|  |  |
| --- | --- |
| Name | Radoslaw Hankiewicz |
| Date | 12/5/2021 |
| Course | Foundation of Programming |
| Title | Assignment 8 |

# **Introduction**

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

The seventh module of the course presents some new information about classes – basic information about fields, constructors, destructors, the keyword self, attributes, properties, methods, the \_\_str\_\_() method, static methods, private methods, decorators, type hints, and doc strings.

**Assignment\_08**

**GitHub link**: https://github.com/Radohan/Assignment\_08

For the assignment 08, a starter script was included, but no code. Our task was to read and understand the pseudocode, then add code to make the application work.

A screenshot of a computer

Description automatically generated

Figure 1 – Assignment\_08 in Spyder

Text

Description automatically generated

Figure 2 – Assignment\_08 in Anaconda

# **Fields**

Fields are the data stores of a class. Fields get created same way as variables.

# **Constructors**

Constructors are a dedicated method invoked when creating an object.

Constructors are a convenient way to ensure proper datatypes in the fields.

They allow for a pre-population (defaults) of values to the fields.

Python’s constructor method is the dunder init method: **\_\_init\_\_()**.

**Destructors**

Destructors are used when an object gets destroyed or de-allocated.

They are responsible of freeing up the memory used by the object, cleaning up references and similar tasks.

In Python, usually you don’t need to use destructors, as the cleanup is done by the runtime.

**Keyword self**

Self is not an official keyword. Its use and convention in the community is so universal.

It is the first parameter in every method.

Every method that is called by an object automatically receives a reference to that object (this way the class knows on which object to use the methods).

# **Attributes**

In Python, Attributes are internal fields or variables that hold data. Below code demonstrates implicitly created fields.

# **Properties**

Properties are special methods allowing controlling validity of values assigned to attributes in a class by making these attributes private and enforce the interaction with them thru methods.

Typically, two methods are created for each attribute: one to set it and one to access it. Most often they are called “getter” or “accessor” for reading (or getting) the attribute and ”setter” for writing (or setting) of the attribute.

In Python, you make an attribute private by pre-pending a double underscore to its name.

# **Methods**

Methods are functions in a script.

Like functions, they allow to organize statements into blocks that can be invoked by calling the method’s name.

The between functions and methods difference is that a method call also submits a reference to the object it’s invoked on, so the first attribute supplied to a method is the “self” reference.

# **Decorators**

Decorators wrap a functionality around a function (or method). It is possible to write custom wrappers (or decorators).

# **Summary**

The module no. 7 presented some new information about some new information about classes – basic information about fields, constructors, destructors, the keyword self, attributes, properties, methods, the \_\_str\_\_() method, static methods, private methods, decorators, type hints, and doc strings.