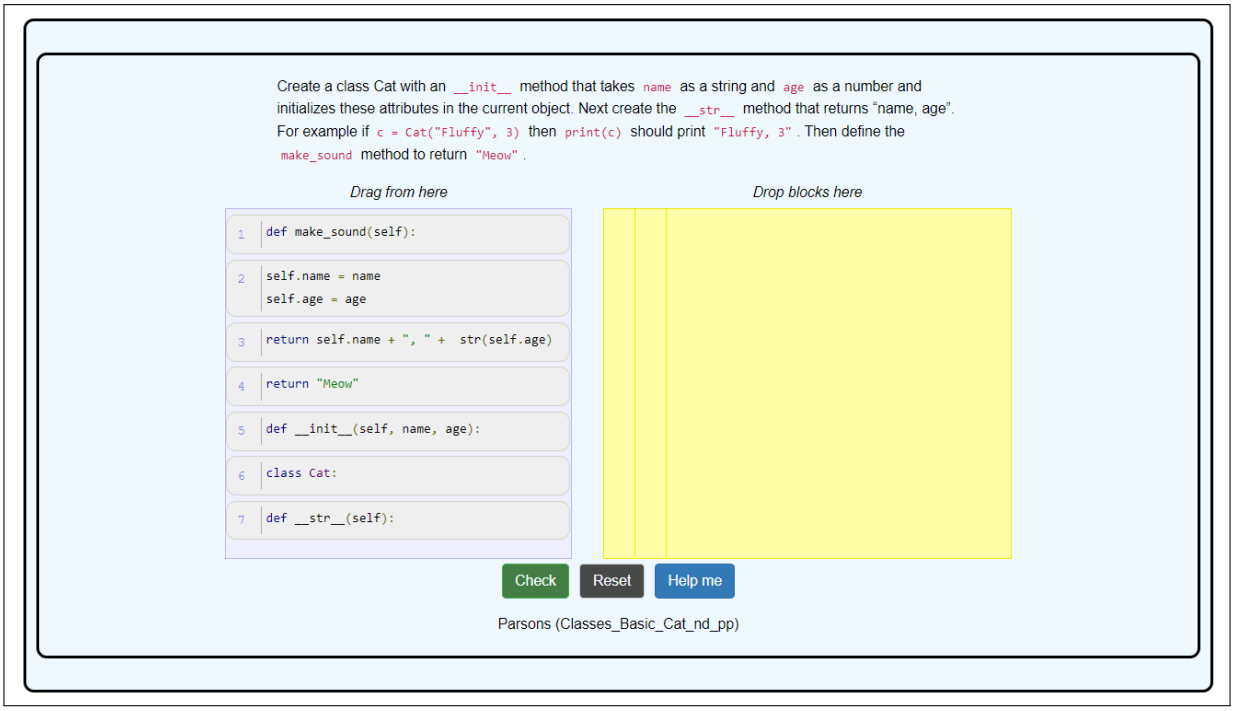

Figure 53: *class-exp* - Define a cat class with name and age attributes and a method for making sound

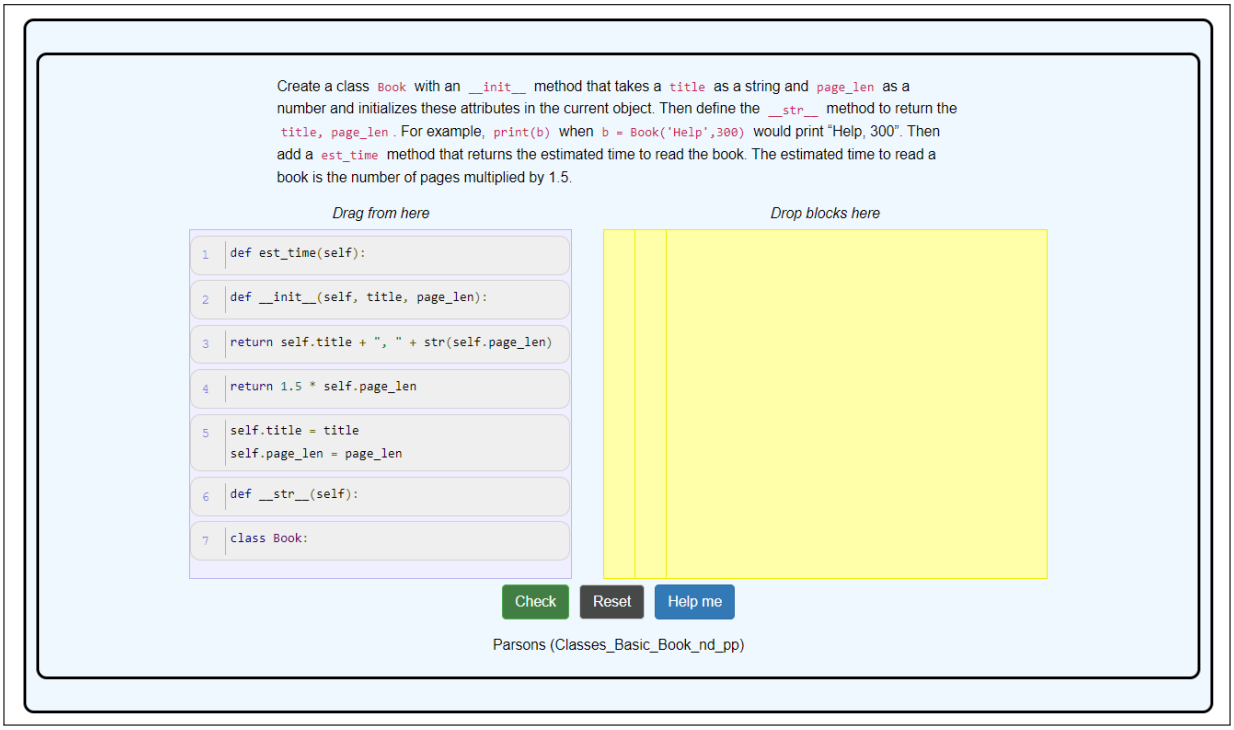

Figure 54: *class-exp* - Define a book class with title and length attributes and a string-conversion method

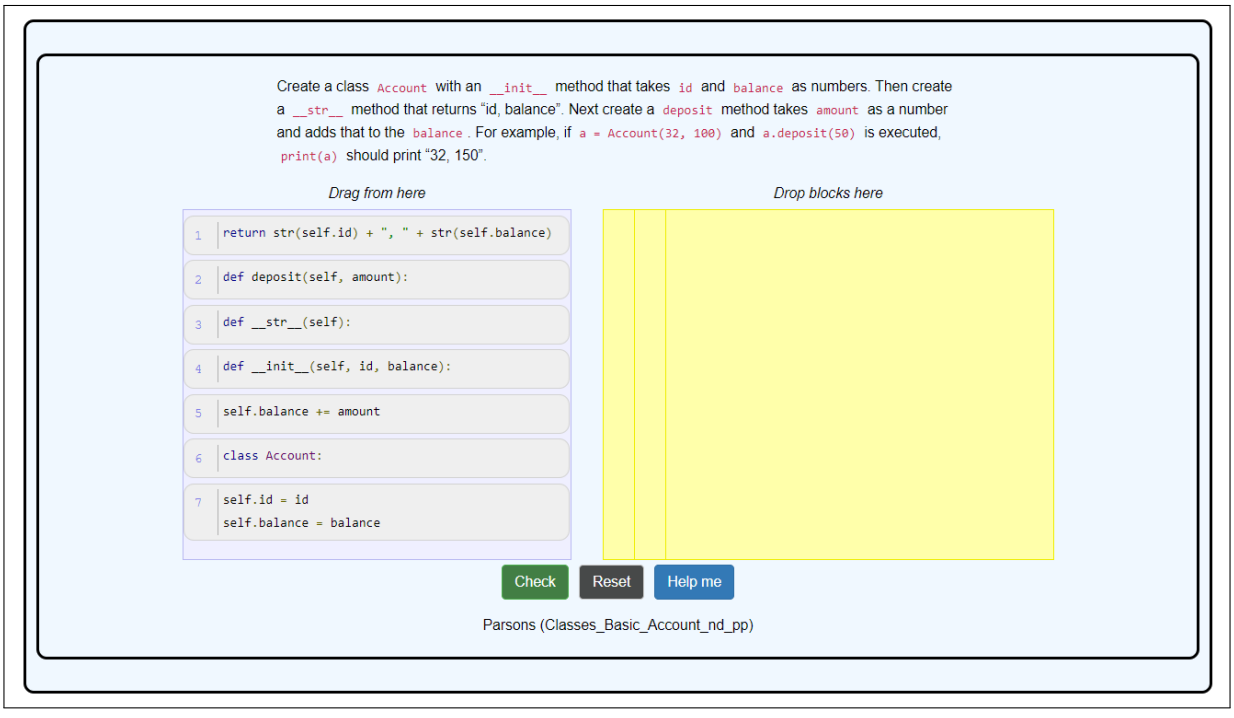

Figure 55: *class-exp* - Define a bank account class with identifier and balance attributes and a deposit method

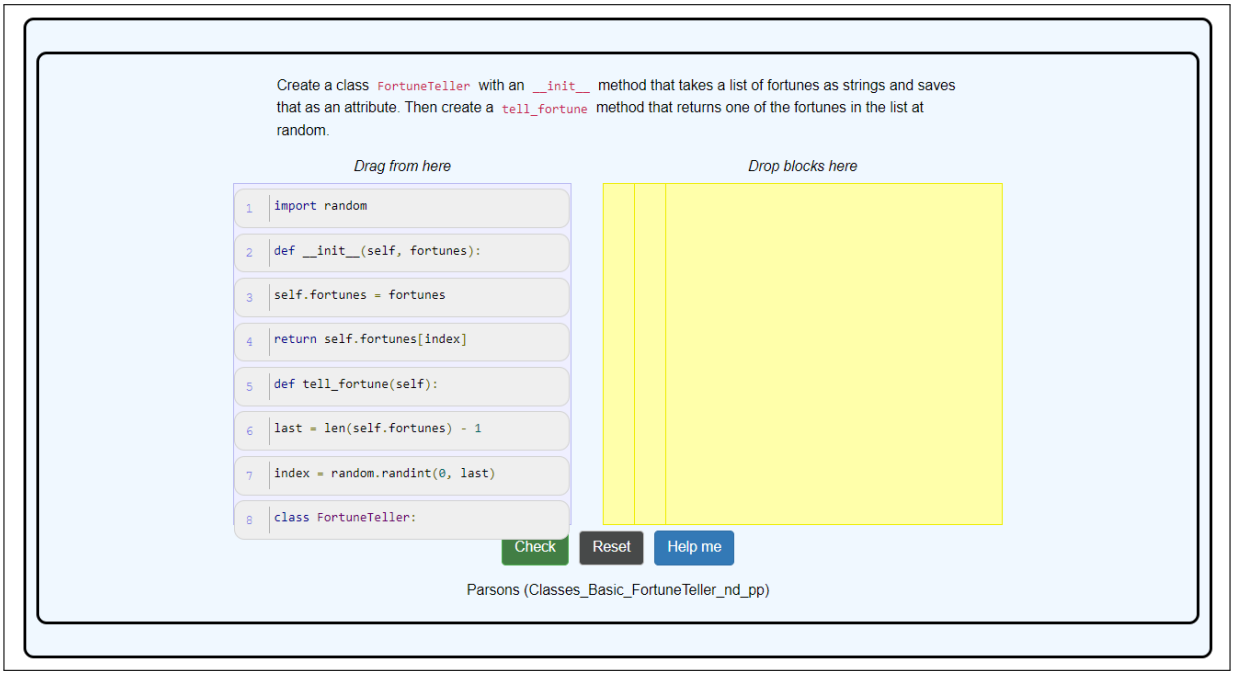

Figure 56: *class-exp* - Define a fortune teller class which randomly selects a fortune from a predefined list

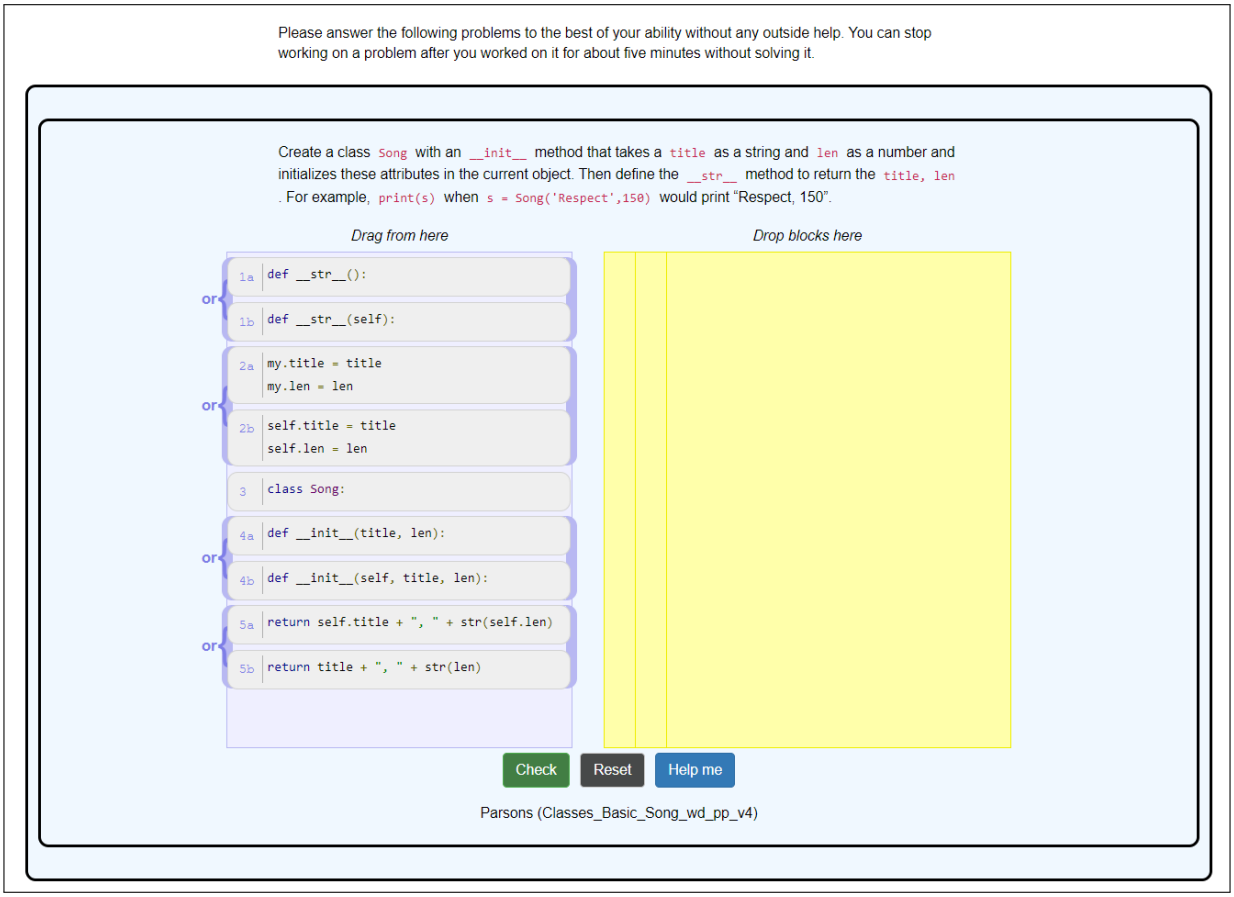

Figure 57: *class-exp* - Define a song class with name and duration attributes (with distractors)

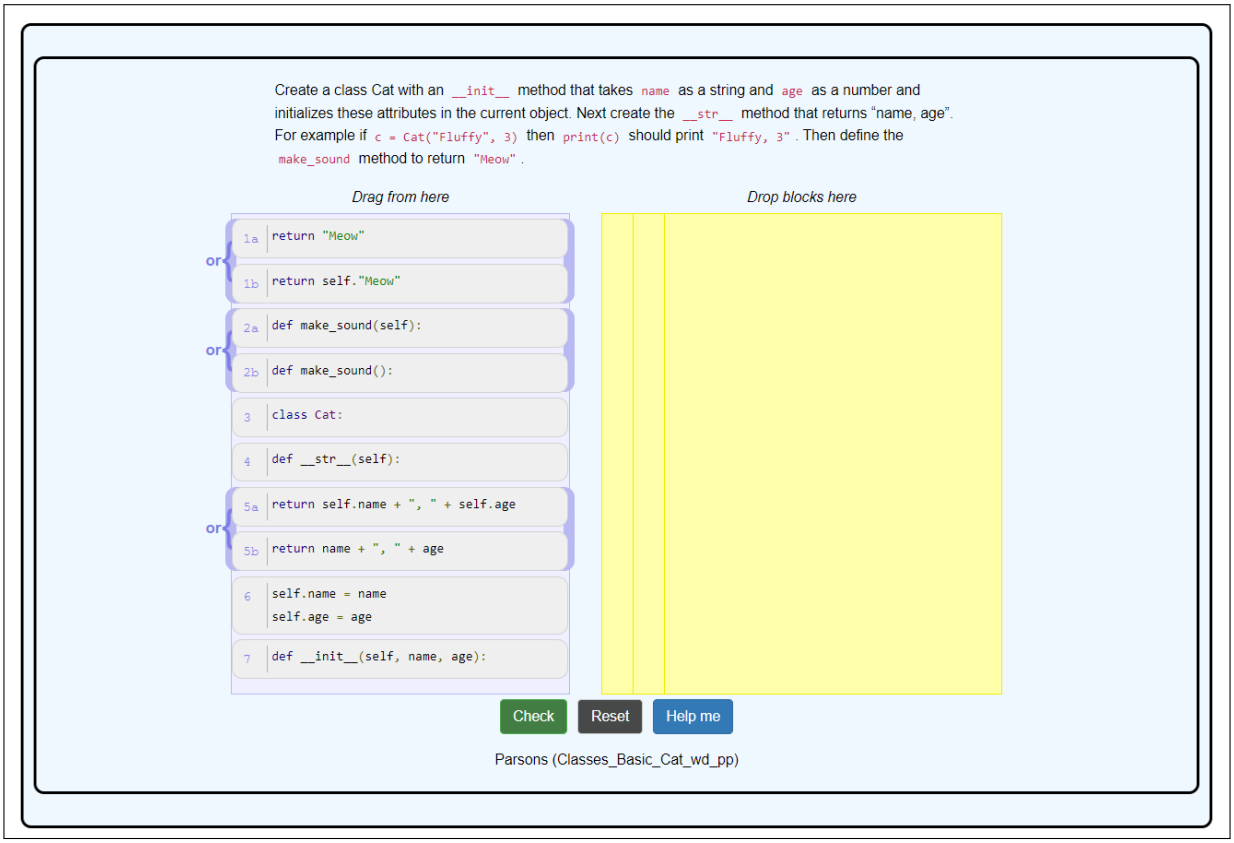

Figure 58: *class-exp* Define a cat class with name and age attributes and a method for making sound (with distractors)

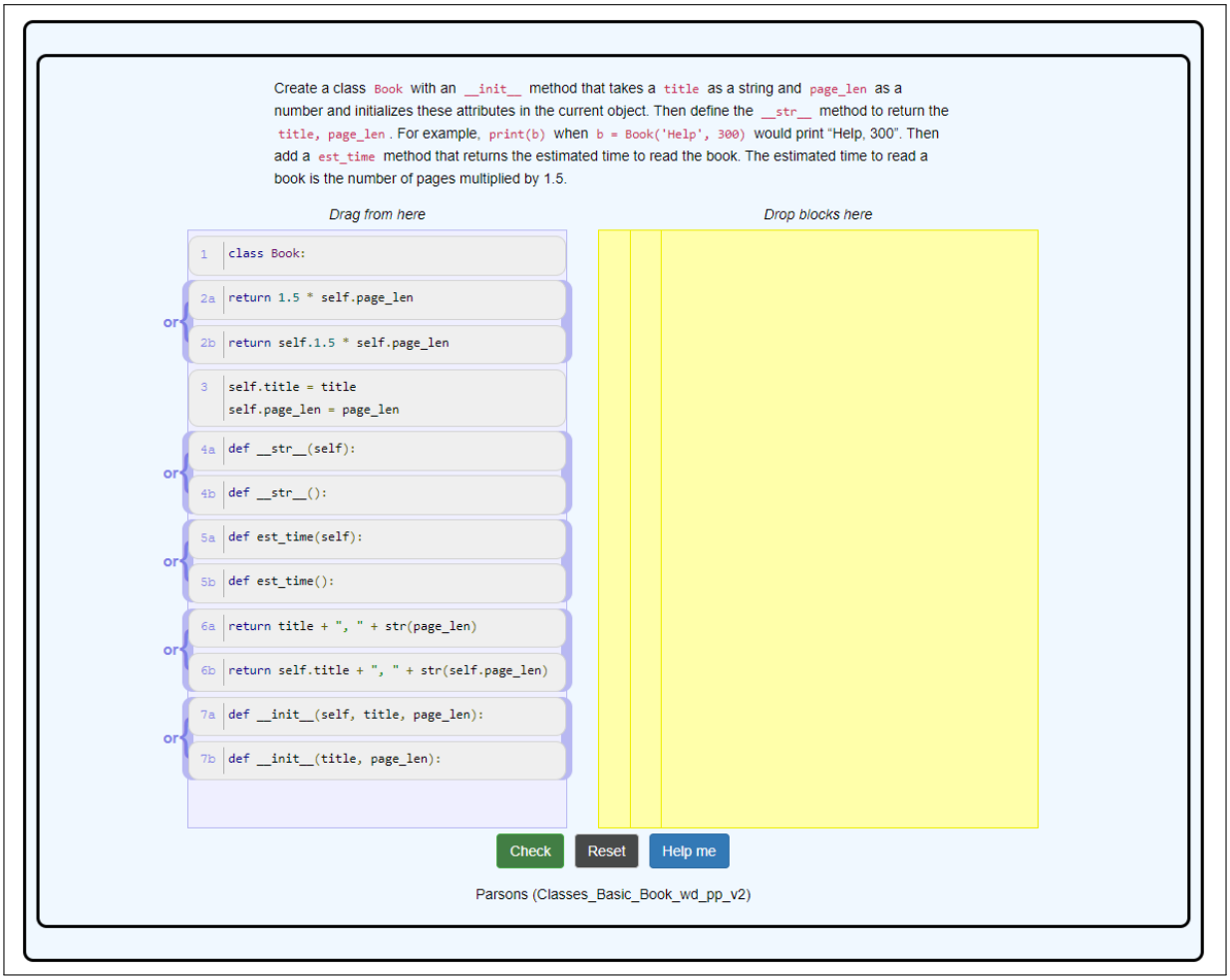

Figure 59: *class-exp*- Define a book class with title and length attributes and with a string-conversion method (with distractors)

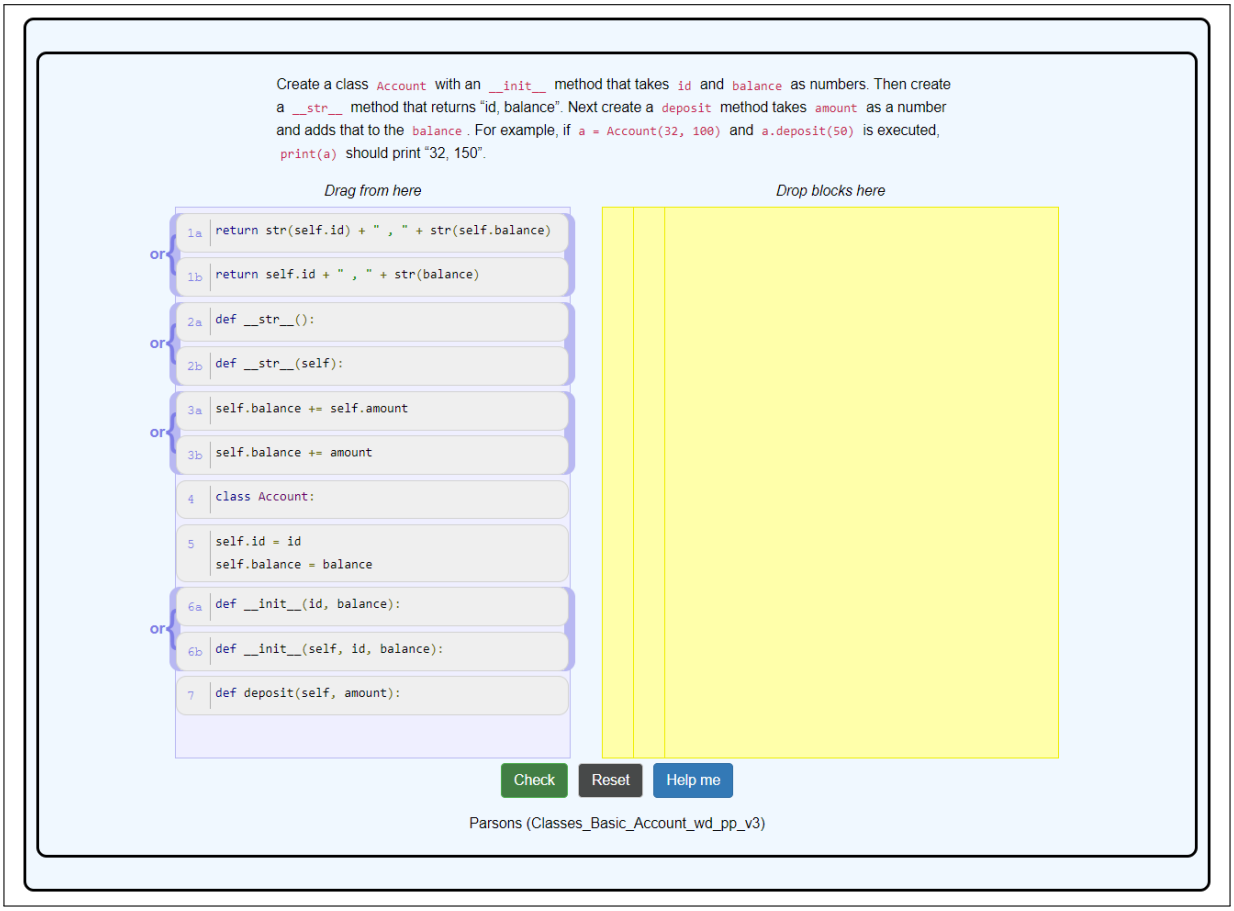

Figure 60: *class-exp* - Define a bank account class with identifier and balance attributes and a deposit method (with distractors)

Figure 61: *class-exp* - Define a fortune teller class which randomly selects a fortune from a predefined list *class-exp* (with distractors)

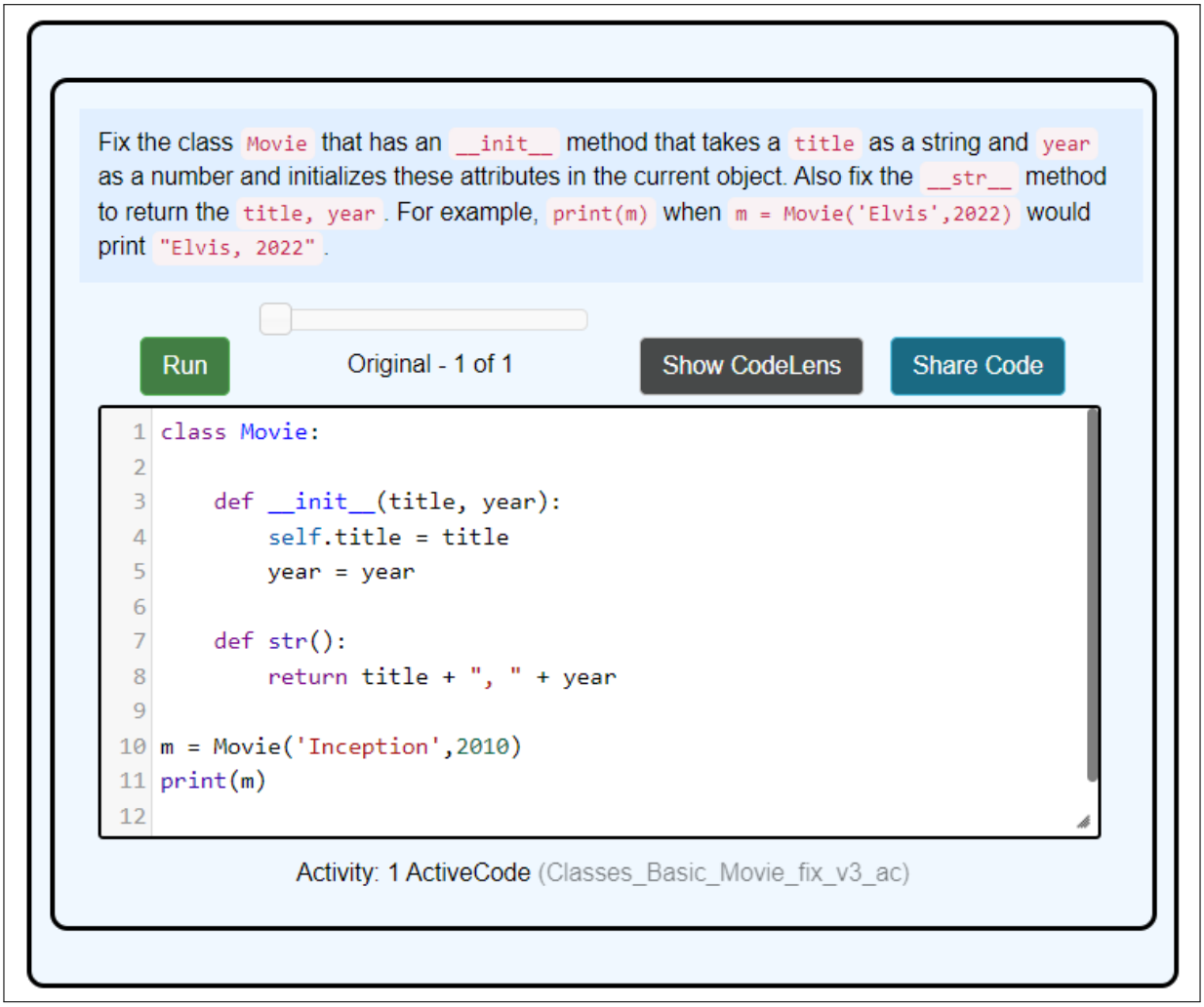

Figure 62: *class-exp* - Posttest task for defining a movie class

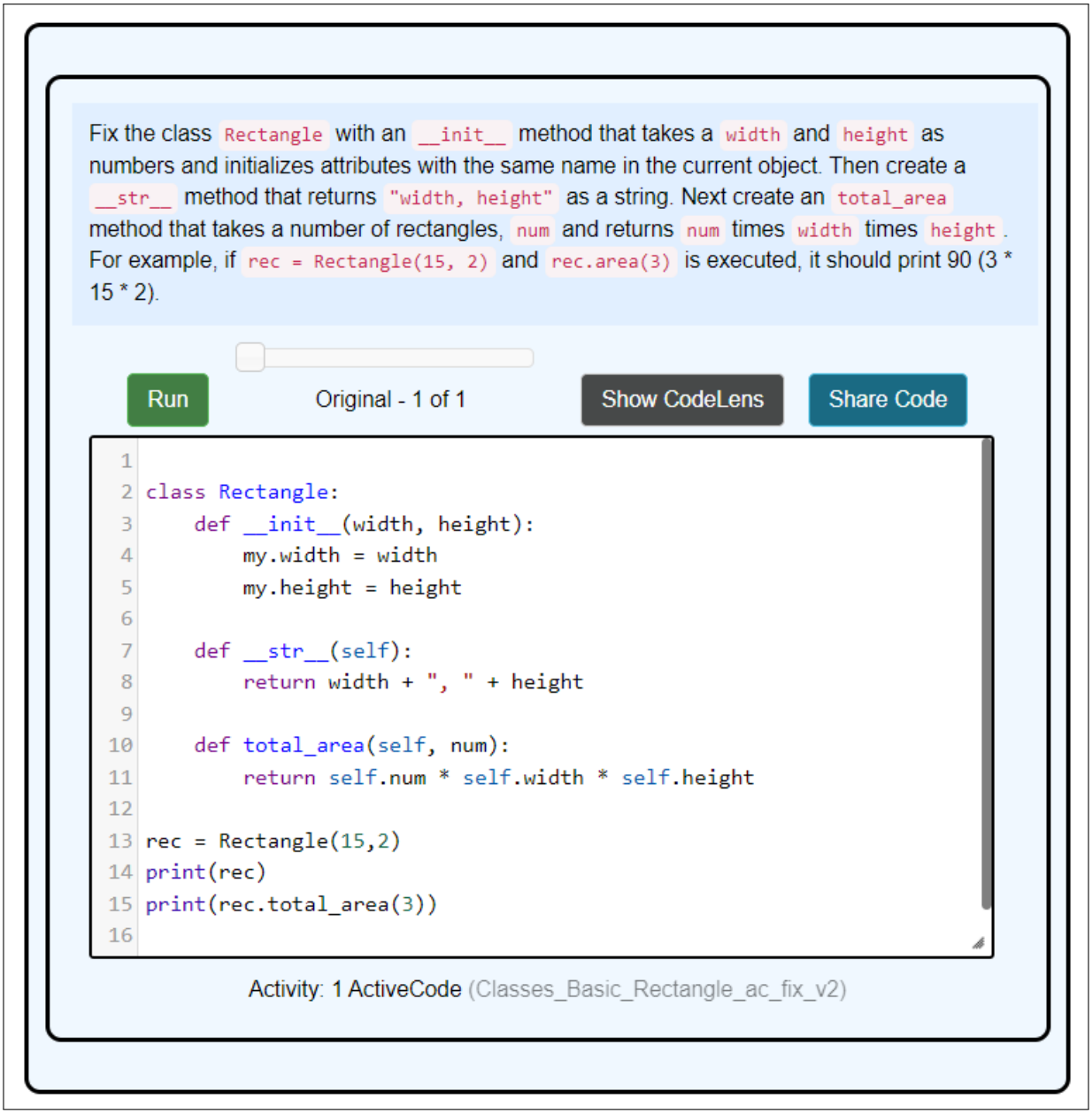

Figure 63: *class-exp* - Posttest task for defining a rectangle class with a string-conversion method

Figure 64: *class-exp* - Posttest task for defining a horse class and a method for making sound

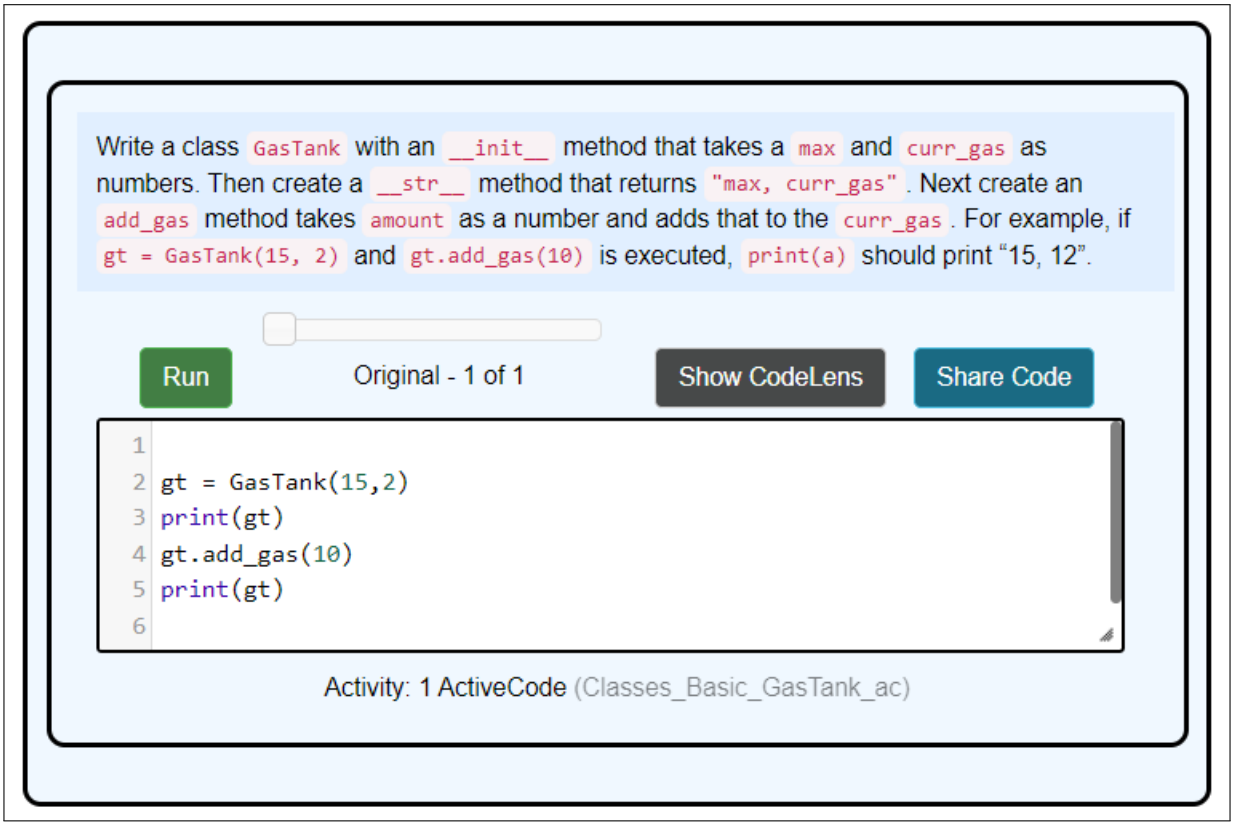

Figure 65: *class-exp* - Posttest task for defining a gas tank class and a method to add gas

Figure 66: *class-exp* - Posttest task for defining a dice class with a given number of sides and a roll method

A.7 *class-tog*

Similar to the previous activity, students are tasked with defining classes, with the same practice and posttest items. However, in this study, students are able to switch freely between a Parsons problem and code using a drop-down menu throughout the practice activities.

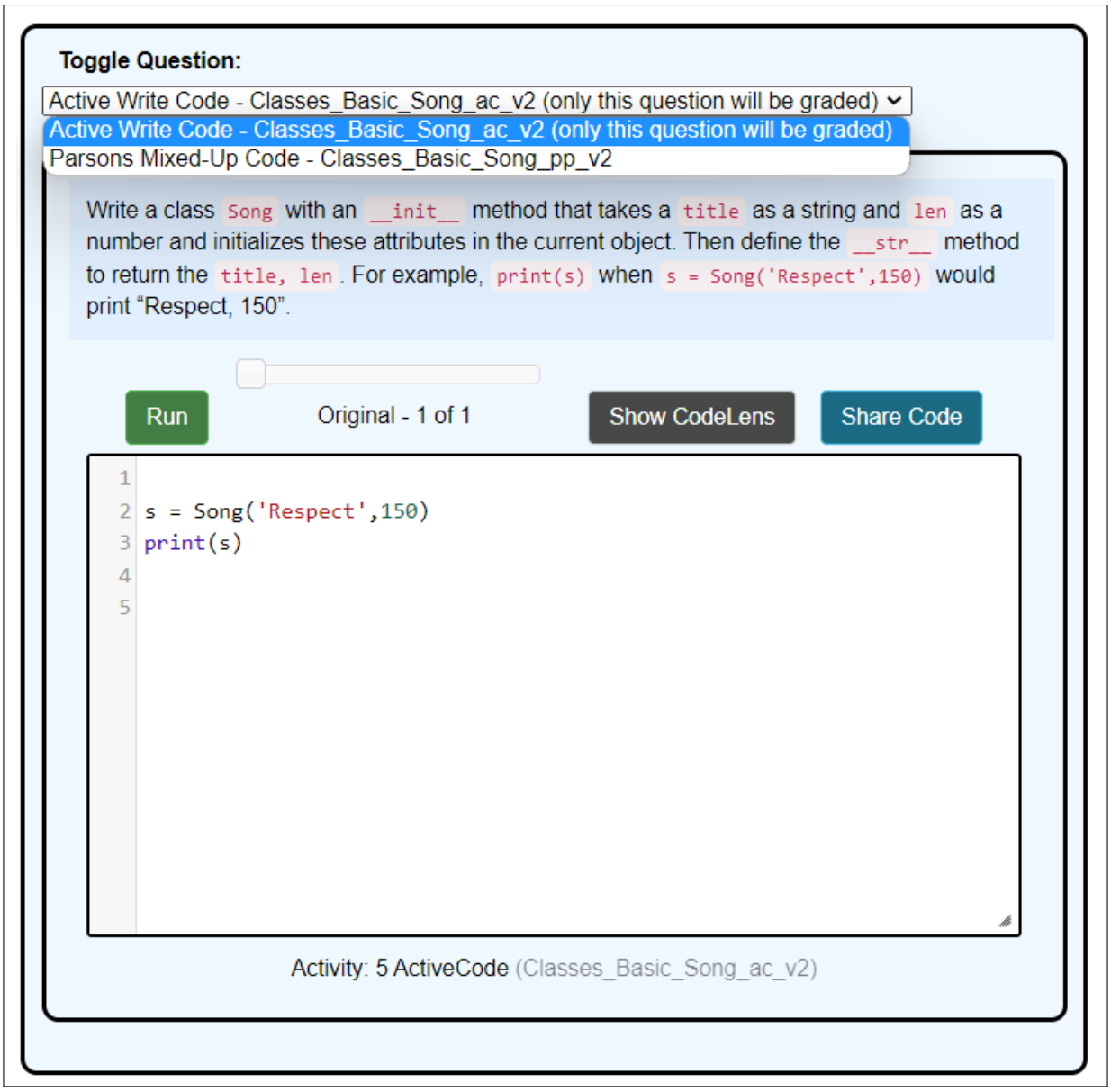

Figure 67: Students have the ability to toggle between code and a Parsons problem

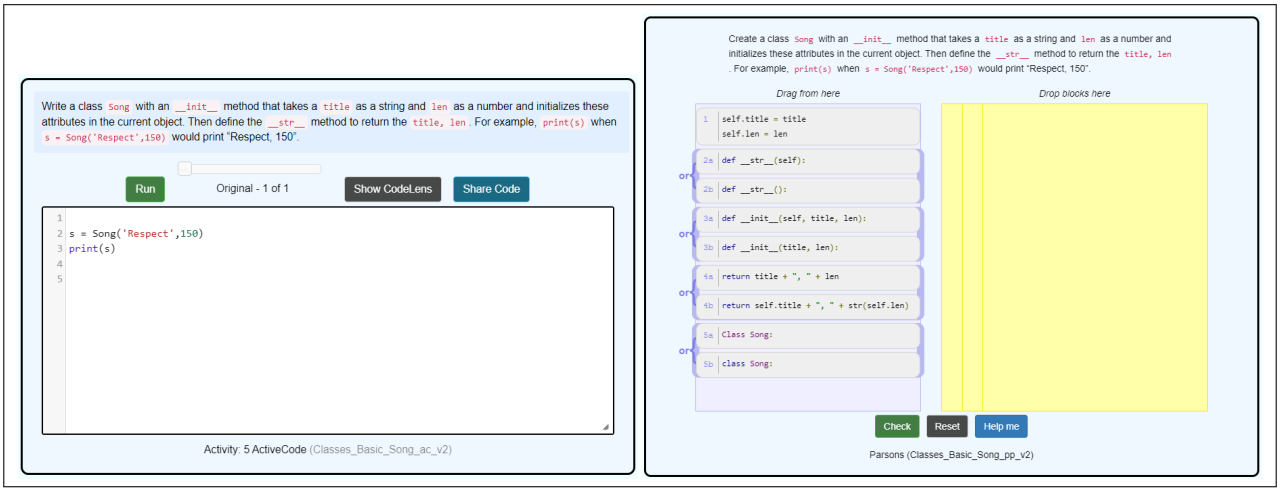

Figure 68: *class-tog* - Define a song class with name and duration attributes

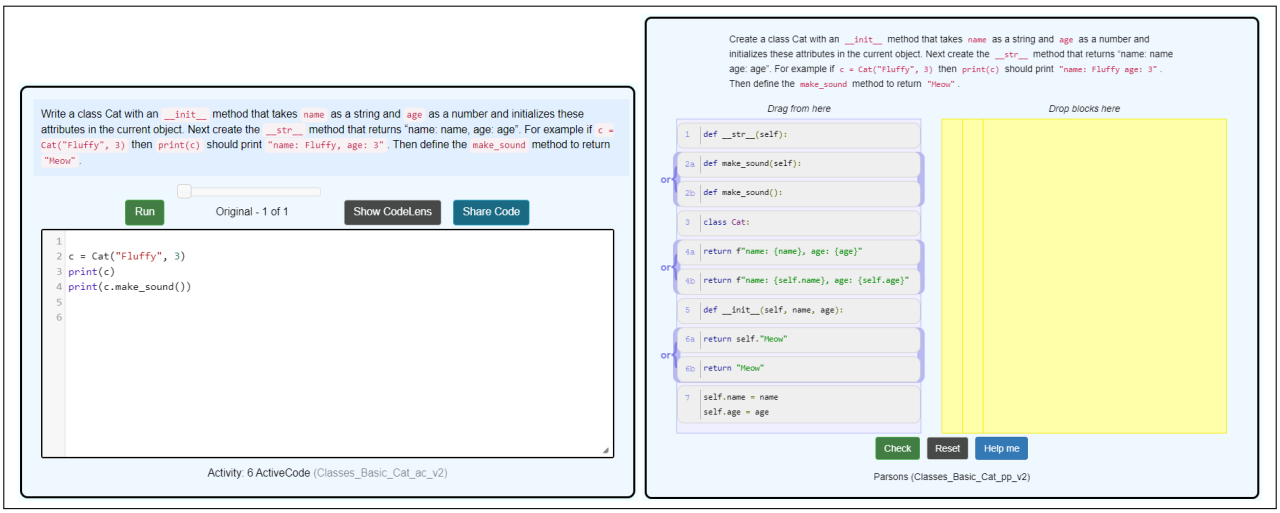

Figure 69: *class-tog* - Define a cat class with name and age attributes and with a method for making sound

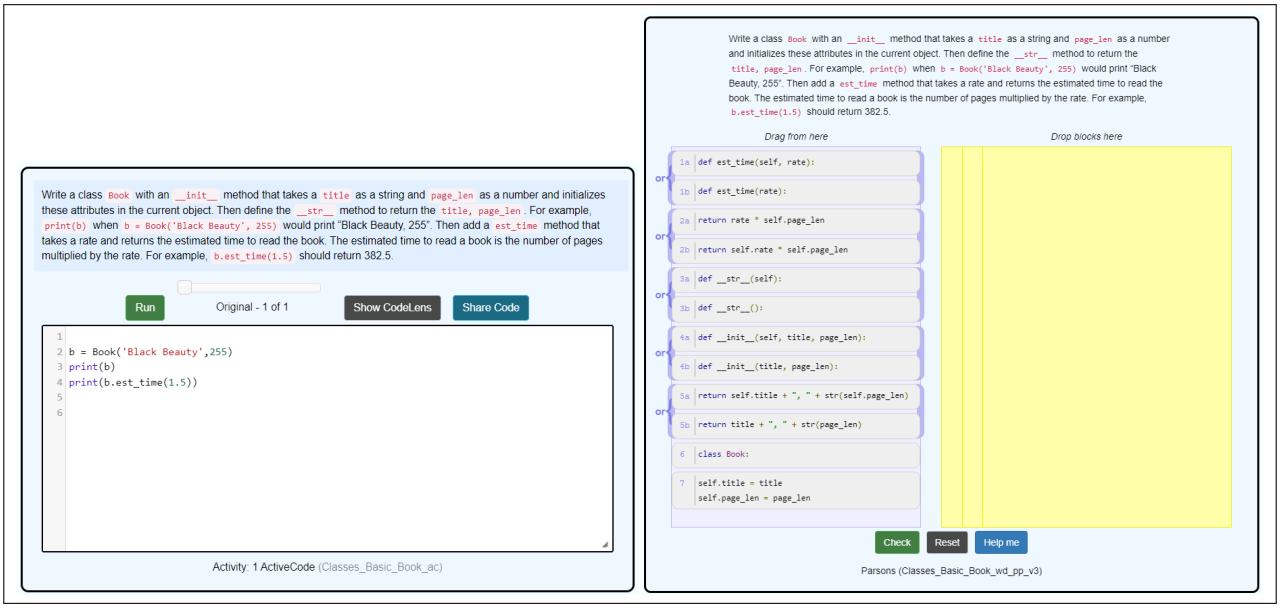

Figure 70: *class-tog* - Define a book class with title and length attributes and a string-conversion method

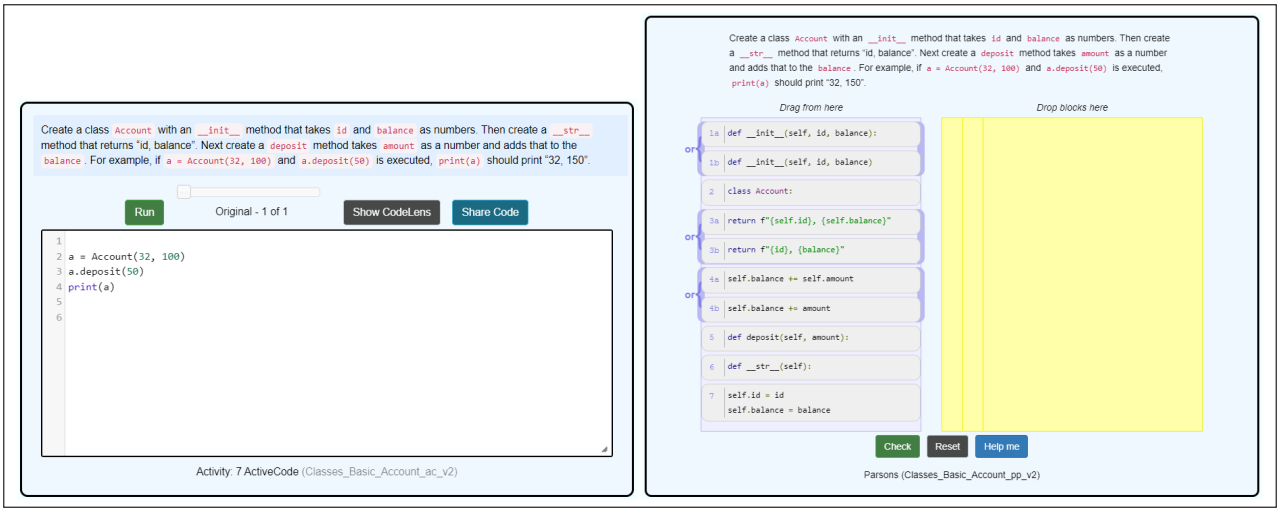

Figure 71: *class-tog* - Define a bank account class with identifier and balance attributes and a deposit method

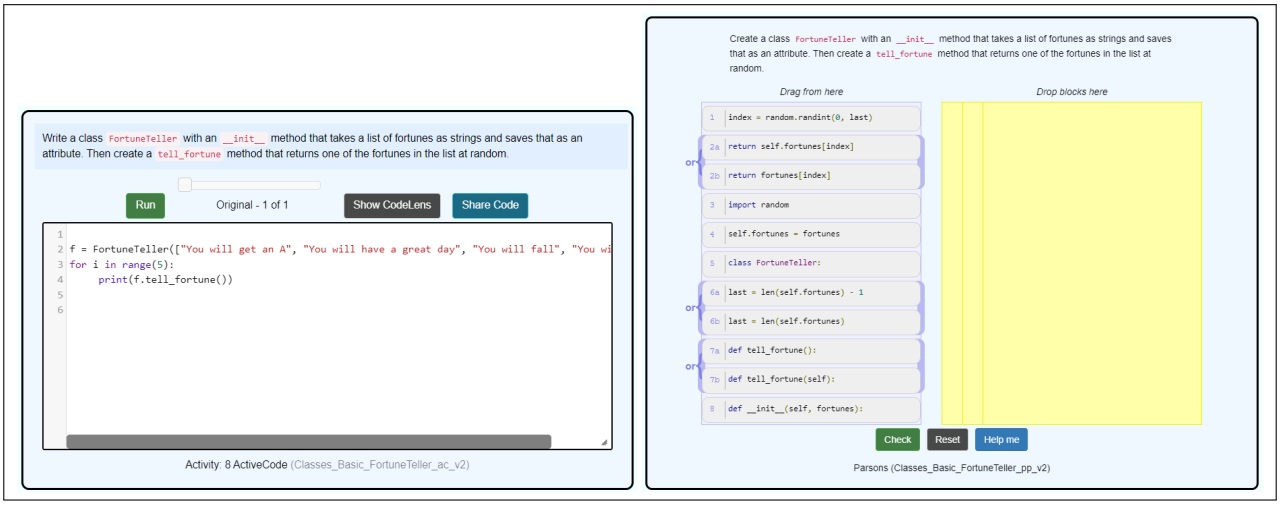

Figure 72: *class-tog* - Define a fortune teller class which randomly selects a fortune from a predefined list (with distractors)

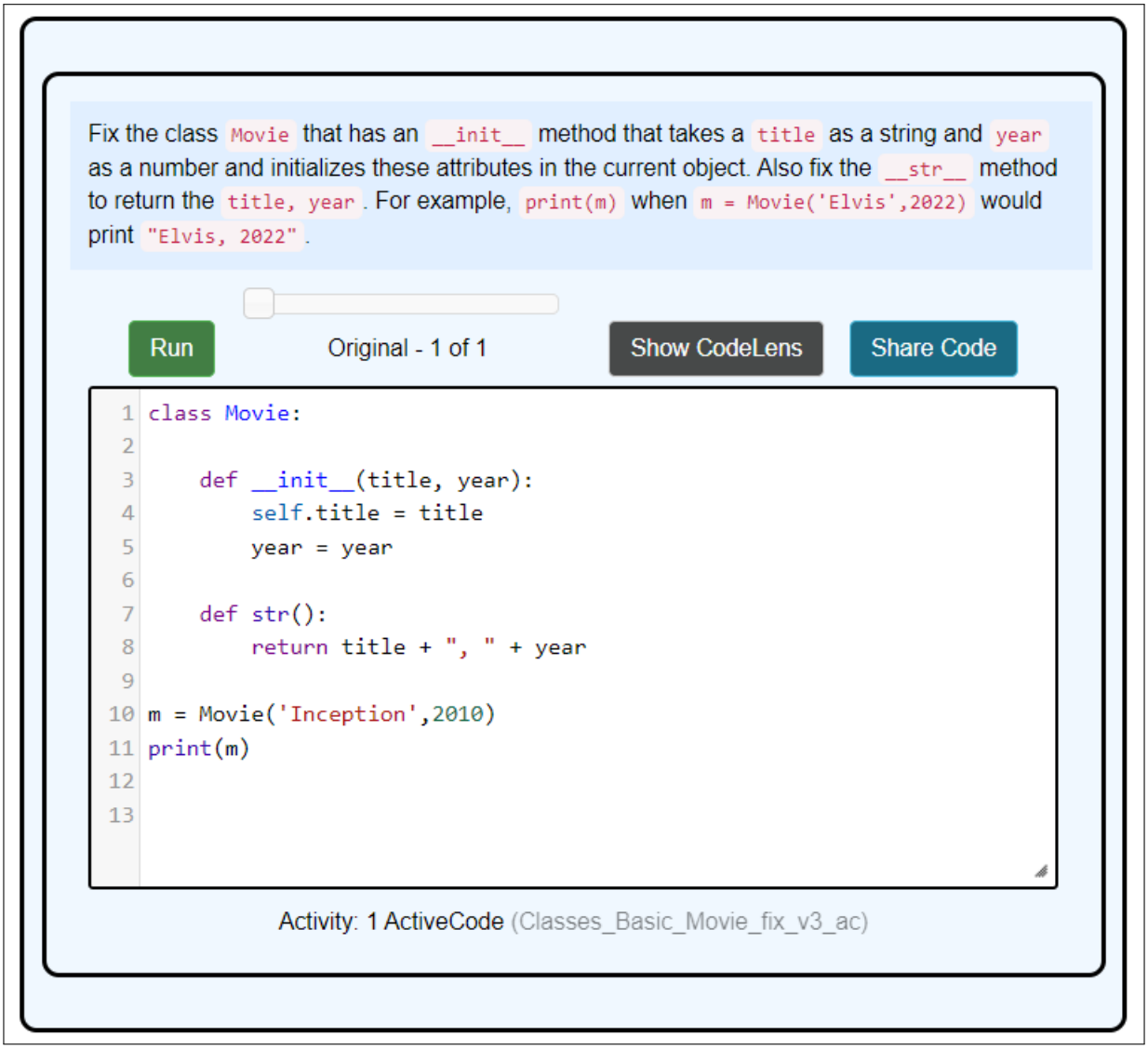

Figure 73: *class-tog* - Posttest task for defining a movie class

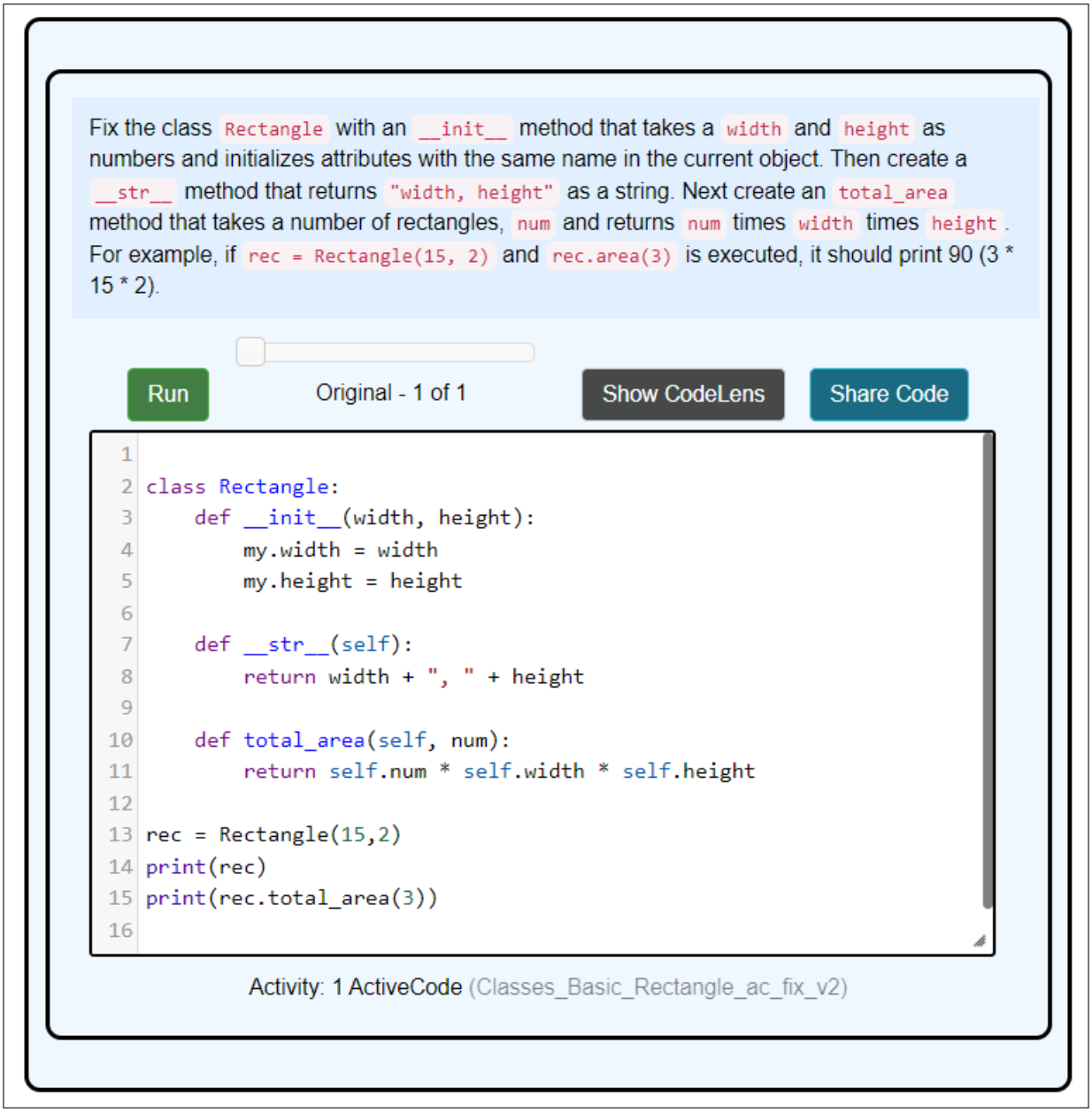

Figure 74: *class-tog* - Post-posttest task for defining a rectangle class and string-conversion method

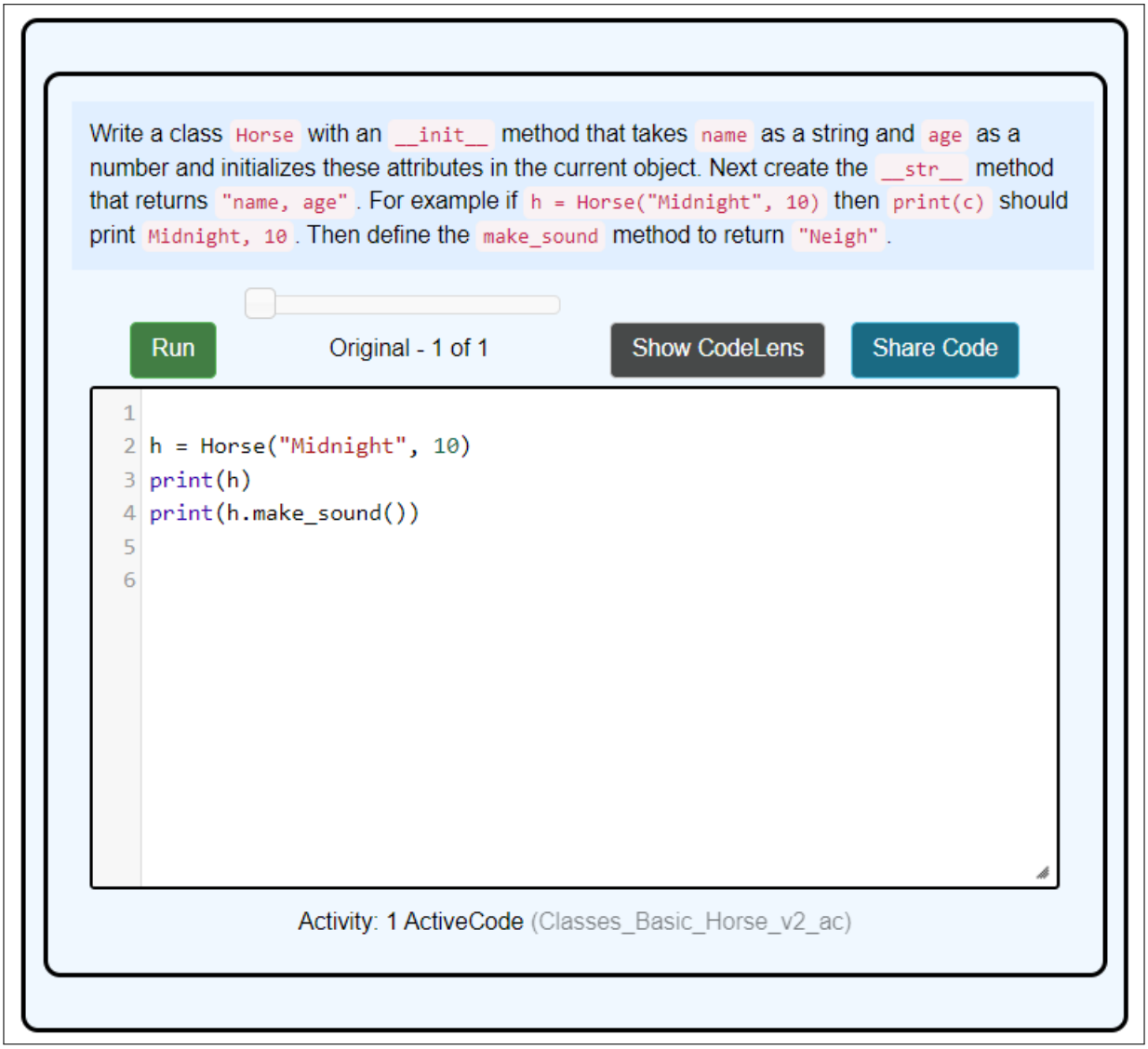

Figure 75: *class-tog* - Post-posttest task for defining a horse class and a method for making sound

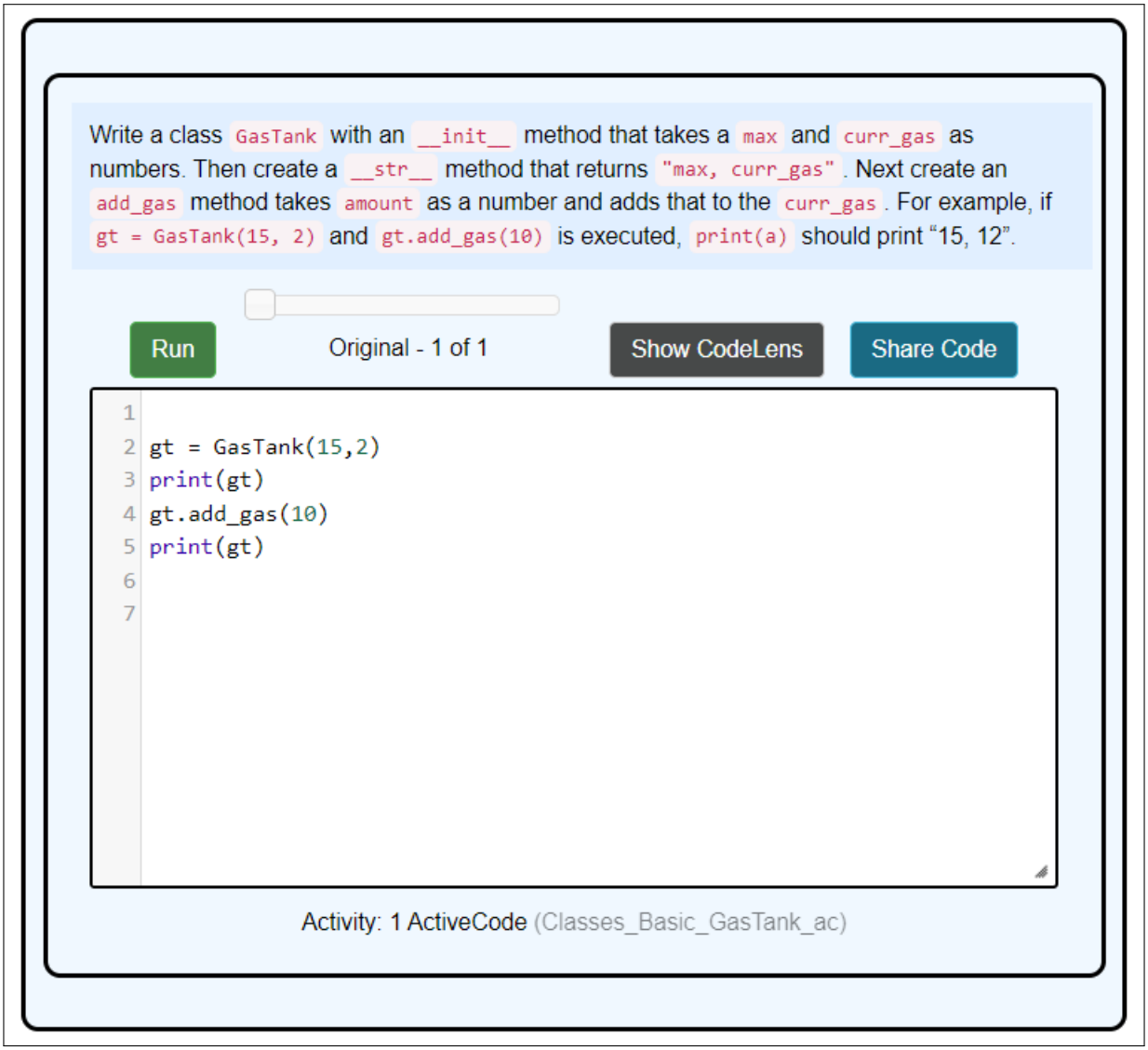

Figure 76: *class-tog* - Posttest task for defining a gas tank class and a method for adding gas

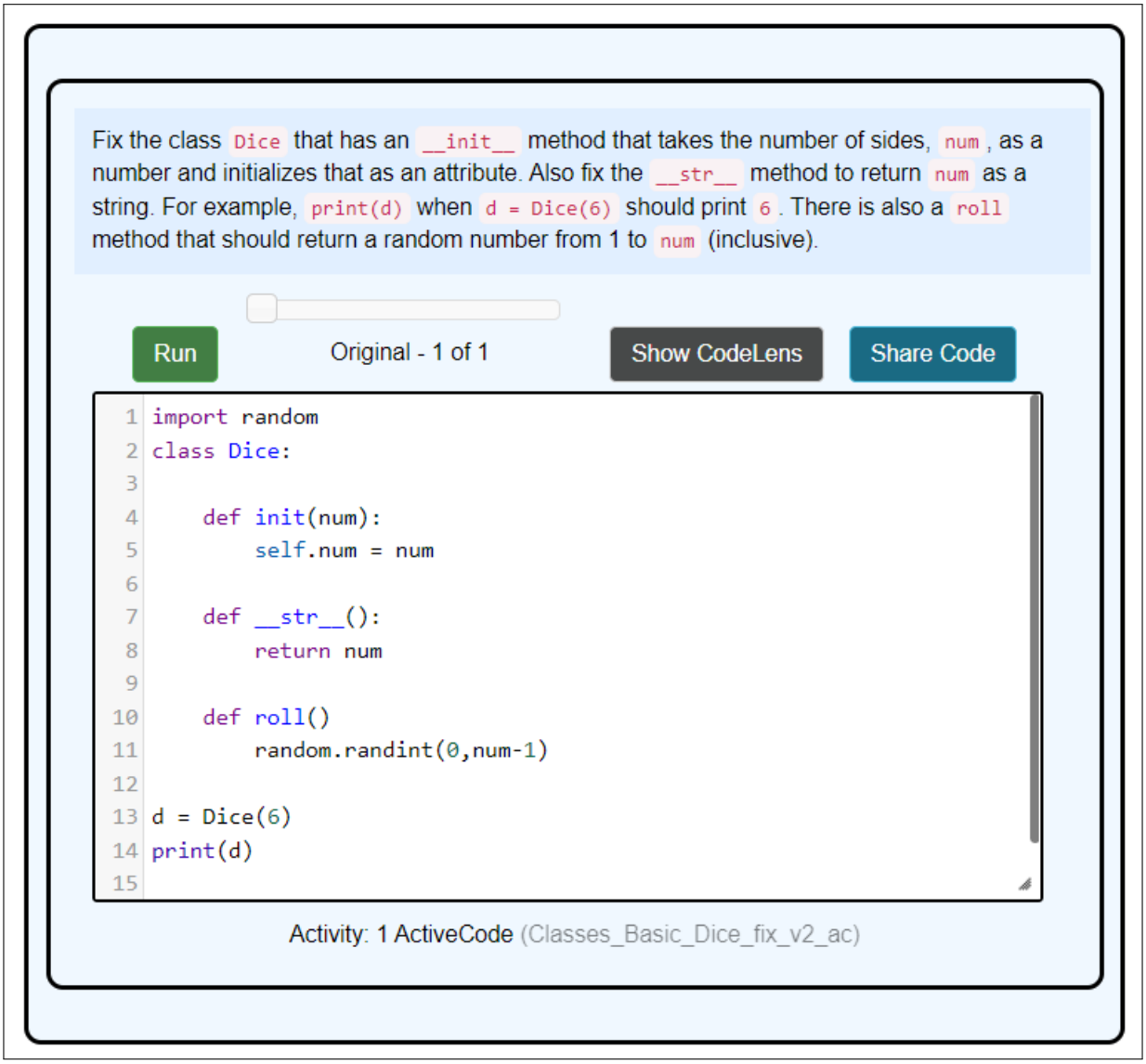

Figure 77: *class-tog* - Posttest task for defining a dice class with a given number of sides and a roll method

A.8 *p3dnd*

The *p3dnd* study is similar to *p3pt* but is intended to have harder practice and posttest problems. The two conditions are Parsons problems with and without distractors.

Figure 78: *p3dnd* - return a character based on what is at the beginning and end of a string (with distractors)

Figure 79: *p3dnd* - return true if a string contains "b*b" where '*' can be any character (with distractors)

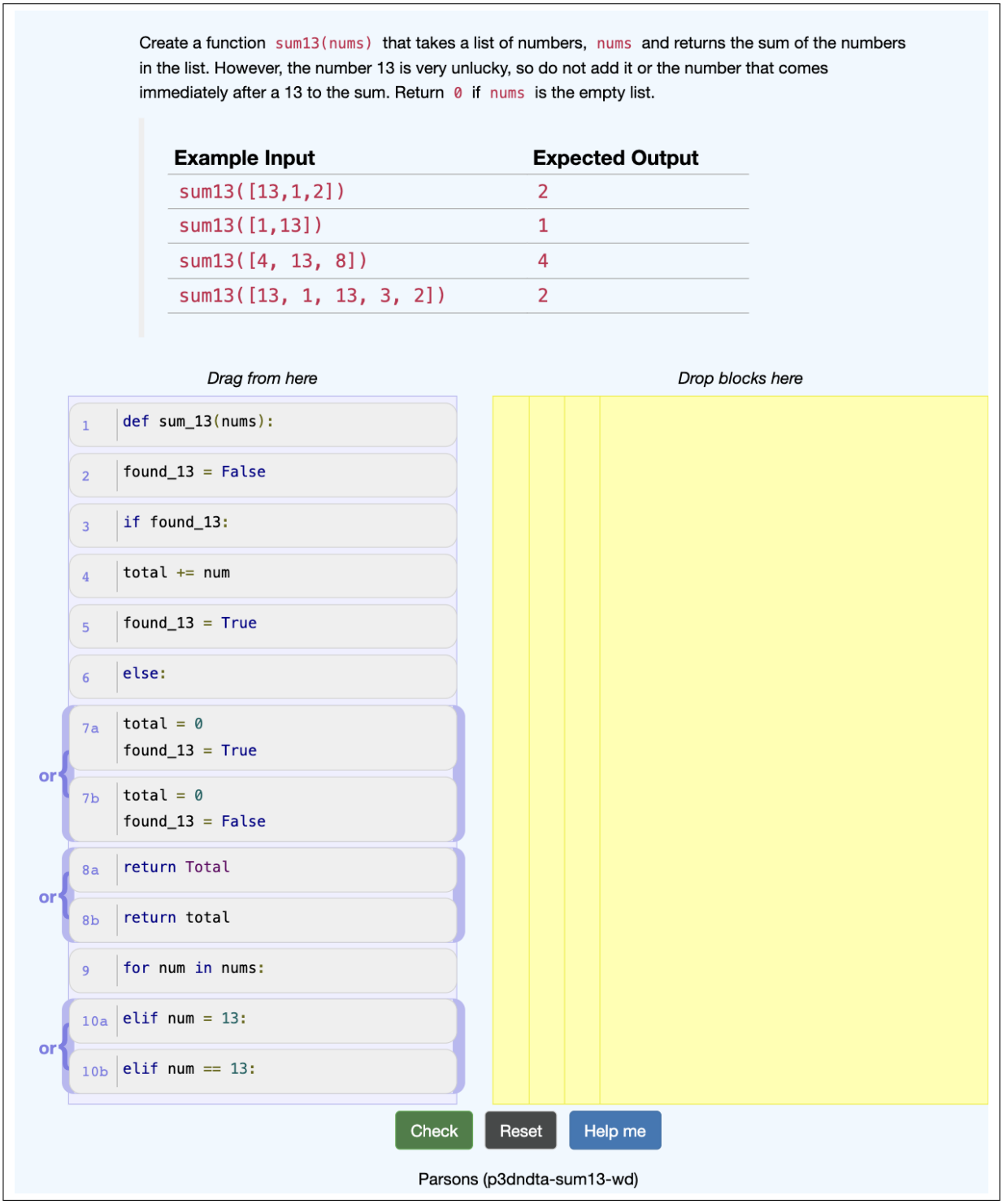

Figure 80: *p3dnd* - return the sum of a list of numbers, but ignore 13 and any number after a 13 (with distractors)

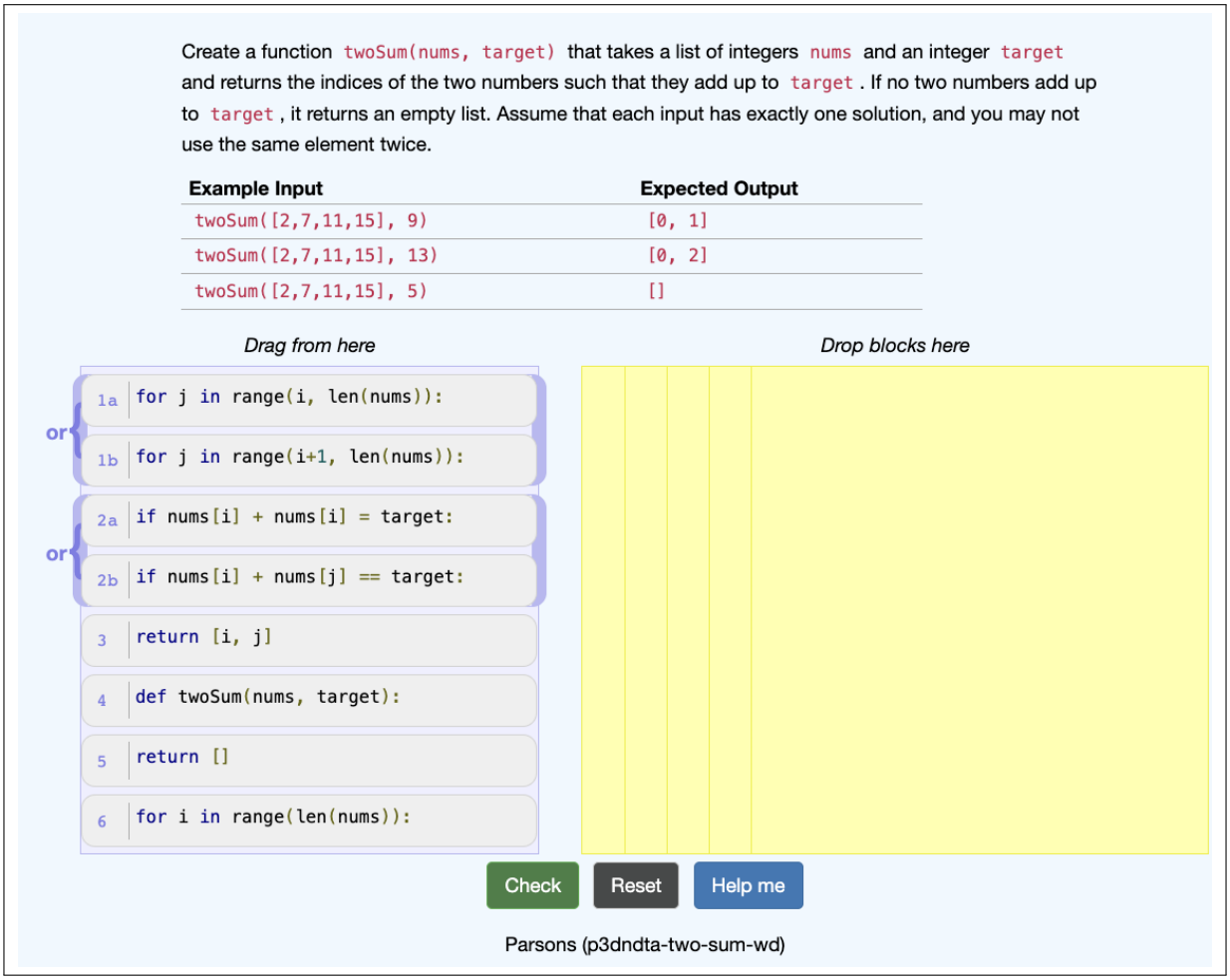

Figure 81: *p3dnd* - returns the indices of two numbers in a list that add up to a passed target value (with distractors)

Figure 82: *p3dnd* - return true if every two in a list of numbers is next to another two (with distractors)

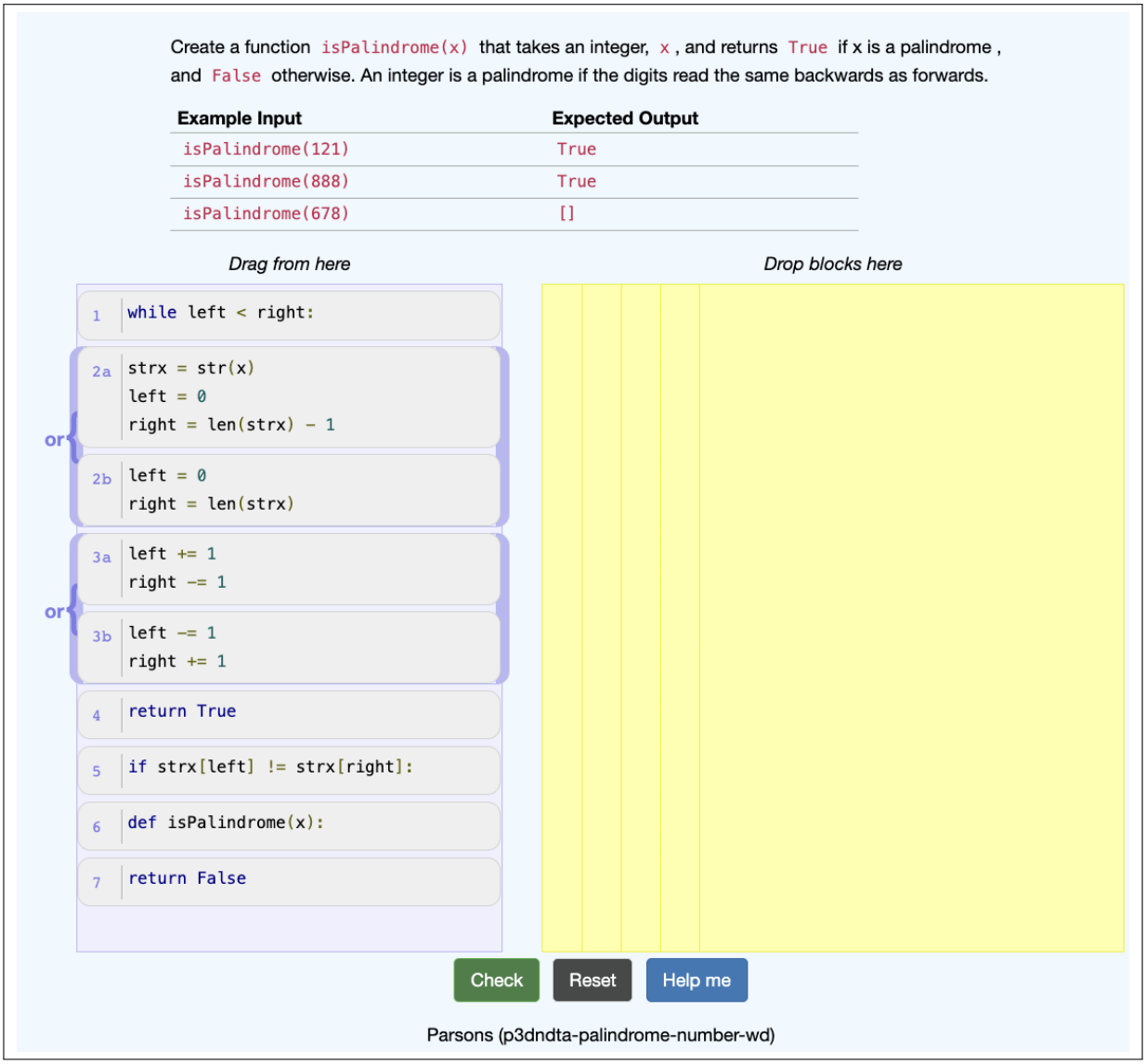

Figure 83: p3dnd - returns true if the digits in a number are a palindrome (with distractors)

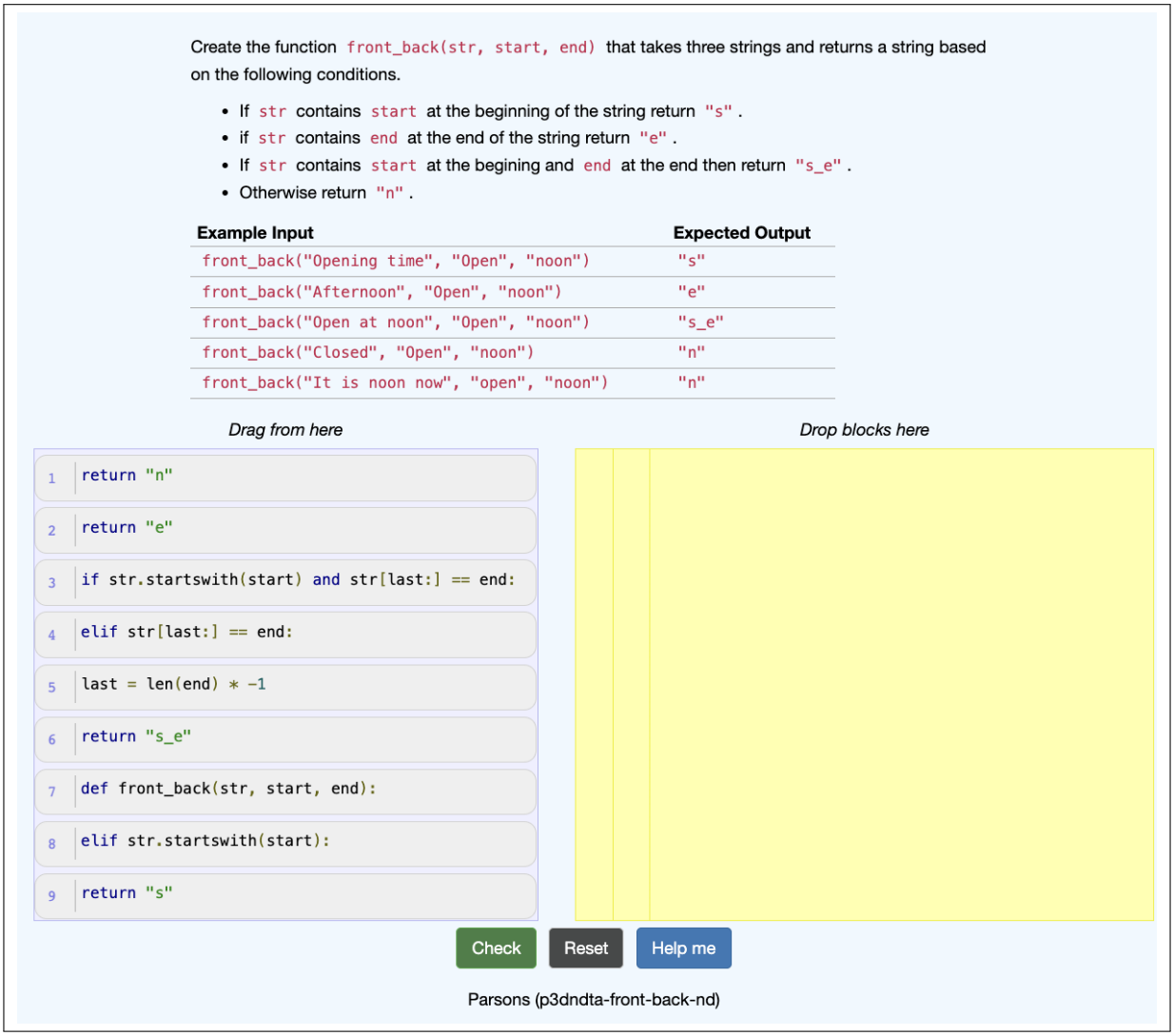

Figure 84: *p3dnd* - return a character based on what is at the beginning and end of a string

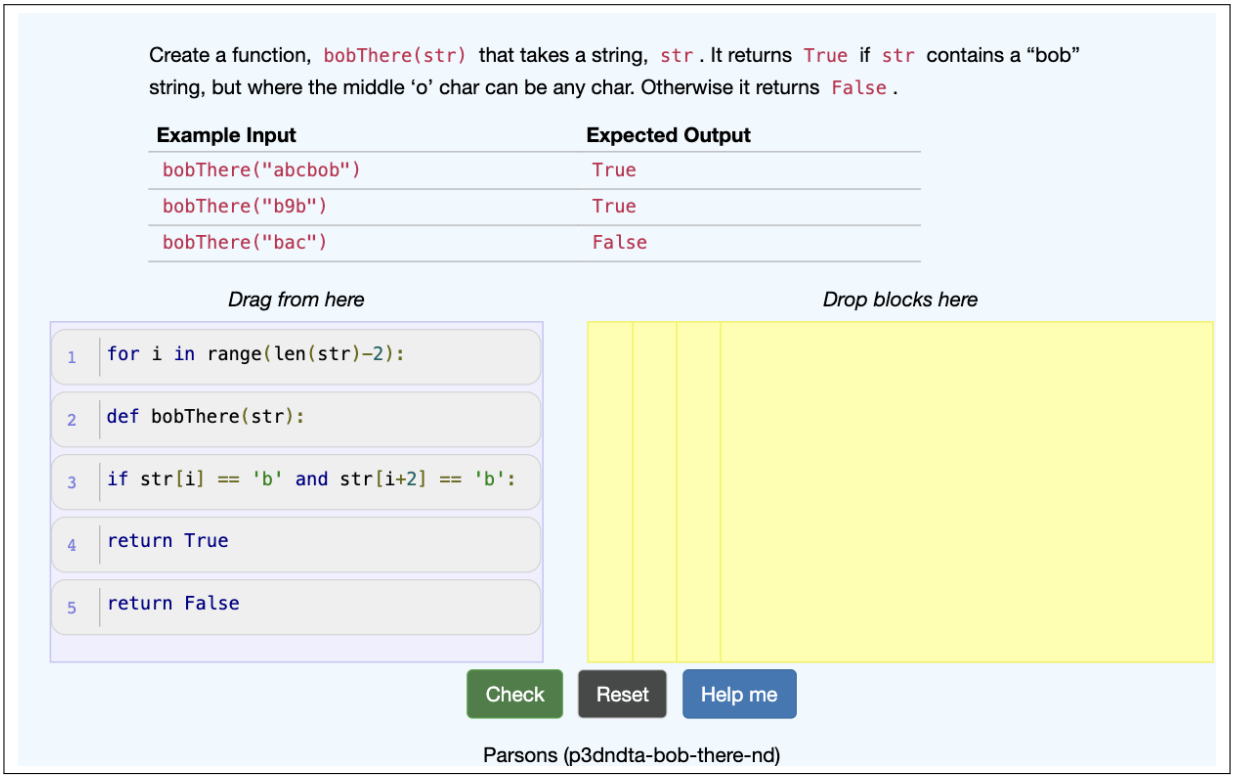

Figure 85: *p3dnd* - return true if a string contains “b*b” where “*” can be any character

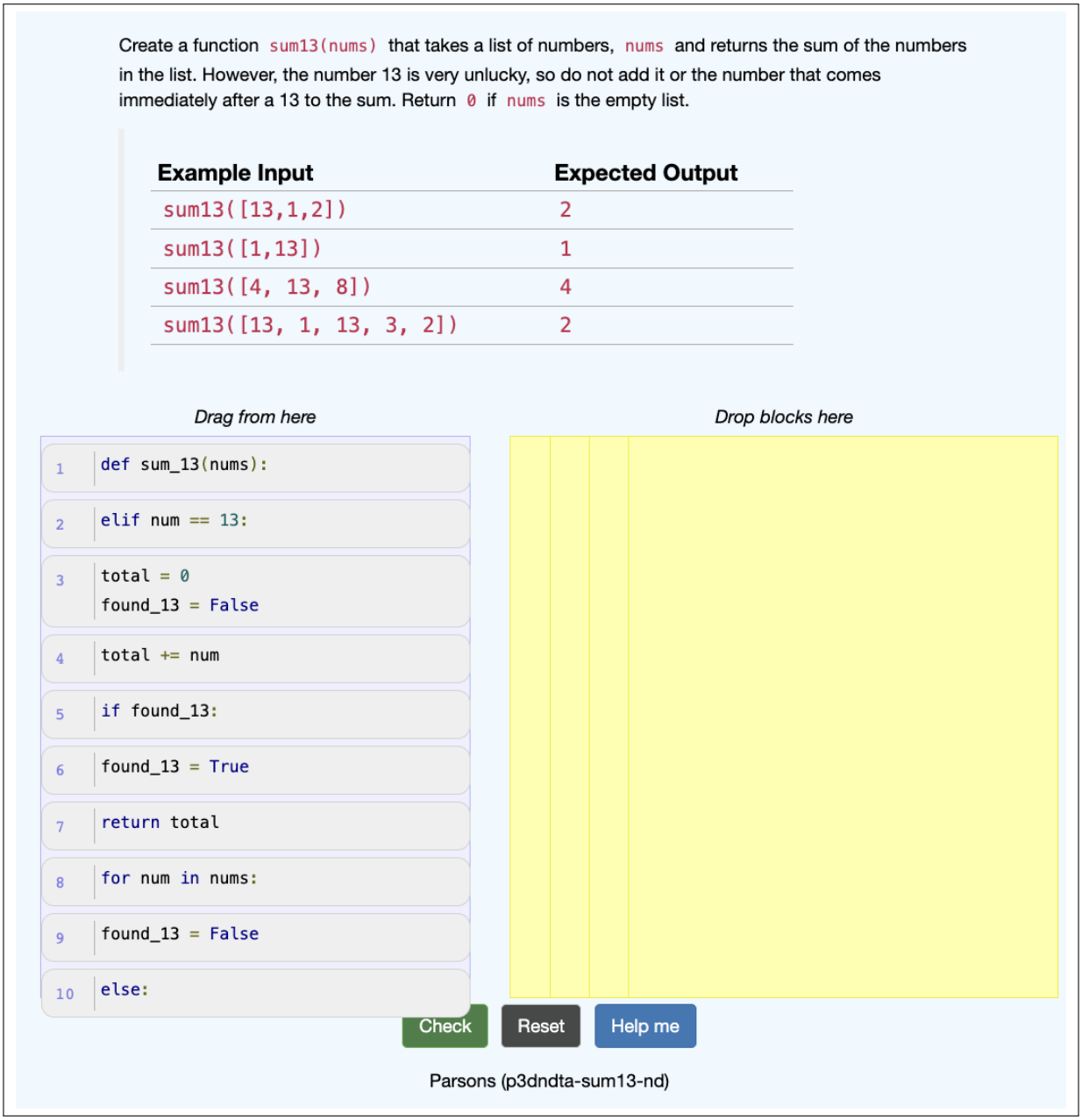

Figure 86: *p3dnd* - return a sum of a list of numbers, but ignore 13 and any number after a 13

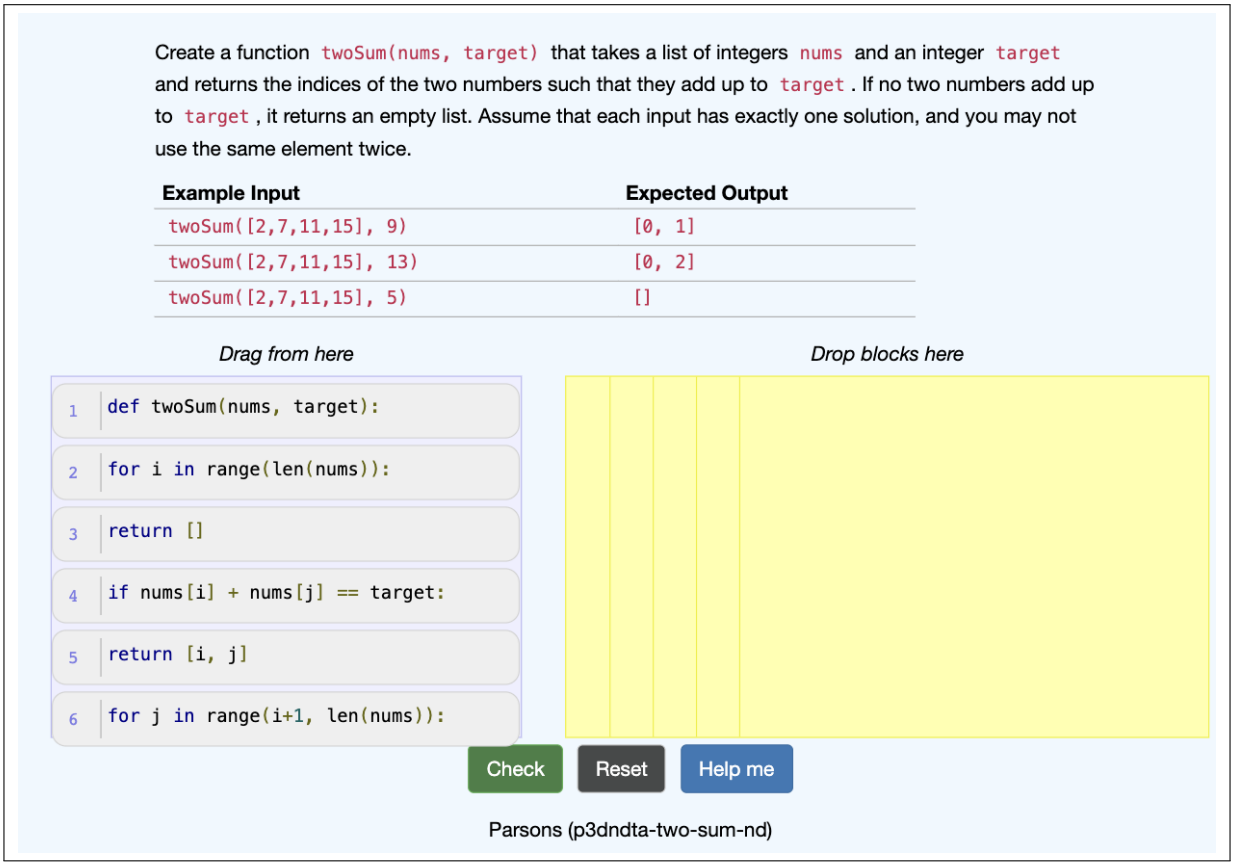

Figure 87: *p3dnd* - returns the indices of two numbers in a list that add up to a passed target value

Figure 88: *p3dnd* - return true if every two in a list of numbers is next to another two

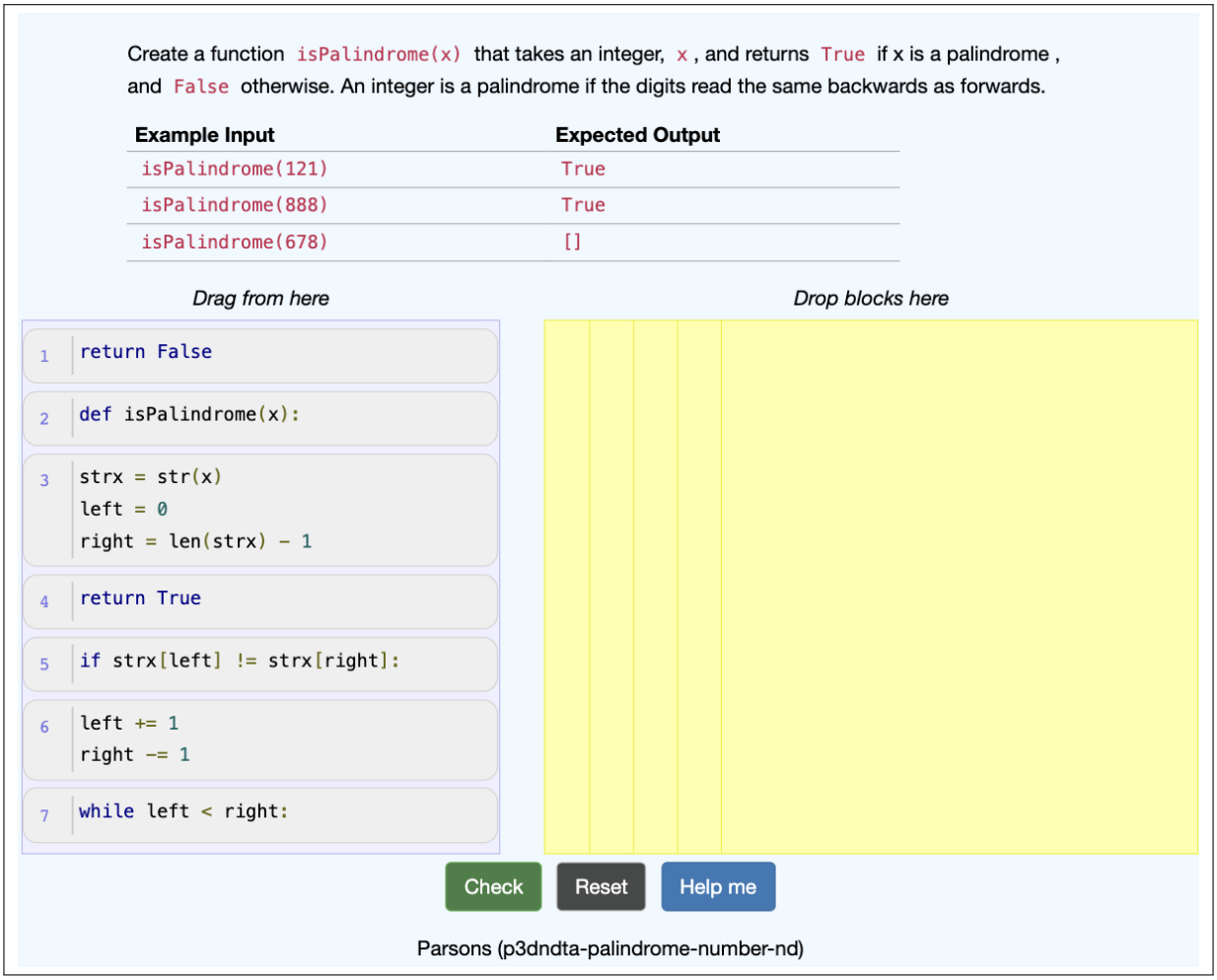

Figure 89: *p3dnd* - returns true if the digits in a number are a palindrome

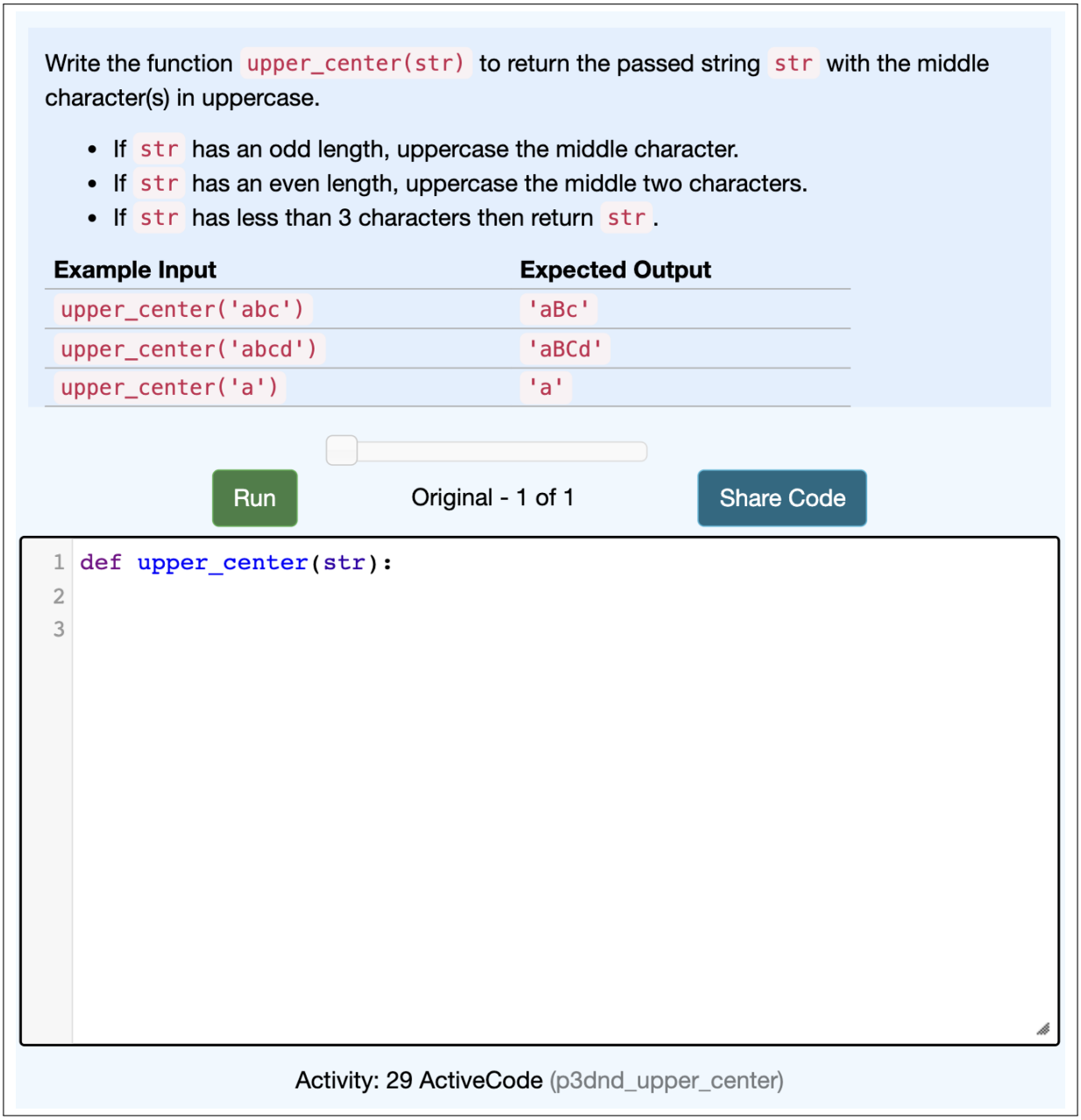

Figure 90: *p3dnd* - return a string with the center characters in uppercase or just the string if the length is less than 3

Figure 91: *p3dnd* - return true if the numbers in a list are sorted in descending order

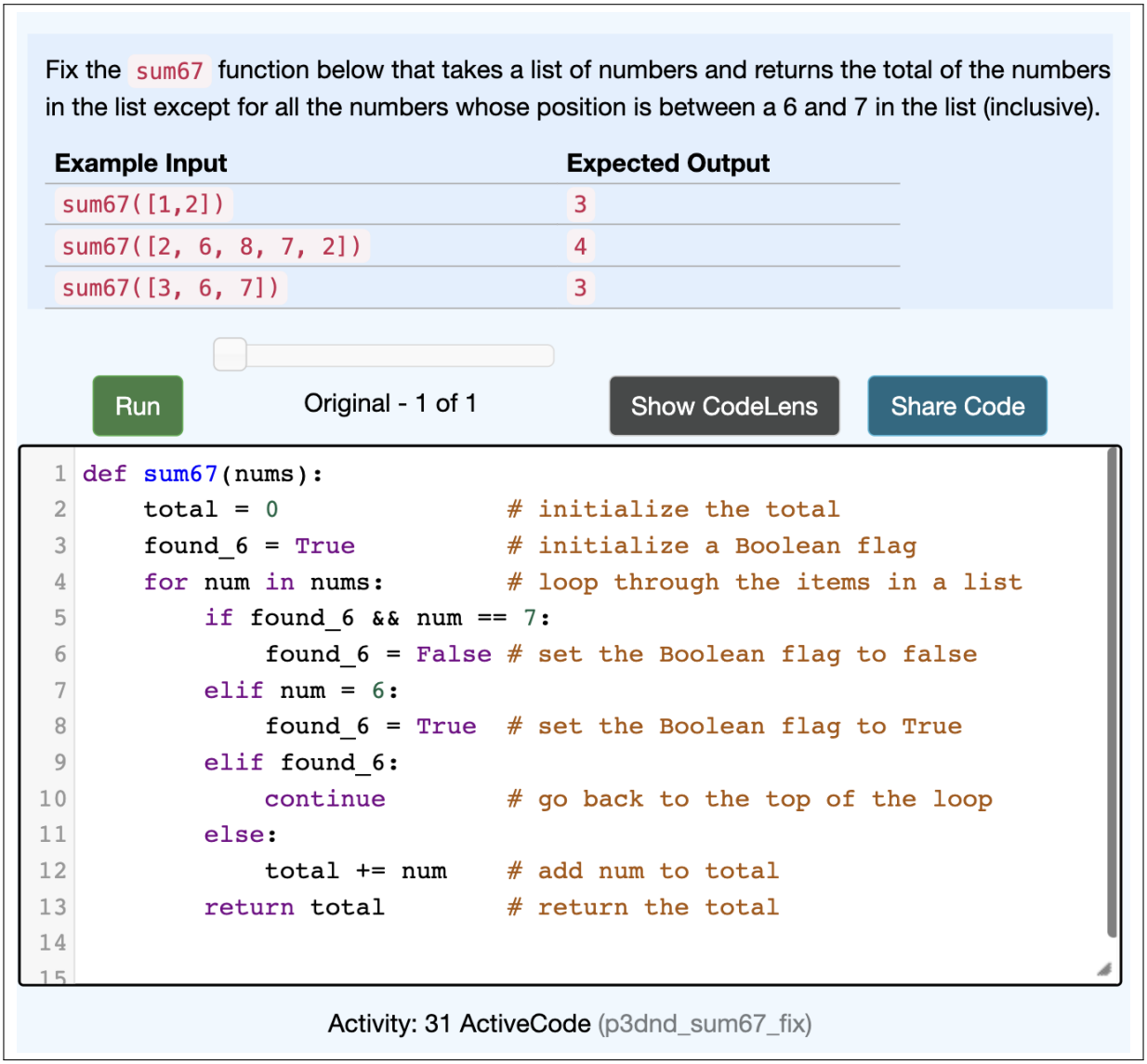

Figure 92: *p3dnd* - returns the total of all numbers in a list except those inclusively between a 6 and 7