Wiki - Level 4 - Local vs. global variables

Example: “circle_area_function”

For language level four, we will broaden the capabilities of Python slightly and consider the kinds of variables that we can use in creating programs. At level one, assignments to variables outside functions generated global variables whose values can be referenced anywhere inside the program, even inside the body of a function. In the example “circle_area_function”, the variable PI references the global version of the variable. (Note the absence of a subscript in *Pystep*.) Global variables can be referenced before, during and after a functional call.

Example: “cylinder_volume”

At language level two, we observed that the parameters of the function have the property that they are local to the body of the function and can only be accessed/updated inside the function body. In many case, we would like to create other variables that are local to the function to temporarily hold value that are useful in the function. In Python, assignment to a variable inside a Python function creates a local version of the variable. In the example “cylinder_volume”, we observe that the assignment to area creates a new variable with subscript one to indicate that this variable is local to the function. Observe that, as we step through this example, once evaluation of the function is completed, the local definition disappears.

Example: “fahrenheit_to_kelvin2”

One obvious question at this point is: What happens if we try to assign to an existing global variable inside the body of a function? In Python, this assignment creates a new local copy of the variable even if a global version of the variable already exists. In the example “fahrenheit_to_kelvin2”, assignment to the variable c inside the function f2k creates a new local copy of c inside f2k. (Note that this local version has subscript one). The global version of c persists through evaluation of the call to f2k and is referenced later in the final print.

Example: “player_is_ready”

However, in some cases, we would like to modify the value of a global variable inside a Python function. Since, by default, an assignment creates a local variable in a function, placing a global statement in that function instructs Python to treat any assignments to the specified variable(s) in the function as assignments to the global version of the variable. In the example “player_is_ready”, assignment to the variable message inside the function ready does not create a new local variable due to the global statement inside the function. We will make use of global variables inside functions substantially in this class since, early in the class, our primary mechanism for communicating information between the various event handlers controlling interactive program will be global variables. Later in the class, we will introduce object-oriented methods for reducing our reliance on global variables.