**Pivot-ED Assessment Part2-Unit1**

**Writing a function to check whether a number is odd or even. Question1 & Question2**

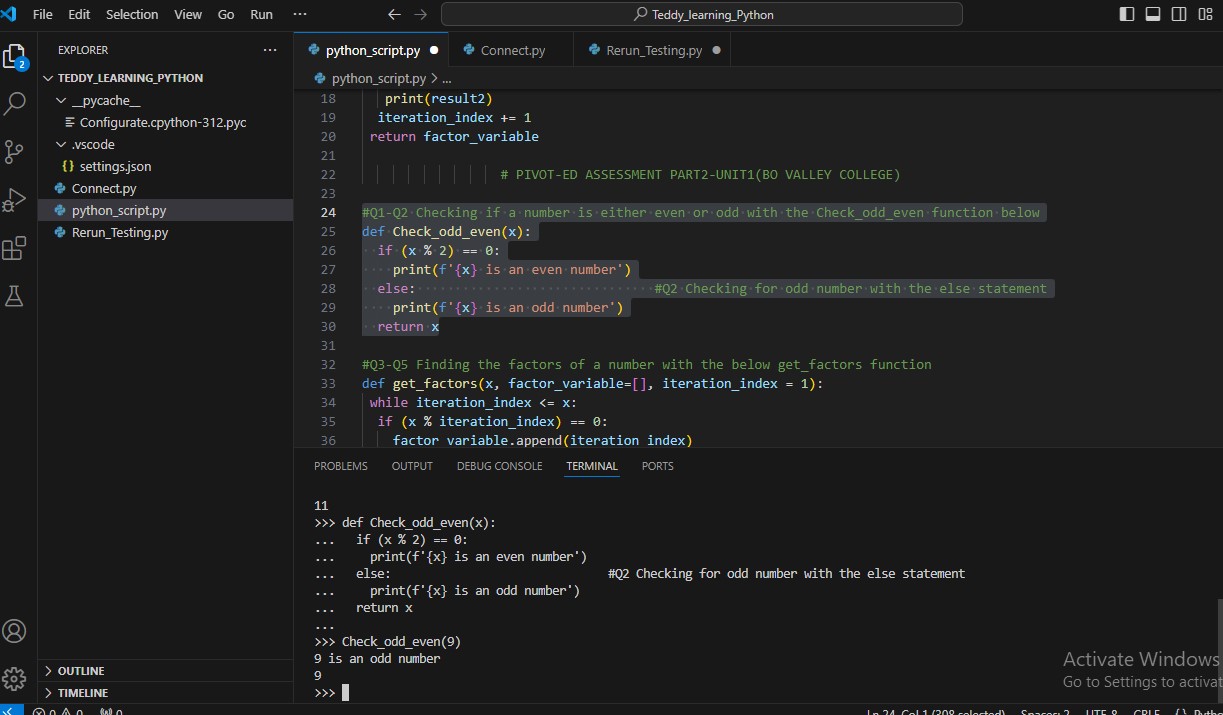
**Steps:**

***def Check\_odd\_even(x):*** **#** Defined Check\_odd\_even function where (x) is number parameter.

**If (x % 2) == 0:** **#** IF statement to check the modulus of (x) is 0.

***12 is an even number*** **#** Output

**Add an else statement** **#** else statement provides a solution incase the if condition(If (x % 2) == 0 ) is not fulfilled, for this solution, it let us know when the if condition is not fulfilled then we have a odd number

Below is a screenshot for the code:

**Writing a function to get the factors of a number. Question3 – Question4**

**Steps:**

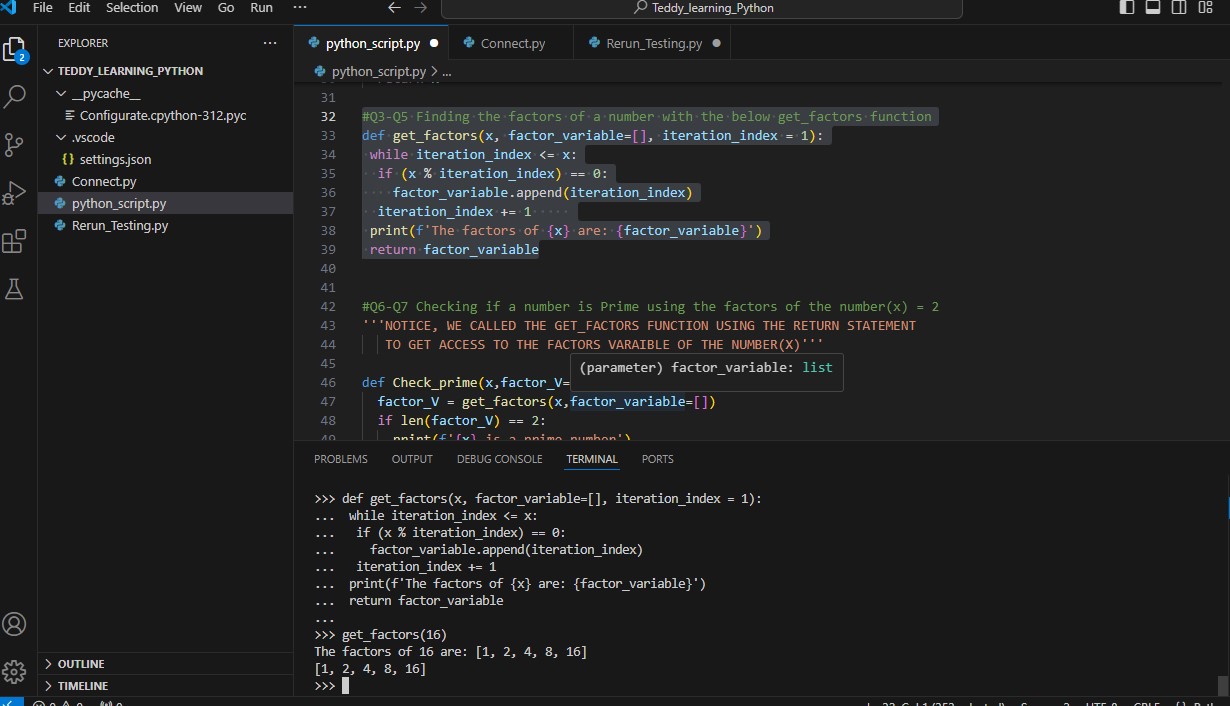
First, we need to define a function called get factors\_function and create a parameter that accepts number, assigning an empty list data type to a Factor\_variable and an Iteration\_index variable with (1) as default

Factors of a number: A number or set of number that divides a certain number with a remainder. In python programming language, it is represented by Modulus (%). For this function the factors of a number are going to be stored in the variable factors variable.

Iteration\_index: the positional value of a number when it increases or decreases at a particular interval

The while loop in python is one of the python inbuilt function that we can use to iterate through a number to a specified index, below screenshot shows it usage

Displaying python code, block of codes or function: The python print() inbuilt function is the most common function to displaying the output of codes, blocks of codes or function in python. In this assessment solution, the print() was used appropriately

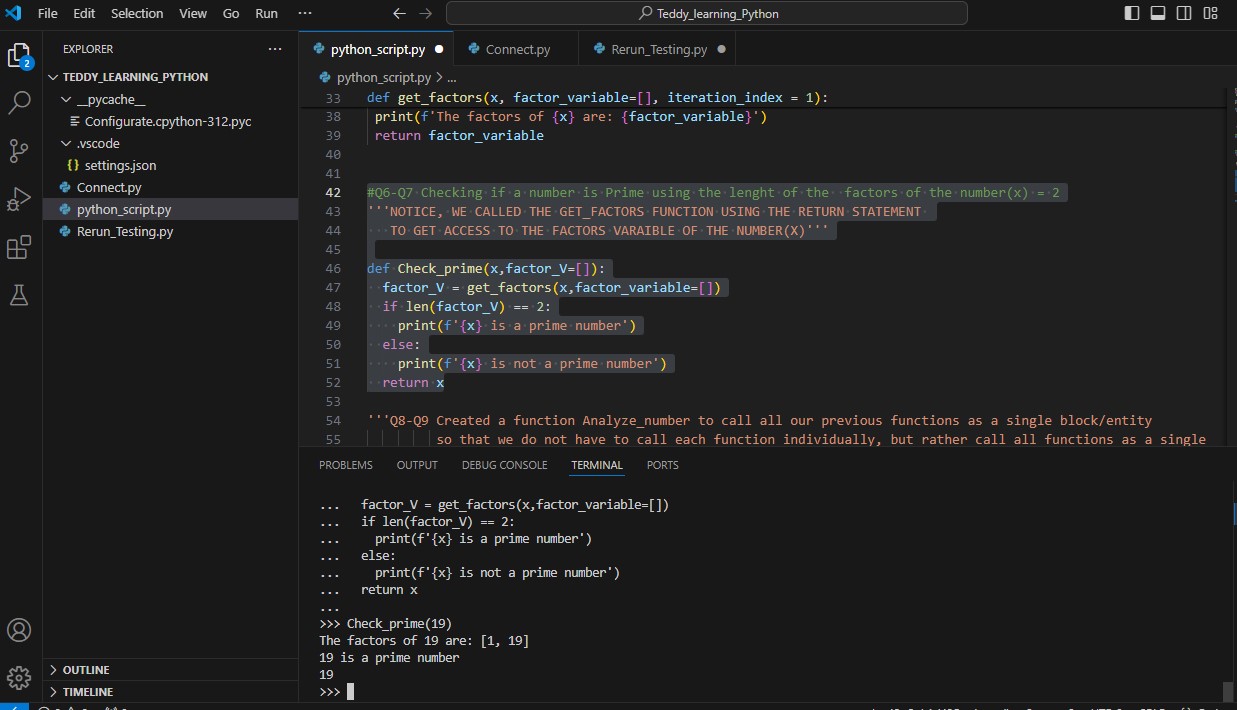
Return Statement: this statement is used if there is any intention to transfer the values of a variable in one function to another function for use, commonly it is known as calling up a function inside another function. The return statement was used in this solution to transfer the values of the Factor\_variable in the get\_factors function to Check\_prime function. Find screenshots below demonstrating the codes of the above explanation.

**Writing a function to check whether a number is prime or number using the factors of the number.**

**Question6-Question7**

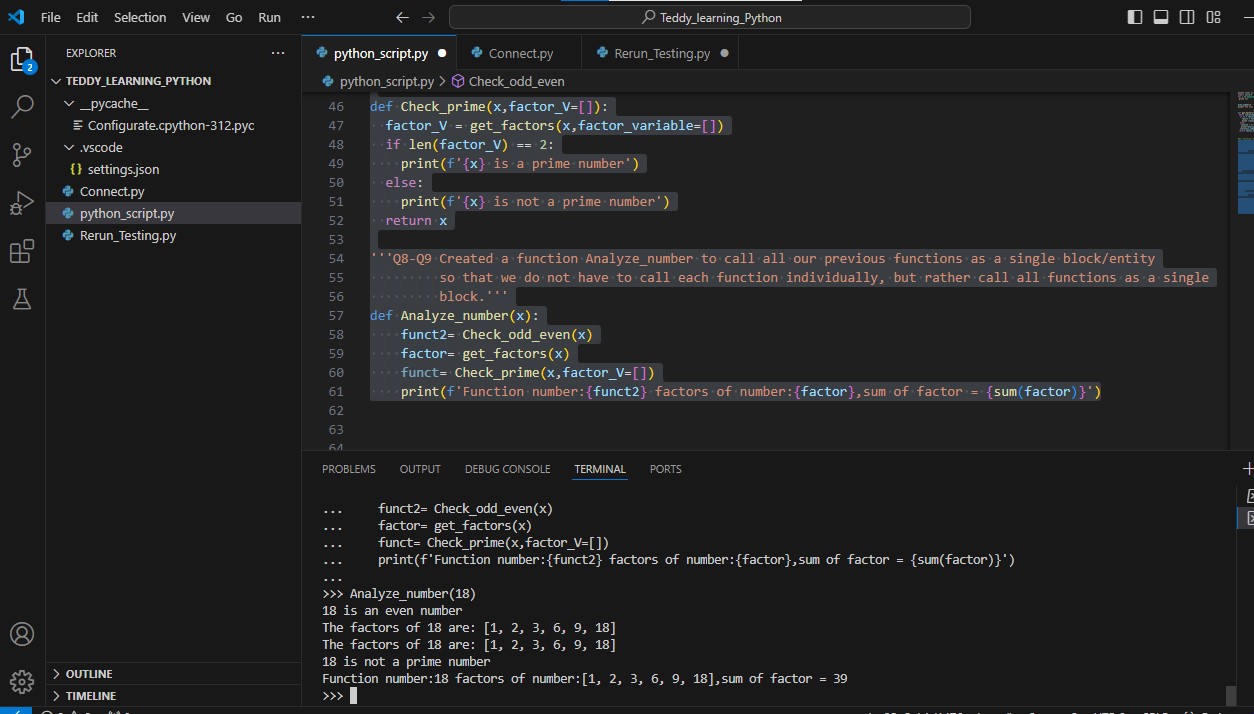
**From the question:**

The factors of the input number must be (i.e. length=2) for it to be called a prime number, and if peradventure the factors are more than (i.e. length>2), then the input number cannot be called a prime number, which the if and else statement in the function took care of.

 Below is a screenshot for the codes:

**Writing function to call the previous function we created earlier as a single block. Question8-Question9**

Created a function Analyze\_number to call all our previous functions as a single block/entity, so that, we do not have to call each function individually, but rather call all functions as a single block.

Below is the screenshots demonstrating this.

**Pivot-Ed Assessment part2- Unit2**

Analyzing business case (Case Study Analysis) is an approach of an intensive study/research of a person, group, event and or business with a aim of devising a systematic solution to a specific or centralized problem affecting the entity ( person, group, event, business). In case analysis thoroughness is inevitable i.e. nearly every aspect involved is analyzed to seek patterns and causes of behavior/trend.

Apropos to business environment, case study analysis is very crucial due to its preponderant effect on corporate Decision-making, performance enhancement and cost reduction. Having said that, it is ideal or high time to delve into a case study of ShopSmart, a major retail outlet in the retail industry: -An industry made of companies that sells goods and services directly to final consumers-.

From the background observation of ShopSmart shop, it was deduced that ShopSmart have some challenges which span from inventory management and product placement to the derived challenges of taunted sales volume which invariably pose a strong threat to the perpetuity of ShopSmart business in the industry, Thus, this case study is made to systematically analyze ShopSmart challenges step by step and proffer a logical solution that will informed the decision-making process of ShopSmart to stay in business going forward.

Furthermore, it is paramount to get acquainted to Corporate Inventory management as a subtopic to familiarize ourselves with inventory methods and practices to know the method that best fit in to solving ShopSmart inventory management lapses

INVENTORY MANAGEMENT: inventory management plays a pivotal role in the success of a business, regardless of its size because it is the process of controlling and closely monitors a company stock level to determine the best and appropriate time to reorder, store or use inventory in the production or business process of a firm ensuring that the right products are available in the right quantities and at the right time. Inventory management is a wide topic when it comes to business management practices, but for the sake of this study we are going to focus on inventory management methods, defining and discussing the pros and cons to further aid our decision of selecting the best and most appropriate inventory method to resolve ShopSmart issues at hand

INVENTORY MANAGEMENT METHOD:

1. JUST-IN-TIME INVENTORY (JIT)
2. MATERIAL REQUIREMENT PLANNING(MRP)
3. ECONOMIC ORDER QUANTITY (EOQ)
4. DAYS SALES OF INVENTORY(DSI)
5. ABC ANALYSIS

**JUST-IN-TIME INVENTORY (JIT)**: - Apparently this method originated from japan in the 1960s and 1970s due to the major contribution of TOYOTA MOTOR to its development. It is an inventory management method where firm receives supplies only when it is needed for production and when there is a demand for or through customer orders.

**Pros:-**

It reduces inventory waste and excesses

It reduces storage cost and insurance cost

It also reduces the risk of inventory obsolesce

**Cons:-**

Relying on JIT can be very risky because during the period of unexpected demand spikes, management may not be able to source the stocks needed to meet up with the spike in demand thereby damaging its reputation with customers which invariably create a strong competition that may send the firm out of business.

**MATERIAL REQUIREMENT PLANNING(MRP): -**

This time of inventory management method is sales-forecast dependent i.e. management must have an accurate sales record in hand to prelude timely planning of inventory needs while communicating this needs to suppliers as soon as possible.

**Pros:-**

It reduces inventory holding cost

It reduces customer lead time

it enables real-time tracking of inventory

improves customers satisfaction

**Con:-**

Relies on accurate data forecast

It could create some inflexibility in the inventory process

High maintenance cost

**ECONOMIC ORDER QUANTITY (EOQ):-**

The EOQ model ensures that the right amount of inventory is ordered per batch so a company does not have to make orders too frequently and there is not an excess of inventory sitting on hand. EOQ reorder quantity per batch is dependent on an assumed constant consumer demand, ordering cost and carrying cost aimed to getting a balance between availability of inventory and cost efficiencies

**Pros:-**

Because its approach is based on forecasted customer demands, hence it improves customer satisfaction

It shields the company against unexpected negative effect of a hike in the price of raw materials or inventories

It reduces ordering and carrying cost

**Cons:-**

It depends widely on accurate data

Sometimes an assumed constant demands is not always realistic

Any fluctuation in demand or reorder cost can cause a great damage that may result to either shortage of inventory or excess inventory issues.

**DAYS SALES OF INVENTORY (DSI): -**

This method is a metric that indicates the average number of days it takes for a firm to turn its inventory into sales, lower DSI values depicts efficient inventory management and faster inventory turnover. DSI is also known as average day of inventory, Days inventory outstanding etc.

**Pros:-**

It improves cash flows

It can improve sales turnover

It also can improve profit

**Con:-**

The high need for accurate sales forecast

The overall reliance on suppliers to meet up with daily inventory demands

There may be difficulty to managing daily cash inflows efficiently

**ABC ANALYSIS: -** it is widely used inventory management technique because it focuses on the importance of inventories because on their contribution to total sales or contribution to total profit of a firm. This technique purport to the maxim “FIRST THINGS FIRST” which means the most important should be given most attention i.e. prioritizing the product which is considered more important in terms of restocking, promotions and development etc.

**Pros:-**

Better decision making

Cost reduction

Efficient inventory management

Focus on high value products

**Cons:-**

It can cause undue product strategy and policies

It can also cause too much focus or attention on a single product line

Going forward, we will be using the Six Sigma approach to analyze ShopSmart datasets for the purpose of answering the imperative six sigma (5W1H) questions to reflect the issues of ShopSmart as well as using the ABC analysis inventory management from the aforementioned methods of optimizing inventory management to solve our problem statement.

1. Who

|  |  |
| --- | --- |
| Questions | Answers |
| Who is involved? | The Data Analyst team of RetailTechSoft |
| Who is affected? | ShopSmart Retail Outlet |
| Who will benefit? | ShopSmart Retail Outlet |
| Who will be Harmed? | Both RetailTechSoft and ShopSmart Retail Outlet |

1. What

|  |  |
| --- | --- |
| Questions | Answers |
| What is your topic narrowed down in a simple phrase/sentence? | ShopSmart retail outlet have a pressing need to improve inventory management or optimize product placement, this needs are borne out of the desire to increase sales turnover to the sole aim of profit maximization. So the topic for this case study is narrowed down to **Optimizing ShopSmart Inventory management** |
| What does your topic involve? (i.e. What are the different parts to it?) | The topic is going to consist of best method to optimize inventory management in ShopSmart e.g. ABC analysis to determine the product with high profit margin to total profit informing decisions on restocking schedule to avoiding stock out in order to keep up with customer demands, This topic also entails creating an inventory table from ShopSmart datasets , using Python Dictionary data structures to enable ShopSmart to track inventory with ease and solve any related customer issues, Sorting product for ShopSmart in their order of sales priority/customer demands using python functions to enhance ShopSmart awareness on the most sort-after inventory. And also visualize output |
| What is it similar to / different from? | The topic is similar to Cost minimization/Profit maximization. |
| What might be affected/changed by your topic? | ShopSmart Sales turnover, ShopSmart inventory management method. |

1. When

|  |  |
| --- | --- |
| Questions | Answers |
| When will it take place? | As soon as possible |
| Does when this takes place affect the topic? | Yes, because the earlier the better for ShopSmart inventory system management to be updated to eradicate the previous lapses, hence deploying the upgrade to meet customers satisfaction and increase sales |

1. Where

|  |  |
| --- | --- |
| Question | Answers |
| Where will it take place? | This project is going to take place at the ShopSmart Inventory management and placement unit because this is where the project is going to be implemented. |
| Does it matter where it takes place? Is it affected by location? | Yes, it does. Because the issues emanated from the inventory management and placement unit of the firm, hence solutions should be implemented at this same unit |

1. Why

|  |  |
| --- | --- |
| Question | Answers |
| Why does this topic matters? | The topic matters because it long-run aim is to maximize profit which is very paramount for ShopSmart to remain in business and not be relegated out of business(perpetuity) |
| What are some causes and effects within the topic? | Causes:  Unsatisfied customers  Inventory mismanagement  Irresponsive product pricing method  Unmotivated workplace practices  Lack of goodwill  Effects:  Positively enhance the value of a firm  Guarantees perpetuity of a business  It can hasten business development  It can create better and advance business and market research  It makes a business a strong competitor to his counterparts. |

1. How

|  |  |
| --- | --- |
| Questions | Answers |
| How does it do what it does? | It does what it does through Python Programming and Python function executions |
| How did it come to be? | Python programming |
| How are those involved affected? | For RetailTechSoft Data Analyst Team—successful project execution and deployment  For ShopSmart Retail Outlet—enhance and improved inventory management and product placement system and ultimately profit maximization  For Customers—greater and better order satisfaction |

**MODEL**

**InventoryTable**

RowID

Orderdate product search function

OrderID

CustomerName inventory management system function

Product

ProductCategory

Profit margin

OrderPriority restocking schedule

OrderQuantity

ShipMode

Price Visualization

Address

**Definition:-**

**InventoryTable:** Table containing inventory values and associated column

**Inventory System Management system:** A python function that optimize inventory management system based on product category, Product priority and profit margin i.e. how much contribution it made to total profit

**Restocking schedule:** this is a python function for restocking ShopSmart inventory based on profit margin or profit contribution to total profit utilizing the ABC analysis inventory management logic to divide restocking decision to three Classes. ClassA: FIRST TO RESTOCK, ClassB: SECOND TO RESTOCK, ClassC: LAST TO RESTOCK

**Visualization:** A pictorial display of inventory management and restocking schedules

**Product search:** A python function that uses Row ID to track complete product details for reference to resolving any customer issues or to track down a particular product

***Below are the python codes for the model***

# To complete this ShopSmart Assessment with Python, we need to first import the tools or libraries

# that we will use to load the chosen dataset into python as well as do our analysis. e.g. Pandas, Numpy

# and a choice, to make my output look more constructive i am going to import Tabulate

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import tabulate

# Below is the code to load the choosen dataset of the assessment into python, for my case(ShopSmart

#Retail data which is an excel or xlsx file)

ShopSmartdata=pd.read\_excel("ShopSmart\_Retail\_Data.xlsx")

ShopSmartdata.head(7)

d=ShopSmartdata.info() # to display the  total number of rows and data types respectively

ShopSmartdata.describe() # this is a quick view at the data predictive stattistics

ShopSmartdata.isnull().sum()# to check for possible null values

# Before we proceed, it is imperative to know our problem i.e. problem statement

# TO OPTIMIZE SHOTSMART RETAIL SHOP INVENTORY MANAGEMENT SYSTEM AND PRODUCT PLACEMENT

# Proceeding further we have to know differentiate our data from category data and quantitative or continous data to determine that columns that

# needed in solving our problem statement

ShopSmartdata.nunique()

# From this above code, it is 14 variables or column that will be used for inventory management and restocking Schedule

# ROW ID

# ORDERDATE

# CUSTOMER NAME

# PRODUCT NAME

# PRODUCT CATEGORY

# PRODUCT BASE MARGIN

# ORDER ID

# ORDER DATE

# ORDER QUANTITY

# ORDER PRIORITY

# UNIT PRICE

# CITY

# STATE

# ZIP CODE

# Removing columns that we do not need in solving our problem statement from ShopSmartdata

not\_needed\_columns = ['Customer Age','Customer Segment', 'Discount','Number of

Records','Product Container',

                      'Product Sub-Category','Profit','Sales','Ship

Date','Shipping Cost','Zip Code']

'''Now we will need to optimize ShopSmart inventory by defining an inventory management function that fetch products based on product category and

order priority and how much profit margin the said product contributes to ShopSmart overall profit. we are defining this function so we can be able to achieve those products in different product category with its accompanying order priority that have a high profit potential and strength in the market

to inform ShopSmart restocking decision. Going forward we are going to set a class on the restocking schedule() class A,B and C) using the ABC

analysis of inventory management method with the logic of how high a product based margin contributes to total profit. if a product has 70

percent and above profit margin ratio such product belongs to the restocking class of A to be restocked first before any other product) and

if the product has between 45 percent to 70 percent profit margin, such product belong to restocking class B and any product with a profit margin below 45 percent belongs to restocking classC (which should be the last set of product

to restock)'''

# First we should create a dataframe to contain the columns that will be used for our further analysis and clean some columns

Address\_Series =tuple(ShopSmartdata['Region']+' '+ShopSmartdata['City']+'

'+ShopSmartdata['State']+ ' ' +ShopSmartdata['Zip

Code'].astype(str))

InventoryTable=pd.DataFrame({'RowID':ShopSmartdata['Row ID'],'ProductCategory':

ShopSmartdata['Product Category'], 'OrderDate':

ShopSmartdata['Order Date'], 'Product':

                              ShopSmartdata['ProductName'],'ProfitMargin':

ShopSmartdata['Product Base Margin'],

                            'OrderID': ShopSmartdata['OrderID],'OrderPriority'

:ShopSmartdata['Order Priority'],

                            'OrderQuantity':ShopSmartdata['Order

Quantity'],'Price':ShopSmartdata['Unit Price'],

'Address': Address\_Series,

                'ShipMode': ShopSmartdata['Ship Mode'],'CustomerName': ShopSmartdata['CustomerName'],'ShipMode':ShopSmartdata['Ship Mode']})

# function to clean the product column of our data

def cleanColumn(df, column):

    cleaned\_column =[]

    for value in df[column]:

        try:

            # Try to convert the value to a string

            cleaned\_value = str(value)

        except (ValueError, TypeError):

            # If conversion fails, set the value to NaN (or any default value)

            cleaned\_value = str.nan

        cleaned\_column.append(cleaned\_value)

    # Replace the original column with cleaned values

    df[column] = cleaned\_column

    return df

Cleanedinventorytable =cleanColumn(InventoryTable,'Product')

print(Cleanedinventorytable)

#defining an inventory management Function that will fetch products based on product category and  order priority requesting user inputs.

def inventory\_management\_system():

    x = input("\nWELCOME TO SHOPSMART INVENTORY OUTLOOK BASED ON PRODUCT CATEGORY AND ORDER PRIORITY\n Please choose and enter a valid product category below you wish to display:\n Furniture\n Office Supplies \n Technology\n")

    y = input(" and proceed with your choice valid order priority below\n Critical\n High\n Medium \n Low \n Not Specified\n")

    # Initialize a list to hold the results

    results = []

    # Loop through the CleanedinventoryTable

    for index in range(len(Cleanedinventorytable)):

        category = Cleanedinventorytable['ProductCategory'].iloc[index]

        priority = Cleanedinventorytable['OrderPriority'].iloc[index]

        profit\_margin = Cleanedinventorytable['ProfitMargin'].iloc[index]

        product = Cleanedinventorytable['Product'].iloc[index]

        row\_id = Cleanedinventorytable['RowID'].iloc[index]

        order\_id = Cleanedinventorytable['OrderID'].iloc[index]

 # Check the conditions

        if category == x and priority == y:

            results.append({'ID': order\_id, 'Product Name': product, 'Profit

Margin': profit\_margin, 'RowID': row\_id})

# Convert results to a DataFrame and print

    if results:

        df = pd.DataFrame(results).set\_index('RowID')

        print(f"\n{y} Priority {x} Inventory:")

        print(df.head(101))

        return df.head(101)

    else:

        print(f"No {y} Priority {x} Orders found.")

        return None

def restocking\_schedule():

# the restocking schedule will be based on how much profit contribution each product from the product category with a specific order priority

# contributes to total profit. we are going to have three restocking schedules of each product category and priority (ClassA,ClassB,ClassC)

    x = inventory\_management\_system()

    if x is None:

        return None

    else:

        print("\nRESTOCKING SCHEDULE ON INVENTORY OUTLOOK BASED ON PROFIT MARGIN RATIO TO SHOPSMART TOTAL PROFIT\n First to restock ClassA:Profit Margin Ratio > 70%\n Second to restock ClassB:Profit Margin Ratio > 45% < 70%\n Last to restock ClassC:Profit Margin Ratio < 45%")

# initialize dictionaries to hold results

    result1 = {'ClassA': [],

                'ClassB': [],

                'ClassC': []}

# Loop through the x = inventory system management returned output

    for index in range(len(x)):

        product = x['Product Name'].iloc[index]

        profit\_margin = x['Profit Margin'].iloc[index]

        if profit\_margin > 0.70:

            result1['ClassA'].append({'product': product,'Profit Margin':

profit\_margin})

elif 0.45 <= profit\_margin <= 0.70:

            result1['ClassB'].append({'product': product,'Profit Margin':

profit\_margin})

        else:

            result1['ClassC'].append({'product': product,'Profit Margin':

profit\_margin})

# loop through the result to print as a single output

    for key in result1:

        if result1[key]:

            df1 =pd.DataFrame(result1[key])

            print(f"\n{key} Products:")

            print(df1.head(71))

# Combine results for all classes to enable us virtualize output using bar chart

    all\_results = pd.concat([pd.DataFrame(result1[key]) for key in result1])

# Plot the combined results

    plot\_bar\_chart(all\_results)

def plot\_bar\_chart(df):

    plt.figure(figsize=(10, 6))

    plt.bar(df['product'], df['Profit Margin'], color=['blue', 'green', 'red'])

    plt.xlabel('Product')

    plt.ylabel('Profit Margin')

    plt.title('Restocking Schedule for ClassA, ClassB, and ClassC Products')

    plt.xticks(rotation=30)

    plt.tight\_layout()

    plt.show()

#     ENDED HERE

# Going forward we will define a function that will use row ID as an input to track a specified product details

# this will enable ShopSmart to easily track their product in the system when resolving any customer issues

def  product\_search():

    try:

        abc = int(input("Enter RowID to search for: "))

    except ValueError:

        print("Please enter a valid integer for RowID.")

        return

# initialize a list to store the ouput

    output = []

#loop through the inventoryTable

    for index in range(len(Cleanedinventorytable)):

            rowID = Cleanedinventorytable['RowID'].iloc[index]

            customer = Cleanedinventorytable['CustomerName'].iloc[index]

            product = Cleanedinventorytable['Product'].iloc[index]

            productcategory =Cleanedinventorytable['ProductCategory'].iloc[index]

            orderDate = Cleanedinventorytable['OrderDate'].iloc[index]

            orderID = Cleanedinventorytable['OrderID'].iloc[index]

            priority = Cleanedinventorytable['OrderPriority'].iloc[index]

            quantity = Cleanedinventorytable['OrderQuantity'].iloc[index]

            price = Cleanedinventorytable['Price'].iloc[index]

            shipMode = Cleanedinventorytable['ShipMode'].iloc[index]

            address = Cleanedinventorytable['Address'].iloc[index]

# set the condition

            if  rowID == abc:

                output.append({'RowID':rowID,'ID':orderID,'Date': orderDate,

'CustomerName': customer,'productName':

product,'Product Category':productcategory,

'Priority':priority,'Quantity':quantity,'Price':

price,'ShipMode':shipMode,'Address':address})

# Convert output to Dataframe and print

if output:

        df=pd.DataFrame(output).set\_index('RowID')

        print(f"\nEntered rowID has the product information below:")

        print(tabulate.tabulate(df,headers=['RowID','ID','Date','CustomerName',

'ProductName','ProductCategory','Priority', 'Quantity',

'Price','ShipMode','Address'],tablefmt='orgtbl'))

    else:

        print(f"No product information found for RowID {abc}.")

        return None

**From the python Code above:**

I will be testing the function by executing a task of first fetch all the products of Furniture in the product category and specific order priority of the furniture product. In this example we will be using “Critical” order priority to fetch all furniture products.

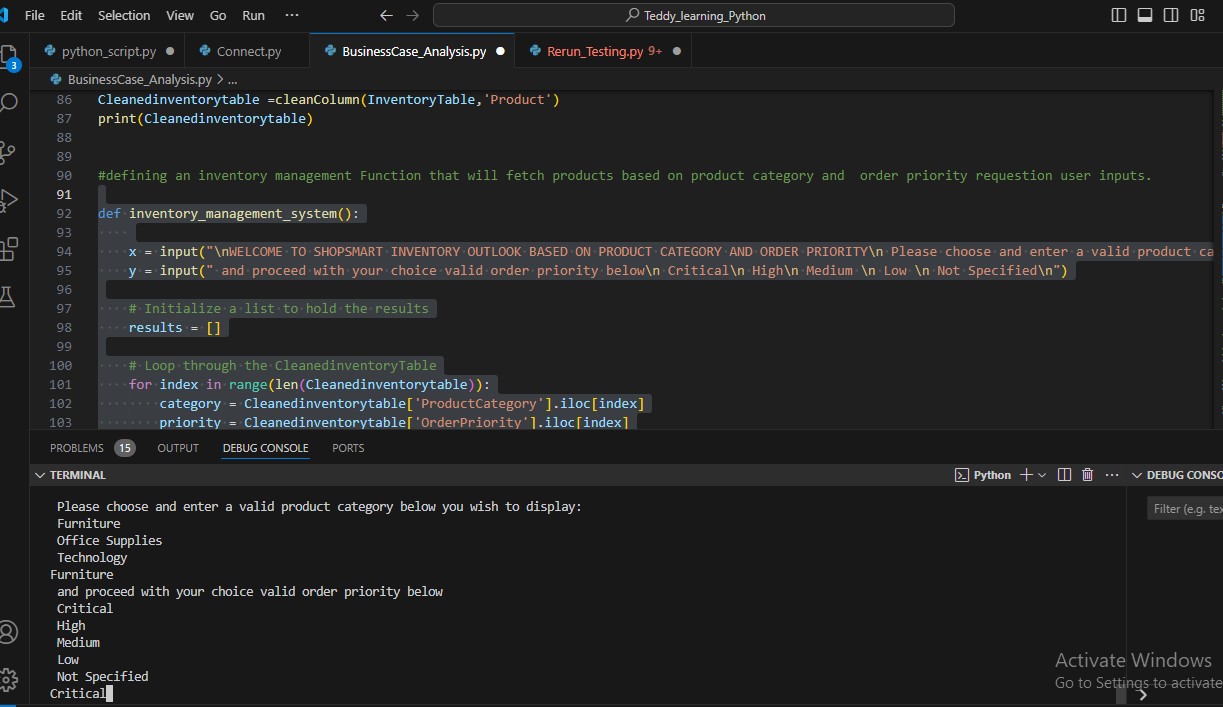
And this example will go as far as telling or showing us the set of Furniture products under the Critical priority that should be considered first to restock due to its profit margin contribution to total profit. A visual presentation will follow to demonstrate the logic

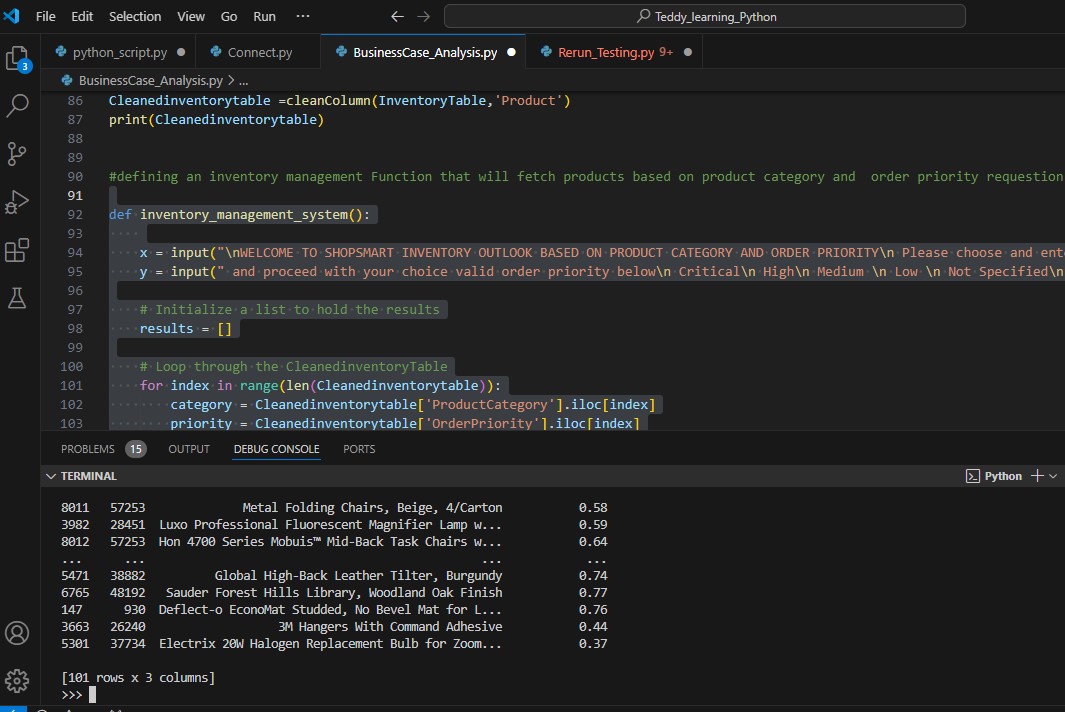
**Running the python Codes:**

**Note:** you can choose any product category and order priority of your choice you wish to display.

Product Category: Furniture

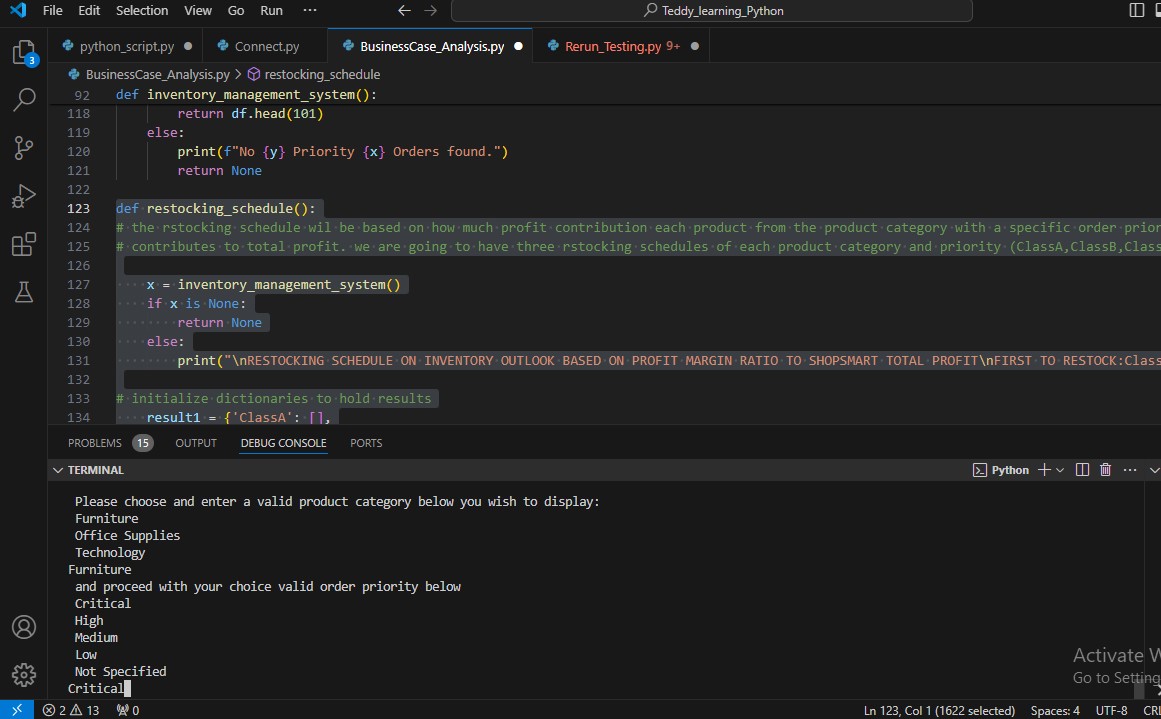
Order priority: Critical

**** Running the inventory management system function to achieve our task of fetching furniture products with Critical order priority only

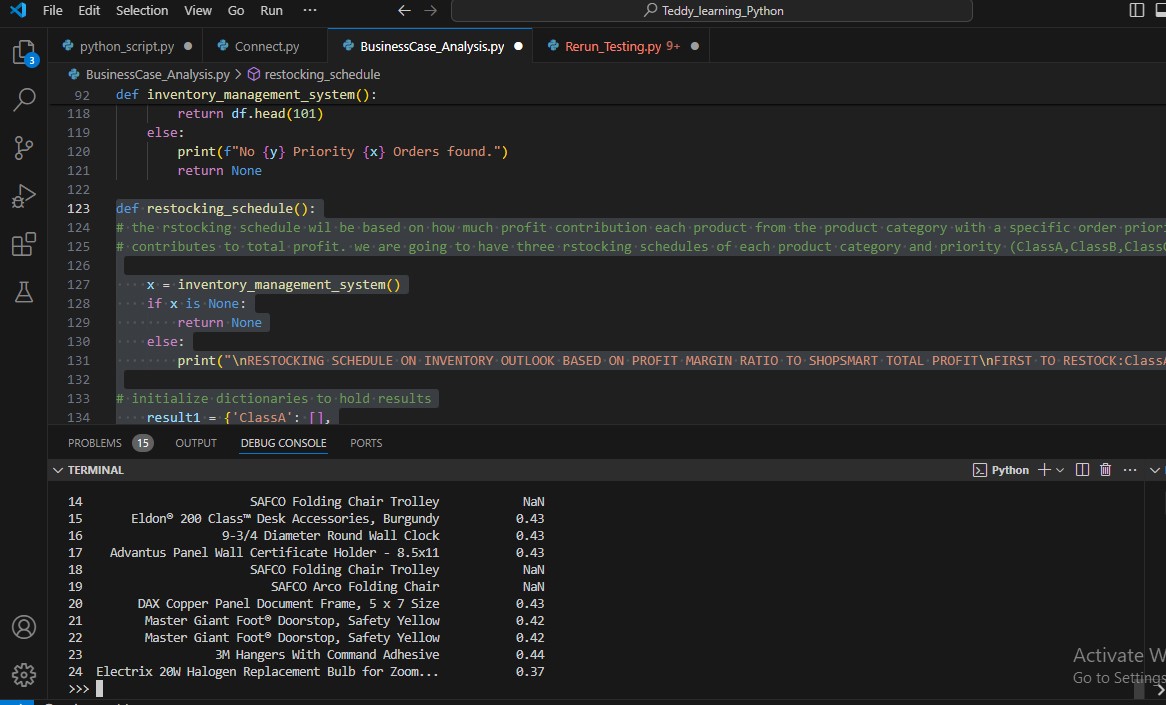
**Output:**

Now: we can proceed to know which set of product belong to our restocking schedule classes of either ClassA, ClassB or Class C to determine which furniture should be restocked first second or last etc. among all the products under this Furniture category with a Critical order priority

To determine this we will run the restocking schedule function



**Output:** Furniture product of Critical priority categorized into ClassA, ClassB and ClassC respectively



Visualizing this output to substantiate our logic, we proceed to running the Plot Bar Chart Function which combine all the classes of product in one display to enable our client see the difference at a glance.



**Explanation:**

Reading this Chart from left to my right, it explains that as furniture products under the Critical priority moves from left to right their contribution to total profit also decreases, using the logic of ABC analysis of inventory management it means ShopSmart should consider restocking products starting from left to right capturing the set of furniture’s from the left to be restocked first and so on

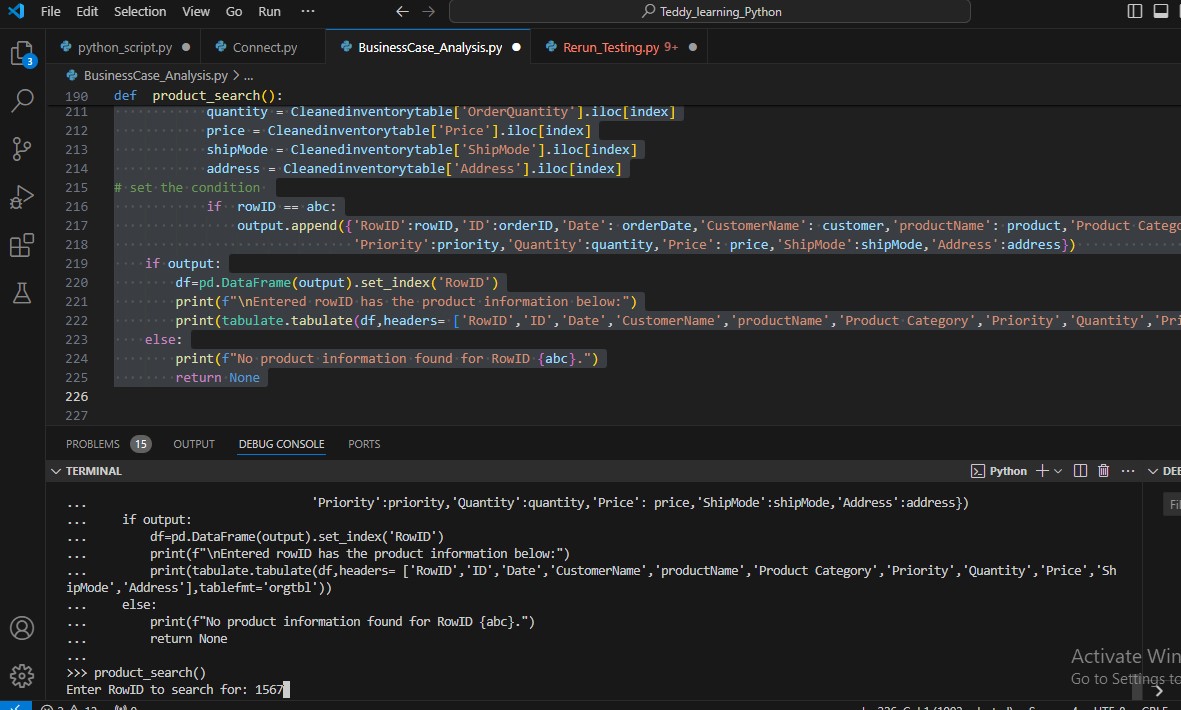
I am optimistic that this can aid ShopSmart to maximize profit as Furniture’s are restocked based on profit contribution to total revenue.

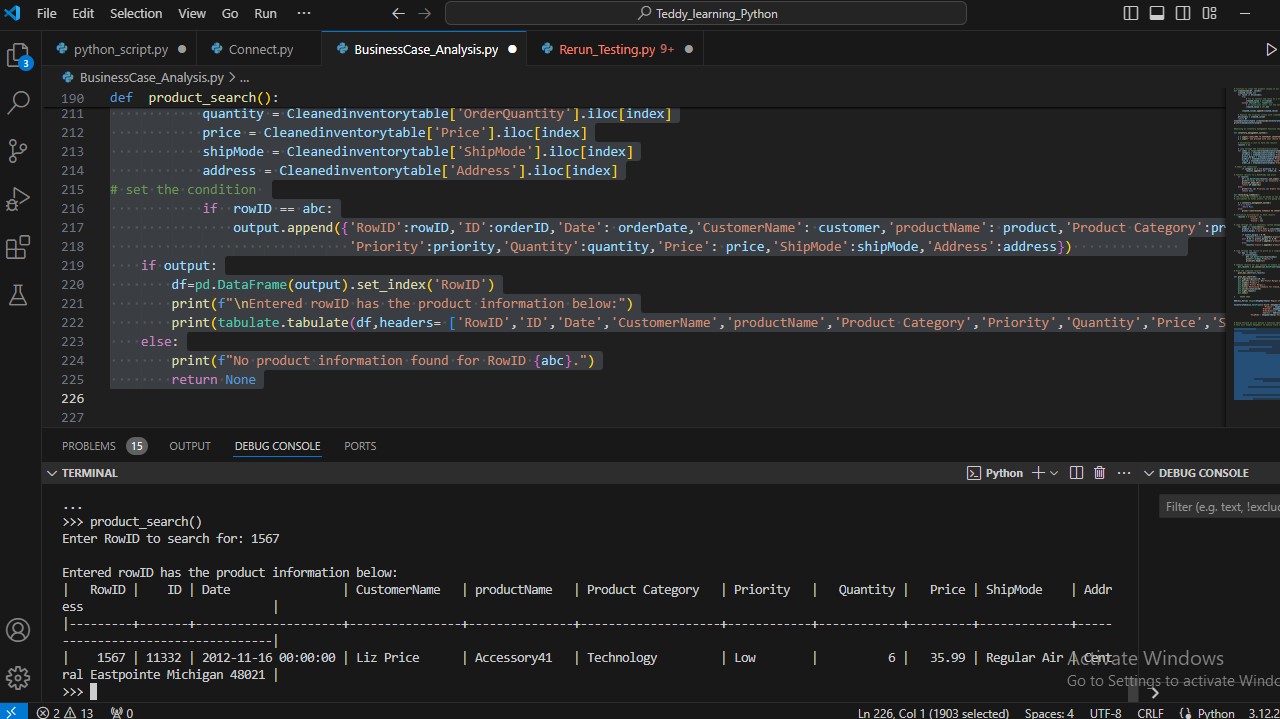
For the product search function, this functions comes in handy when handling customers issues or when product details needs to be obtained for references or other related reasons

**Example**

If ShopSmart retail shop wants to know when a particular product was ordered or what type of ship mode was used or want to know that customer name of such products, the product search function becomes an easy solution to such problems. Hence, say we want to know the above details of a particular product whose rowID is **1567**.

We proceed to running and using the product search function to solve this





From our output after running the product search function, we can see the product with rowID **1567** is a Technology product which was order by Liz Price on 16-11-2012 and was shipped by Regular Air

This operation by the product search function have just given ShopSmart a one stop solution to finding out the product information without passing through any rigorous process