

Multi-Paradigm Programming

Shop Assignment

Killian Foley | Multi-Paradigm Programming | December 2021 | G00387875

Lecturer: Dr. Dominic Carr

# **Summary**

This report describes some of the differences and similarities between the procedural and object oriented (OOP) programming methods in the context of the shop program constructed using the C language (procedural) and Python (procedural & OOP) as part of the course Multi-Paradigm Programming.

## **Introduction**

The assignment for the module Multi-Paradigm Programming required the creation of a shop program using languages C and Python, and two distinctive programming paradigms, procedural and object oriented (OOP). This report details some of the similarities and differences observed between the two programming paradigms and will explain how each of the programs are similar and how they fundamentally differ in construction.

**Programming Paradigm**

Paradigm in the context of programming is the style or method of thinking about and approaching problems. How computer code is written and structured is heavily affected by the chosen paradigm. It can have a large influence in how the problem explored is solved by the programmer. The nature of programming paradigms makes it that some map more easily to certain types of problems than others, there is no one size fits all approach. Each paradigm has its advantages and disadvantages some of which will be explored in this report. Regardless of paradigm or language chosen, computers work and understand things at a very low level i.e., binary, or a series of 1’s and 0’s. Some of the various programming paradigms include

* Imperative/ procedural
* Functional
* Object-oriented
* Declarative
* Data Flow.

The focus of this assignment and hence report is to learn and understand how to program in both paradigms and how to identify differences and similarities between procedural and object-oriented paradigms.

**Procedural Programming**

Imperative programming is the method of programming with an explicit sequence of commands that update the state of the program. Procedural programming is an extension of imperative programming with the addition of using procedure calls often referred to as methods, or functions. Procedural programming is derived from structured programming whereby procedures are called in a specific order. It follows a top-down approach not too dissimilar to that of a cooking recipe whereby one would start from the top of the recipe and read down through the recipe which is broken into a collection of discrete steps or methods. To reduce complexity and avoid repetition procedural programming groups instructions into re-usable elements called procedures when can then be called multiple times when required during the execution of the program. An example of a method or function from the shop.c program is the printProduct function below

void printProduct(struct Product p)

The void keyword indicates that the function does not return a value. What it does is execute a set of commands based on the input arguments inside the parentheses (the product struct). These commands can be re-used without typing them repeatedly by calling the function example below. In this case the function is contained inside a for loop iterating through the products in the shop printing the product name and the current stock held by the shop.

printProduct(shop->stock[i].product);

**Object-Oriented Programming**

In object-oriented programming the program is divided into small parts called objects. An object has both a state and functionality. It is more real world than procedural in that an Animal may be an object. It can have an instance of the object such as a cat which has various states such as species, age, colour, gender etc. Object-Oriented programming builds up libraries of reusable Objects (code) which is demonstrated in the shop\_oop.py program.

**Comparisons between Shop Procedural and Shop Object-Oriented**

In the shop\_oop.py program there is a class defined as Product. It has objects which are the products stocked in the shop such as Bread, Jam etc. example of code below. They states which are name and price

class Product:

def \_\_init\_\_(self, name, price=0):

self.name = name

self.price = price

Inside this class is a representation method (\_\_repr\_\_) below which allows for printing of the product in this case.

def \_\_repr\_\_(self):

return f'Product: {self.name} Price: €{self.price}'

This is in contrast to the previously mentioned method of printing a product in the shop.c program. In the shop.c program there is a “struct” Product. A struct is a grouping of variables under a single name. It has state but not functionality. Modifications to the state are performed by functions external to the struct which is a major difference between these two programming paradigms. In order to print information about a product such as name or price and printProduct method must be created similar to outlined in the procedural section of this report.

The object oriented method allows for classes to be inherited over various levels. In the shop\_oop.py program there is a ProductStock class which inherits the product name and price from the “Product” class. Inside the ProductStock class are methods which return certain states of Product objects such as price, name and cost. The code required to create the methods is a lot more simplistic than that of the procedural method however the theory is actually the same. In OOP the method is contained within the class allowing for easier debugging and changes to code. In procedural programming the methods are created external to the class (struct in C) but the principle is the same in that the data exists and the method performs a set of tasks on or with this data.

In OOP the use of the parameter “self” is quite common place. It is used to represent the instance of the class. Often thought to be a keyword in Python, however anything can be used. Self is just preferred to increase readability of the code. Procedural programming differs in that methods require independent parameters be passed in to perform any operations rather than have them perform on themselves.

The procedural nature of execution i.e. top down, is notable in the procedural paradigm (C program) in comparison to the OO paradigm most notably when constructing and testing the code. When compiling the program in C if a function has not occurred previously, or been seen by the compiler, the program will not run and an error will be raised. It cannot jump forward and backwards as is the way in OOP. In Object oriented however the line number of the method does not matter and it does not require a defined “order”.

Object oriented programs due to their modularity allow for new data objects to be created easily from existing objects. This was noted by the writer when writing code to update stock. An item that was previously not stocked would be created and have its stock updated easily in the shop\_oop program. This was not the case in the procedural programs.

One similarity is the use of functions themselves. Although the technique differs as discussed previously there is some common ground in how the functions can be constructed and prevent duplication of code.

Another notable similarity between the two paradigms is the importance of state and how it affects the behaviour of the program. This is demonstrable in the shop program whereby the stock of a product must be updated after every order so as to avoid a future customer attempting to buy a product that is not in stock. Another example of state is the customers budget and the cash available to the shop. These are two examples of state which must be updated to ensure accurate functionality of the shop program.

The execution of the code can be seen as a similarity. In all three cases a “main” function exists whereby the program is instructed how to call the initial functions and how to proceed. After the initial execution of the program however, the differences are apparent. The point here is method of initiating the program via a “main” method is the same.

**Conclusion**

The purpose of this assignment was to explore the differences and similarities between object-oriented programming paradigm and procedural paradigm by constructing a virtual shop using C (Procedural) and Python programming languages (Procedural and OOP). Using the three programs created as part of the assignment, some of the key similarities and differences were detailed in report above. The learning outcomes of this assignment were

* Ability to compare different programming paradigms
* Write programs using a variety of different paradigms

**References:**

Dr. D. Carr. GMIT – Course notes accessed via Moodle MPP page December 2021.

* Imperative/ Procedural Programming
* Object Oriented Programming.
* The C Programming Language

Website: object-oriented programming (OOP) (https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP)

Website: Wikipedia. Programming paradigm. https://en.wikipedia.org/wiki/Programming\_paradigm

Website: Imperative programming. https://en.wikipedia.org/wiki/Imperative\_programming

Website: Real Pythons Instance Class… https://realpython.com/instance-class-and-static-methods-demystified/

Website: Geeksforgeeks.org. https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/

Website: Tutorials Point. https://www.tutorialspoint.com/differences-between-procedural-and-object-oriented-programming