TASK

Programming With Functions — Defining Your Own Functions

Visit our website

Introduction

WELCOME TO THE PROGRAMMING WITH FUNCTIONS TASK!

This is an introduction to **Functions** in Python.

A function is a reusable and organised block of code that is used to perform a single action or specific task. Functions can either be user-defined or built-in.

In this task, you will be introduced to functions that are built into the Python language itself and are readily available for us to use as well as introduce 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.

Get in touch

Connect for support

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/portal** 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!

A note from the

HyperionDev Team

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 developer should know about can be found **here**.*

WHAT ARE FUNCTIONS?

A function is a reusable and organised block of code that is used to perform a single action or specific task. Functions are also sometimes referred to as 'Methods' (if you have used Java before, functions in Python are very similar to Java methods).

Functions in programming also relate to mathematical functions — perhaps you recall f(x) in mathematics. A mathematical function took some input, in this case x, did some computation with it, and returned a value, normally called y. i.e. y = f(x) is a function that when called with the parameter x returns the value y.

Functions can either be user-defined or built-in. Built-in functions are built into the Python language itself and are readily available for us to use. You have actually already been using some built-in functions such as, `print()`, `input()`, `len()`, `int()`, `str()` and `float()`. See a list of available built-in functions **here**. The `'write'` command, i.e. `ofile.write(name+"\n")`, is in fact also a function, implemented by some programmer, that tells Python how to write the content you give it to the file ofile.

There are thousands of functions already implemented in Python that you can use to get things done. Programmers have already written the logic for many common and even complex tasks and sometimes you can find the exact built-in function that you need to complete a task.

However, you are not limited to these functions. You can also create your own functions to meet your own needs, these are what are known as user-defined functions.

IMPORTING OUTSIDE MODULES

We mentioned earlier that there are thousands of already written functions, but how do you get access to them?

Modules are the answer. Modules, or libraries, are pieces of code already written by other programmers that you can 'import' into your program. Though you don't see their code directly, you can access them. You import these modules using the 'import' keyword followed by the module's name.

An example of a very useful module is the math module which has many pre-built-in functions for you to use, such as the one considered in the example below.

Say you want to find the square root of a number. It would be tedious to try to write code to implement that (and a bit hard if you think about it!).

Instead, let's just use the `sqrt` function already implemented by some programmer years ago in the math library.

```
import math
# This imports the math module. You do not need to install the math module
separately; it comes with Python but must be specifically imported into a
program that uses it.

print("The square root of 25 is: " + str(math.sqrt(25))+".")
# We are now using the sqrt function of the math module.
```

Here, math.sqrt returns a FLOAT which is just an int that has decimal places (for example, 1.5). That makes sense, given that the square root of numbers isn't always a simple integer. The square root of 25 may be “an easy 5”, but the square root of, say, 3 has many decimal digits.

We must thus convert the result of math.sqrt(25) to a string before we will be able to print it out since the print function only works on strings. This explains why we have `str()` around the call to the function — the result of the `sqrt` function is being cast from float to string and printed out.

Another example of using built-in functions would be:

```
import math

maximum = max(1,5,7,3,6)
print(maximum)

minimum = min(1,4,7,1)
print(minimum)

absolute = abs(-42)
print(absolute)
```

When you run this code, you'll notice how the functions automatically do the mathematical operation for you.

Read and run the example.py file in this task folder to see many more examples of importing modules, particularly math modules. You can also visit **https://docs.python.org/3.7/library/math.html** to read more about the math module and its functions. Python modules and their functions are usually very well documented online.

Now let's talk about defining our own functions.

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 **return**s 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 defining 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.

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

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

# 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

Make sure to first read example.py comprehensively, as it contains several examples of built-in and defined functions.

Open it using Notepad++ (Right-click the file and select 'Edit with Notepad++') or IDLE.

Compulsory Task 1

Follow these steps:

- Create a new Python file in this folder called **binary.py**
- Write a program that can convert a binary number to a decimal number.
- A binary number is a number that is made up entirely of 0s and 1s (e.g 101101). You can represent any amount you would like using binary.
- Ask the user to enter a binary number and convert that number to a decimal number.
- You can visit the following website to find out how to convert from binary to decimal: **http://www.rapidtables.com/convert/number/how-binary-to-decimal.htm**
- Print out the decimal value of the number.
- Remember to make use of the built-in functions found in the math module as well as lists (remember to import the math module into your program!).

Compulsory Task 2

Follow these steps:

- Create a Python file called **my_function.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 3

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 its compatibility with different options.

Completed the task(s)?

Ask your mentor to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that 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.