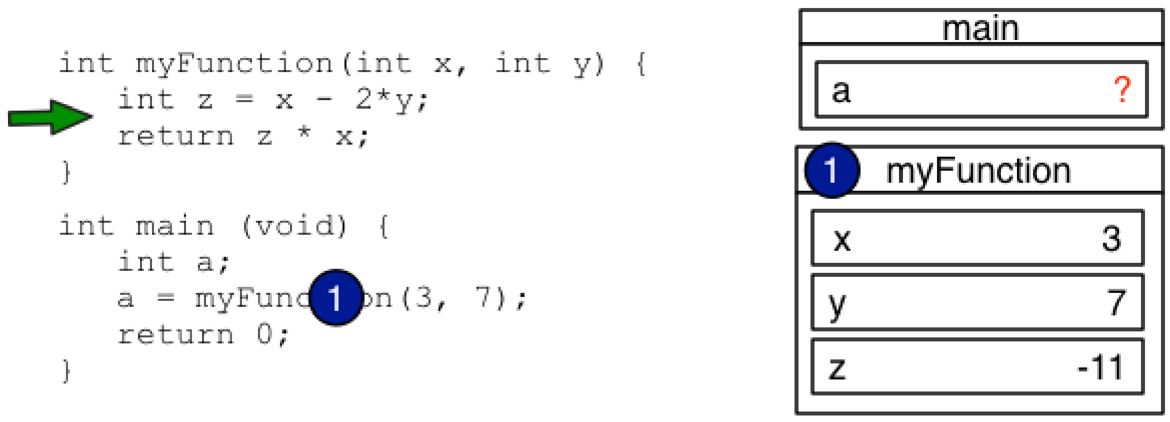
How to Evaluate a Function

coursera.org/learn/programming-fundamentals/supplement/Zouek/how-to-evaluate-a-function

Evaluating a function call is more complex than evaluating the other kinds of expressions that we have seen so far—it may take many steps of executing the code in the function to determine its answer. In fact, code may call one function, which itself may call other functions before finally coming up with an answer. While this may seem daunting, we can do it properly by following a few rules for executing function calls by hand.

As a first step towards reading code with function calls, we must first group together the variables belonging to one particular function into a larger box, labeled with the function's name, which is called a *frame* (or *stack frame*, since they are located on the call stack). This figure shows an example of this organization.



```
int myFunction(int x, int y) {
    int z = x - 2*y;
    return z * x;
}
int main (void) {
    int a;
    a = myFunction(3, 7);
    return 0;
}
```

Notice that in the example shown in the figure, one of the functions is named **main**. The function named **main** is special—execution of a program starts at the start of **main**. We start by drawing an empty frame for **main** and putting the execution arrow right before the first line of code in **main**. We then execute statements of the code until **main** returns, which ends the program.

Calls to functions may appear in expressions, in which case we must evaluate the function to determine its return result. To do this evaluation, we take the following steps:

1. Draw a frame for the function being called. Place a box in that frame for each parameter that this function takes.

2. Initialize the parameters by evaluating the corresponding expressions in the function call and copying the resulting values into the parameter's box in the called function's frame.

3. Mark the location of the function call, and note that location in the corner of the function's frame.

4. Move the execution arrow immediately before the first line of code in the called function.

5. Evaluate the lines of code inside the called function.
6. When you reach a return statement, evaluate its argument to a value. Note down this return value.
7. Return the execution arrow back to where the function was called—you know this location because you noted it in the corner of the frame. You will return the arrow to the middle of the line of code (rather than the typical "between them") because that line of code is part-way done.
8. Erase the frame for the called function.
9. Use the return value of the function as the value of the function call in the expression in which it appears.

A function call may also be used as a statement by itself, in which case, it is evaluated the same as above, except that its return value is not used for anything.

The next video demonstrates the execution of code with function calls.