# Multi Paradigm Programming Project

**Procedural Programming language:**

One of the most common programming methods is Procedural programming which follows a set of commands in an order. Fundamentally, the procedural code is the one that directly instructs a device on how to finish a task in logical steps.[2]

As its name stands out, procedural programming style involves writing a list of steps in an order that should be followed by the program/computer to finish a task. This approach is called a top-down approach and that treats data and procedures as two different entities.

With this programming usually we start with main functions and this part holds the main procedures to be used or called to do the task. To do the tasks, the program is then divided into small components that will solve the main function call.

A programmer can import Pre-defined functions written in 'high level' languages from the library or the registry instead of re-writing the functionality which saves costs and time.

A set of instructions can be grouped together into a reusable element called procedures which can then be called when needed during program execution. It is also known as routines, subroutines, or functions. We can include parameters to these procedures which can be done through 'Pass by value,' 'pass by reference,' 'pass by result,' 'pass by value-result' and 'pass by the name.'

**Key features of Procedural Programming language:**

1. Predefined functions
2. Programming libraries
3. Local and Global variables
4. Procedures and Parameter passing
5. Modularity

**Advantages:**

- Procedural Programming is excellent for general-purpose programming and easy to learn and implement.
- The code can be reused in various parts of the program, without the need to copy it
- The program flow can be tracked easily
- It consumes less memory of the computer
- It is easier for tracking the flow of the codes

**Disadvantage:**

- It focuses on functions rather than data
- While preparing a large program, it is difficult to identify the usage of global data
- Even though global data is easier handle it is error prone and very cost effective when it comes to maintenance and enhancements
- The modification of global data requires modification of all the functions which are using the global data
- It is also exceedingly difficult to track the errors when the program has too many procedures and calls.

**Procedural programming key features used in C and Python program:**

**Shop Program written in C Language:**

As per Top-down model this Shop program has been written in the following order to achieve the functionalities said in the assignment

1. Importing the header files with extension .h which holds C function declaration and macro definitions

   ```
   #include <stdio.h>
   #include <string.h>
   #include <stdlib.h>
   #include <dirent.h>
   #include <errno.h>
   #include <ctype.h>
   ```
2. Global variable declaration part to hold the global variables that are used in the program
3. Structure definition part – Struct is a composite data type with a group of variables under one block of memory
   I have few structs to group a list of variables used to collect/ store the values of a data

   ```
   E.g., struct Product {
           Char* name.
           Doubtle price.
           }
   Struct ProductStock {
           Struct Product product;
           Int quantity.
           }
   ```
   From the above Product Structure has two variables that are used to store name and price. This is how the attributes/ variable of the data type Product will be stored together.

   The second one is ProductStock which has Product Struct as one of its data. Here we must use Struct keyword in front of Product to refer to the Product Struct.

   Most important part while using Struct in the program is, we must declare the Structs in the first place before referring them from another Structs.

4. Function declaration part of all functions used in the program

   Once all the data type are declared we have to declare the functions that are used in the program. This is to tell the compiler about the function’s name, return type and parameters.

   e.g., void printProduct(struct Product);
   void printCustomer(struct Customer, struct Shop *);

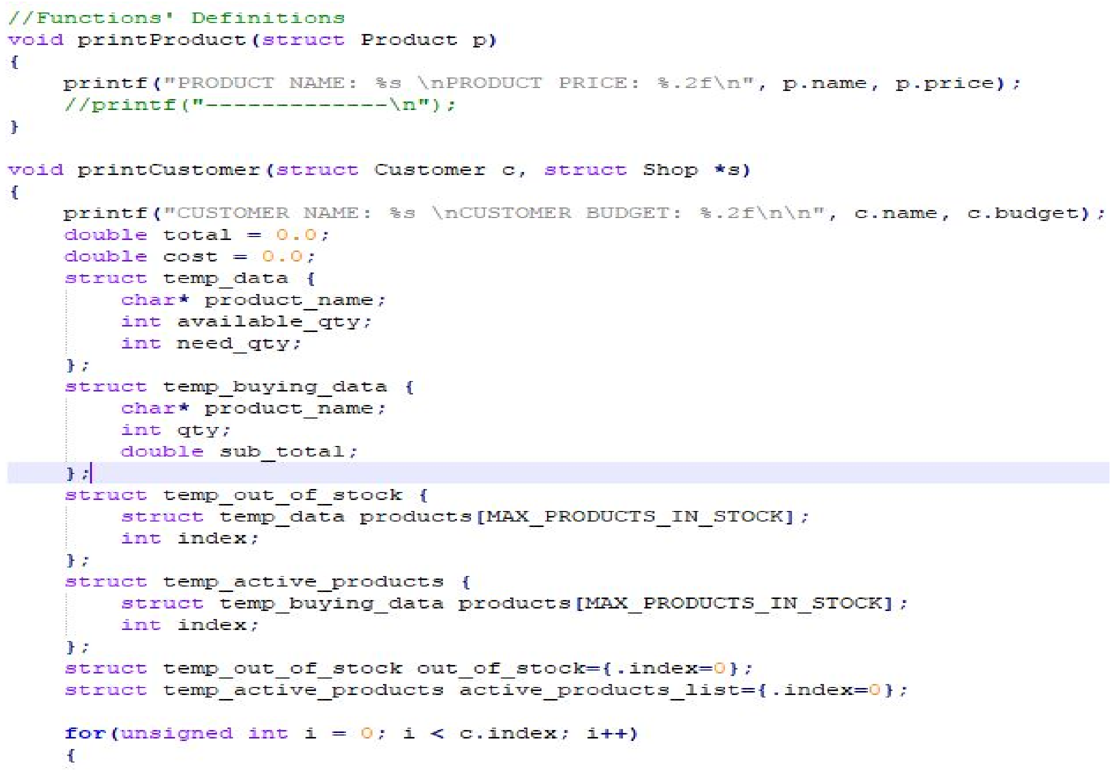
5. Function definition part of all functions to process / implement the steps to produce the result back to the Main program

   e.g., PrintProduct will take the Struct Product as its parameter and prints the details on the screen as per the steps included in it.

   e.g., PrintCustomer will take two different Structs Customer and Shop as parameters and process steps included inside the function.

   This function has local variables which are only used by this Function and are not shared by other functions.

   e.g, double total = 0.0; [ A double type variable “total” is declared with default value 0.0]



```
//Functions' Definitions
void printProduct(struct Product p)
{
    printf("PRODUCT NAME: %s \nPRODUCT PRICE: %.2f\n", p.name, p.price);
    //printf("-------------\n");
}

void printCustomer(struct Customer c, struct Shop *s)
{
    printf("CUSTOMER NAME: %s \nCUSTOMER BUDGET: %.2f\n\n", c.name, c.budget);
    double total = 0.0;
    double cost = 0.0;
    struct temp_data {
        char* product_name;
        int available_qty;
        int need_qty;
    };
    struct temp_buying_data {
        char* product_name;
        int qty;
        double sub_total;
    };
    struct temp_out_of_stock {
        struct temp_data products[MAX_PRODUCTS_IN_STOCK];
        int index;
    };
    struct temp_active_products {
        struct temp_buying_data products[MAX_PRODUCTS_IN_STOCK];
        int index;
    };
    struct temp_out_of_stock out_of_stock={.index=0};
    struct temp_active_products active_products_list={.index=0};

    for(unsigned int i = 0; i < c.index; i++)
    {
```

6. Main function – it acts as the entry point of the program it is from here that execution begins.

   I have the main Menu items for this program to get user input and based on their choice a Switch case will be activated to call the required functions to process the user input.

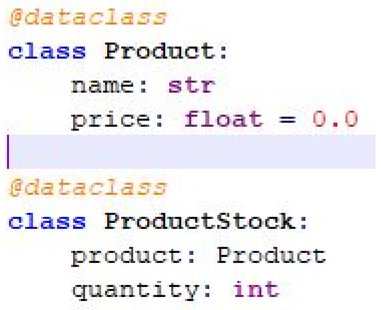
**Shop Program written in Python:**

As per Top-down model this Shop program has been written in the following order to achieve the functionalities said in the assignment

1. Importing the standard modules to use scripts from another python modules

   from dataclasses import dataclass, field
   from typing import List
   import csv
   import re
   import os
   from glob import glob

2. Dataclass definition part – dataclass is a composite data type with a group of variables under one block of memory
   I have few dataclasses to group a list of variables used to collect/ store the values of a data



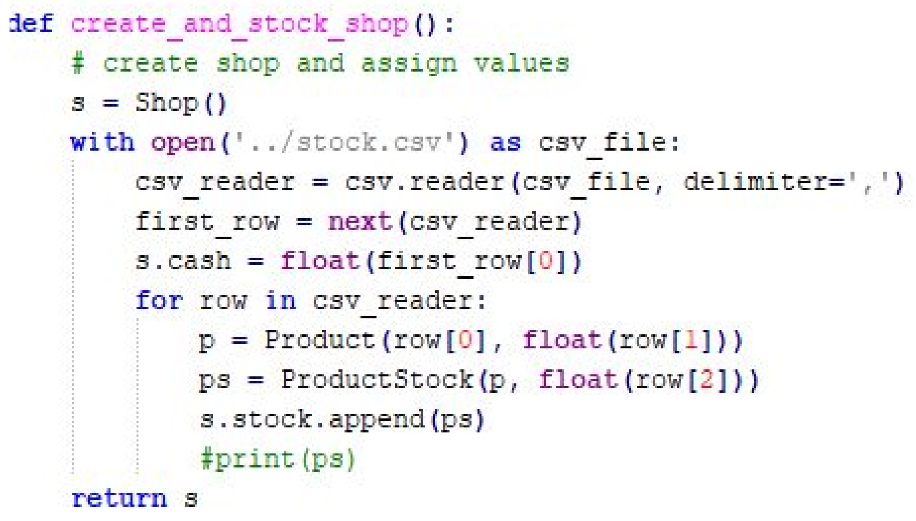
```
@dataclass
class Product:
    name: str
    price: float = 0.0

@dataclass
class ProductStock:
    product: Product
    quantity: int
```

   From the above Product dataclass has two variables that are used to store name and price. This is how the attributes/ variable of the data type Product will be stored together.

   The second one is ProductStock which has Product dataclass as one of its data. Here we must use dataclass keyword in front of Product to refer the Product dataclass.

   The most important part while using Struct in the program is, we must declare the dataclass in the first place before referring them from another dataclass.

3. Function definition part: Function definition part of all functions to process / implement the steps to produce the result back to the Main program


```
def create_and_stock_shop():
    # create shop and assign values
    s = Shop()
    with open('../stock.csv') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        first_row = next(csv_reader)
        s.cash = float(first_row[0])
        for row in csv_reader:
            p = Product(row[0], float(row[1]))
            ps = ProductStock(p, float(row[2]))
            s.stock.append(ps)
            #print(ps)
    return s
```

4. Main function of the program

   I have the main Menu items for this program to get user input and based on their choice a Switch case will be activated to call the required functions to process the user input.

**Object Oriented Programming:**

Object oriented is all about encapsulating data and its behavior into objects. An object is a component of a program that knows how to perform certain actions and how to perform certain actions and how to interact with other elements of the application. It is an abstract data type created by the developer and includes multiple properties and methods. A method in OOP is like a procedure in procedural programming but the difference is that the method is a part of an object. A class is a template to create an object that has both variables(data) and behaviors (functions or methods).

Added benefits of using OOP increases reusability of code, scalability, and efficiency. Abstraction, encapsulation, inheritance, and polymorphism are the basic concepts of Object-oriented programming.

**Abstraction**: It means using simple things to handle complexity by hiding internal details from the user. Users just need to know which methods of the object are available to use in the call statements and what are the attributes to be used to pass the details to get the required result from it. User does not need to know what the internal procedures are used in the methods.
**Encapsulation**: It refers to the bundling of data in a single unit along with the methods that operate on that data. It hides the implementation details from users and if a data member of a class is declared as private to that class means it can only be accessed within the same class. Any other classes in the program can access a 'private' data member of other class

**Inheritance**: One of the most important key concepts in OOP is Inheritance. It is the procedure in which one class inherits the attributes and methods of another class. The main class that holds the methods and properties that are inherited by other classes is called Parent class. The rest of the classes that are inheriting the behaviors of the Parent class are called child classes. With inheritance we can increase the functionality of the program by reusing the code and often it fastens the implementation time as well.

**Polymorphism**: Ability to have functionality inherited and implemented from parent class to sub classes and allows to implement multiple methods within the same class with the same name but a different set(s) of parameters. This is called method overloading

**Advantages:**

- Reuse of code by implementing Inheritance and flexibility through Polymorphism.
- Reliability and sustainability across different platforms. Security by hiding data and implantation methods in the form of abstraction.
- Easier to debug the code and error track to cut down the coding and debugging time.

**Disadvantages:**

- Planning and implantation could be difficult in the early stage of coding especially for those who are from Procedural programming.
- Typically slower than Procedural programs

**Object Oriented programming key features used in Python program:**

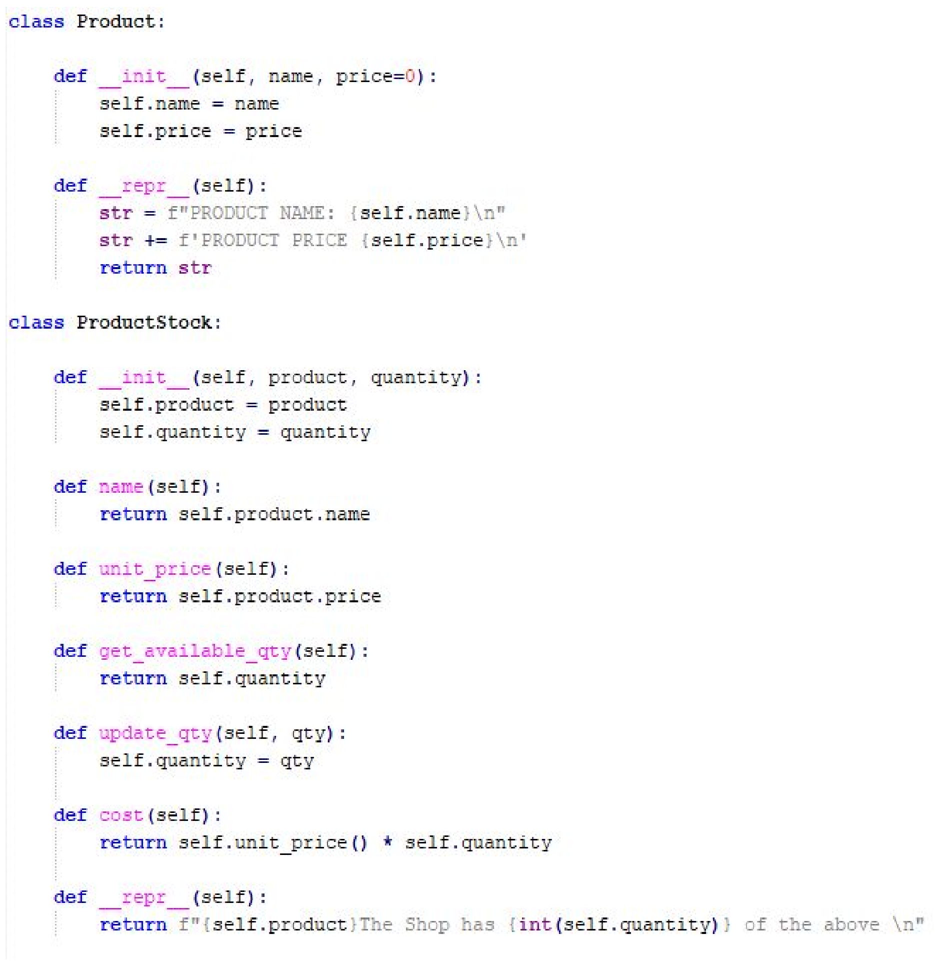
**Shop Program written in Python Language:**

With object-oriented programming model this Shop program has been written in the following order to achieve the functionalities said in the assignment

1. Importing the standard modules to use scripts from another python modules

   from typing import List
   import csv
   import re
   import os
   from glob import glob

2. Class definition part – As per OOP model I have added Classes which are used to create the data structures. Attributes may be data or functions.



```
class Product:

    def __init__(self, name, price=0):
        self.name = name
        self.price = price

    def __repr__(self):
        str = f"PRODUCT NAME: {self.name}\n"
        str += f'PRODUCT PRICE {self.price}\n'
        return str

class ProductStock:

    def __init__(self, product, quantity):
        self.product = product
        self.quantity = quantity

    def name(self):
        return self.product.name

    def unit_price(self):
        return self.product.price

    def get_available_qty(self):
        return self.quantity

    def update_qty(self, qty):
        self.quantity = qty

    def cost(self):
        return self.unit_price() * self.quantity

    def __repr__(self):
        return f"{self.product}The Shop has {int(self.quantity)} of the above \n"
```

From the above Product class has two variables that are used to store product name and price. The second one is ProductStock which has a subclass product class as one of its data.

__init__ is an object constructor that defines the default values upon calling the class.

Order of class definition is not a problem, but the required classes must be defined before calling them. From the above example, order of classes Product and ProductStock can be shuffled but creating the object of ProductStock it is required to define the Product class.

3. Main function of the program

   I have the main Menu items for this program to get user input and based on their choice a Switch case will be activated to call the required functions to process the user input.

**Analysis of similarities:**

| | **Procedural Programming** | **Object-oriented Programming** |
|---|---|---|
| Programming approach | Uses high-level language | Uses high-level language |
| Import of standard modules | Must be placed in the first place | Must be placed in the first place |
| Function definition | Functions must be defined at the first place before making a call | Classes must be defined at the first place before making a call |
| Referring another object | Using Struct we can refer another Struct as its data | Using class object, we can refer another class object as its data |

**Analysis of differences:**

| | **Procedural Programming** | **Object-oriented Programming** |
|---|---|---|
| Programming approach | Top-down | Bottom-up |
| | Procedure/Structure oriented | Object oriented |
| Order of Function definition | It must be in an order. There is no forward-declare available and you cannot call a function before defining it | There is no order of the classes required but all the classes must be defined before making a call |
| Function and Data | Function is more important than data | Data is more important than function |
| | Adding a new data and function is not easy | Adding a new data and function is easy |
| | Program is divided into small parts called functions | Program is divided into small parts called objects |
| | Attributes will be data | Attributes may be data or functions |
| Access Specifier | There is no access specifier | OOP has private, public and private access specifiers |
| Inheritance | Not allowed | Allowed |

Reference:

- Structured programming, https://www.techopedia.com/definition/16413/structured-programming
- Procedural Programming, https://hackr.io/blog/procedural-programming
- OOP vs Procedural Programming, https://study.com/academy/lesson/object-oriented-programming-vs-procedural-programming.html
- Multi Paradigm Programming, Class Material, https://learnonline.gmit.ie/pluginfile.php/136445/mod_resource/content/0/MPP%20-3%20-%20Imperative%20Procedural.pdf
- Encapsulation, https://www.sumologic.com/glossary/encapsulation/
- Inheritance, https://www.programiz.com/python-programming/inheritance