**PRACTICAL FILE**

**MODELING AND SIMULATION LAB**

**(CS 603)**

**BE CSE 6TH SEM**

**(GROUP-4)**

****

**University Institute of Engineering and Technology (UIET), Panjab University, Chandigarh, India- 160014**

**Under the guidance of Submitted By**

Priyanka Mam Ojas Arora

Department of Computer Science and Engineering Roll No: UE223073

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SNO.** | **DATE** | **TITLE** | **PAGE NO** | **REMARKS** |
| 1. | 29-01-2025 | Working with 1) Branching statements,2) Loops,3) Datatypes, 4) Functions, 5) Plots, 6) Array and Cell Array, 7) Inputs and Outputs etc. | 3-12 |  |
| 2. | 05-02-2025 | Implementation of Traffic Light Simulation Using MATLAB. | 13-14 |  |
| 3. | 19-02-2025 | Rainfall Prediction using Monte Carlo Simulation | 15-17 |  |
| 4. | 19-03-2025 | MATLAB code to simulate queueing system at ATM using Poisson Distribution. | 18-20 |  |
| 5. | 19-03-2025 | MATLAB code to simulate Zigzag Walking person. | 21-23 |  |
| 6. | 19-03-2025 | MATLAB code to simulate fire spread in a forest using cellular automata. | 24-26 |  |
| 7. | 19-03-2025 | MATLAB code for estimating π (Pi) using Monte Carlo simulation. | 27-29 |  |
| 8. | 02-04-2025 | Simulation code for Shortest Remaining Time First (SRTF) Scheduling Algorithm. | 27-29 |  |
| 9. | 09-04-2025 | MATLAB code for Simulation of a Water Tank Filling and Draining System over the time. | 30-24 |  |
| 10. | 16-04-2025 | MATLAB-Based Simulation of Inventory Management for Supply Chain Optimization. | 35-38 |  |

**Practical 1**

**Aim**

**I**. Introduction to MATLAB

**II.** Working with:

**1.** Branching statements

**2.** Loops

**3.** Datatypes

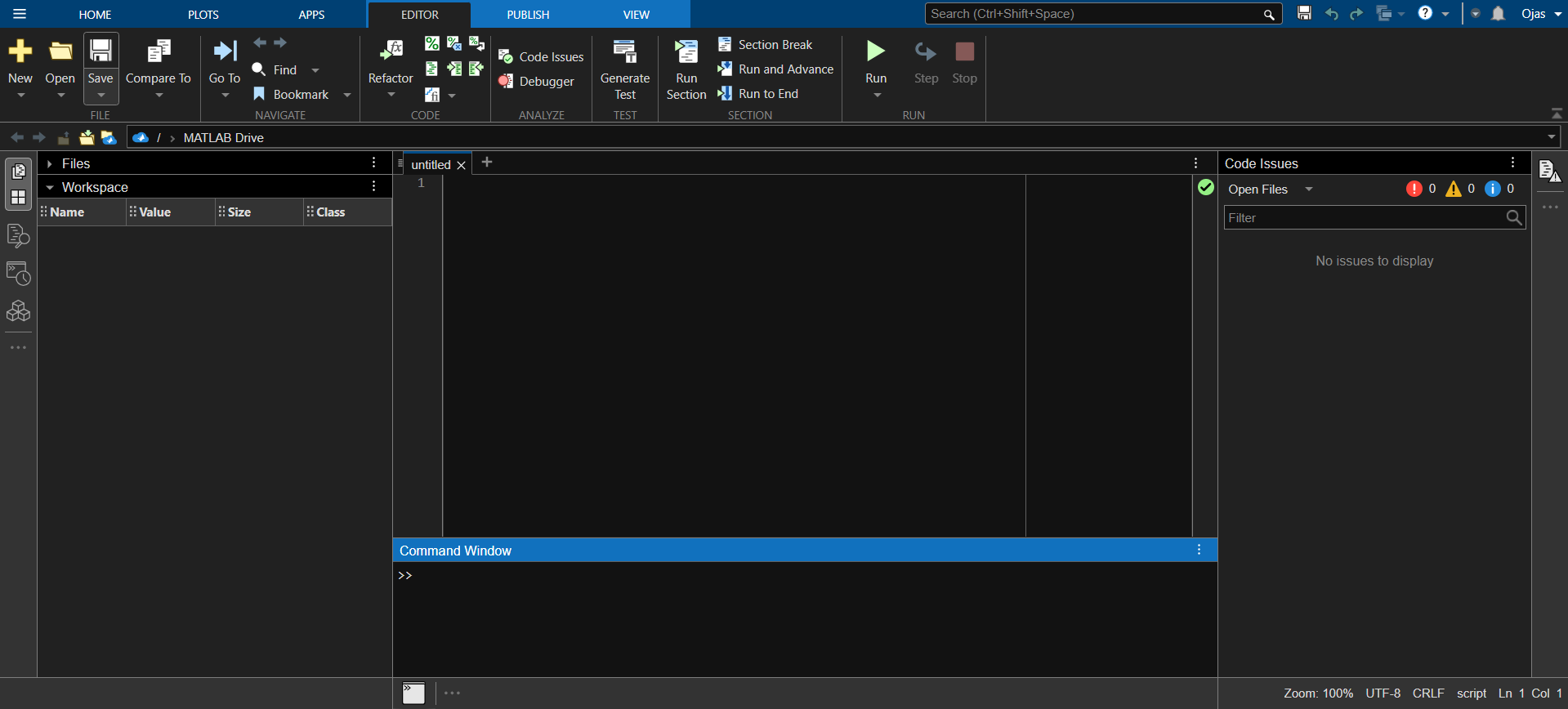
**4.** Functions

**5.** Plots

**6.** Array and Cell Array

**7.** Inputs and Outputs

**I. Introduction to MATLAB**



**MATLAB** stands for Matrix Laboratory. It is a high-performance language that is used for technical computing. It allows matrix manipulations, plotting of functions, implementation of algorithms and creation of user interfaces.

**Getting Started with MATLAB**

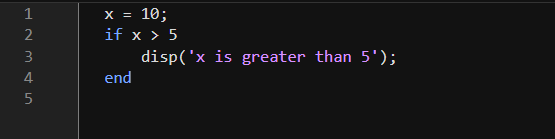
It is both a programming language as well as a programming environment. It allows the computation of statements in the command window itself.

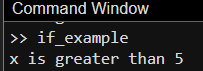
* **Command Window:** In this window one must type and immediately execute the statements, as it requires quick prototyping. These statements cannot be saved. Thus, this is can be used for small, easily executable programs.
* **Editor (Script):** In this window one can execute larger programs with multiple statements, and complex functions These can be saved and are done with the file extension ‘.m ‘.
* **Workspace:** In this window the values of the variables that are created in the course of the program (in the editor) are displayed.

**II. Working with:**

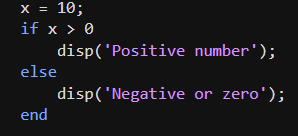
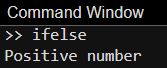
**1. Branching statements**

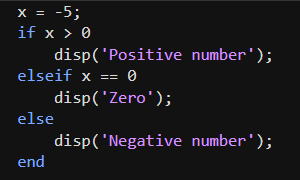
In MATLAB, branching statements are used to execute different code based on conditions. They allow for decision-making in the program. The primary branching statements in MATLAB are:

**A. if:** Executes a block of code if a condition is true.



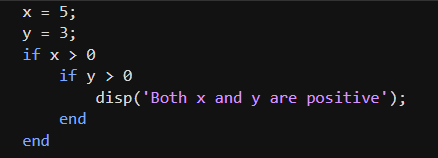
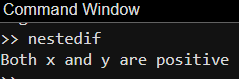
**B. if-else:** Allows you to execute one block of code if a condition is true and another block of code if the condition is false.

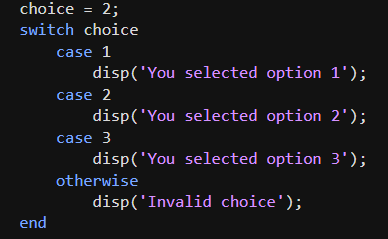


**C. if-else-if:** used to check multiple conditions in sequence and execute different code blocks depending on which condition is true.



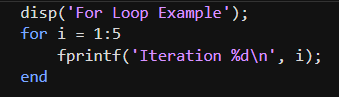
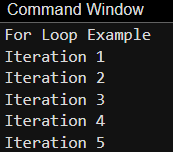
**D. nested-if:** statement placed inside another if statement. It allows for more complex decision-making by evaluating multiple conditions in a hierarchical manner.

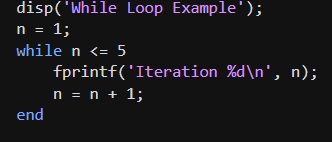
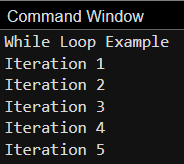


**E. switch:** used to evaluate a variable or expression against multiple possible values. It provides a cleaner alternative to using multiple if-else statements when you need to compare a variable with several different values.

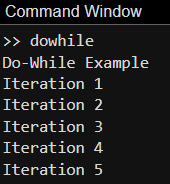
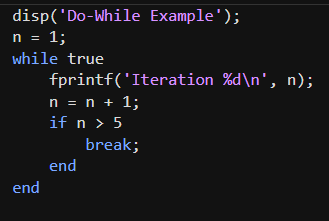
**2. Loops**

In MATLAB, loops are used to repeatedly execute a block of code. The different types of Loops in MATLAB are:

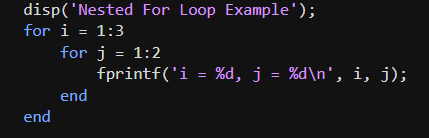
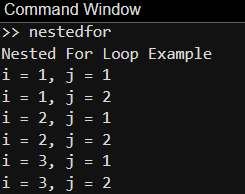
**A. For Loop:** Used to iterate a specific number of times over a range or array.

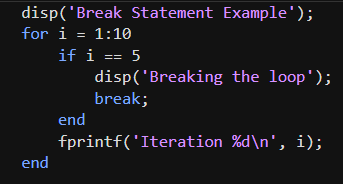
**B. While Loop:** Repeats the code block as long as the specified condition is true.

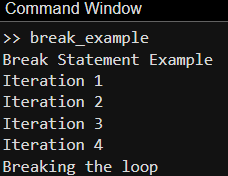
**C. Do-While Loop:** Executes the code block at least once and then checks the condition.

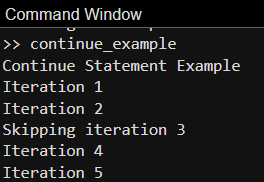


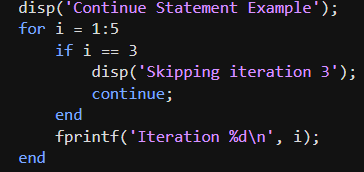
**D. Nested-For Loop:** Loops inside another loop, used for multidimensional operations.



**E. Break Statement:** Exits the loop immediately, regardless of the loop condition.



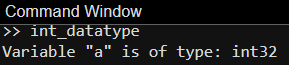
**F. Continue Statement:** Skips the current iteration and moves to the next one.

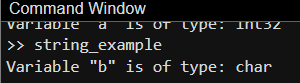


**3. Datatypes**

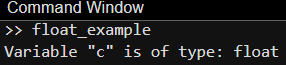
In MATLAB, data types refer to the kind of data that can be stored and manipulated. Common data types include:

**A. Int:** refers to integer data types that store whole numbers. These are used when you want to work with numerical values without any decimal points.

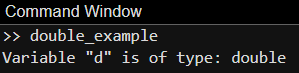


**B. String:** a sequence of characters used to represent text.

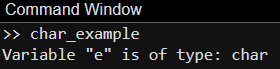
**C. Float:** In MATLAB, a float (short for floating-point number) refers to a data type used to represent real numbers (i.e., numbers that may have decimal points).



**D. Double:** In MATLAB, double is the default data type used to store floating-point numbers. It provides double-precision (64-bit) representation.



**E. Char:** In MATLAB, char (short for character) is a data type used to store text as a character array. Each character in the array is stored as an individual element.

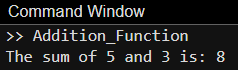
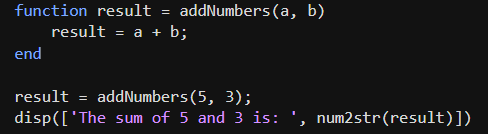
****

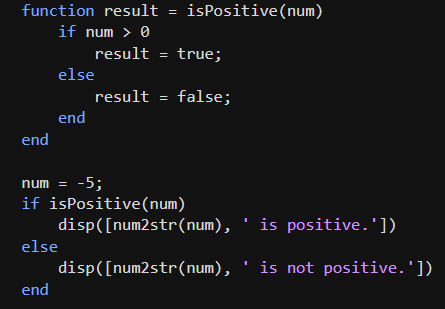
**4. Functions**

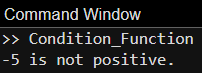
In MATLAB, functions are blocks of code designed to perform specific tasks and can be reused throughout a program. Functions take input arguments, process them, and return output values.

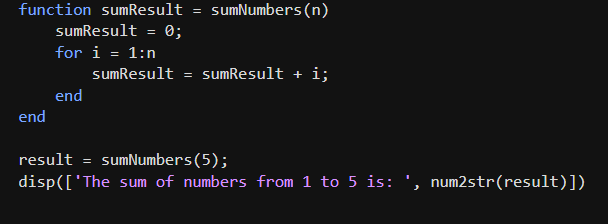
**Types of Functions in MATLAB:**

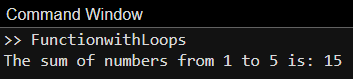
**A. User-defined function:** a custom function created by the user to perform specific tasks that are not built into MATLAB. These functions help in organizing code, avoiding redundancy, and improving program readability.

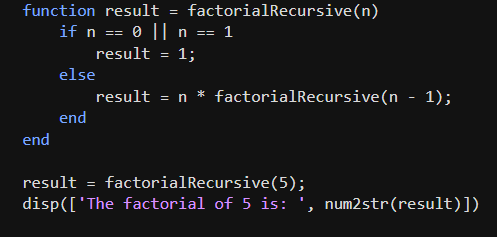


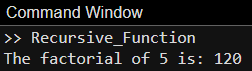
**B. Condition Function:** a user-defined or built-in function that uses conditional statements to make decisions based on certain criteria. It typically evaluates conditions using if, else, and elseif constructs.

****

**C. Function with Loops:** contains repetitive operations using for or while loops. Loops inside functions allow repeated execution of tasks until a condition is met or for a fixed number of iterations.

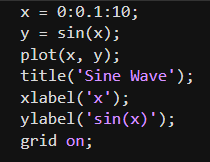
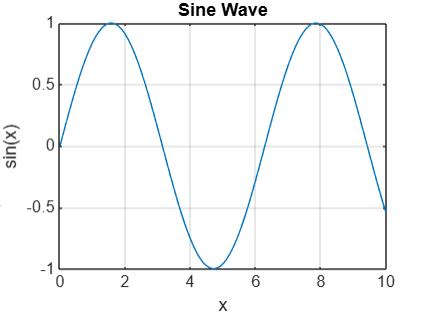
****

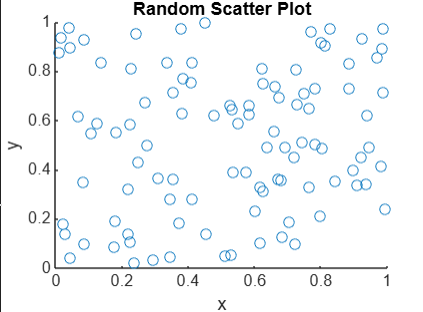
**D. Recursive Function:** function that calls itself in order to solve smaller instances of the same problem.

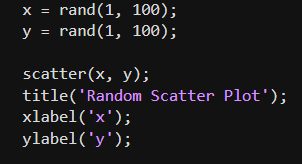
****

**5. Plots**

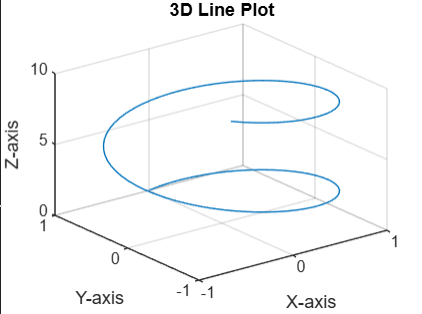
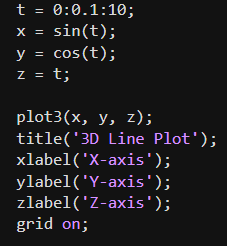
In MATLAB, plotting is a fundamental feature used for visualizing data. You can create various types of plots such as line plots, scatter plots, bar plots, and more. Here are some common plotting functions and their usage:

**A. Line Plot:** A line plot is a type of graph that represents data points as dots connected by straight lines. It is commonly used to show trends or continuous data over a variable.

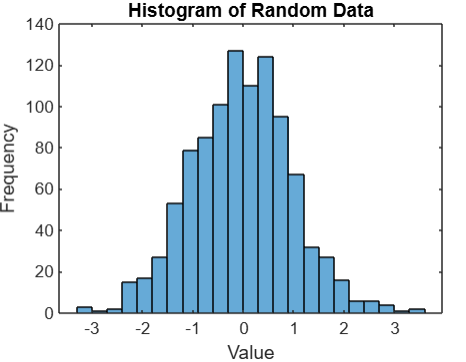
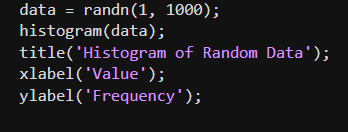
**B. Scatter Plot:** A scatter plot is a type of graph that displays individual data points on a two-dimensional plane. It is useful to visualize the relationship between two variables.

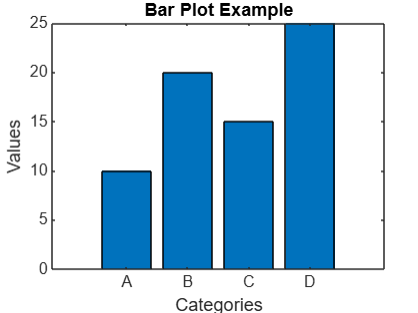


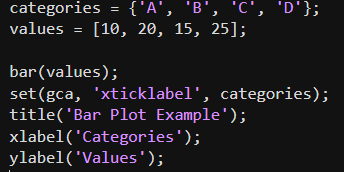
**C. 3D Line Plot:** A 3D line plot is used to represent data in three dimensions. It plots data points in 3D space and connects them with lines.



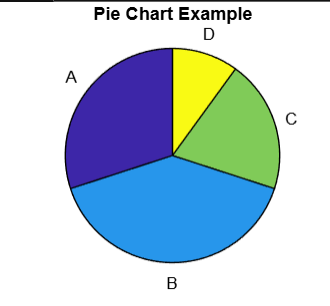
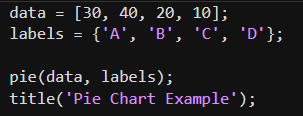
**D. Histogram:** A histogram is a graphical representation of the distribution of a dataset. It shows the frequency of data points within certain ranges or bins.



**E. Bar Plot:** A bar plot represents data with rectangular bars where the length of each bar is proportional to the value of the data. It is commonly used for categorical data.



**F. Pie Chart:** A pie chart is a circular statistical graphic that is divided into slices to illustrate numerical proportions. It is often used to represent parts of a whole.



**6. Array and Cell Array**

In MATLAB, arrays and cell arrays are two commonly used data structures, but they are used for different purposes and have different characteristics.

**A. Array:** An array in MATLAB is a collection of elements of the same data type stored in a single variable. Arrays can be 1D (vectors), 2D (matrices), or even higher-dimensional. Types of Arrays are:

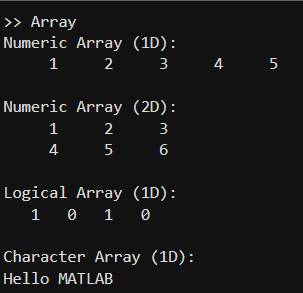
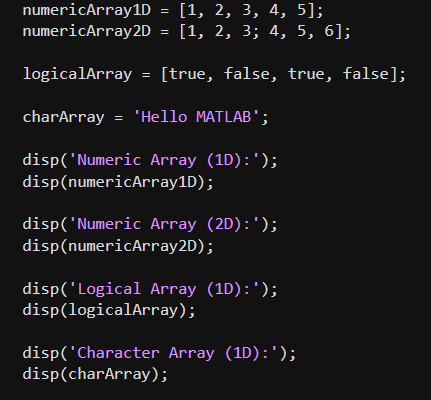
**a) Numeric Array**: A numeric array in MATLAB is an array that contains numbers, either integers or floating-point values. Numeric arrays are the most commonly used type for storing and manipulating numerical data.

* **1D Numeric Array (Vector):** A one-dimensional array with a sequence of numbers.
* **2D Numeric Array (Matrix):** A two-dimensional array (matrix) of numbers arranged in rows and columns.

**b) Logical Array:** A logical array contains logical values: true (1) or false (0). Logical arrays are typically used for storing conditions or performing logical operations such as comparisons or filtering.

* **1D Logical Array:** A one-dimensional logical array storing true or false values.

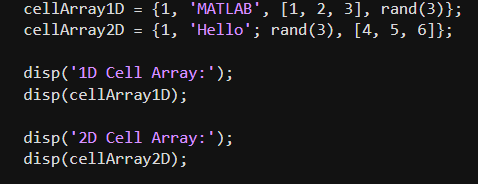
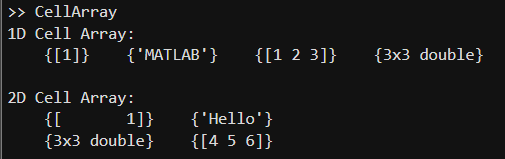
**c) Character Array**: A character array in MATLAB is used to store characters (individual letters, symbols, or words). It is essentially a sequence of characters and is also used for strings in earlier versions of MATLAB.

* **1D Character Array (String):** A single string represented by an array of characters.

**B. Cell Array**

A cell array is a data structure in MATLAB that allows you to store data of different types and sizes in a single array. Each element in a cell array can be of any data type (numeric, string, array, another cell array, etc.), making it extremely flexible.

* **1D Cell Array:** A one-dimensional cell array where each element can store different data types, such as numbers, strings, or arrays.
* **2D Cell Array:** A two-dimensional cell array, where each cell can contain different types of data.

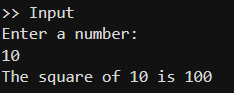
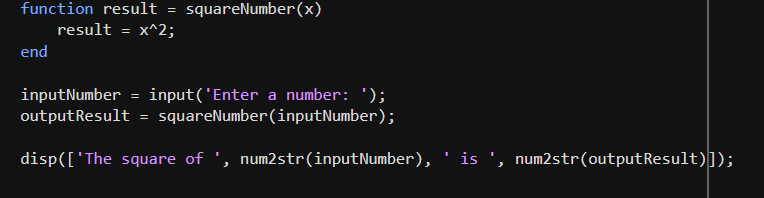


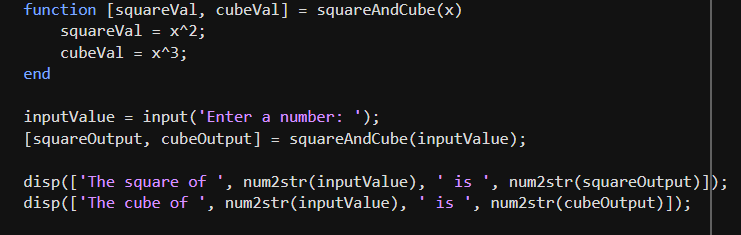
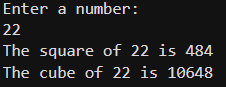
**7. Inputs and Outputs**

In MATLAB, handling inputs and outputs is an essential part of creating functions and scripts. The concept of inputs and outputs allows functions to accept data and return results.

**A. Inputs in MATLAB:** Inputs in MATLAB are variables or data passed into functions for processing. These inputs allow a function to operate on different data each time it is called.

**B. Outputs in MATLAB:** Outputs in MATLAB are the results returned by a function. These outputs are the processed data that is produced from the function’s input arguments.

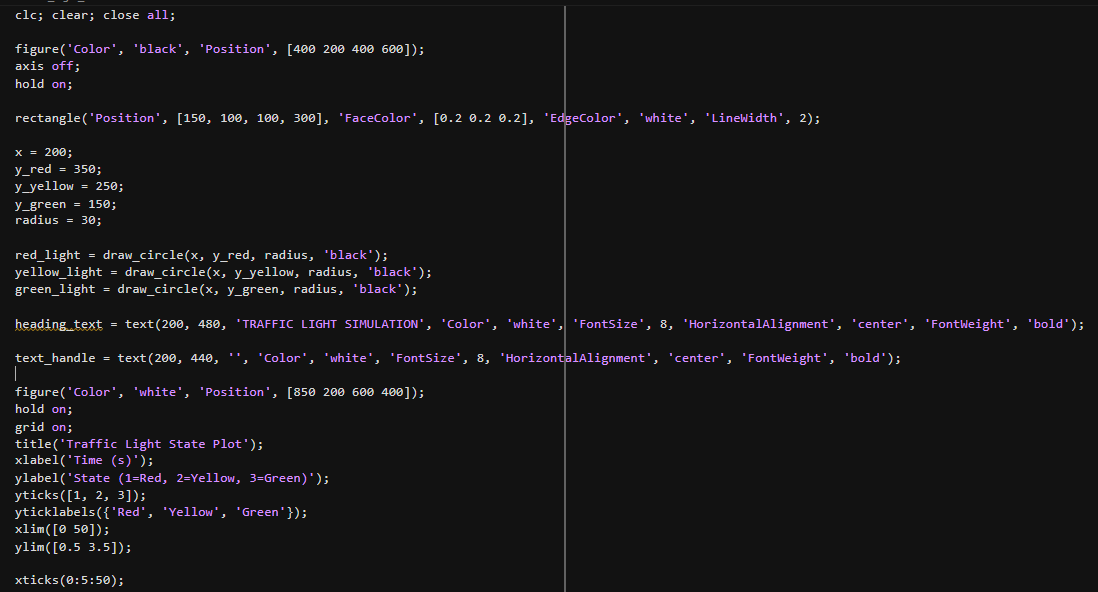


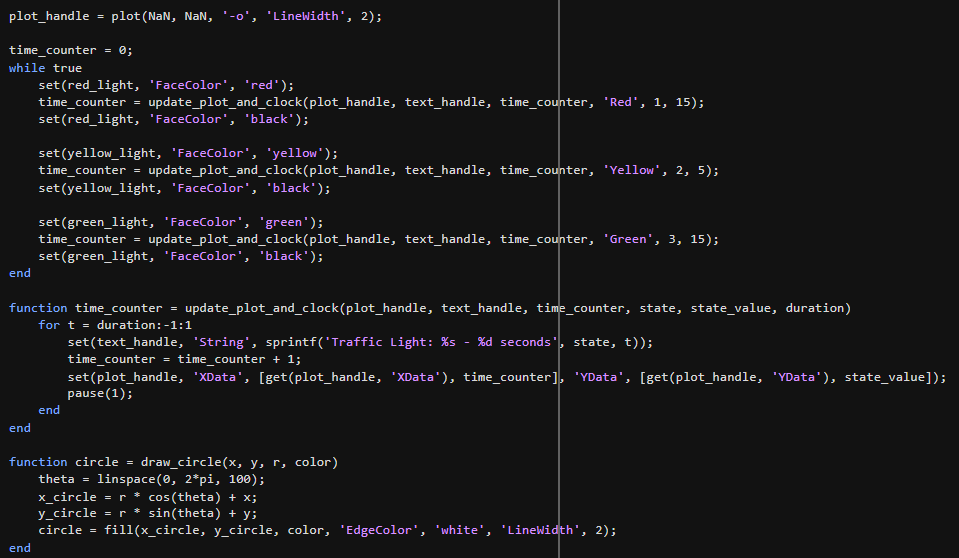


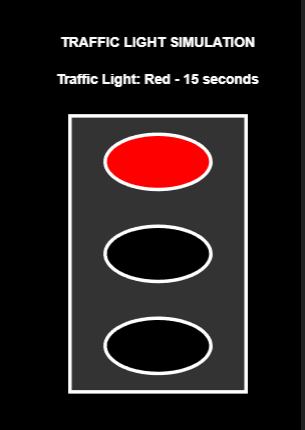
**Practical 2**

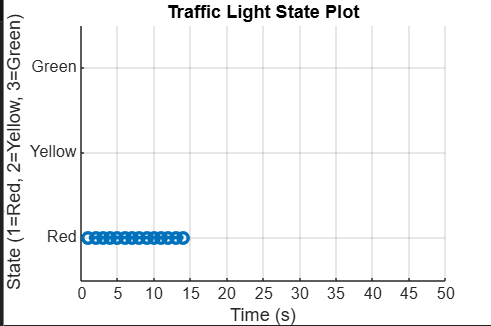
**Aim**

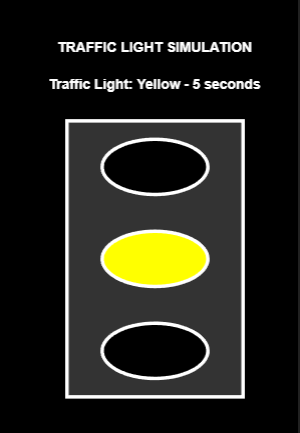
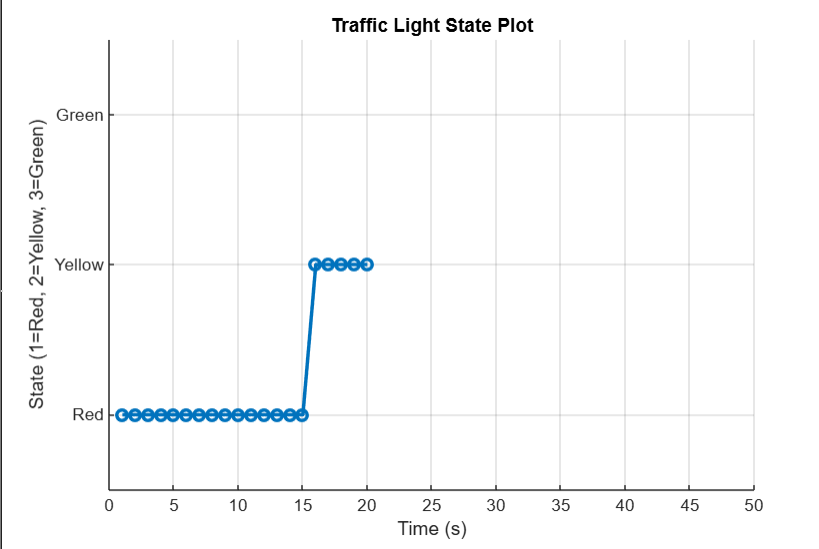
Implementation of Traffic Light Simulation Using MATLAB.

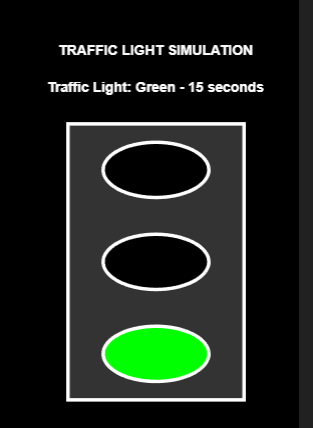
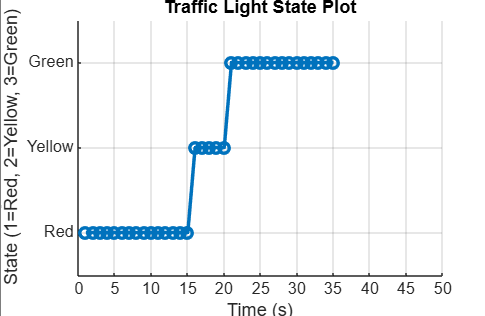
**Code for Implementation of Traffic Light Simulation Using MATLAB**



**Output**

**1. Red Light**

**2. Yellow Light**



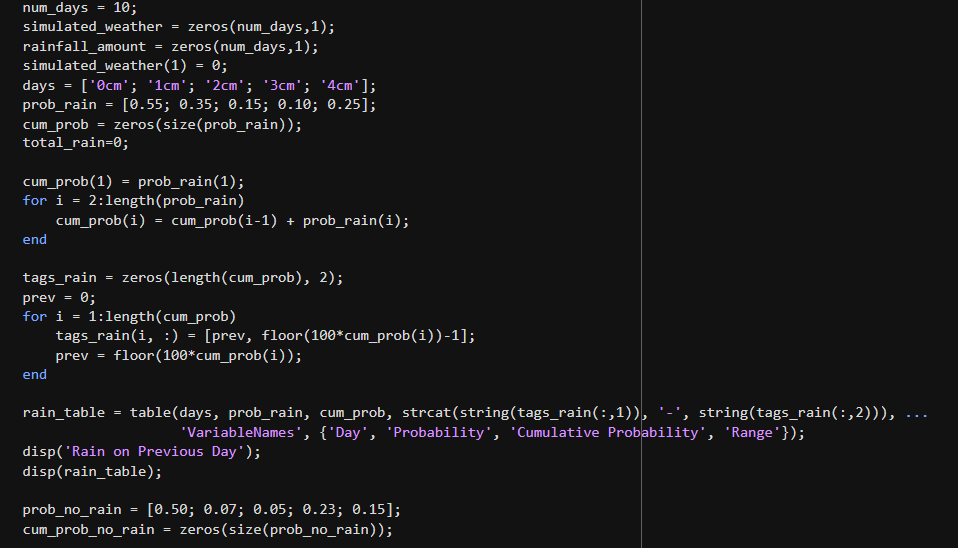
**3. Green Light**

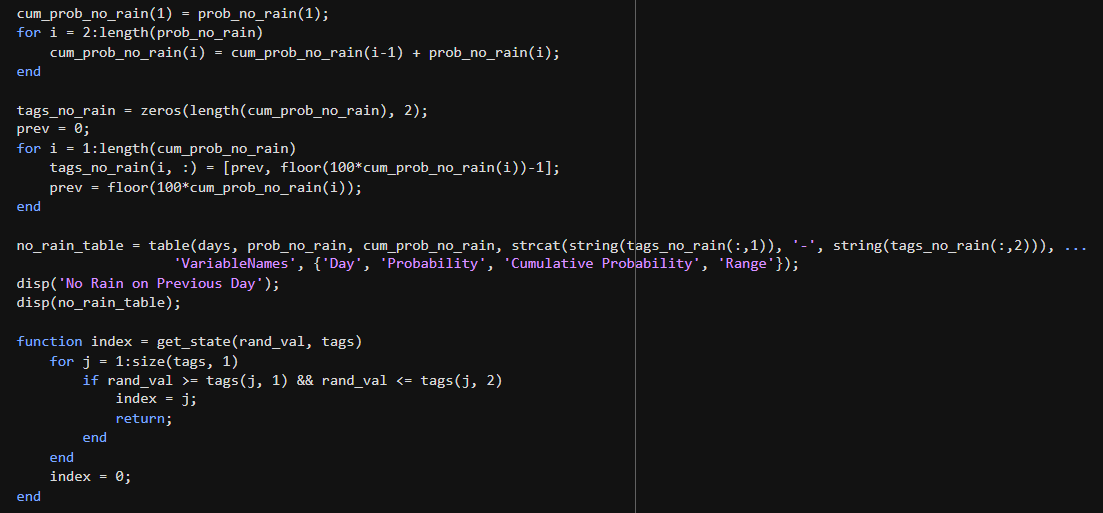
**Practical 3**

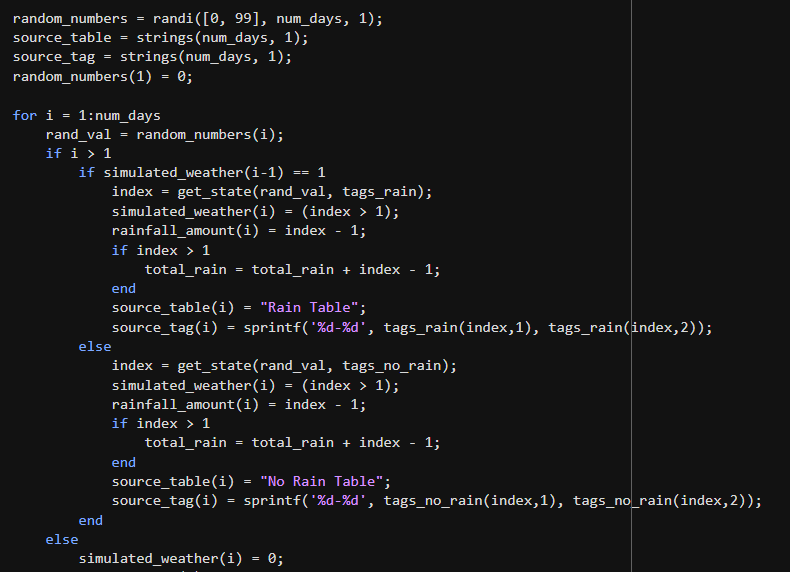
**Aim**

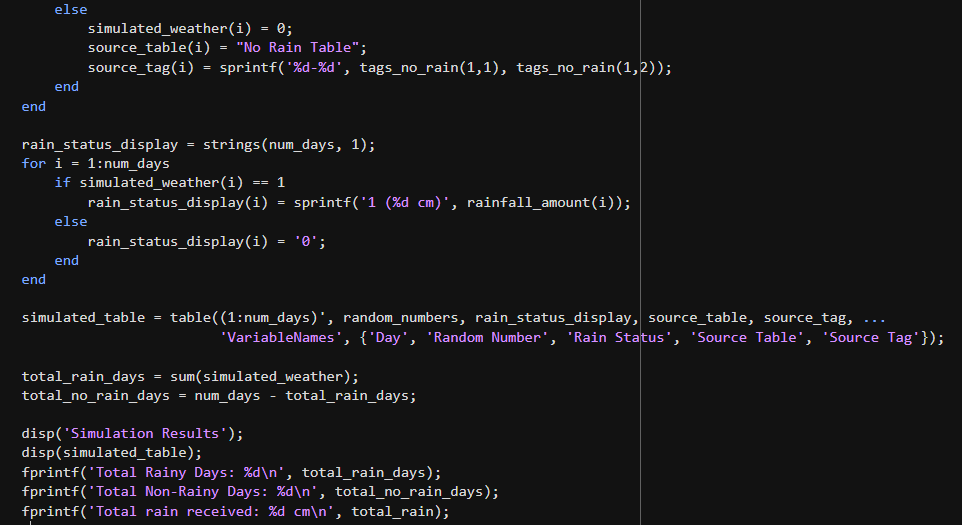
Rainfall Prediction using Monte Carlo Simulation

**Code for Implementation of Rainfall Prediction using Monte Carlo Simulation** **Using MATLAB**

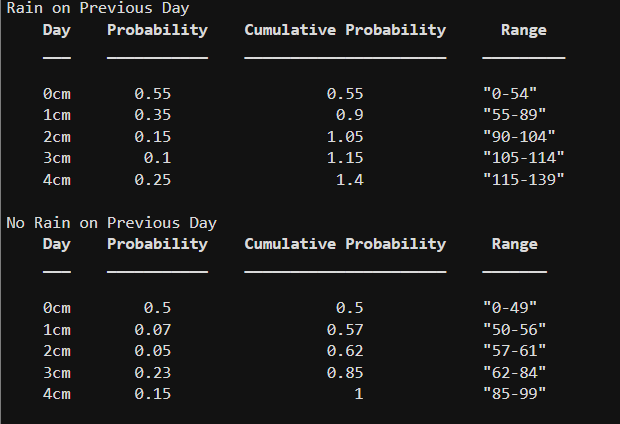


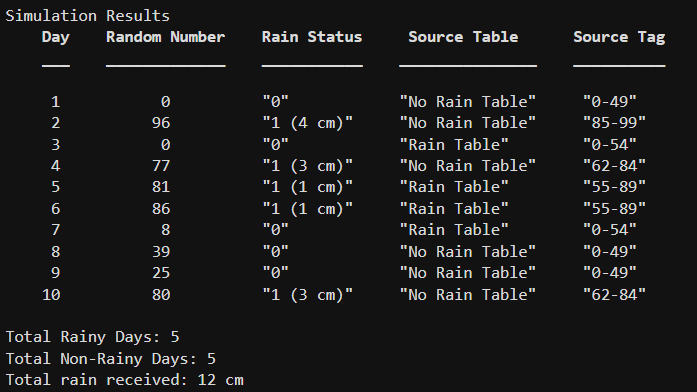






**Output**





**Practical 4**

**Aim**

Simulate queueing system at ATM using Poisson Distribution.

**Introduction to ATM Queueing System using Poisson Distribution**

An Automated Teller Machine (ATM) queueing system can be analysed using **Queueing Theory** and **Poisson Distribution**, which helps in understanding how customers arrive at the ATM and how long they take to get served.

**Implementation of Queuing Systems**

The implementation of a queuing system involves several key steps:

**1. Generating Random Arrival and Service Times:**

* The inter-arrival times are generated using an exponential distribution based on the Poisson process.
* Service times are also generated following an exponential distribution.

**2. Computing Customer Arrival Times:**

* The cumulative sum of inter-arrival times determines when each customer arrives at the system.

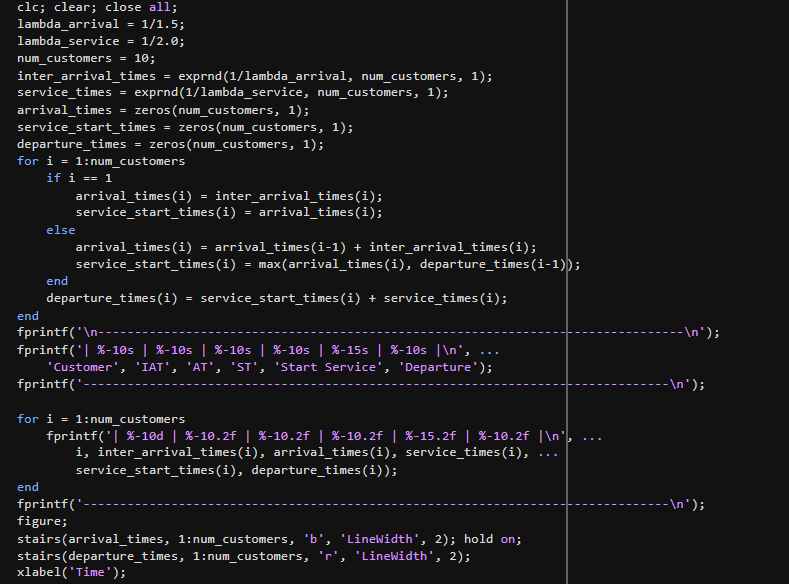
**3. Determining Service Start and Departure Times:**

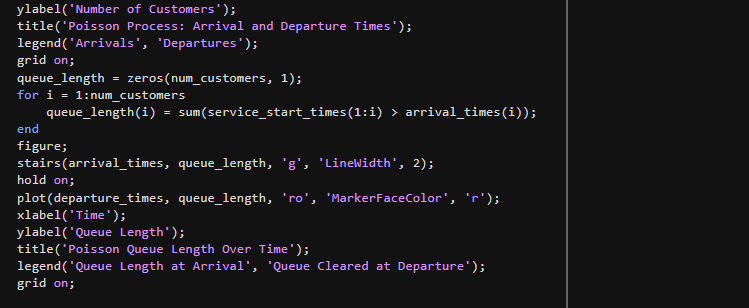
* The service start time for each customer is determined by checking the availability of the server.
* If the server is busy, the customer has to wait until the previous customer departs. o Departure time is computed as the sum of service start time and the respective service duration.

**4. Measuring Server Utilization:**

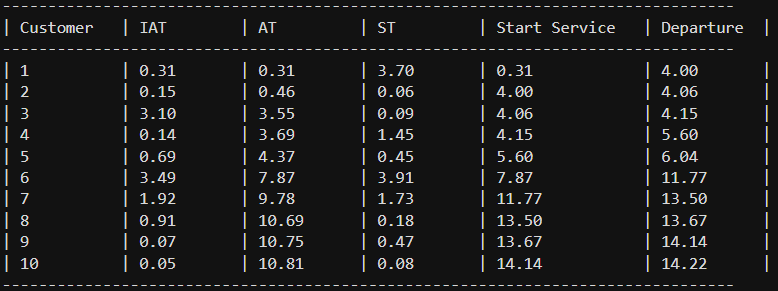
* The total time the server is busy is recorded.
* The proportion of busy time to total elapsed time is calculated to determine server efficiency.

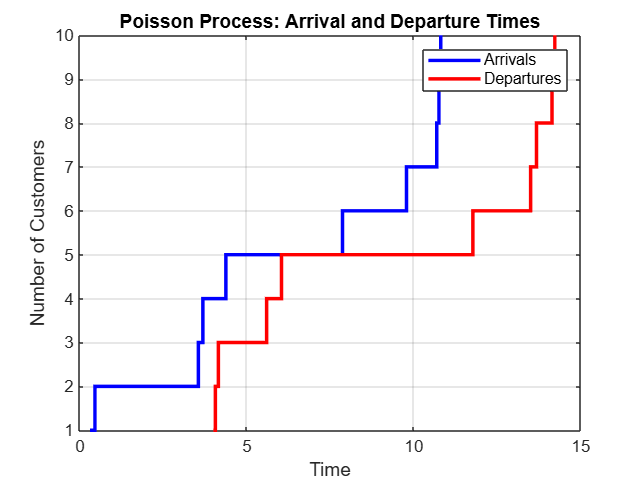
**Code for Implementation of Simulating queueing system at ATM using Poisson Distribution**

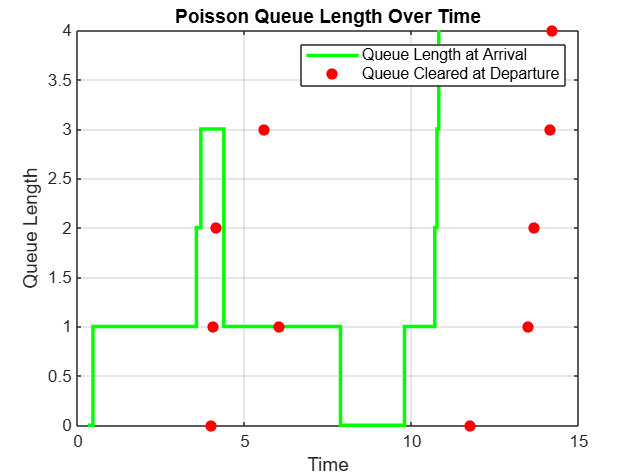




**Output**







**Practical 5**

**Aim**

Simulation of Zigzag Walking Person.

**Introduction to Zigzag Walking Person**

The **Zigzag Walking Person** is a simulation that demonstrates a structured movement pattern where a person moves in a predefined zigzag manner. Unlike random walking, where the direction is unpredictable, zigzag walking follows a fixed sequence of movements (Left, Forward, Right, Forward, and repeats).

This type of movement is commonly seen in:

* **Everyday life** – A person navigating through a crowded space in a zigzag pattern.
* **Sports & Athletics** – Athletes performing agility drills using zigzag running.
* **Robotics & AI** – Autonomous robots using systematic zigzag scanning to explore areas.
* **Nature** – Animals moving in zigzag paths to evade predators or track prey.

**Concept of Zigzag Walking Person**

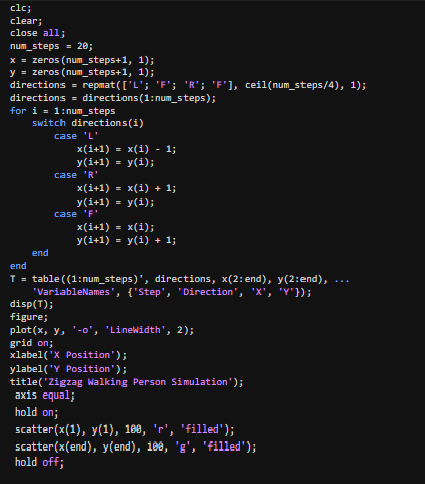
The movement pattern consists of three primary directions:

1. **Left (L):** Move one step left.
2. **Forward (F):** Move one step forward (upward).
3. **Right (R):** Move one step right.

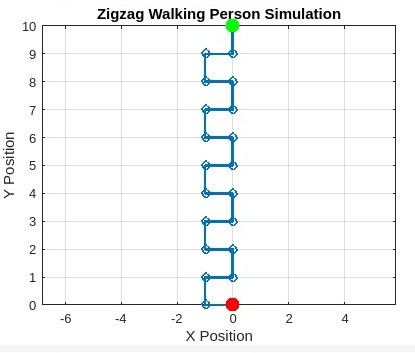
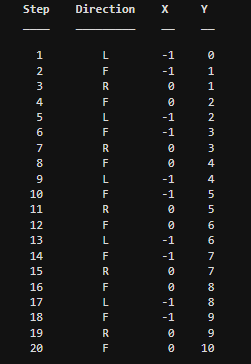
This pattern repeats, creating a zigzag motion on a 2D plane.

.

**Code for Implementation of Simulation of Zigzag Walking Person**

****

**Output**

****

**Practical 6**

**Aim**

Simulation of Fire Spread in a Forest using cellular automata.

# **Introduction to Fire Spread in a Forest Simulation**

Forest fires are a natural phenomenon that can cause widespread destruction to ecosystems. To study and predict the behavior of fire spread, **cellular automata (CA)** can be used as a computational model. **Cellular automata** are grid-based systems where each cell follows simple rules based on its neighbors, leading to complex emergent behavior.

# **Concept of Cellular Automata in Fire Spread**

A **cellular automaton** consists of:

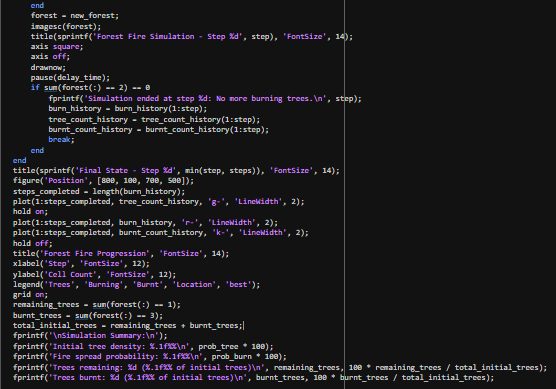
* A **grid** where each cell represents a portion of the forest.
* A set of **states** (e.g., Tree, Burning, Empty).
* **Rules** determining how fire spreads between neighboring cells.

Each tree can be in one of the following states:

* **Tree (Green):** Unburned and susceptible to catching fire.
* **Burning (Red):** Currently on fire and will spread flames.
* **Empty (Gray):** Already burned down or cleared.

At its core, this simulation helps in understanding **how fire propagates through different terrains** and the **factors that influence its speed and intensity**. The **spread of fire is not random**—it follows physical principles**.**

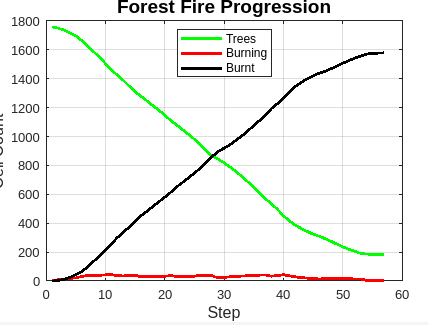
# **Code for Implementation of Simulation of Fire Spread in a Forest using Cellular Automata**



**Output**

**IGNITION POINTS**

OutputIGNITION POINTS



**Practical 7**

**Aim**

Estimating **π (Pi)** using Monte Carlo simulation.

**Introduction to Determination of value of Pi by Monte Carlo Method**

The value of Pi can be determined by using the relation for area of a circle.

Area of circle= pi\*r^2 Area of the quadrant is pi\*r^2/4 =pi/4

AII points satisfying the equation x^2 + y^2 <= 1 x, y >= 0 lie in this quadrant. Now if we have a pair of random numbers R1 and R2 in the range (0, 1), then the point R1 and R2 may lie within the quadrant or outside the quadrant but within the square enclosing the quadrant. If we generate a large number of such points (say N) by taking pairs of random numbers, and out of them M lie within the quadrant, then the ration M/N will approach the area under the curve, which is pi/4.

This method uses a **randomly generated set of points** to approximate the area of a circle inscribed within a square. The ratio of points that fall inside the circle to the total number of points helps us determine the value of **π**.

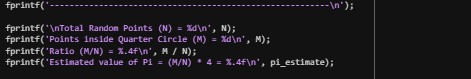
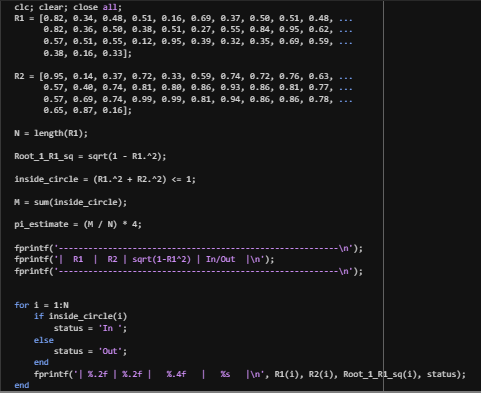
The computations for 33 points are being used in the code. For each pair of random numbers, the point is within the quadrant, when R1^2 + R2^2 <= 1.

# **R2 <=square root 1-R1^2**

The **Monte Carlo method** is a **stochastic (randomized) computational technique** used to solve mathematical and scientific problems through **random sampling**. It is widely applied in fields such as **numerical integration, optimization, risk analysis, and physics simulations**.

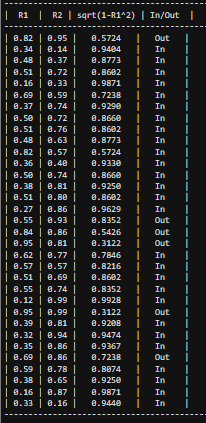
In the context of **π estimation**, the Monte Carlo method works by **randomly generating points** and checking whether they lie inside a **unit circle** inscribed within a **square**. By analyzing the ratio of points inside the circle to the total number of points, we can derive an approximation of **π**.

**Code for Implementation of Estimating π (Pi) using Monte Carlo simulation**



.

**Output**

****

**Practical 8**

**Aim**

To simulate code for Shortest Remaining Time First (SRTF) Scheduling Algorithm.

**Introduction to Shortest Remaining Time First (SRTF) Algorithm**

The **Shortest Remaining Time First (SRTF)** is a preemptive version of the **Shortest Job First (SJF)** algorithm. It selects the process with the shortest remaining CPU burst time for execution.

**Key Characteristics of SRTF:**

* It continuously checks the CPU burst length of arriving processes and preempts the current process if a shorter process arrives.
* It minimizes the average waiting time, making it an optimal scheduling algorithm in theory.
* The main challenge is predicting the exact CPU burst time for each process.

**Working of SRTF:**

* If no process is running, the CPU selects the process with the shortest burst time.
* If a new process arrives with a burst time shorter than the remaining time of the current process, the CPU switches to the new process.
* The cycle continues until all processes are executed.

Since SRTF is preemptive, it ensures that shorter tasks do not get delayed by longer ones, leading to better responsiveness in multitasking environments.

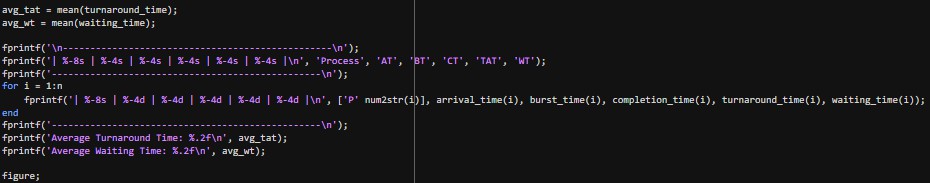
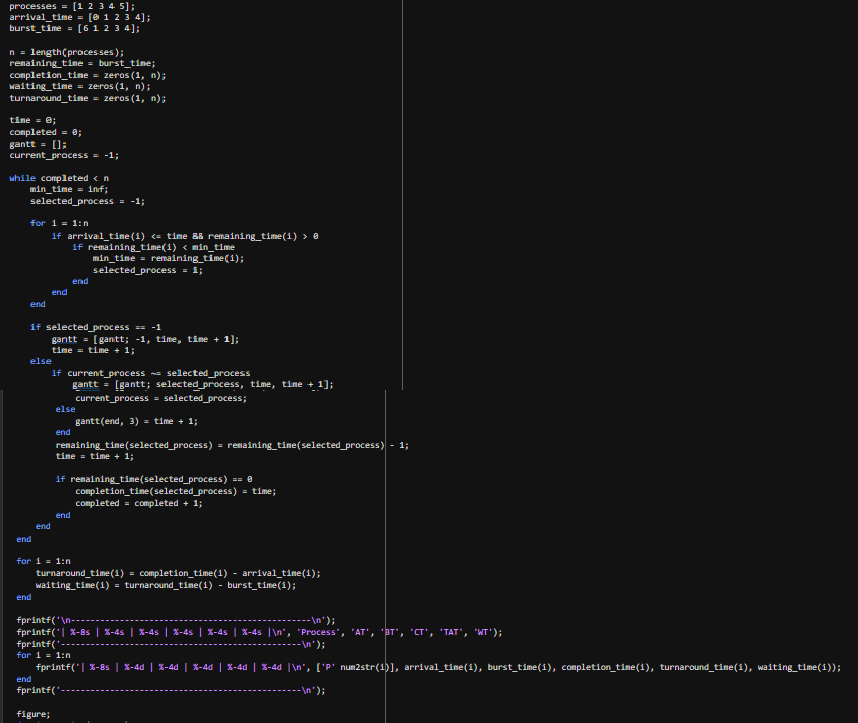
**Advantages of SRTF:**

* Ensures **shorter processes** are executed quickly.
* **Reduces average waiting time** compared to other scheduling algorithms.
* More efficient in **time-sharing** and **interactive systems**.

**Disadvantages of SRTF**:

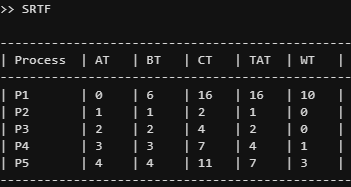
* **Difficult to implement** as predicting exact burst times is complex.
* Can cause **starvation** for longer processes, as new short processes may continuously preempt them.
* Frequent **context switching** leads to increased overhead.

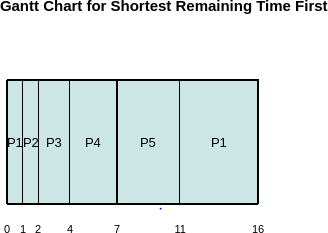
**Code for Simulating Shortest Remaining Time First Algorithm**



.

**Output**

****

****

**Practical 9**

**Aim**

**Simulation of a Water Tank Filling and Draining System over the time.**

**Introduction to Simulation of a Water Tank Filling and Draining System:**  
This project focuses on simulating a water tank system that undergoes both filling and draining processes over time. It aims to provide a simple yet effective representation of how water levels vary based on inflow and outflow rates.

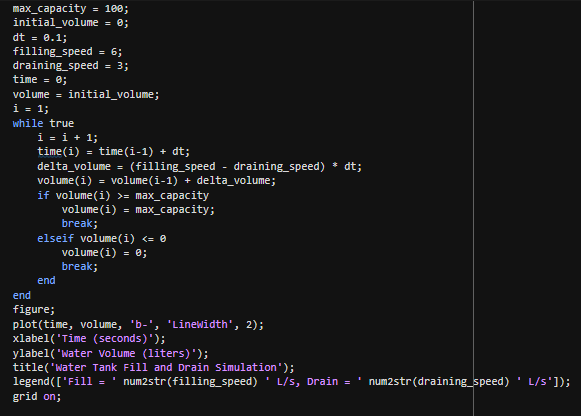
**Objective:**  
The main goal of this simulation is to understand the dynamic behaviour of water levels in a tank. By changing parameters like flow rate and time intervals, we can observe how the tank responds to different situations.

**Applications:**  
Such simulations are widely used in real-life applications like water resource management, automatic tank filling systems, industrial process control, and smart irrigation systems. It helps in designing systems that require efficient water level monitoring.

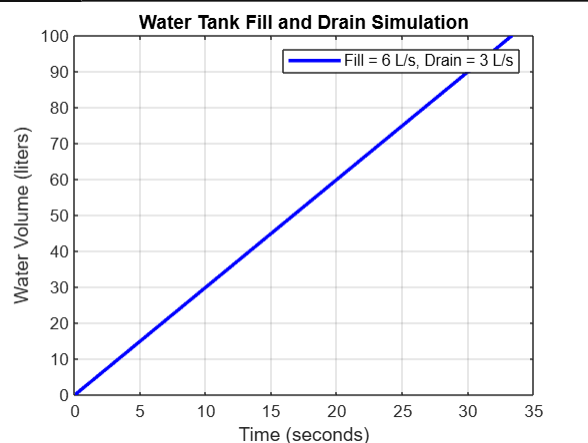
Moreover, this kind of model has practical applications in various fields, including agriculture (for automated irrigation), civil engineering (for water supply planning), and smart home systems (for efficient water usage). It can also serve as an educational tool for students learning about control systems, fluid dynamics, or environmental science.

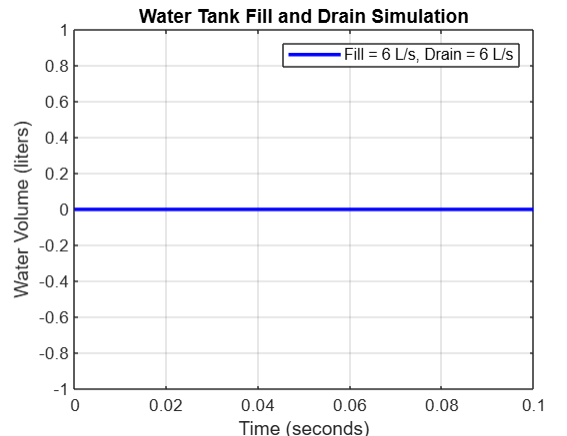
The simulation output is typically presented in the form of visual graphs or animations that show the tank's water level over time. This makes it easier to analyse trends and detect any inconsistencies or inefficiencies in the system.

**Code for Implementation of Simulating Water Tank Filling and Draining**



**Output**





**Practical 10**

**Aim**

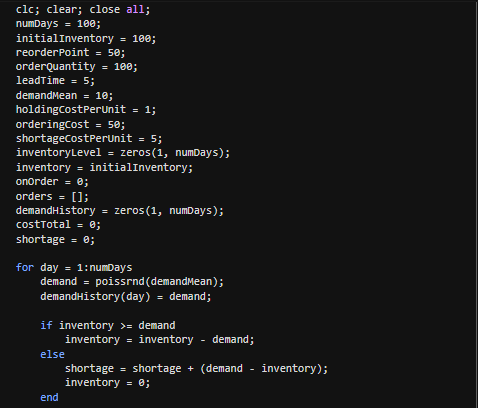
Simulation of Inventory Management for Supply Chain Optimization.

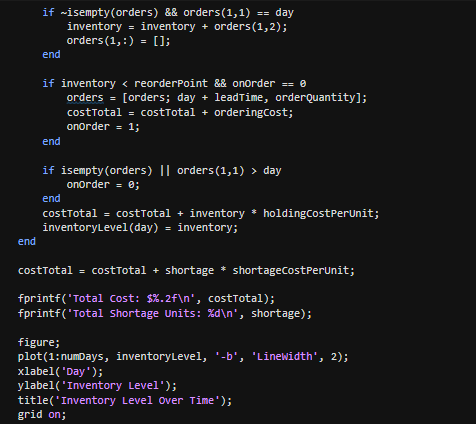
**Simulation of Inventory Management for Supply Chain Optimization**

Simulation-based inventory management is a modern approach that leverages computational models to replicate real-world supply chain behaviours. Instead of relying solely on historical data or fixed rules, simulation allows businesses to visualize and test various inventory strategies under dynamic and uncertain conditions such as fluctuating demand, variable lead times, and supply disruptions.

By virtually experimenting with different policies—like reorder points, safety stock levels, or batch ordering—organizations can predict outcomes and fine-tune operations without real-world risks. This empowers data-driven decisions that lead to reduced stockouts, minimized holding costs, and improved customer satisfaction.

Simulation acts as a digital twin of real-world supply chain systems. It provides a risk-free environment to analyse the impact of various inventory decisions before implementing them. From warehouse replenishment to demand forecasting, simulation enables businesses to assess the best-fit policies that minimize cost and maximize service levels.

**Code for Implementation of Simulation of Inventory Management for Supply Chain Optimization**



**Output**

