Procedural and Object Oriented Approach to Coding

Procedural Programming

Procedural programming is generally the first method people learn when they begin their coding career. It is a list of instructions beginning at the top and executing line by line one after another and is based on the concept of calling procedures.

Procedures are blocks of code stored under a name that can be called at anytime to execute the listed instructions. Arguments can be also passed into the procedures to alter the results obtained. These named blocks of code are also known as method, functions routines or subroutines.

Some examples of procedural based languages are C, BASIC, Java, Python and Pascal.

Object-Oriented Programming

Object-oriented programming (OOP) is a programming paradigm based on objects. Objects are representation entities. It focuses on specific states and the functionality of the object and offer reusable code for future projects. The classes used in OOP are used as a blueprint for creation of the objects.

Some OOP languages are known as pure languages such as Ruby where everything in the language is an object. Java and C++ would be examples of impure languages as they have some primitive data types that are not used as objects such as int, char, float and double.

Inheritance is another key feature of object oriented programming. Inheritance is a relation between two classes. The inherited is known as the superclass and the inheritor is known as the subclass. The subclass takes on the properties of the superclass. This inheritance feature promotes reusability of the code.

Python is a multi-paradigm language in that it can be used in both a procedural and an OOP style. For the assignment given to us in the multi paradigm programming module of the data analytics course we were tasked with creating a shop in procedural C, python and OOP python.

Analysis of similarities

The same expressions and syntax can be used in both procedural and OOP in Python such as the use of if __name__=”main”: indicating the start of the code to be read on execution. This would be

expected as it is the same language but a few adjustments do need to be made for calling functions or objects depending on the requirement of the expression or returned results. For example in the OOP shop code each class had a def __repr__ function that returns a string result that is a representation of the object.

In both programming styles the larger programme is divided up into several parts. In OOP these are known as objects and in procedural they are known as functions. They share traits such as both calling on functions (known as methods in OOP) to perform required actions on arguments or objects called.

The same expressions and syntax can be used in both procedural and OOP in Python. This would be expected as it is the same language but a few adjustments do need to be made for calling functions or objects depending on the requirement of the expression or returned results. For example to return a string result in OOP that is a representation of the object the def __repr__ method needs to be used.

In OOP objects can communicate with each other using member methods. This can be seen from our programme where the customer object communicates with the shop object through its methods to determine the price of the product in the shopping list and that there is adequate stock in the shop from checking the Shop object calling shop.stock.

In procedural programming data is moved freely between functions in a similar manner with the stock check and costs calculation functions bringing in the previously stocked shop details and customer details in using the arguments of the returned values cust and shop from the previously run functions of def print_customer(cust, shop) and def print_shop(shop).

Analysis of differences

From a first glance through our code for the assignment we can see there are several differences between OOP and procedural programming. From the top we can see that the classes in the OOP do more than just hold the information as shown in the procedural python. The classes in OOP hold information but also contain the methods required to manipulate the information received and provide required outputs.

OOP is based on real world objects where the data as opposed to the functions and procedures is given the importance. This is shown with the focus of the OOP shop being on manipulation of the objects eg. shop, customer etc. Procedural programming on the other hand places importance on the functions and sequences of execution over the data itself. Logical grouping of functions and

methods is not required within the procedural programming section itself as the execution is a step by step methodology. As programmes get larger it becomes less practical to use a procedural method as making any changes to the code becomes a harder process.

Procedural programming doesn't offer any way of hiding its code making the code less secure than its OOP counterpart. While we didn't need to use the security measures in OOP in the shop assignment there are what's known as access specifiers for private, public and protected for limiting how much access users should have to the code itselfand also to give an idea of the thought process behind the code to any future developers who may be working on the code.

In OOP programming a constructor is called whenever a new instance of an object is created. This is shown in our code within each class by the def __init__ method. This is not required by the procedural programming.

The procedural file relied on the various functions and variables to calculate new instances of the requested information while the OOP used classes with the methods within the class itself to create the object and then print out a text representation of the object. The self parameter is used in the OOP to represent the instance of the class to access the attributes and methods within the class. This is not used with the procedural section of the code as the classes in this section are only used for storage of information.

Sources-

[1] Kiang H., W. (2020). towardsdatascience.com, Python: Procedural or Object-Oriented Programming? [Online]. Available at: https://towardsdatascience.com/python-procedural-or-object-oriented-programming-42c66a008676

[2] Patel, P. (2019). geeksforgeeks.org. Differences between Procedural and Object Oriented Programming. [Online]. Available at: https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/

[3] (2020). vivadifferences.com. Understanding Procedural Vs. Object Oriented Programming With Example [Online]. Available at: https://vivadifferences.com/difference-between-procedural-and-object-oriented-programming-with-example/

[4] Frankenmint. (2016). codementor.io. Comparing Programming Paradigms: Procedural Programming vs Object-oriented Programming [Online]. Available at:

https://www.codementor.io/learn-programming/comparing-programming-paradigms-procedural-programming-vs-object-oriented-programming

[5] Malik, U. (2018). stackabuse.com. Object Oriented Programming in Python [Online]. Available at: https://stackabuse.com/object-oriented-programming-in-python/