Hyperion Development

Task: Beginner Programming With Functions -

Defining Your Own Functions

www.hyperiondev.com

Introduction

Welcome to The Beginner programming With Functions - Using Built in Functions Task!

This Task reintroduces functions and will focus on teaching you how to create your own functions. It will also show you how functions can be used to compute certain values using list elements and/or text file contents.

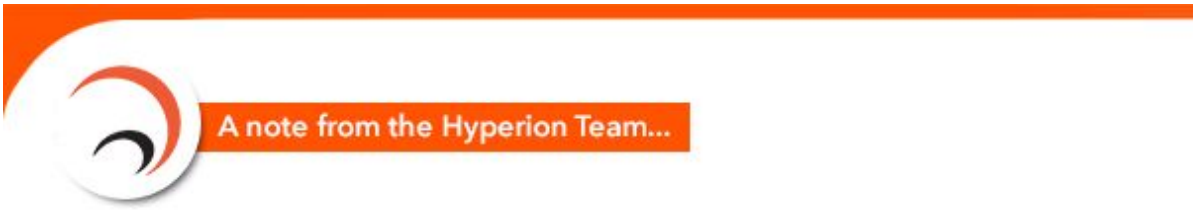
Connect with your mentor

CONNECT

Remember that with our courses - you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to www.hyperiondev.com/support to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!



Very few things just work out of the box. As developers we know this and are ready for it. When you dive into Python for the first time, the number of things that do indeed just work is amazing. Why then a post about more packages to use?

These 5 packages will give you the ability to move beyond what you think is possible, and in a couple of lines of code, allow you to grab information from web pages, do complex calculations on it, and push your findings onto your own web app in a matter of minutes not weeks, and as an added bonus, you get to sleep at night knowing your code is doing what it should be doing.

The 5 Python Packages every South African developer should know about can be found HERE

How to Define a Function

A function is defined as follows:

```
def addone(x):
    y = x+1
    return y
```

The function that has been created in the example above is named **'addone.'** It takes as input **the parameter 'x'**. A parameter is a variable that is declared in a function definition. Parameters store the data needed to perform the logic that the function specifies. Parameters are filled when data is passed to the function as the function is called (which you will learn about soon). The code indented under 'def addone' is the logic of the function. It defines what happens when the function is called. Simply put, the function in the example above, computes a new variable, y, which is the value stored in variable x with 1 added to it. It then 'returns' the value, y.

The general syntax of a function in Python is as follows:

```
def functionName(parameters):
      statements
      return (expression)
```

The def and return Keywords

Note the **'def'** keyword. Python knows you're *def*ining a function when you start a line with this keyword. After the keyword 'def' you will then put a function name, its input parameters and then a colon, with the logic of the function indented underneath.

Note the **'return'** keyword. Python will expect this at the end of your function, but it doesn't always have to be there. The value after the keyword 'return' will be returned/passed back to whatever code called the function.

Calling a Function

In order to execute a function, you need to 'call' it. You call a function by using the function's name followed by the values you would like to pass to the parameters within parentheses. The values that you pass to the function are referred to as **arguments**.

In the example below, the function defined above ('addone') is called. In this example, we pass the value '10' as an argument to the function 'addone'. Since we created a parameter called 'x' when we defined the function 'addone', passing the argument 10 to the function 'addone' will result in the parameter 'x' being assigned the value 10.

```
numPlusOne = addone(10)
# The result that the 'addone' function 'returns' is stored in the numPlusOne
variable.

print ("10 plus 1 is equal to: " + str(numPlusOne)+".")

# Or even:
print ("10 plus 1 is equal to: " + str(addone(num))+".")
```

Think of a call to the function (e.g. addone(num)), as a 'placeholder' for some computation. The function will go off and run its code and return its result in that place.

You can define a function, but it will not run unless called somewhere in the code. For example, though we have defined the function 'addone' above, the code indented underneath it would never be executed unless there was another line that called 'addone' with the command 'addone(some_variable)' somewhere in the main body of your code.

Function Parameters

In the function definition, you put the parameters between the parentheses after the function name. You can have more than one of these variables or parameters. Simply separate them by commas. When you call a function, you place the value you would like to pass to the function in parentheses after the function name. This value is passed to the function and stored in the corresponding function parameter variable. When calling a function, be sure to place the values you are passing to the function in the same order as the corresponding function parameters in the function definition.

Instructions

First read 'example.py'. Open it using Notepad++ or IDLE.

- 'example.py' should help you understand some simple Python. Every task will have example code to help you get started. Make sure you read all of 'example.py' and try your best to understand.

- You may run 'example.py' to see the output. Feel free to write and run your own example code before doing the Task to become more comfortable with Python.

Compulsory Task 1

Follow these steps:

- Create a Python file called "myFunction.py" in this folder.
- Create your own function that prints all the days of the week.
- Create your own function that takes in a sentence and replaces every second word with the word "Hello"

Compulsory Task 2

Follow these steps:

- Create a Python file called **holiday.py** in this folder.

- You will need to create four functions:
    - HotelCost - This function will take the number of nights a user will be staying at a hotel as an argument and return a total cost for the hotel stay (You can choose the price per night charged at the hotel).

    - PlaneCost - This function will take the city you are flying to as an argument and return a cost for the flight (Hint: use if/else if statements in the function to retrieve a price based on the chosen city).

    - CarRental - This function will take the number of days the car will be hired for as an argument and return the total cost of the car rental.

- HolidayCost - This function will take three arguments: the number of nights a user will be staying in a hotel, the city the user will be flying to and the number of days that the user will be hiring a car for. Using these three arguments, you can call all three of the above functions with respective arguments and finally return a total cost for your holiday.

- Print out the value of your Holiday function to see the result!

- Try using your app with different combinations of input to show it's compatibility with different options.

Give your thoughts..

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes. Think the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.