|  |  |
| --- | --- |
| Name | Radoslaw Hankiewicz |
| Date | 11/21/2021 |
| Course | Foundation of Programming |
| Title | Assignment 6 |

# **Introduction**

The purpose of this document is to present what I learned in the sixth module of the course.

The sixth module of the course presents some new information about functions, functions parameters, return values, and multiple return values, positional and names arguments, reference DataTypes in arguments, keyword None, difference between local and global variables, as well as some information regarding shadowing global variables, function document headers and classes.

**Assignment\_06**

**GitHub link**: <https://github.com/Radohan/Assignment_06>

In the Assignment\_06 we were asked to follow the instructions and modify the starter script, so it uses functions more + add docstrings and new information to the header and save it as CDInventory.py.

A screenshot of a computer

Description automatically generated

Figure 1 – Assignment\_06 in Spyder

Text

Description automatically generated

Figure 2 – Assignment\_06 in Anaconda

**Function parameters**

Parameters (arguments) can be passed in with functions.

They pass in values for processing.

It’s possible to pass in as many arguments as necessary.

# **Naming conventions**

Naming conventions are sets of rules for choosing the character sequence should be used for identifiers which denote variables, types, functions, and other entities in source code and documentation.

In Python we use:

* **UpperCamelCase** – class names,
* **CAPITALIZED\_WITH\_UNDERSCORES** – constants,
* **lowercase\_separated\_by\_underscores** – other names.

**Return values**

# The return values can be:

* consumed instantly (e.g., print statement),
* assigned to a variable (allows to use the results multiple times without having to call the function each time).

# **Multiple return values**

# Return values can be:

* **single item**,
* **multiple items** – when being returned, they should be bundled into a collection, in Python: **Tuple**.

# **Positional vs. named arguments**

To call a function 2 types of arguments can be used:

* **named arguments**: when the name of the parameter was included (as defined in the function declaration);
* **positional arguments**: used when parameters implicitly use the arguments and assign them in the same sequence as the parameters are defined in the function declaration.

It is possible to mix positional and named arguments.

# **Default parameter values**

It's possible to set default values for a parameter.

This value will be used in case no value for the parameter is given when the function is being called.

# **Overloaded functions**

Python does not allow you to have two functions with the same name but allows to use default values to accomplish similar result as with the multiple functions.

# **The None keyword**

The **None** keyword is a special **DataType**.

It has one value: **None**.

It is used to indicate the **absence of values** and is also possible as a parameter value.

When used as default argument it allows for simple checks to change the behavior of functions.

# **Reference DataTypes in attributes**

In Python, values with:

* **immutable DataTypes** (string, integers, floats, and tuples) pass to a function as **values;** make a copy of the values;
* **mutable DataTypes** (lists and dictionaries) pass to a function as **reference;** make a copy of the reference to the values.

# **Local vs. global variables and variable scope**

The variable scope is the region of a computer program where the binding is valid: where the name can be used to refer to the entity.

In order to be able to change variables that are defined on the parent level inside a child, in Python the keyword **global** can be used.

# **Shadowing a global variable**

Shadowing a global variable means that a variable declared within a certain scope has the same name as a variable declared in an outer scope.

# **Function document headers (docstring)**

In Python, frequently an explanatory header is added at the beginning of a function (known as **docstring**). The way the information is formatted in the docstring depends on the project, team, etc.

# **Classes and functions**

# Classes are used to group functions, variables, and constants.

# The **class** keyword is used to indicate the class object.

# **Summary**

The module no. 6 presented some new information about functions, functions parameters, return values, and multiple return values, positional and names arguments, reference DataTypes in arguments, keyword None, difference between local and global variables, as well as some information regarding shadowing global variables, function document headers and classes.