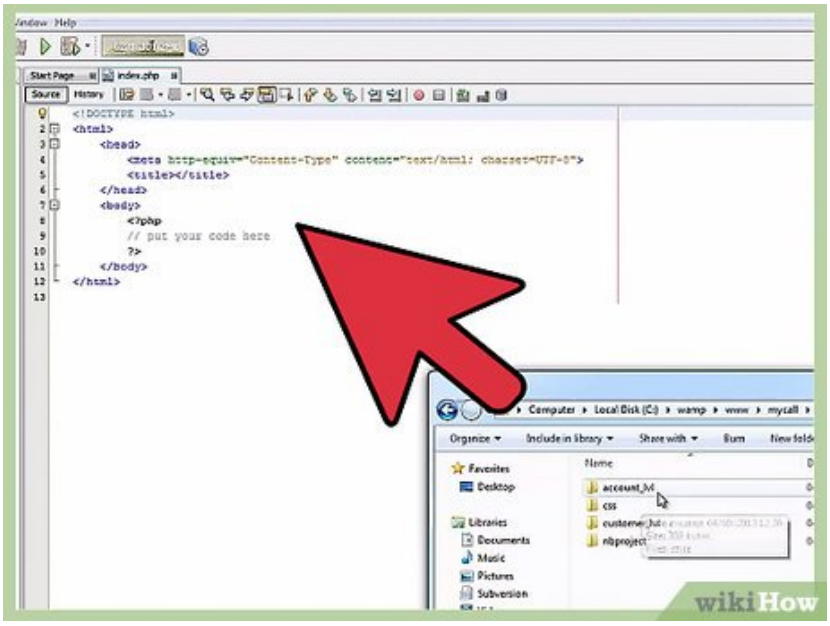
How to Create and Call PHP Functions

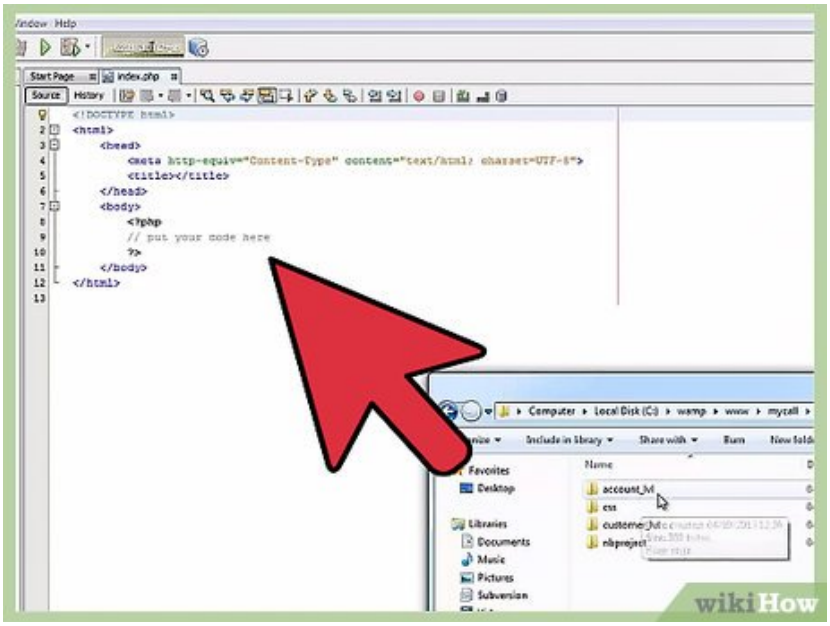
Perhaps you have learned the basics of writing PHP scripts. But sometimes your code may be long and repetitive. PHP functions are a flexible and easy way to consolidate code. This tutorial will you teach the basics of PHP functions. (Note:...

Part 1 of 3:

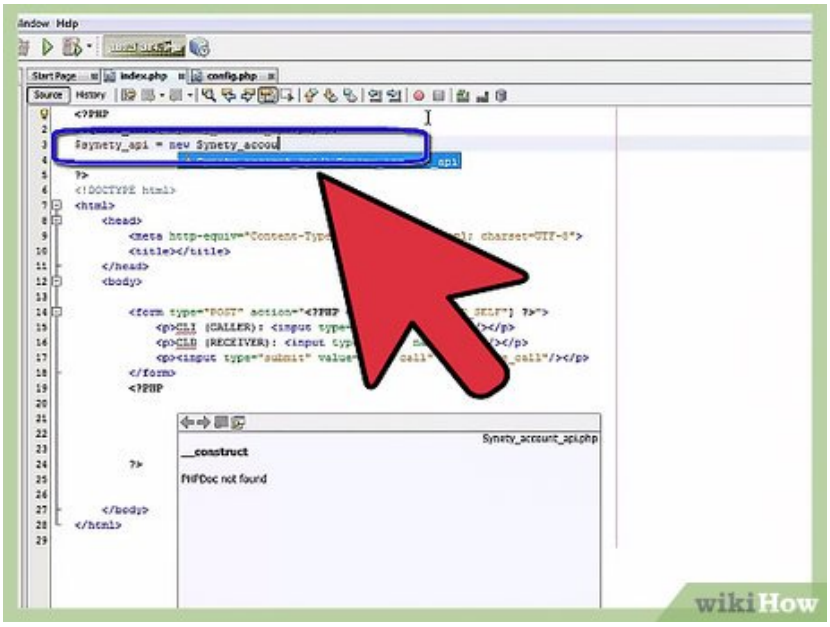
Creating a Function

1. 

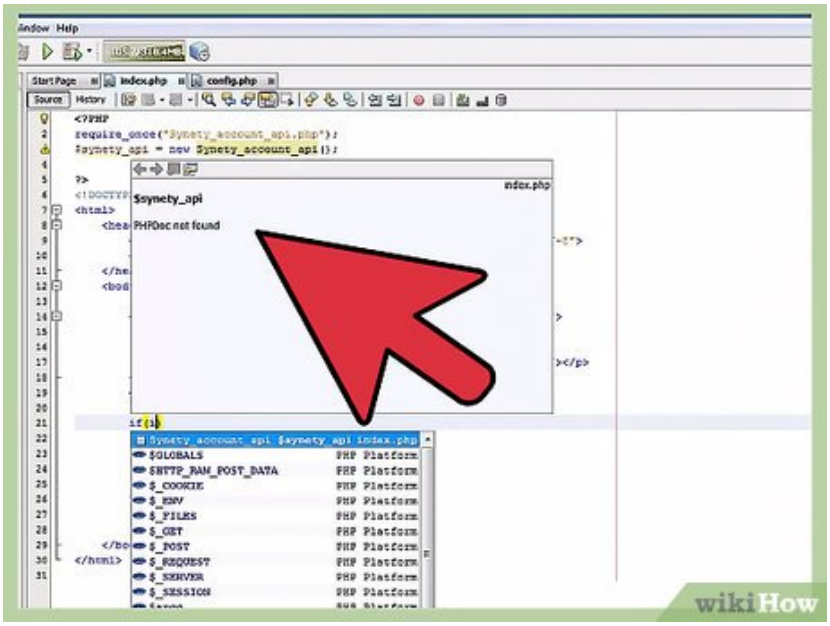
Create a new PHP file on your web server, and open it in your favorite text editor.

2. 

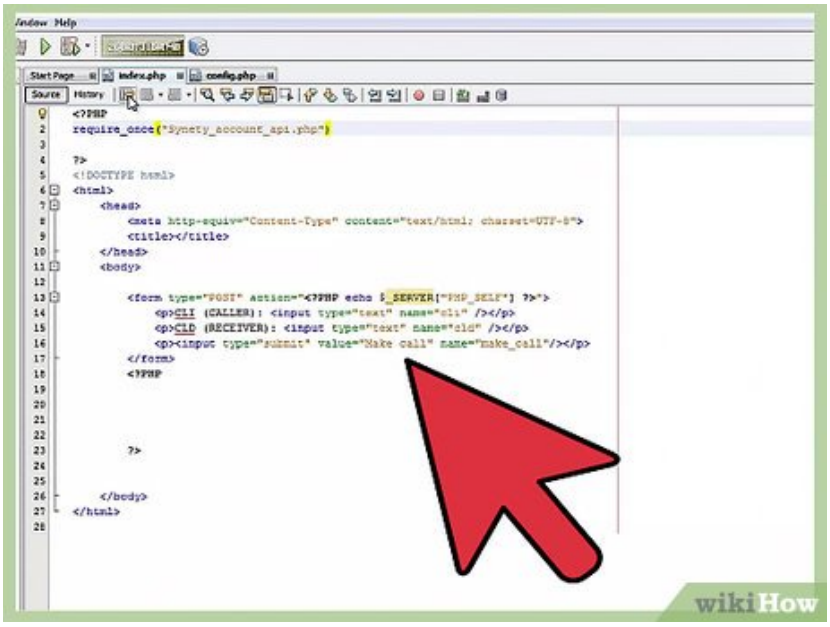
Start your file by typing the open and close PHP tags with some space in between to work with.

3. 

Type this on a new line between the two PHP tags.

4. 

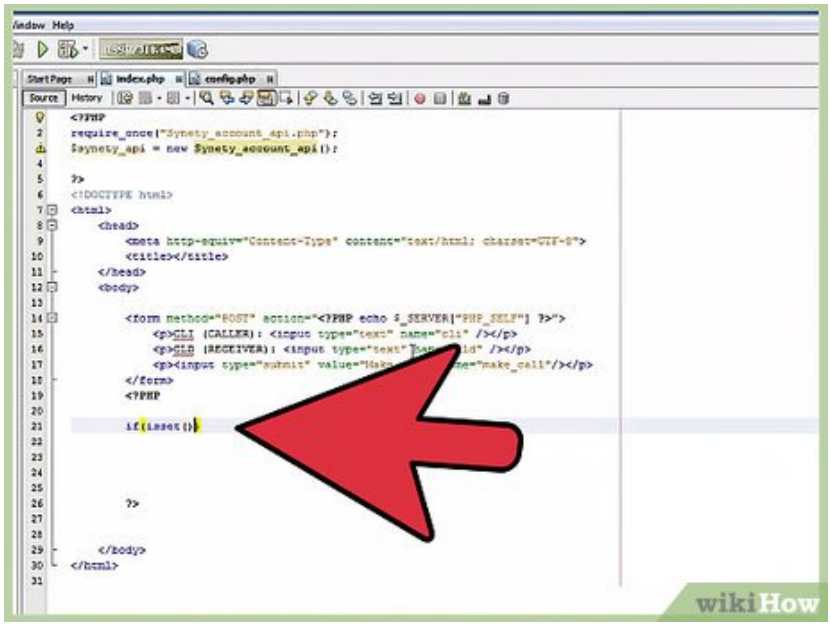
Whenever you want to create a function, always start on a new line with the word "function", hit the space bar and type its unique and descriptive name followed by two parenthesis, then type an open curly brace. In this case i chose to call the function "my_function," in your scripts you should give your functions better names.You may have noticed the "$input" variable in between the parenthesis. This is called an argument. It is the input that the function will return as an output. PHP functions can have multiple arguments, as long as they are separated by commas. Observe the example above.

5. 

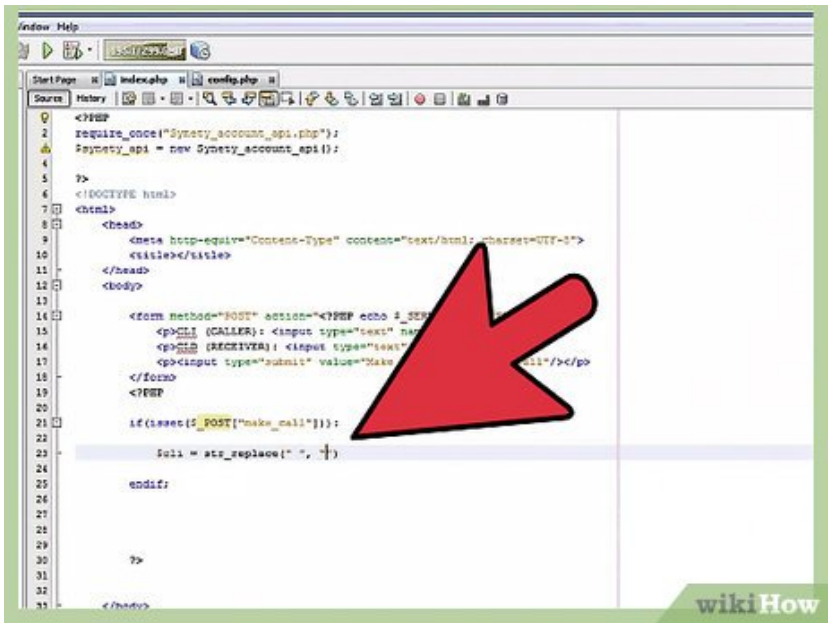
Now type this code inside the curly braces. The command "return" does exactly what is says, it returns the output. In this case the output is $input minus two, times 10. Take note that once the output has been returned the function is exited, and no code after it will be executed.

Part 2 of 3:

Calling a Function

1. 

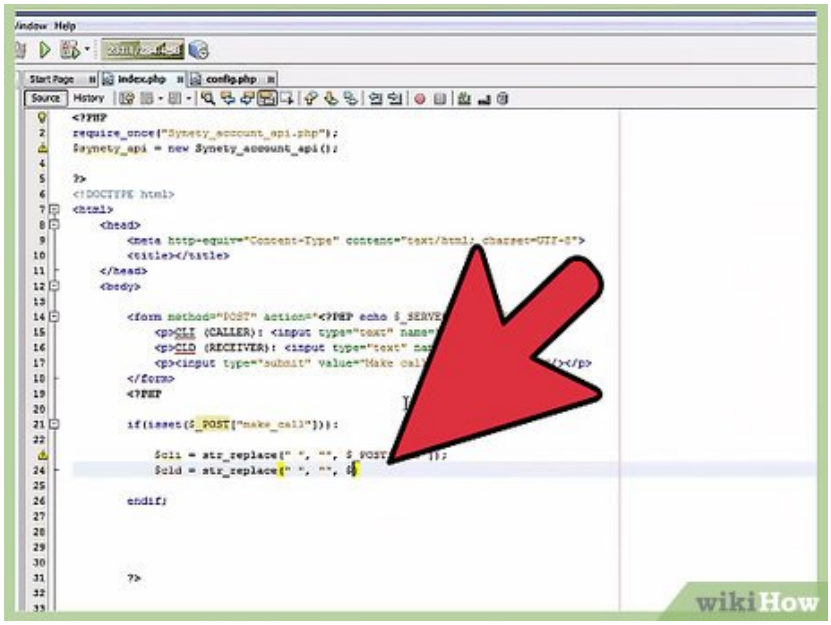
Type the code below on a new line after the closing curly brace of the function. This line of code is making a call to "my_function." In this case we will give my_function an input of 8, and PHP will echo the returned output value. Test the script, you will get a value of 60.

2. 

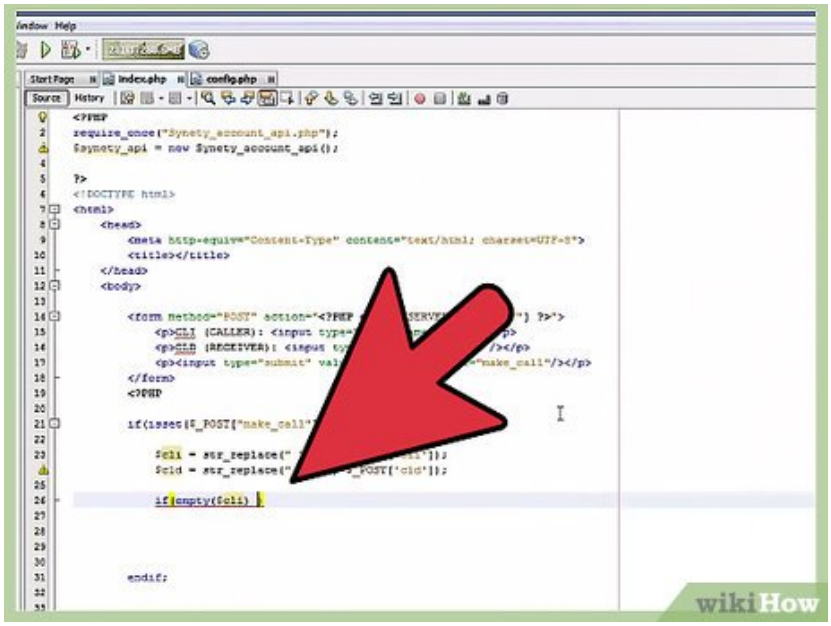
Now type this code on a new line. The magic of coding with functions is the ability to reuse and recycle the same code over and over. We are now calling my_function two more times and reusing its code. Also notice how the in the code above the call to my_function is treated like a number. Always treat calls to functions as the datatype you expect to get back (whether it is a number, string, boolean, or resource). Test the script, you will get a value of 60 followed by a value of 260.

Part 3 of 3:

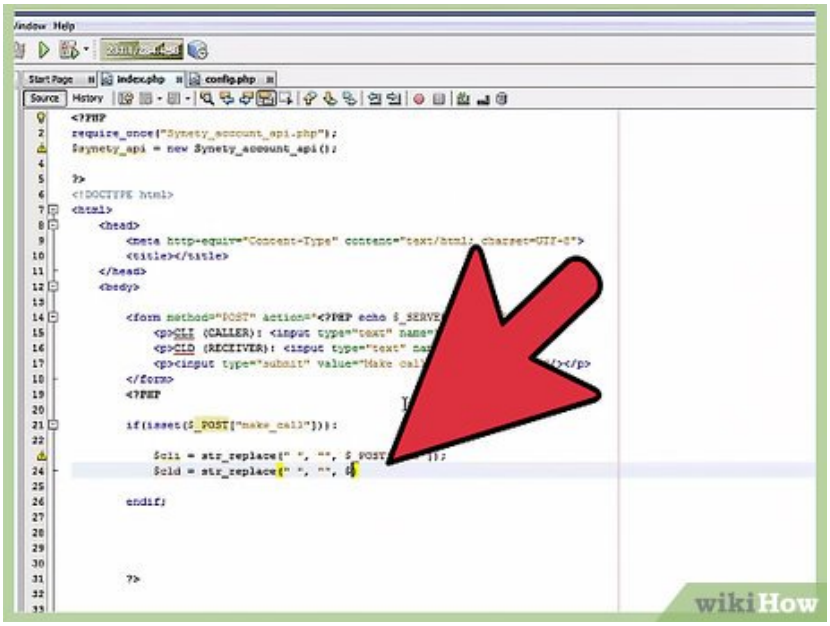
Functions and Variables

1. 

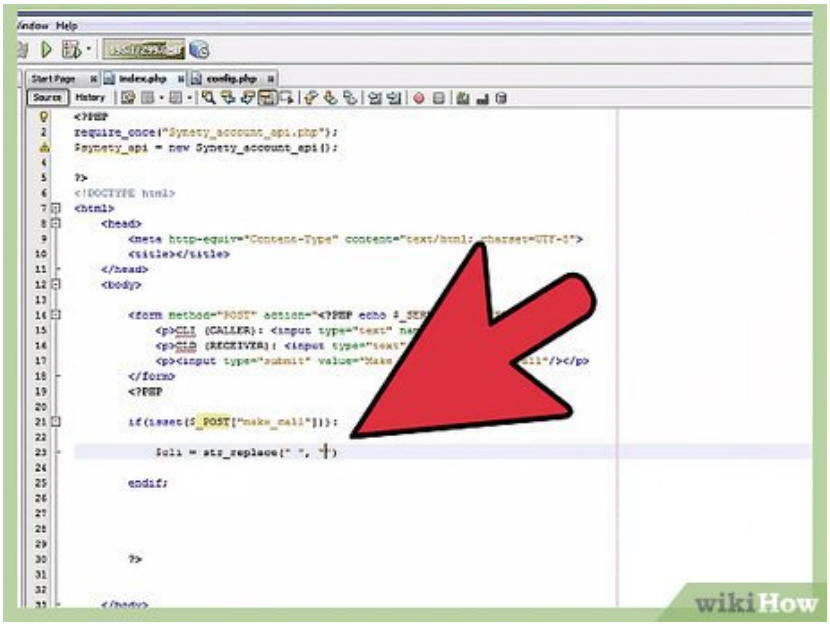
Add three lines of code to your script.

2. 

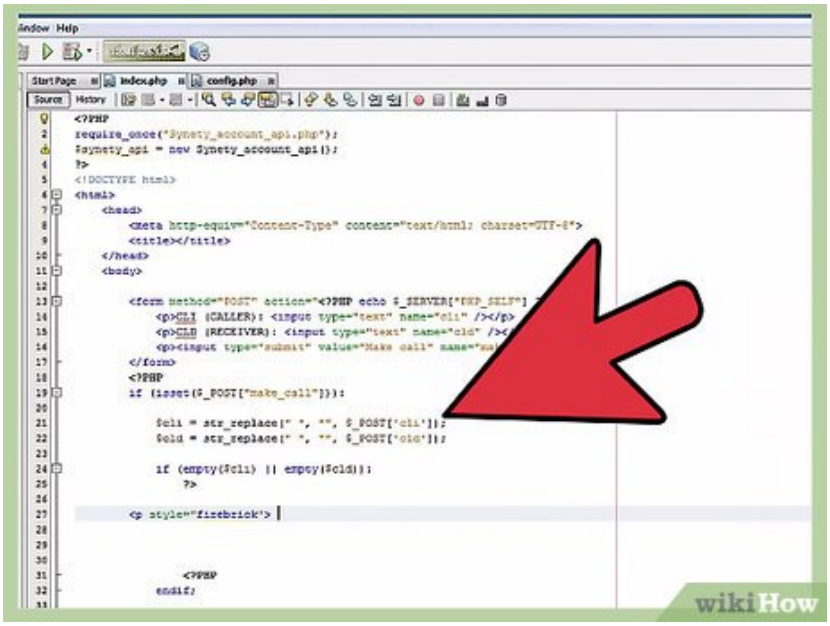
Add these before any other code at the beginning of your function.

3. 

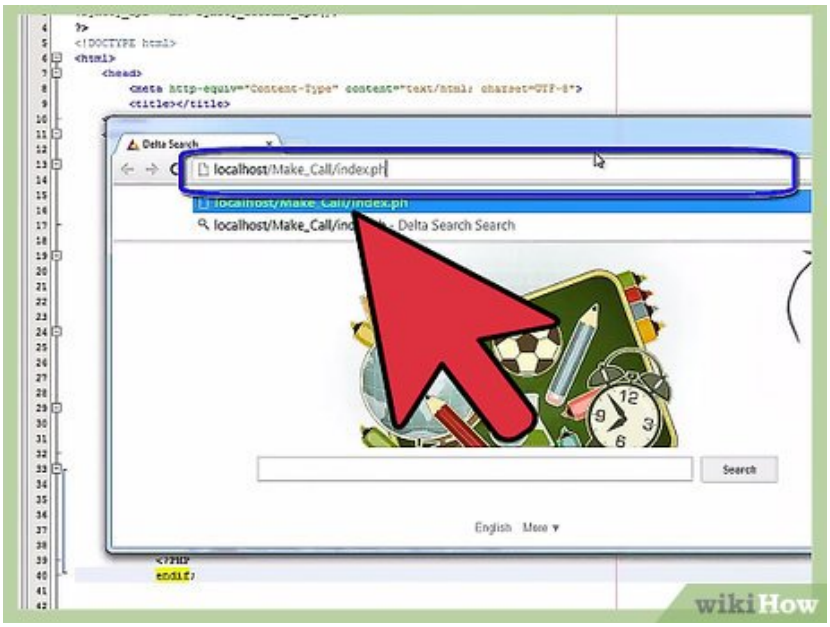
Add this line right after the closing curly brace of your function.

4. 

For sake of clarity here is the entire code sample.

5. 

A variable declared inside a function cannot be used outside the function. For example if i tried to echo $input *outside of the function*, PHP would throw an error because i have not created that variable *outside of the function*. So note that variables created in a function (including the arguments) can only be used inside *that* function. Variables declared outside of a function can only be used outside of functions (unless you use them as an argument of course).That said, there is a command that allows you, in a function, to use a variable created outside a function. That command is "global", in the code above we use global with the variable $num so we could change it or access its value in our function. If you test the code, you will get the values 50 and 240.

6. 

Play around with functions! Build your own function that is really useful. Use them in your future PHP projects. Using functions can improve code modularity and the size of the overall project.

tips and guides. Thank you for reading and for following us regularly.