Task: Beginner Programming With Functions - Using Built in Functions

www.hyperiondev.com

Introduction

Welcome to The Beginner Programming With Functions - Using Built in 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. We are going to be exploring some of the more useful and most common Python functions.

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!

A note from the Hyperion Team...

Why does 19200 Big Macs a year equal potential job security? Why can South African software developers buy more Big Macs than their American counterparts, and, what on earth do Big Macs and software development have in common anyway? All these questions are answered HERE when considering 5 key reasons as to why software development is a great career to find yourself in.

-The Hyperion Team

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, len(), int(), str() and float(). 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 know 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. To find out more about the math module and its functions, go to https://docs.python.org/2/library/math.html Python modules and their functions are usually very well documented online.

Instructions

Before you get started we strongly suggest you start using Notepad++ or IDLE to open all text files (.txt) and python files (.py). Do not use the normal Windows notepad as it will be much harder to read.

First read example.py, open it using Notepad++ (Right click the file and select 'Edit with 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.

- You are not required to read the entirety of Additional Reading.pdf, it is purely for extra reference.

Compulsory Task

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!).

Things to look out for:

1. Make sure that you have installed and setup all programs correctly. You have setup **Dropbox** correctly if you are reading this, but **Python or Notepad++** may not be installed correctly.
2. If you are not using Windows, please ask your mentor for alternative instructions.

Give your thoughts..

RATE

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.