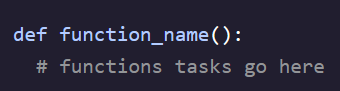
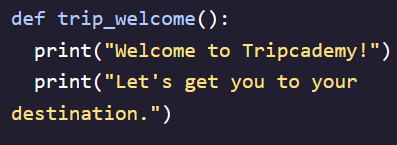
**[Functions:](https://www.codecademy.com/resources/docs/python/functions?page_ref=catalog)**

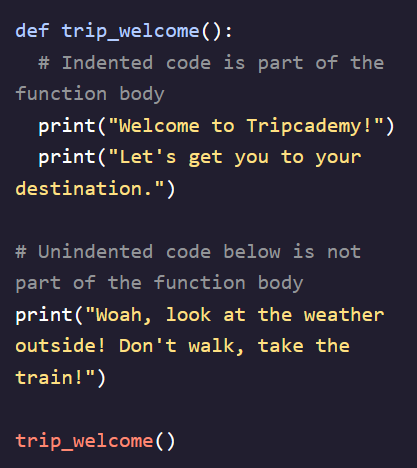
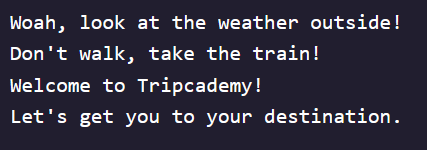
- Used when we have to repeat *the same set of steps* in many different places in our program  
- Allow us to write the code in one location and then reuse that function multiple times in our program  
- Functions begin with *def <function\_name> ():*- Begins with **def** function header  
- Followed by snake\_case **<function\_name>** that explains what function does  
- Followed by **( )** that can hold input values known as *parameters*- Ends with **:   
-** Need one or more valid python statements in function body after header (must be indented to show it is apart of function)  
- Functions need to be called later in a program otherwise nothing will output to terminal 

**Calling a Function:**

- The process of executing the code located within the body of a function inside of the program  
- To call a function type out the *<function\_name>* followed by *( )* with no indentation

**Execution Flow:**

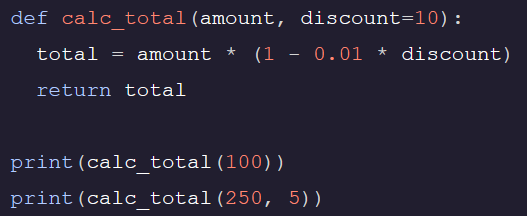
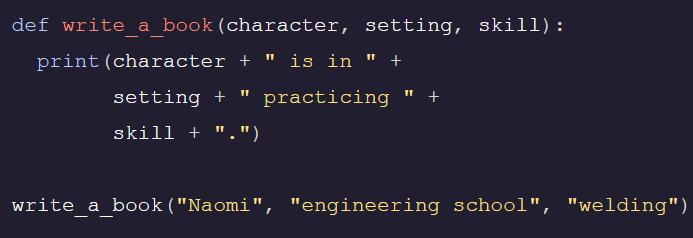
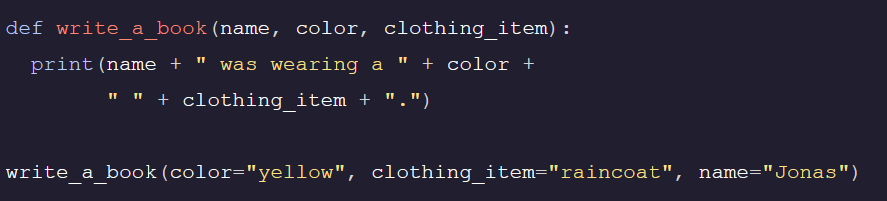
- *Execution Flow* is the order a program executes in Python from top to bottom   
- Can use whitespace and indentation to create layered functions that don’t all run at once when called  
- The *scope* of a variable is the part of a program where it can be accessed (in this case only inside *def* function)

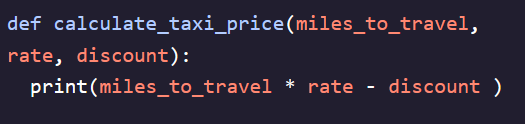
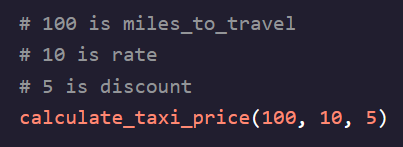
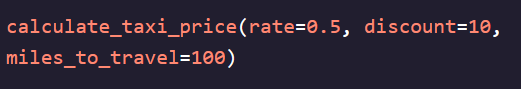
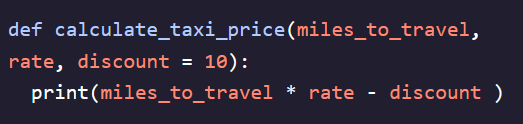
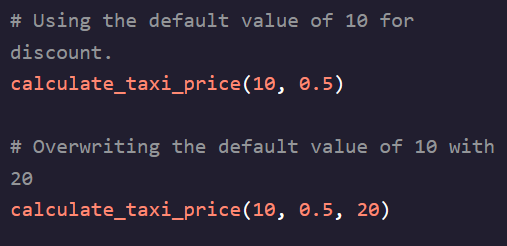
If you unindent a higher print statement that one will print first before the next unintended one

In this code, the third *print ()* statement is at the same level as *def …*   
When the program executes it goes outside in and prints *def* and third print statement first, then inner print statements

[**Parameters**](https://www.codecademy.com/resources/docs/python/functions/arguments-parameters) **& Arguments:**

- *Parameters* are variables that are declared in the function  
- When a function is called, each parameter is assigned the value which was passed as the corresponding argument during the call  
- Can be initialized to a default value in the function header itself  
  
- When a function is called, an *argument* is passed into it to correspond with each parameter in the order which they appear   
- *Keyword Parameters* allow users to input variables directly into parameters regardless of order  


**Types of Arguments**

- **Positional Arguments –** Arguments that can be called by their position in the function definition  
 - Assignment depends on position in function call  
  **- Keyword Arguments –** Arguments that can be called by their name **-** Assigned by explicitly assigning a keyword to each value   
 **- Default Arguments –** Arguments that are given default values  
 - Values are assigned to functions in function header as default values   
 - Can skip default value in function call **OR** overwrite it by assigning our own value   
 

**Returns:**

- A *return* stores a value in the program for access later on  
- Allows us to reuse this value throughout the rest of the program  
- The value returned in this way is called a *returned function value*- Can return multiple values in function by separating them with a comma  
