

|  |  |
| --- | --- |
| Using Inventory Management to Combat Supply Chain Disruptions - Global  Trade Magazine  Internship report  **Universidad Politécnica de Valencia(UPV)** | Abstract  Report to explain how the code works and the theoretical part of the model  Ahmadou oury Diallo  Student in 2nd year of engineering school at IMT Atlantique |

Contents

[I. Inventory Management 2](#_Toc144999019)

[A. Importance of inventory : 2](#_Toc144999020)

[B. Function of inventory : 2](#_Toc144999021)

[C. Managing inventory : 2](#_Toc144999022)

[a. How inventory items can be classified (ABC analysis) 2](#_Toc144999023)

[b. How accurate inventory records can be maintained 2](#_Toc144999024)

[II. Inventory Models : 3](#_Toc144999025)

[D. Inventory Model for independent demand 3](#_Toc144999026)

[Model 1 : Basic Economic Order Quantity ( EOQ) : 3](#_Toc144999027)

[Model 2 : Production Order Quantity ( POQ) : 6](#_Toc144999028)

[Model 3 : Quantity Discount 8](#_Toc144999029)

[Model 4 : Single Period model 10](#_Toc144999030)

[Model 5 : Deterministic Demand 12](#_Toc144999031)

[Model 6 : Stochastic Demand 14](#_Toc144999032)

[III. Bibliography : 20](#_Toc144999033)

# Inventory Management

The objective of inventory management is to strike a balance between inventory investment and customer service.

## Importance of inventory :

* One of the most expensive assets of many companies representing as much as 50% of total invested capital.
* Operations managers must balance inventory investment and customer service

## Function of inventory :

* To provide a selection of goods for anticipated demand and to separate the firm from fluctuations in demand.
* To decouple or separate various parts of the production process
* To take advantage of quantity discounts
* To hedge against inflation

## Managing inventory :

### How inventory items can be classified (ABC analysis)

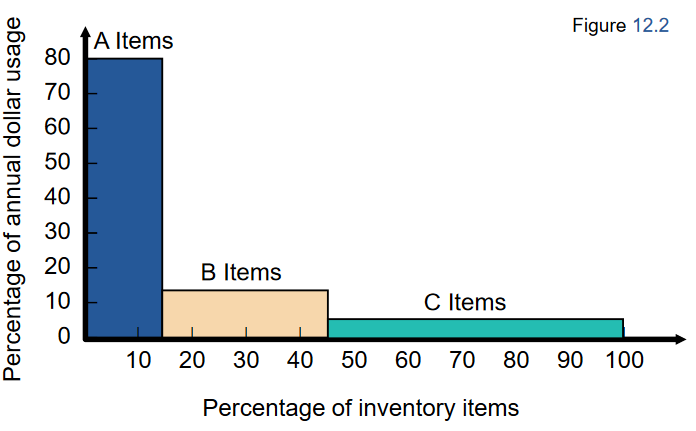
Divides inventory into three classes based on annual dollar volume

▶ Class A - high annual dollar volume

▶ Class B - medium annual dollar volume

▶ Class C - low annual dollar volume

Used to establish policies that focus on the few critical parts and not the many trivial ones



1.Principles of operation management by HIEZER & RENDER

### How accurate inventory records can be maintained

#### Record Accuracy :

Accurate records are a critical ingredient in production and inventory systems

► Periodic systems require regular checks of inventory

► Two-bin system

► Perpetual inventory tracks receipts and subtractions on a continuing basis

► May be semi-automated

► Stockrooms should be secure

► Necessary to make precise decisions about ordering, scheduling, and shipping

#### Cycle Counting

Items are counted and records updated on a periodic basis

1. Eliminates shutdowns and interruptions

2. Eliminates annual inventory adjustment

3. Trained personnel audit inventory accuracy

4. Allows causes of errors to be identified and corrected

5. Maintains accurate inventory records

# Inventory Models :

▶ Independent demand - the demand for item is independent of the demand for any other item in inventory

▶ Dependent demand - the demand for item is dependent upon the demand for some other item in the inventory

▶ Holding costs - the costs of holding or “carrying” inventory over time

▶ Ordering costs - the costs of placing an order and receiving goods

▶ Setup costs - cost to prepare a machine or process for manufacturing an order

▶ May be highly correlated with setup time

## Inventory Model for independent demand

Need to determine when and how much to order

1. Basic economic order quantity (EOQ) model
2. Production quantity model
3. Quantity discount model

### Model 1 : Basic Economic Order Quantity ( EOQ) :

Important assumptions

1.Demand is known, constant, and independent

2. Lead time is known and constant

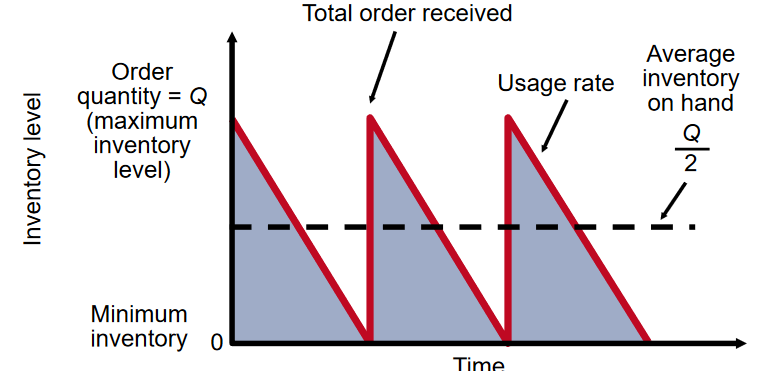
3. Receipt of inventory is instantaneous and complete

4. Quantity discounts are not possible

5. Only variable costs are setup (or ordering) and holding

6. Stockouts can be completely avoided

* **Inventory usage over time :**



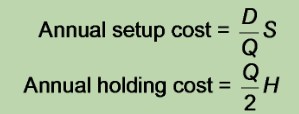
2..Principles of operation management by HIEZER & RENDER

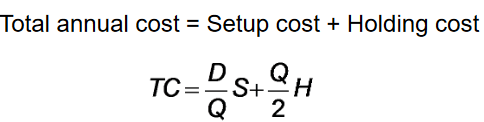
* **Minimizing costs : objective is to minimise total costs :**

A diagram of a function

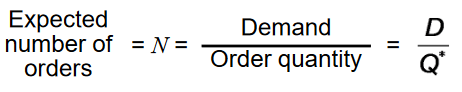
Description automatically generated

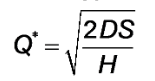
3..Principles of operation management by HIEZER & RENDER





A black text on a white background

Description automatically generatedOptimal order quantity is found when annual setup cost equals annual holding cost :



**Example** :

Determine optimal number of needles to order :  
D = 1000 units ; S = $10 per older ; H = $0.50 per unit per year

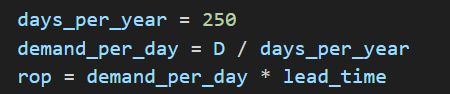
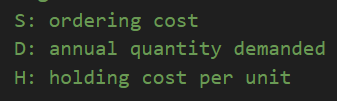
Q\* = Number of oders N =

Time between orders T = =

Total annual cost = Setup cost + Holding cost = = $100

**Programmation** :

file: Economic\_order\_quantity.py

days\_per\_year : working day per year

A screenshot of a computer program

Description automatically generated

Result :

A blue circle with black text

Description automatically generatedA screenshot of a computer

Description automatically generated

A graph of a number of blue lines

Description automatically generated with medium confidence

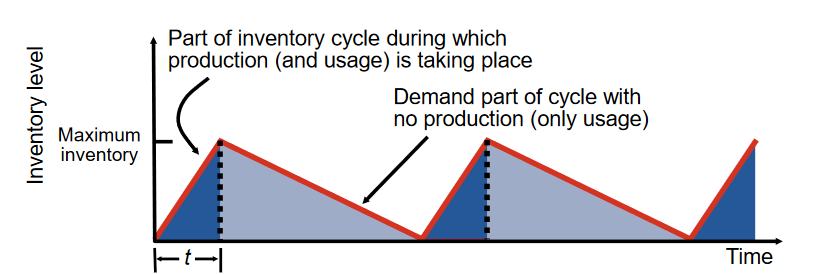
A graph of a number of colored lines

Description automatically generated

### Model 2 : Production Order Quantity ( POQ) :

1. Used when inventory builds up over a period of time after an order is placed

2. Used when units are produced and sold simultaneously



4..Principles of operation management by HIEZER & RENDER

Q = Number of pieces per order p = Daily production rate

H = Holding cost per unit per year d = Daily demand/usage rate

t = Length of the production run in days

A math equations with numbers

Description automatically generated with medium confidenceA black square with black text

Description automatically generated

**Example** :

D = 1000 units ; S =10$ ; H =0.50 $ ; p = 8 unit/day, d = 4 unit/day

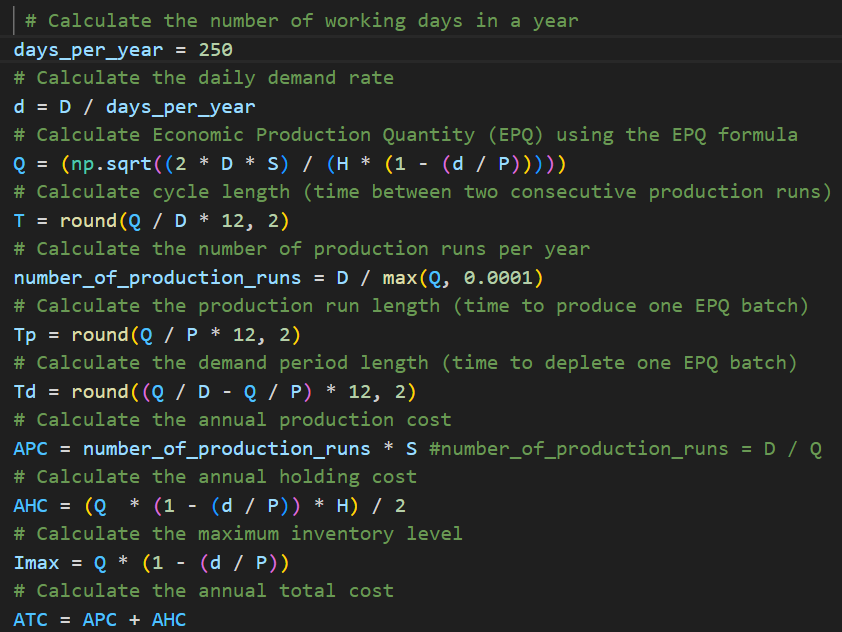
Q\*p = 283 ; d =

**Programme :**

File : Production\_order\_quantity.py

A black background with green text

Description automatically generated



Result :

A blue circle with white text

Description automatically generatedA screenshot of a computer

Description automatically generated

A graph of a graph

Description automatically generated with medium confidence

### Model 3 : Quantity Discount

▶ Reduced prices are often available when larger quantities are purchased

▶Trade-off is between reduced product cost and increased holding cost

A red and white discount schedule

Description automatically generated

A diagram of a price

Description automatically generatedA math equation with black text

Description automatically generatedA diagram of a function

Description automatically generated with medium confidence

5..Principles of operation management by HIEZER & RENDER

**Example** :

D = 5000 ; S = 49$ ; I = 20%; P = 5 $

Q\* = 700

**Programme :**

File : Quantity\_discounts\_model.py

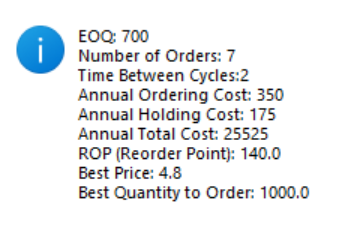
A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generatedA table with red and white text

Description automatically generated

A screenshot of a calculator

Description automatically generatedA blue circle with black text

Description automatically generated

### Model 4 : Single Period model

The Single-Period Model, also known as the Newsboy Problem, is a classic inventory management problem that focuses on making a one-time purchase decision for a perishable product. In this model, a retailer must decide how much inventory to order for a single selling season, given uncertain demand and limited shelf life. The goal is to maximize expected profit by finding the optimal order quantity that minimizes the trade-off between overstocking and stockouts.

Key elements of the Single-Period Model include:

**Demand:** The uncertain customer demand for the product during the selling season. It is typically described by a probability distribution.

**Sales Price:** The price at which the product is sold to customers.

**Cost per Unit:** The cost incurred to purchase each unit of the product from the supplier.

**Salvage Value:** The value obtained for each unsold unit of the product at the end of the selling season. This salvage value could represent, for instance, the price at which the product can be sold or returned to the supplier.

The objective is to determine the optimal order quantity that maximizes expected profit or, equivalently, minimizes expected costs. The decision involves finding a balance between the cost of overordering (stocking excess inventory that might not be sold) and the cost of underordering (stockouts leading to lost sales or unsatisfied customers).

The **Optimal Stocking Level** in the Single-Period Model can be calculated as follows:

Optimal Stocking Level = Average Demand + (Z)(Standard Deviation of Demand)

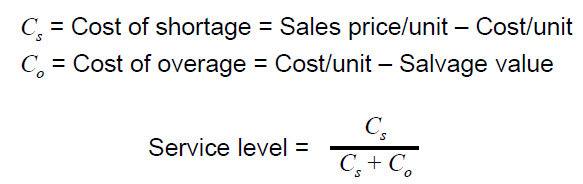
Where Z is the value from the standard normal distribution that corresponds to the desired service level. The service level represents the probability of not encountering stockouts during the selling season.

The **Stockout Risk** is given by:

**Stockout** **Risk** = 1 - Service Level

It represents the probability of experiencing a stockout during the selling season.

The Single-Period Model is often used in situations where products have a short shelf life, such as fashion items, fresh produce, or seasonal goods. It helps businesses make informed decisions about how much inventory to order for a single selling period to maximize their profit or minimize their losses.

A diagram of a service level

Description automatically generated

**Programme** :

File : Single\_period\_model.py

A screen shot of a computer code

Description automatically generated

A computer screen shot of text

Description automatically generated

A screenshot of a computer

Description automatically generated

### Model 5 : Deterministic Demand

This part contains a discrete-event simulation (DES) model of a single product inventory system.

What is deterministic ?

 In a deterministic system, outcomes are entirely predictable and can be precisely determined based on known inputs and initial conditions.

 Every time the same input is provided to the system, it will produce the same output.

 There is no randomness or uncertainty involved in the process

#### Problem statement :

The company starts the period with an initial inventory of 60 units. There is a fixed cost associated with placing an order of $32,  Each individual items costs the company $3 to order The company incurs a holding cost of $1 per month for every item in inventory, shortage cost of $3 per month for every item backlogged.

**Programme** :

MEAN\_IAT = 0.1                        # average time between customer demands (months)

DEMANDS = [5]              # predefined customer demand sizes

START\_INVENTORY = 60.               # units in inventory at simulation start

COST\_ORDER\_SETUP = 32.              # fixed cost of placing an order

COST\_ORDER\_PER\_ITEM = 3.            # variable cost of ordering an item

COST\_BACKLOG\_PER\_ITEM = 5.          # monthly cost for each item in backlog

COST\_HOLDING\_PER\_ITEM = 1.          # monthly cost for each item in inventory

SIM\_LENGTH = 120.                   # duration of the simulation (in months)

* **InventorySystem Class**: This class represents the inventory system. It initializes the system with various attributes and methods related to managing inventory and costs.

**\_\_init\_\_(self, env, reorder\_point, order\_size):** Constructor method that initializes the attributes of the inventory system.

**place\_order(self, env, units):** Places an order for a specified number of units and schedules the arrival of the order.

# update ordering costs

        self.ordering\_cost += (COST\_ORDER\_SETUP

                              + units \* COST\_ORDER\_PER\_ITEM)

**review\_inventory(self, env):** Regularly reviews the inventory level and places orders if needed.

 def review\_inventory(self, env):

        """ Check inventory level at regular intervals and place

        orders inventory level is below reorder point """

**receive\_order(self, env, units):** Receives orders and updates transit orders.

**update\_cost(self, env):** Updates holding and shortage costs at each inventory movement.

**get\_demand(self):** Returns the demand at the current time.

DEMANDS = [5]

Deterministic demand is fixed

def get\_demand(self):

        """ Get the demand at the current time """

        return DEMANDS[int(self.env.now) % len(DEMANDS)]

**demands(self, env):** Generates demand at regular intervals and updates the inventory level.

**calculate\_ioh(self):** Calculates the Inventory On Hand (IOH) at each time step.

* **run(reorder\_point, order\_size, sim\_length, lead\_time, display\_chart)**: This function runs the inventory system simulation and returns the simulation results in a dictionary.

**reorder\_point**: Inventory level triggering replenishment.

**order\_size:** Number of units to order at each replenishment.

**sim\_length:** Duration of the simulation in months.

**lead\_time:** Lead time for order delivery.

**display\_chart:** Boolean indicating whether to display a chart of inventory level.

**step\_graph(inventory)**: This function displays a step line chart of the inventory level and related metrics.

**inventory:** An instance of the InventorySystem class.

**run\_simulation()**: This function is the callback for the "Run Simulation" button in the GUI. It gathers input values from the GUI, runs the simulation, and displays the results.

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A graph of a graph and a graph of a graph

Description automatically generated with medium confidence

### Model 6 : Stochastic Demand

In a stochastic simulation, randomness and uncertainty are introduced into the model. Random variables and probabilistic distributions are used to model variables such as demand, lead times, or other factors that have inherent variability. This approach allows us to study the impact of uncertainty on the system's behaviour and performance. In our current code, we introduced randomness in the demand generation and lead time processes, making the simulation more realistic and reflective of real-world scenarios.

* **Library** : I use a lot of library

**Numpy :** is a powerful library for numerical computing in Python. It provides support for working with arrays and matrices, along with a wide range of mathematical functions. In my code, NumPy is used for generating random numbers, particularly for modelling demand and lead times.

**Simpy:**  is a discrete-event simulation library in Python. It allows us to model and simulate complex systems using discrete events (e.g., arrivals, departures, etc.). SimPy is often used for simulation of systems that involve entities (like customers, orders, etc.) and resources (like machines, servers, etc.). In your code, SimPy is used to create and run simulations of inventory systems.

**Tkinter :** is the standard GUI (Graphical User Interface) library for Python. It provides tools for creating windows, dialogs, buttons, labels, and other graphical elements to build interactive applications with graphical interfaces. In my code, Tkinter is used to create a graphical interface for users to input simulation parameters and view simulation results.

#### Policy ( s, Q ) :

○ Definition: This policy triggers an order of size Q when the inventory position reaches or falls below the reorder point s.

○ Pros: Allows you to maintain a relatively constant level of inventory. The fixed order size makes it easier to plan and manage inventory.

○ Cons: Can cause large fluctuations in stock levels, as the order is triggered as soon as the reorder point is reached, regardless of the actual level of demand. May lead to stockouts if demand exceeds expectations.

#### Problem statement

It’s the same problem statement we use for the deterministic demand.

**Programme** :

**Simulation Parameters:**

* Reorder point: 20 units
* Order size: 30 units
* Simulation duration: 10 months
* Average demand: 3 units
* Demand standard deviation: 0.5 units

**InventorySystem Function:**

* This class represents a single inventory management system with a fixed reorder point and a fixed order size policy.
* The methods **place\_order, review\_inventory, receive\_order, update\_cost, get\_demand, and demands** handle events and costs associated with the simulation.
* Variables like **ordering\_cost, holding\_cost, shortage\_cost, history, ioh\_history, num\_orders**, etc., are used to track costs and stock levels over time.

**run\_simulation Function:**

* This function is called when the "Run Simulation" button is pressed in the graphical user interface (GUI).
* It retrieves the user-entered parameters and calls the run function to execute the simulation.
* The results are then displayed in the GUI.

**step\_graph Function:**

* This function generates graphs to visualize simulation results, including stock levels over time, demand, costs, etc.
* The graphs include stock levels over time, demand, Inventory On Hand (IOH), and other relevant information.

**Graphical User Interface (GUI):**

* I have created a simple graphical interface to allow users to input simulation parameters and display results

**Result** :

A screenshot of a computer

Description automatically generated

A computer screen shot of numbers

Description automatically generated

A graph of a diagram

Description automatically generated with medium confidence

#### Policy ( s, S ) :

○ Definition: This policy triggers an order whose size is the difference between the reorder level S and the current inventory position when the inventory position reaches or falls below the reorder point s.

○ Pros: Allows better adaptation to actual demand, as the size of the order is adjusted according to the current inventory level. Can help maintain a better balance between inventory levels and demand.

○ Cons: Determining replenishment level S can be complex and require extensive demand and cost analysis. May result in more complex inventory management due to variation in order size.

#### Problem statement

It’s the same problem statement we use for the deterministic demand

**Programme** :

Simulation constant :

MEAN\_IAT = 0.1                       # average time between customer demands (months)

START\_INVENTORY = 60.               # units in inventory at simulation start

COST\_ORDER\_SETUP = 32.              # fixed cost of placing an order

COST\_ORDER\_PER\_ITEM = 3.            # variable cost of ordering an item

COST\_BACKLOG\_PER\_ITEM = 5.          # monthly cost for each item in backlog

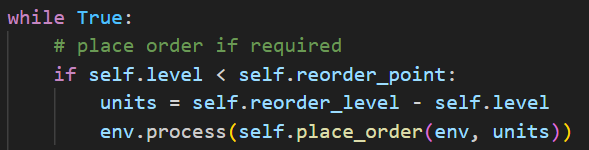
COST\_HOLDING\_PER\_ITEM = 1.          # monthly cost for each item in inventory

Class **InventorySystem**:

**\_\_init\_\_(self, env, reorder\_point, reorder\_level, demand\_mean, demand\_std):** Initializes the inventory system with the specified parameters. It sets up the initial values, launches the processes for checking inventory and generating demands.

**place\_order(self, env, units):** Places and receives orders, updates ordering costs, determines order lead time, and updates inventory level.

**check\_inventory(self, env):** Regularly checks the inventory level and places orders if the inventory falls below the reorder point.



**receive\_order(self, env, units):** Receives orders and updates stock in transit.

**calculate\_stock\_in\_transit(self):** Calculates the stock currently in transit.

**update\_cost(self, env):** Updates holding and shortage costs based on inventory changes.

**get\_demand(self):** Generates the demand at the current time following a normal distribution.

 def get\_demand(self):

        """ Get the demand at the current time """

     return max(0, int(np.random.normal(loc=self.demand\_mean, scale=self.demand\_std)))

**demands(self, env):** Generates demand at random intervals and updates the inventory level.

**calculate\_ioh(self):** Calculates the inventory on hand (IOH) at each point in time.

**calculate\_fill\_rate(self):** Calculates the fill rate, which is the percentage of demand fulfilled.

**Function run(reorder\_point, reorder\_level, sim\_length, demand\_mean, demand\_std, display\_chart=False):**

Runs the inventory system simulation for a specified length and returns simulation results in a dictionary.

**Function step\_graph(inventory):** Displays a step line chart of the inventory level, IOH, and demand over time.

**Function run\_simulation():** Callback function for the "Run Simulation" button in the GUI.

Reads input values from the GUI entries, runs the simulation using the run function, calculates the fill rate, and displays the results in the GUI.

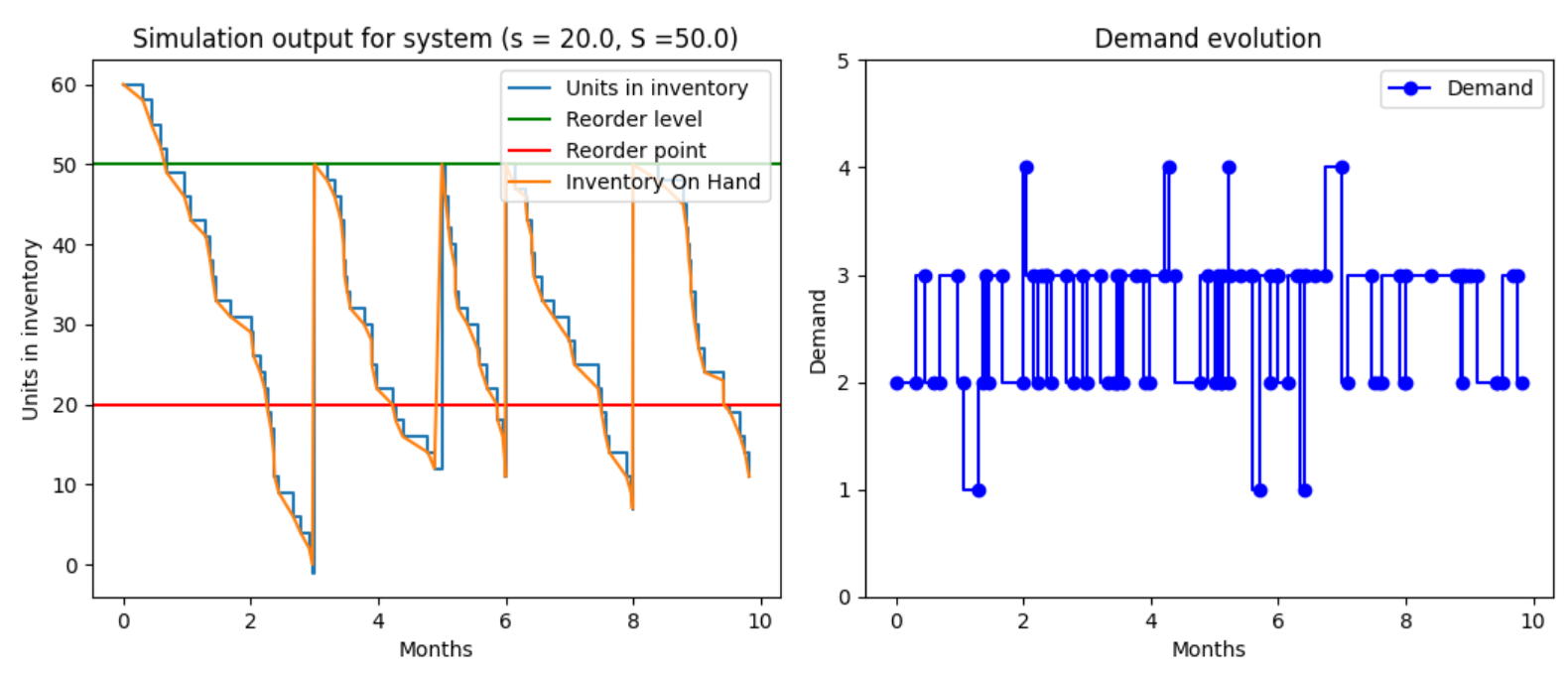
GUI-related Code:

This part of the code creates a simple graphical user interface (GUI) using the tkinter library.

Result :

A screenshot of a computer

Description automatically generated



#### Policy ( s, S ) :

○ Definition: This policy involves periodic reviews of the inventory position at regular intervals R. During each review, an order is placed to achieve the replenishment level S minus the current inventory position.

○ Advantages: Allows more regular inventory management by carrying out periodic reviews. Ordering is based on the gap between the inventory position and the replenishment level S, which takes into account variations in demand.

○ Disadvantages: May cause delays in ordering and replenishment because the order is placed only at scheduled review points. May require careful planning of exams to avoid stock-outs or overstocks.

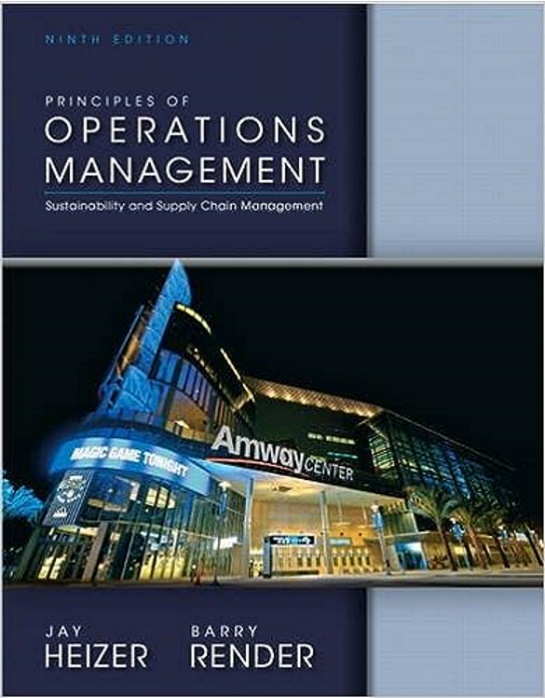
#### Policy ( s, S ) :

○ Definition: This policy combines aspects of (s, Q) and (R, S) policies. The inventory position is checked at intervals of R. An order is triggered only if the inventory position reaches or falls below the reorder point s. The order size is determined by the difference between the reorder level S and the current inventory position.

○ Benefits: Allows you to combine the benefits of (s, Q) and (R, S) policies by taking into account both actual demand and periodic reviews. Allows more flexible inventory management by adjusting order size based on current inventory situation.

○ Disadvantages: May require careful planning of examinations and continuous monitoring of inventory position. Improper estimation of reorder point s can lead to errors in ordering decisions

# Bibliography :

A book cover with a picture of a crane and a person

Description automatically generated

A close-up of a inventory management

Description automatically generated

<https://blog.scallog.com/reapprovisionnement-logistique>

<https://towardsdatascience.com/inventory-management-for-retail-deterministic-demand-311682c02518>

<https://asadali047.medium.com/inventory-simulation-for-beginners-7ea55eb6c4f8>

<https://victorjacobsen.com/?p=1740>

<https://www.linkedin.com/advice/3/what-main-differences-between-deterministic>

<https://www.slideshare.net/hakimizaki/c9-inventory-management>