# Week 6 MATLAB Activities

## Contents

Week 6 MATLAB Activities ........................................................................1

Reading ........................................................................1

Tutorial 1: Comparing Strings ........................................................................2

Tutorial 2: Decisions, Decisions, Decisions ........................................................................2

Tutorial 3: Loops ........................................................................6

Tutorial 3: Histogram ........................................................................11

Assignment 1: Club Bouncer for Taylor Swift ........................................................................13

Assignment 2: MATLAB rocks! ........................................................................14

Assignment 3: 100 Positive Integers ........................................................................14

Assignment 4: Histogram ........................................................................15

Assignment Submission ........................................................................16

🚦 **Red light: No AI**

Time required: 120 minutes

1. Create a MATLAB script named **Wk06Lastname.m**
2. Save all programs in this script.
3. Include your name and date at the top of the script file as comments.
4. Put a Section Break between each program.

# Reading

Matlab A Practical Introduction to Programming and Problem Solving (Stormy Attaway)

Sections 5.1, 5.2

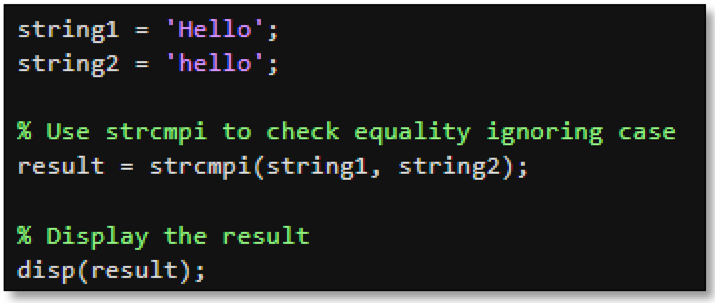
## Tutorial 1: Comparing Strings

The **strcmpi** function in MATLAB is used to compare two strings for equality, ignoring the case sensitivity. The function returns a logical value (true or false) based on whether the two input strings are equal when case differences are disregarded.

Here's the basic syntax:

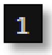
```
result = strcmpi(str1, str2);
```

- **str1** and **str2 -** The strings you want to compare.
- **result** - A logical value (1 for true, 0 for false) indicating whether the strings are equal, disregarding case.



```
string1 = 'Hello';
string2 = 'hello';

% Use strcmpi to check equality ignoring case
result = strcmpi(string1, string2);

% Display the result
disp(result);
```

Example run:



```
1
```

In this case, **result** would be 1, indicating that the two strings are considered equal when case differences are ignored.

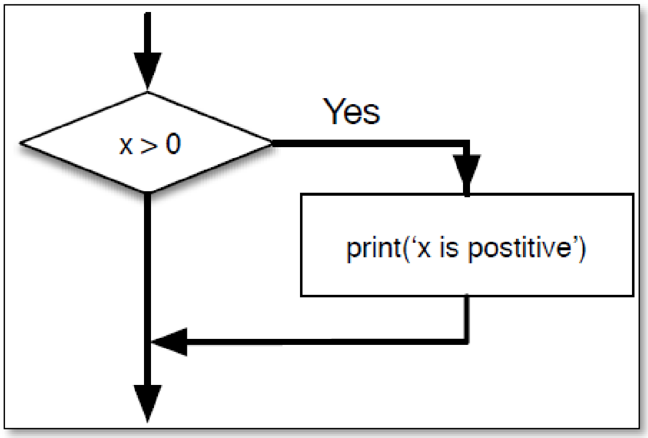
## Tutorial 2: Decisions, Decisions, Decisions

if statements in MATLAB are used to execute code based on a condition. They are fundamental for controlling the flow of a program.

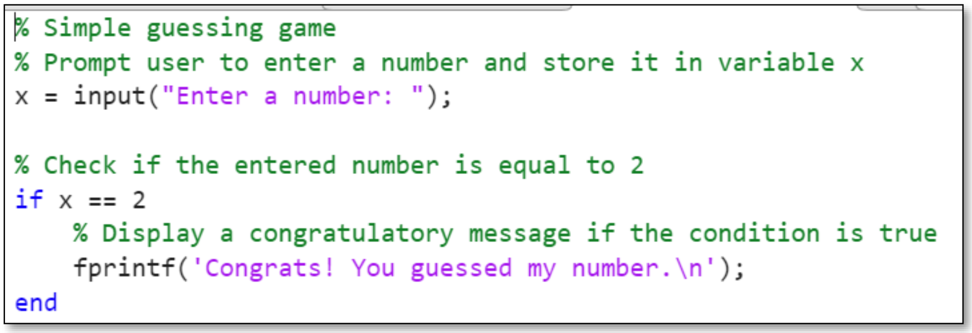
**if**

The basic syntax of an if statement in MATLAB is:

```
if condition
    % Code to execute if condition is true
end
```

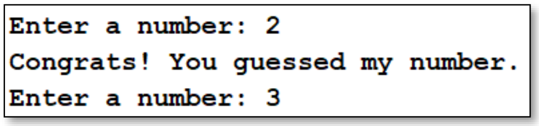


A basic if statement used in a guessing game



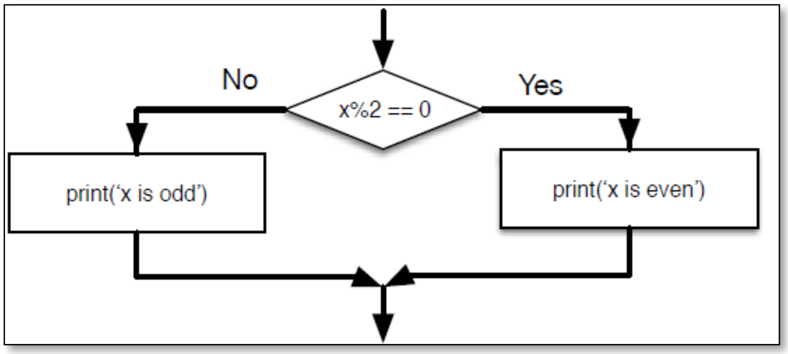
```
% Simple guessing game
% Prompt user to enter a number and store it in variable x
x = input("Enter a number: ");

% Check if the entered number is equal to 2
if x == 2
    % Display a congratulatory message if the condition is true
    fprintf('Congrats! You guessed my number.\n');
end
```

Example run:

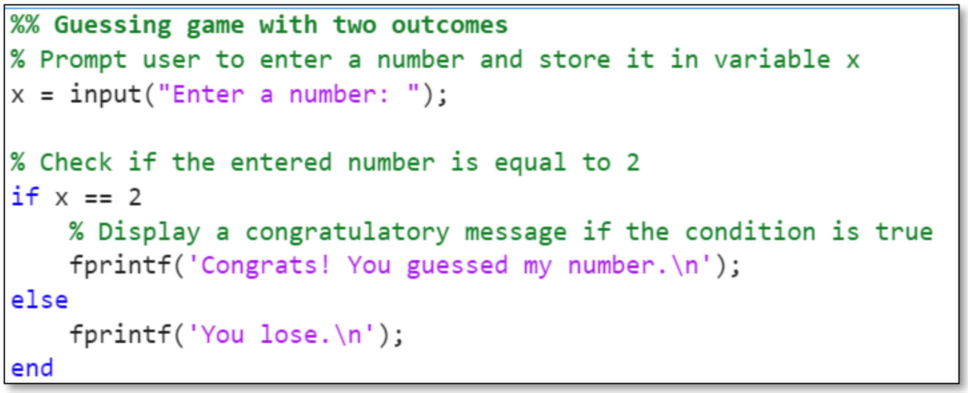


```
Enter a number: 2
Congrats! You guessed my number.
Enter a number: 3
```

**if else**

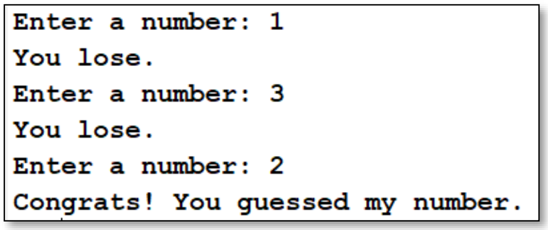


Here, the message "x is non-positive" will be displayed because the condition x > 0 is false.



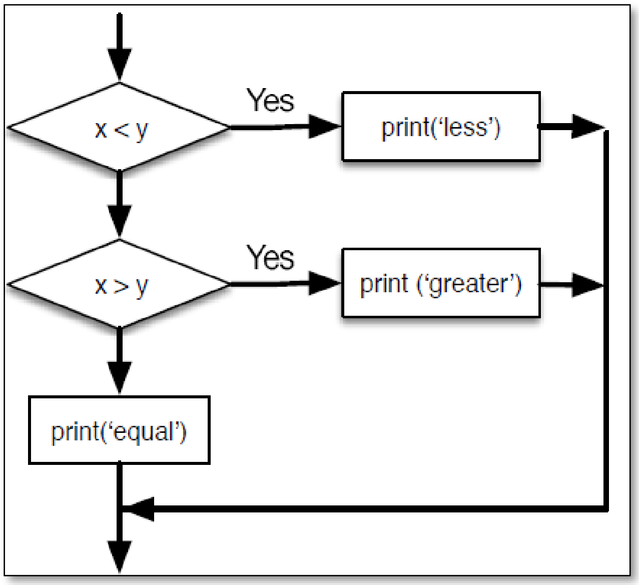
```
%% Guessing game with two outcomes
% Prompt user to enter a number and store it in variable x
x = input("Enter a number: ");

% Check if the entered number is equal to 2
if x == 2
    % Display a congratulatory message if the condition is true
    fprintf('Congrats! You guessed my number.\n');
else
    fprintf('You lose.\n');
end
```

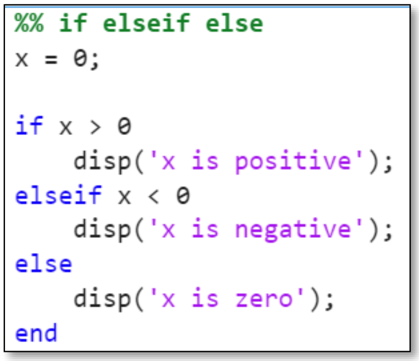
Example run:



```
Enter a number: 1
You lose.
Enter a number: 3
You lose.
Enter a number: 2
Congrats! You guessed my number.
```

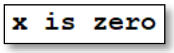
**if elseif else**





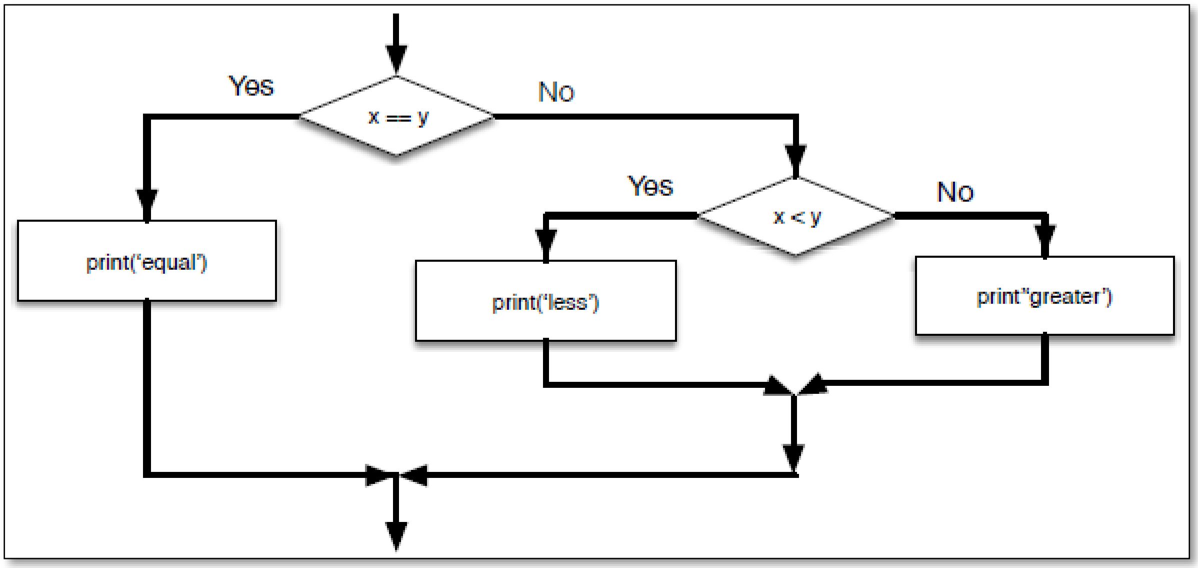
```
%% if elseif else
x = 0;

if x > 0
    disp('x is positive');
elseif x < 0
    disp('x is negative');
else
    disp('x is zero');
end
```

**Example run:**

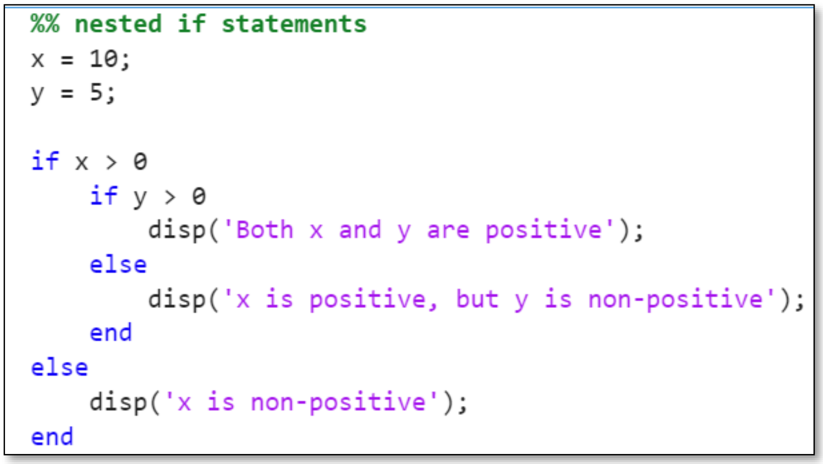


```
x is zero
```

**Nested if statements**

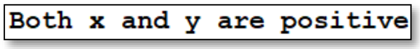


This example demonstrates how if statements can be nested within each other. The message "Both x and y are positive" will be displayed.



```
%% nested if statements
x = 10;
y = 5;

if x > 0
    if y > 0
        disp('Both x and y are positive');
    else
        disp('x is positive, but y is non-positive');
    end
else
    disp('x is non-positive');
end
```

Example run:



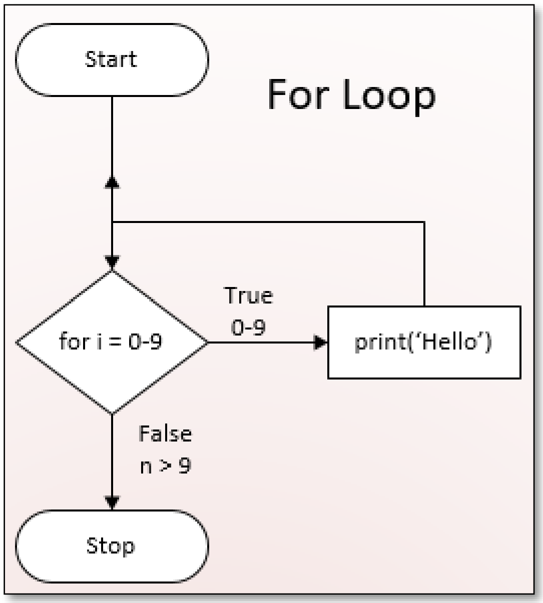
```
Both x and y are positive
```

## Tutorial 3: Loops

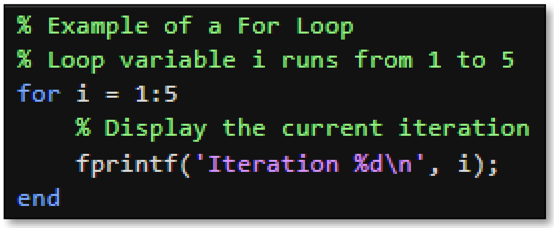
Loops in MATLAB are structures that enable the repeated execution of a block of code, providing a powerful mechanism for automating repetitive tasks. There are two primary types of loops in MATLAB: **for** and **while**.

**For Loop:**

- **Syntax:** **`for variable = start:step:end`**
- Used when the number of iterations is known in advance.
- The loop variable takes values from `start` to `end` with the specified step.
- Ideal for tasks with a fixed number of repetitions.



**For Loop**: A for loop is used when the number of iterations is known in advance.

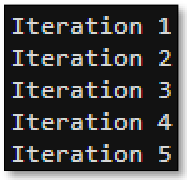


```
% Example of a For Loop
% Loop variable i runs from 1 to 5
for i = 1:5
    % Display the current iteration
    fprintf('Iteration %d\n', i);
end
```

Explanation:

- **for i = 1:5** - Initializes a loop that iterates over values of i from 1 to 5.
- **fprintf('Iteration %d\n', i);** Displays the current iteration using fprintf.

Example run:

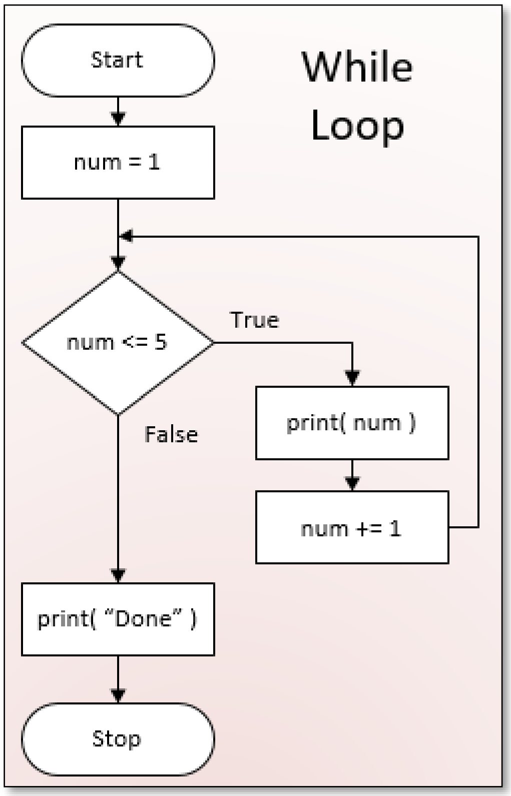


```
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
```

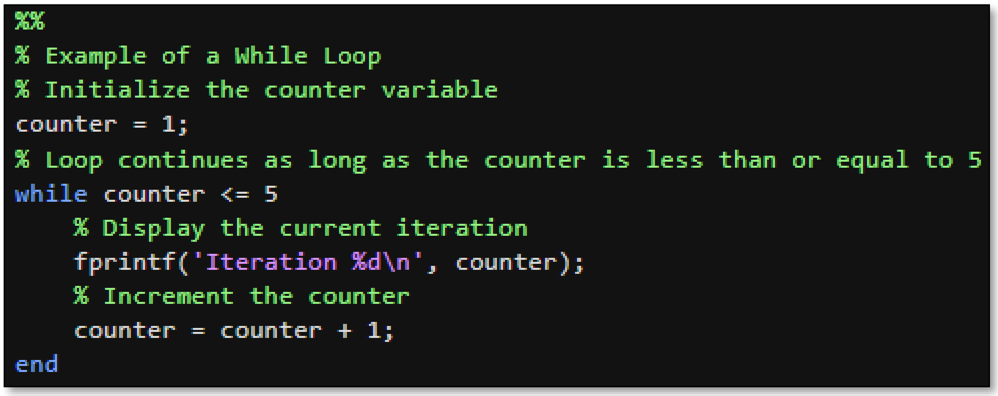
**While Loop:**

- **Syntax:** `while condition`
- Executes a block of code as long as the specified condition is true.
- Suitable when the number of iterations is not known beforehand.

- Requires an external condition to control loop termination.



**While Loop:** A **while** loop is used when the number of iterations is not known in advance, and the loop continues until a specific condition is met.

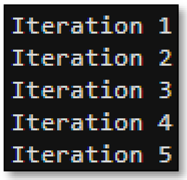


```
%%
% Example of a While Loop
% Initialize the counter variable
counter = 1;
% Loop continues as long as the counter is less than or equal to 5
while counter <= 5
    % Display the current iteration
    fprintf('Iteration %d\n', counter);
    % Increment the counter
    counter = counter + 1;
end
```

Explanation:

- **counter = 1;** - Initializes a counter variable outside the loop.
- **while counter <= 5 -** Specifies the condition for the loop to continue.
- **counter = counter + 1; -** Increments the counter within the loop.

Example run:



```
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
```

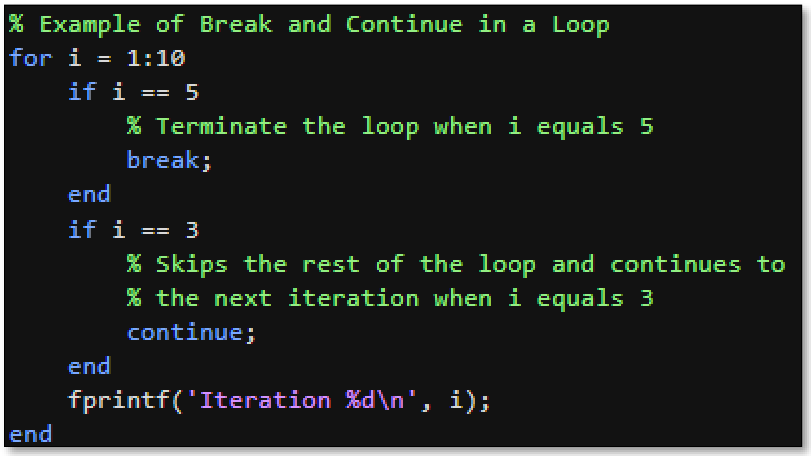
**Loop Control Statements:**

**Break:**

- **Syntax:** `break`
- Terminates the loop prematurely based on a specific condition.

**Continue:**

- **Syntax:** `continue`
- Skips the remaining code in the loop for the current iteration and moves to the next iteration.

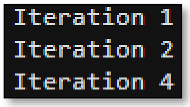


```
% Example of Break and Continue in a Loop
for i = 1:10
    if i == 5
        % Terminate the loop when i equals 5
        break;
    end
    if i == 3
        % Skips the rest of the loop and continues to
        % the next iteration when i equals 3
        continue;
    end
    fprintf('Iteration %d\n', i);
end
```

Explanation:

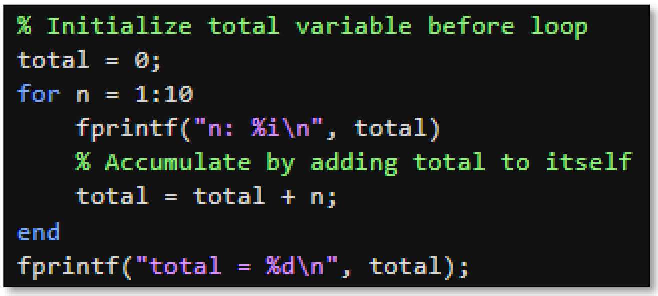
- **break; -** Exits the loop prematurely.
- **continue; -** Skips the rest of the loop and moves to the next iteration.

Example run:



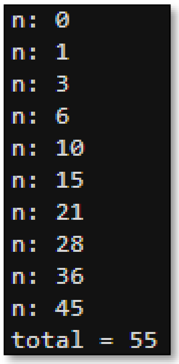
```
Iteration 1
Iteration 2
Iteration 4
```

There are times when we want to acculate many data points into a total.



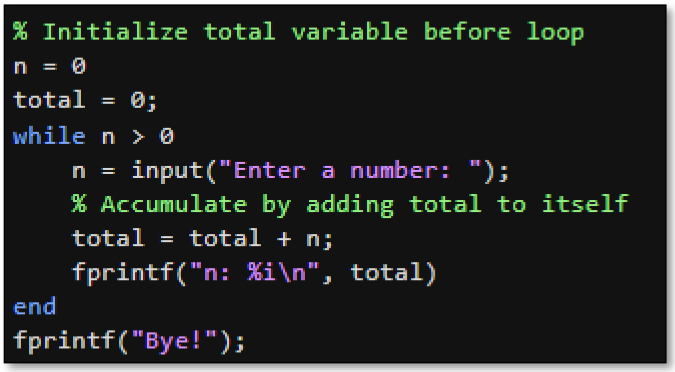
```
% Initialize total variable before loop
total = 0;
for n = 1:10
    fprintf("n: %i\n", total)
    % Accumulate by adding total to itself
    total = total + n;
end
fprintf("total = %d\n", total);
```

Example run:



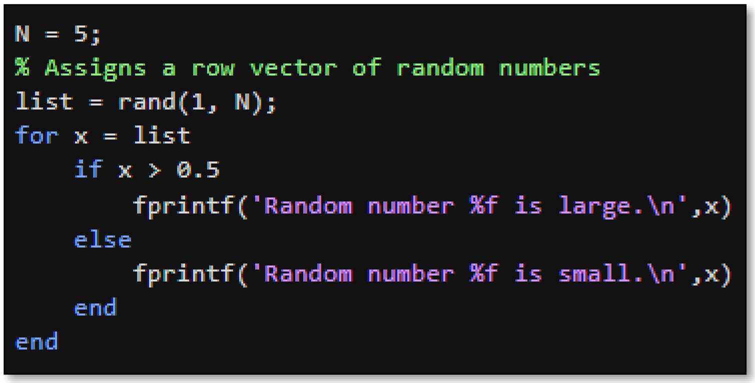
```
n: 0
n: 1
n: 3
n: 6
n: 10
n: 15
n: 21
n: 28
n: 36
n: 45
total = 55
```

The same with a while loop.



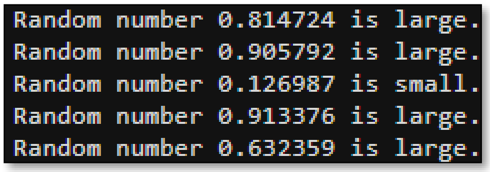
```
% Initialize total variable before loop
n = 0
total = 0;
while n > 0
    n = input("Enter a number: ");
    % Accumulate by adding total to itself
    total = total + n;
    fprintf("n: %i\n", total)
end
fprintf("Bye!");
```

You can combine a loop and a decision.



```
N = 5;
% Assigns a row vector of random numbers
list = rand(1, N);
for x = list
    if x > 0.5
        fprintf('Random number %f is large.\n',x)
    else
        fprintf('Random number %f is small.\n',x)
    end
end
```

Example run:

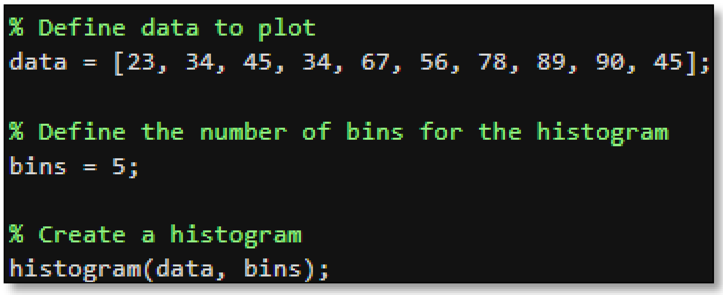


```
Random number 0.814724 is large.
Random number 0.905792 is large.
Random number 0.126987 is small.
Random number 0.913376 is large.
Random number 0.632359 is large.
```

## Tutorial 3: Histogram

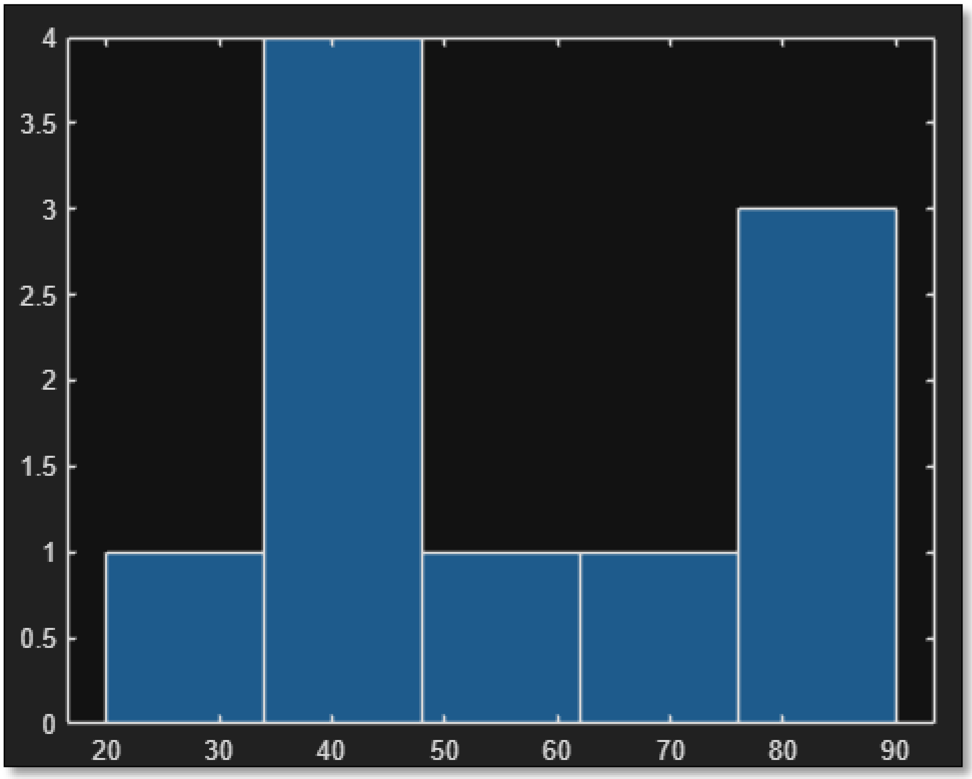
A histogram is a graphical representation of data distribution.

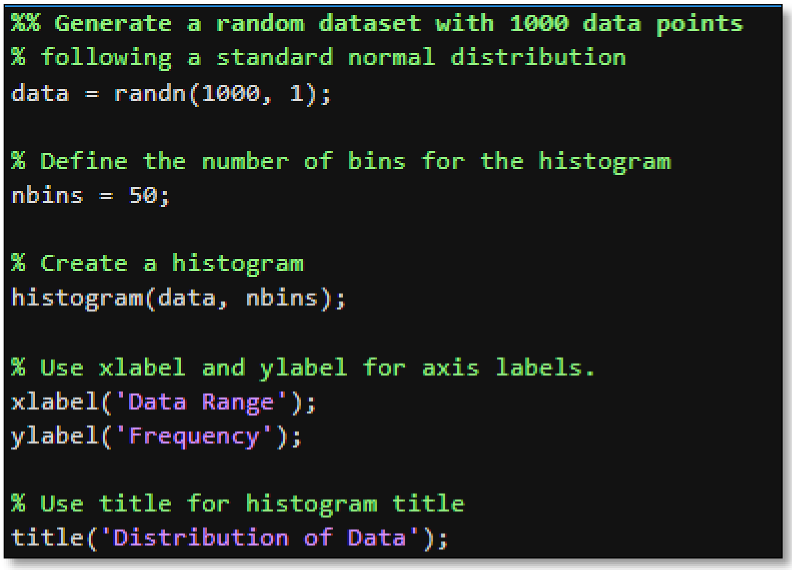
It consists of bars that represent the frequency or count of data within specific intervals. The data are grouped into equal intervals or bins.



```
% Define data to plot
data = [23, 34, 45, 34, 67, 56, 78, 89, 90, 45];

% Define the number of bins for the histogram
bins = 5;

% Create a histogram
histogram(data, bins);
```

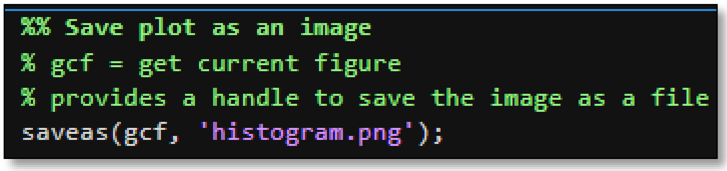
Example run:





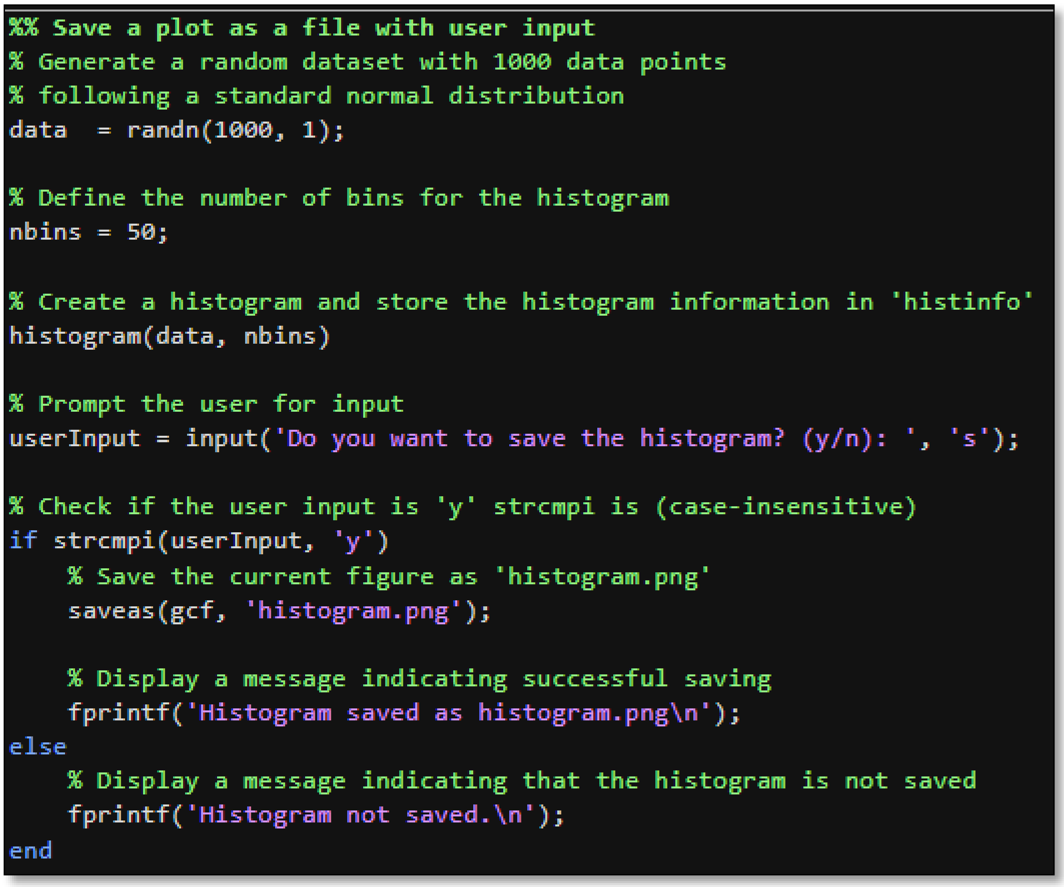
```
%% Generate a random dataset with 1000 data points
% following a standard normal distribution
data = randn(1000, 1);

% Define the number of bins for the histogram
nbins = 50;

% Create a histogram
histogram(data, nbins);

% Use xlabel and ylabel for axis labels.
xlabel('Data Range');
ylabel('Frequency');

% Use title for histogram title
title('Distribution of Data');
```

Add the following code to the previous code to save the plot as a file.



```
%% Save plot as an image
% gcf = get current figure
% provides a handle to save the image as a file
saveas(gcf, 'histogram.png');
```

This program displays a histogram and allows the user to choose to save the image.



```
%% Save a plot as a file with user input
% Generate a random dataset with 1000 data points
% following a standard normal distribution
data  = randn(1000, 1);

% Define the number of bins for the histogram
nbins = 50;

% Create a histogram and store the histogram information in 'histinfo'
histogram(data, nbins)

% Prompt the user for input
userInput = input('Do you want to save the histogram? (y/n): ', 's');

% Check if the user input is 'y' strcmpi is (case-insensitive)
if strcmpi(userInput, 'y')
    % Save the current figure as 'histogram.png'
    saveas(gcf, 'histogram.png');

    % Display a message indicating successful saving
    fprintf('Histogram saved as histogram.png\n');
else
    % Display a message indicating that the histogram is not saved
    fprintf('Histogram not saved.\n');
end
```

## Assignment 1: Club Bouncer for Taylor Swift

Taylor Swift is coming to town. She is playing at the local school auditorium. You want to buy a ticket. Before you can do that, you must write a program to determine who can enter the concert.

When you are testing a single variable for a range of values, when you have eliminated the upper range, you don’t have to test that any more.

For example: If someone is not older than 21, you don’t have to test again for not being older than 21.

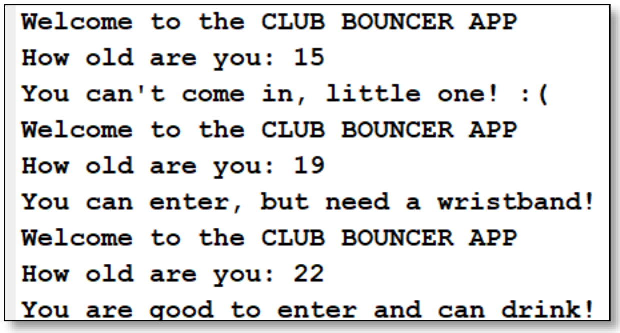
**NOTE:** You can’t have more than one age. You would not use multiple if statements. We will rarely if ever use multiple if statements for a single variable.

1. Ask for the user’s age.
2. If they are over 21 → they can enter and have a drink.

3. Else if they are between 18 and 21 inclusive (18, 19, 20, or 21) → They can enter but can’t have a drink. They must wear a wristband.

4. Otherwise, they are younger than 18 → They are too young.

Example run:

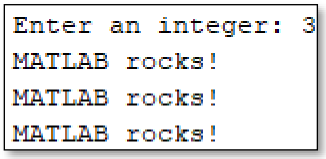


```
Welcome to the CLUB BOUNCER APP
How old are you: 15
You can't come in, little one! :(
Welcome to the CLUB BOUNCER APP
How old are you: 19
You can enter, but need a wristband!
Welcome to the CLUB BOUNCER APP
How old are you: 22
You are good to enter and can drink!
```

## Assignment 2: MATLAB rocks!

Prompt the user for an integer num and print “MATLAB rocks!” num times.

Example run:

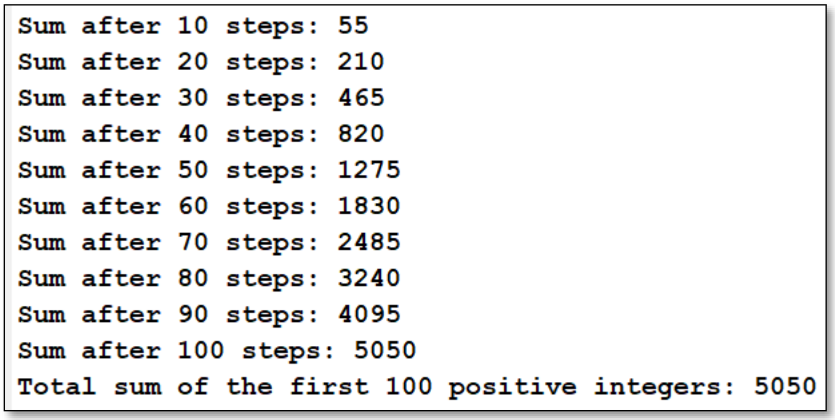


```
Enter an integer: 3
MATLAB rocks!
MATLAB rocks!
MATLAB rocks!
```

## Assignment 3: 100 Positive Integers

Write a MATLAB script to calculate the sum of the first 100 positive integers using a for loop. The script should also display the intermediate sums at every 10th step.

Example run:



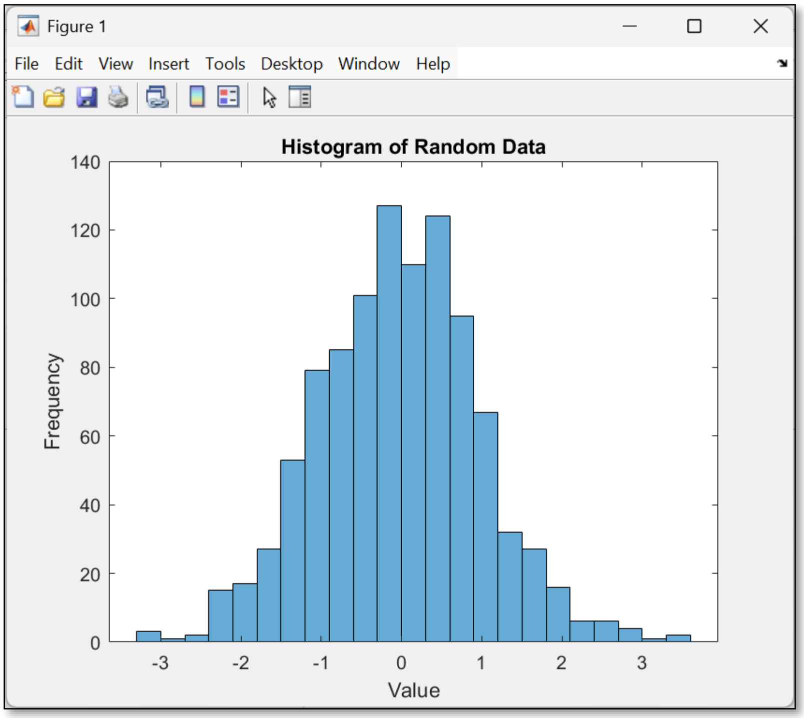
```
Sum after 10 steps: 55
Sum after 20 steps: 210
Sum after 30 steps: 465
Sum after 40 steps: 820
Sum after 50 steps: 1275
Sum after 60 steps: 1830
Sum after 70 steps: 2485
Sum after 80 steps: 3240
Sum after 90 steps: 4095
Sum after 100 steps: 5050
Total sum of the first 100 positive integers: 5050
```

## Assignment 4: Histogram

Generate and plot a histogram of 1000 random numbers drawn from a normal distribution.

1. Generate 1000 random numbers using the randn function.
2. Plot the histogram using the histogram function.
3. Label the axes and add a title to the plot.

Example run:



## Assignment Submission

1. Submit properly named and commented script file.
2. Attach a screenshot of the Command Window showing the successful execution of each script.
3. Attach all to the assignment in Blackboard.