Python-3

April 18, 2019

1 Python Functions

In Python, function is a group of related statements that perform a specific task.

Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.

Furthermore, it avoids repetition and makes code reusable.

1.0.1 Syntax of functions

def function_name(parameters): """“docstring”""" statement(s)

1.0.2 Docstring

The first string after the function header is called the docstring and is short for documentation string. It is used to explain in brief, what a function does.

Although optional, documentation is a good programming practice. Unless you can remember what you had for dinner last week, always document your code.

In the above example, we have a docstring immediately below the function header. We generally use triple quotes so that docstring can extend up to multiple lines. This string is available to us as **doc** attribute of the function.

1.0.3 Type of functions

Types of Functions Basically, we can divide functions into the following two types:

Built-in functions - Functions that are built into Python. User-defined functions - Functions defined by the users themselves.

1.0.4 Function Arguments

1. Default Arguments
2. Keyword Arguments
3. Arbitrary Arguments

1.0.5 Recursive Function

A function can call other functions. It is even possible for the function to call itself. These type of construct are termed as recursive functions.

1.0.6 Anonymous Function

In Python, anonymous function is a function that is defined without a name.

While normal functions are defined using the def keyword, in Python anonymous functions are defined using the lambda keyword.

Hence, anonymous functions are also called lambda functions.

1.0.7 Scope of variables

Scope of a variable is the portion of a program where the variable is recognized.

2 Python Modules

Modules refer to a file containing Python statements and definitions.

A file containing Python code, for e.g.: example.py, is called a module and its module name would be example.

We use modules to break down large programs into small manageable and organized files. Furthermore, modules provide reusability of code.

We can define our most used functions in a module and import it, instead of copying their definitions into different programs.

3 Python Packages

We don't usually store all of our files in our computer in the same location. We use a well-organized hierarchy of directories for easier access.

Similar files are kept in the same directory, for example, we may keep all the songs in the "music" directory. Analogous to this, Python has packages for directories and modules for files.

As our application program grows larger in size with a lot of modules, we place similar modules in one package and different modules in different packages. This makes a project (program) easy to manage and conceptually clear.

Similar, as a directory can contain sub-directories and files, a Python package can have sub-packages and modules.

A directory must contain a file named **init**.py in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file.