ENGR 0012 – Engineering Problem Solving

Goals for this week:

- Perform matrix operations in MATLAB
    - Load a file
    - Create a script
- Introduce looping/branching notation

Please submit your HW!

Please submit through both new and old submission systems

Slide prepared by Irene Mena
Please do not copy or distribute

Matrix addition (or subtraction) consists of adding (or subtracting) the elements in the same location from each matrix

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} = \begin{bmatrix} (A_{11}+B_{11}) & (A_{12}+B_{12}) & (A_{13}+B_{13}) \\ (A_{21}+B_{21}) & (A_{22}+B_{22}) & (A_{23}+B_{23}) \\ (A_{31}+B_{31}) & (A_{32}+B_{32}) & (A_{33}+B_{33}) \end{bmatrix}$$

$$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 8 & 9 & 10 \end{bmatrix} + \begin{bmatrix} -1 & 4 & 8 \\ 1 & 0 & -3 \\ 5 & -2 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 8 & 14 \\ 4 & 5 & 4 \\ 13 & 7 & 12 \end{bmatrix}$$

Matrices being added/subtracted must have same dimensions!

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

In matrix multiplication, element (3,2) of the product is the sum of the products of row 3 of A and column 2 of B

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$= \begin{bmatrix} (a_{11}b_{11}+a_{12}b_{21}+a_{13}b_{31}) & (a_{11}b_{12}+a_{12}b_{22}+a_{13}b_{32}) & (a_{11}b_{13}+a_{12}b_{23}+a_{13}b_{33}) \\ (a_{21}b_{11}+a_{22}b_{21}+a_{23}b_{31}) & (a_{21}b_{12}+a_{22}b_{22}+a_{23}b_{32}) & (a_{21}b_{13}+a_{22}b_{23}+a_{23}b_{33}) \\ (a_{31}b_{11}+a_{32}b_{21}+a_{33}b_{31}) & \boxed{(a_{31}b_{12}+a_{32}b_{22}+a_{33}b_{32})} & (a_{31}b_{13}+a_{32}b_{23}+a_{33}b_{33}) \end{bmatrix}$$

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

For example: What is the product of these two matrices?

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \\ -1 & -2 & -3 \end{bmatrix}$$

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

For example: What is the product of these two matrices?

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \\ -1 & -2 & -3 \end{bmatrix}$$

$$= \begin{bmatrix} [(1\cdot 2)+(2\cdot 1)+(3\cdot -1)] & [(1\cdot 4)+(2\cdot 3)+(3\cdot -2)] & [(1\cdot 6)+(2\cdot 5)+(3\cdot -3)] \\ [(4\cdot 2)+(5\cdot 1)+(6\cdot -1)] & [(4\cdot 4)+(5\cdot 3)+(6\cdot -2)] & [(4\cdot 6)+(5\cdot 5)+(6\cdot -3)] \\ [(7\cdot 2)+(8\cdot 1)+(9\cdot -1)] & [(7\cdot 4)+(8\cdot 3)+(9\cdot -2)] & [(7\cdot 6)+(8\cdot 5)+(9\cdot -3)] \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 & 7 \\ 7 & 19 & 31 \\ 13 & 34 & 55 \end{bmatrix}$$

From slides by N. Vidic

Slide prepared by Irene Mena

Some additional reminders about matrix multiplication:

- **AB ≠ BA** (Matrix multiplication is NOT commutative)
- To multiply matrices, same dimensions are not necessary
- Left matrix must have same number of columns as right matrix has rows
- Product matrix dimension will be number of rows of left matrix by number of columns of right matrix

$$\mathbf{A}_{\mathbf{4x3}} \mathbf{\ x\ B}_{\mathbf{3x2}} \mathbf{= C}_{\mathbf{4x2}}$$

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

Some additional reminders about matrix division:

- Division is **NOT** possible!
- To divide matrices, you multiply by the inverse
- The inverse of A is such that $A*A^{-1} = I$ → use inv

Slide prepared by Irene Mena
Please do not copy or distribute

Try this!

- Let A = [5 6 3], B= [7 5 2] and C=[1;2;3]

Use MATLAB to determine the result of:

A*C

A*B

A-B

6*A

C*B

Do you understand why you get those results?

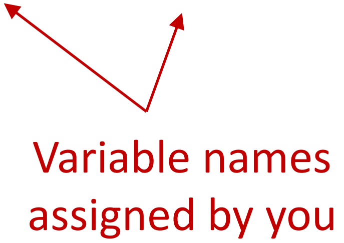
Slide prepared by Irene Mena
Please do not copy or distribute

In addition to regular matrix multiplication, MATLAB also allows you to multiply element by element

A .* B

Some useful commands:

- who
- whos
- sum(MyArray)
- prod(MyArray)
- sort(MyArray)
- min(MyArray) → same with max
- [MyValue MyLocation]=min(MyArray) → same with max



Variable names assigned by you

Some useful commands:

- round(MyArray)
- ceil(MyArray)
- fix(MyArray)
- floor(MyArray)

Try this!

The table shows the hourly wages, hours worked and output (number of widgets produced) in one week for five widgets makers.

	W1	W2	W3	W4	W5
Hourly Wage ($)	5	5.50	6.50	6	6.25
Hours Worked	40	43	37	50	45
Output (widgets)	1000	1100	1000	1200	1100

Use MATLAB to answer these questions:

- How much did each worker earn in a week?
- What is the total salary amount paid out?
- How many widgets were made?
- How many widgets does each worker produce in one hour?
- Which worker is the most efficient?

Hint: Create an array for each variable, then use matrix operations!

To create a matrix with string data, make sure they all have the same number of characters

- See more in your textbook

In MATLAB, you can create cells (similar to a spreadsheet)

- A cell array "is a collection of containers called cells in which you can store different types of data"[1]

Element 1,1	Element 1,2	Element 1,3
Element 2,1	Element 2,2	Element 2,3

In matrix form: Each number or letter is an element

Cell 1,1	Cell 1,2	Cell 1,3
Cell 2,1	Cell 2,2	Cell 2,3

In cell form: Matrices/numbers or words per cell

[1] From course textbook

Your turn!

1. Given the data and results from the problem we worked on, create the variable WidgetsCells. The following information should be stored in each cell:

Array with hourly wages	Array with amount each worker earned in a week
Array with hours worked	Total salary paid that week
Array with number of widgets (output)	Total number of widgets produced that week
Worker names (5 workers – names of your choice)	Array with widgets per hour

2. Extract data from the cells to display the name of worker 3 and how many widgets worker 3 produced, as follows:

"The name of worker 3 is *name* and this worker produced *number* widgets"

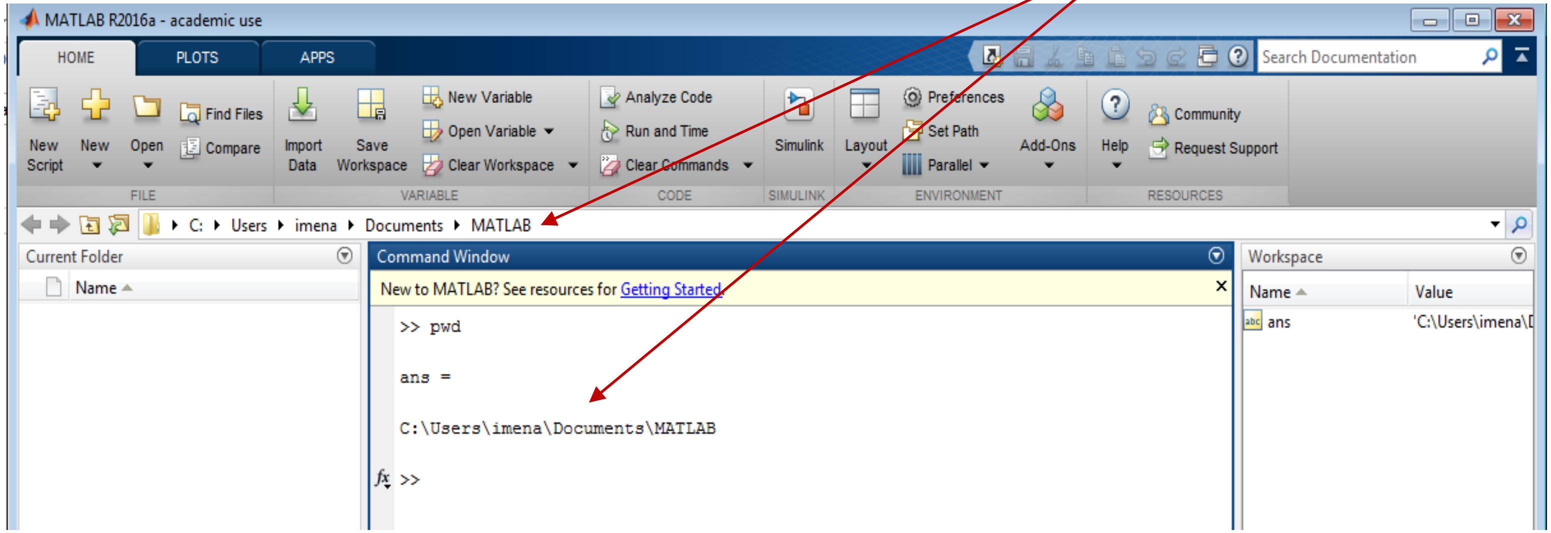
Slide prepared by Irene Mena
Please do not copy or distribute

MATLAB also lets you create a third type of data: structures

- Similar to cell arrays – they provide a convenient way to group related data of different types
- They access elements by using field names instead of index number
- See more in your textbook!

MATLAB allows you to import a data file, but make sure MATLAB can access it!

See the path at the top, or use pwd to get it



New to MATLAB? See resources for Getting Started.

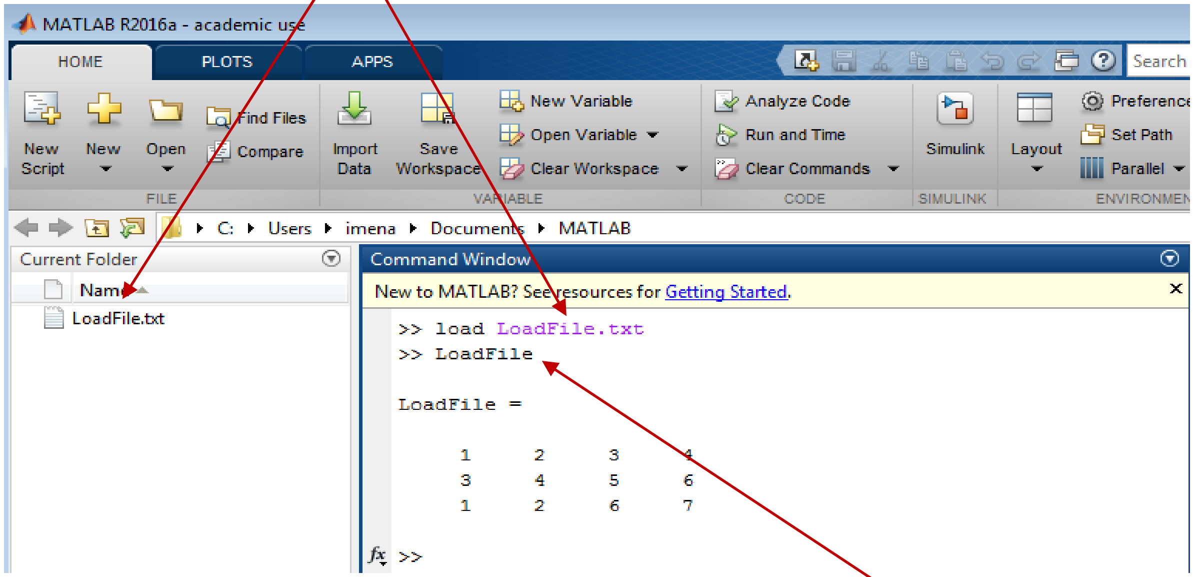
```
>> pwd

ans =

C:\Users\imena\Documents\MATLAB

>>
```

Slide prepared by Irene Mena
Please do not copy or distribute

Example

To load a file, use the load command

load LoadFile.txt



New to MATLAB? See resources for Getting Started.

```
>> load LoadFile.txt
>> LoadFile

LoadFile =

     1     2     3     4
     3     4     5     6
     1     2     6     7

>>
```

To display, type the file name with NO extensions (NO .txt)

Slide prepared by Irene Mena
Please do not copy or distribute

To extract data, specify the column(s) and row(s), or use the colon

- MyData(2,4)

- MyData(:,4)

- MyData(2,:)

Slide prepared by Irene Mena
Please do not copy or distribute

Try this!

Create the following file and save it as LoadFile.txt:

```
5       5.5     6.5     6       6.25
40      43      37      50      45
1000    1100    1000    1200    1100
```

Each column represents one worker (W1-W5). The first row represents the hourly wage, the second row represents the hours worked, and the third row represents the output (number of widgets) per week.

- Load the file into Matlab
- Display the data
- Create and display an array called HourlyWage, which should have the hourly wage
- Create and display an array called HoursWorked, which should have the hours worked
- Create and display an array called Widgets, which should have the number of widgets produced

Use MATLAB to answer these questions:

- If each widget is sold for $5, how much revenue did each worker bring in? → Variable called RevenuePerWorker
- What is the total revenue for that week? → Variable called TotalRevenue

In MATLAB, you can create scripts or programs (m-files)

- These include multiple commands

- This is efficient – you can run multiple commands easily

Programs generally follow these steps:

1. Take the input (for example: variable values, data, …)

2. Process the input

3. Generate output (for example: variable values, plots, …)

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

Programs will generally have these components:

- Variables
- Arithmetic operations & functions
- Input
- Looping/branching
- Output

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

Some good programming practices include:

- Include comments!
- Define problem and desired output
- Break problem down into primary components
- Write code in components and debug as you go, never try to write an entire program (especially a large one) and then run it!

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

Example

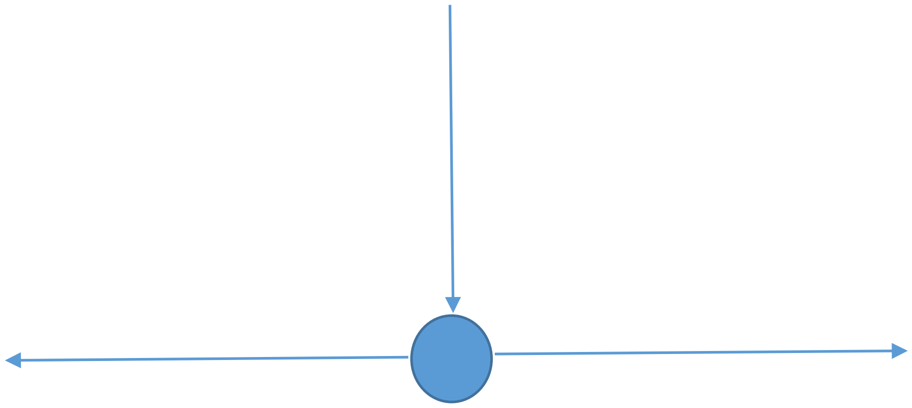
Try this!

a) Create this script

```
% your name(s)
% date
% in-class example to learn MATLAB

%Assign values to variables
a=10;
b=20;
c=30;
d=12;
e=13;
f=14;

%Add the variables
sum1=a+b+c
sum2=d+e+f
```

b) Create your own simple script!

Sometimes we need scripts that will perform certain functions depending on certain conditions – this is branching



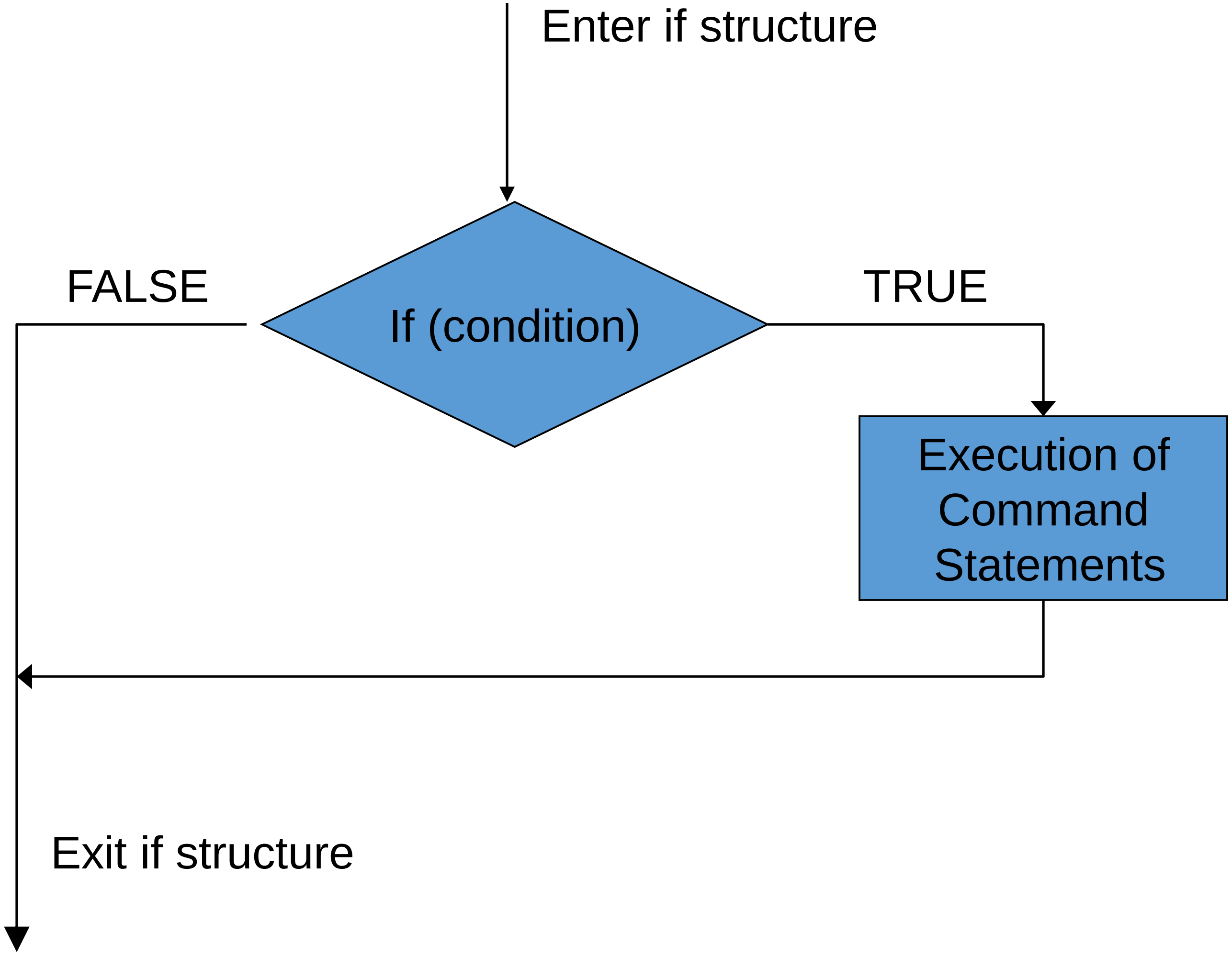
Left or right? It depends…

At the specified point, you evaluate the condition and decide if true (1) or false (0)

Slide prepared by Irene Mena
Please do not copy or distribute

The *if statement* tests one condition

```
if (condition1)
    command1
    command2
    …
end
```

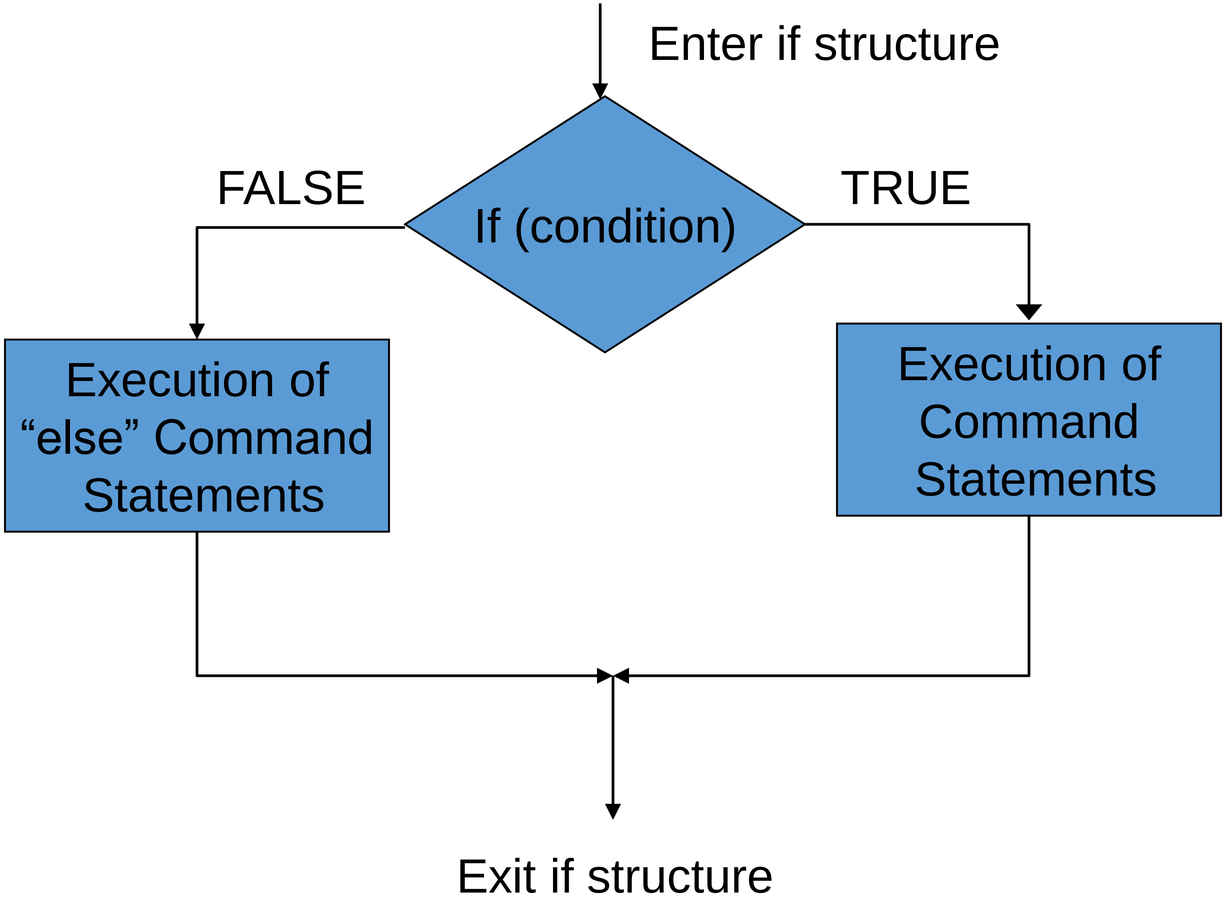


From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

With the *if-else statement*, one condition is tested, but there are two different possible outcomes

```
if (condition1)
    command1
    command2
    …
else
    command1
    command2
      …
end
```



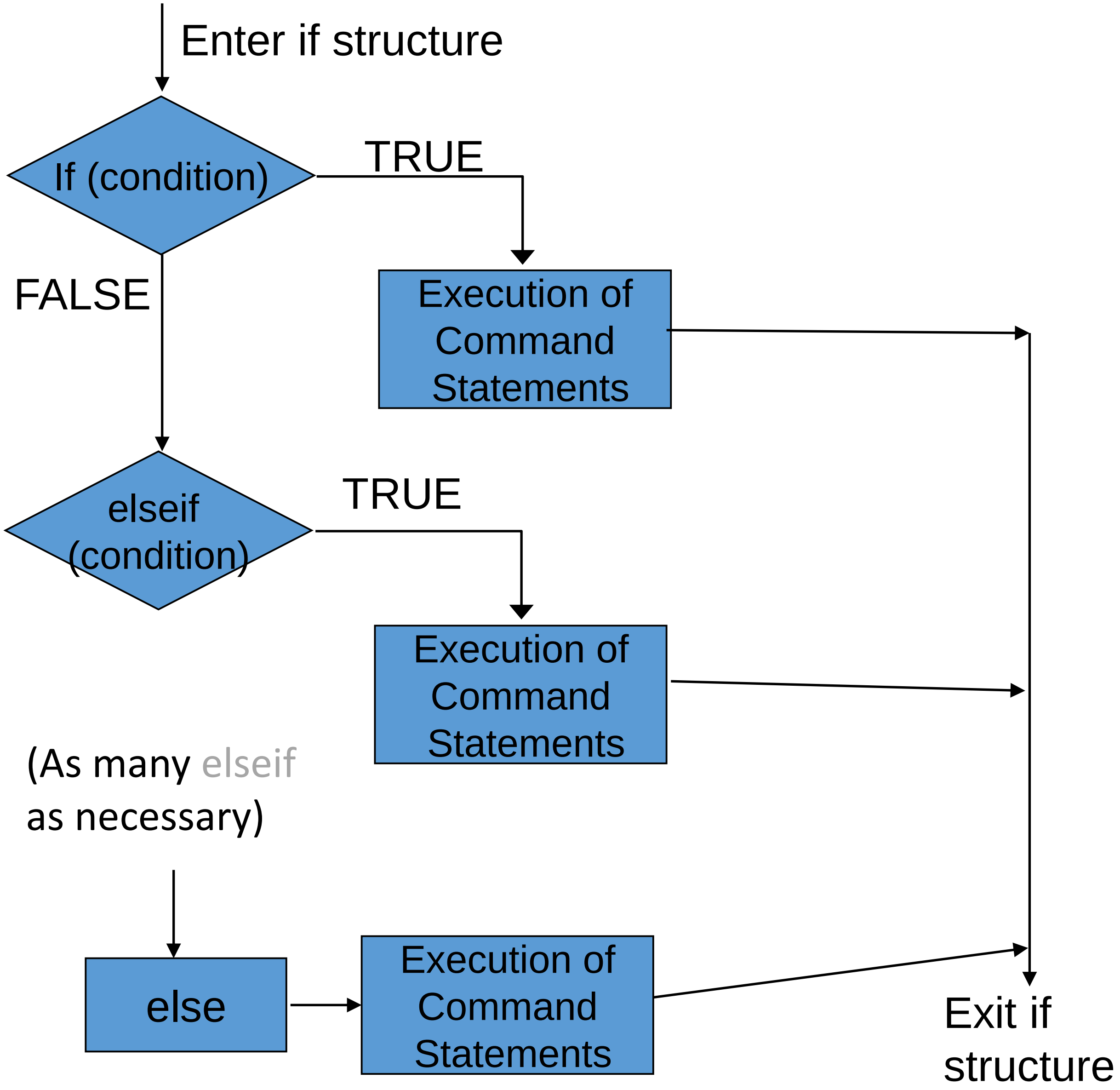
Enter if structure
FALSE
If (condition)
TRUE
Execution of “else” Command Statements
Execution of Command Statements
Exit if structure

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

With the *if-elseif statement*, multiple conditions are tested

```
if (condition1)
    commands
elseif (condition2)
    commands
elseif (condition 3)
    commands
elseif (condition 4)
    commands
else
    commands
end
```



Enter if structure
If (condition)
TRUE
FALSE
Execution of Command Statements
elseif (condition)
TRUE
Execution of Command Statements
(As many elseif as necessary)
else
Execution of Command Statements
Exit if structure

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

When using this branching, (a) start with if and end with end, (b) include a conditional expression

- A conditional expression will include operators

<   >   ==   ~=

- See more operators in your textbook

A script can ask for user input

- For a numerical input:

```
AGE = input(‘How old are you?’)
```

- For a string (text) input:

```
NAME = input(‘What is your name?’, ’s’)
```

Notice additional argument (‘s’)
required for string input

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

Let’s write a script (“program_apples”)
Notice a few things:

- Include comments
- input
- disp
- Indentations

Program should ask user how many apples they would like, and how much they cost.

Then, calculate the total spent.

Then, display whether the total will be less than $10, between $10 and $20, or greater than $20.

From slides by N. Vidic

Slide prepared by Irene Mena
Please do not copy or distribute

Run the script several times with different inputs

```
% Your names
% Group number
% date
%
% in class example

%Requesting info from user
apples=input(‘How many apples do you want?');
cost=input(‘How much does an apple cost?’);

%Calculate total
total=apples*cost;
```

```
%Determine appropriate output
if (total>=20)
   disp(‘This will cost more than $20')
   disp(total)
elseif (total>=10)
   disp(‘This will cost $10 or more but not more than $20’)
   total
else
   disp(‘This will cost less than $10’)
   total
end
```

From slides by N. Vidic

Slide prepared by Irene Mena

Logical operators can also be used, for example:

& | ~

(See table in textbook)

You can figure out if statements without MATLAB

```
a=3;
b=3;
c=5;
d=2;
e=1;
f=4;

Sum1=a+b+c;
Sum2=d+e+f;
Sum3=c+f;

if Sum1>Sum2
        Subtotal1=Sum1+Sum2
        Subtotal2=Sum1+Sum3
else
        Subtotal1=Sum1-Sum2
        Subtotal2=Sum1-Sum3
end
```

Sum1=11

Sum2=7

Sum3=9

Is 11>7? Yes!

Subtotal1=11+7=18

Subtotal2=11+9=20

Your turn: Figure out if statements without MATLAB

```
a=11;
b=3;
c=16;
d=7;
e=11;
f=10;

Sum1=a+b+c;
Sum2=d+e+f;
Sum3=c+f;

if Sum1==Sum2
            Subtotal1=Sum1+Sum2
            Subtotal2=Sum1+Sum3
elseif Sum1>Sum2
            Subtotal1=Sum1-Sum2
            Subtotal2=Sum1-Sum3
else
            disp('We're done!')
end
```

Sum1=30

Sum2=28

Sum3=26

Is 30=28? No!

Is 30>28? Yes!

Subtotal1=2

Subtotal2=4

Slide prepared by Irene Mena
Please do not copy or distribute

Looping allows you to run the program a certain number of times, depending on the type you use:

- While loop: you test a condition

- For loop: you specify number of times

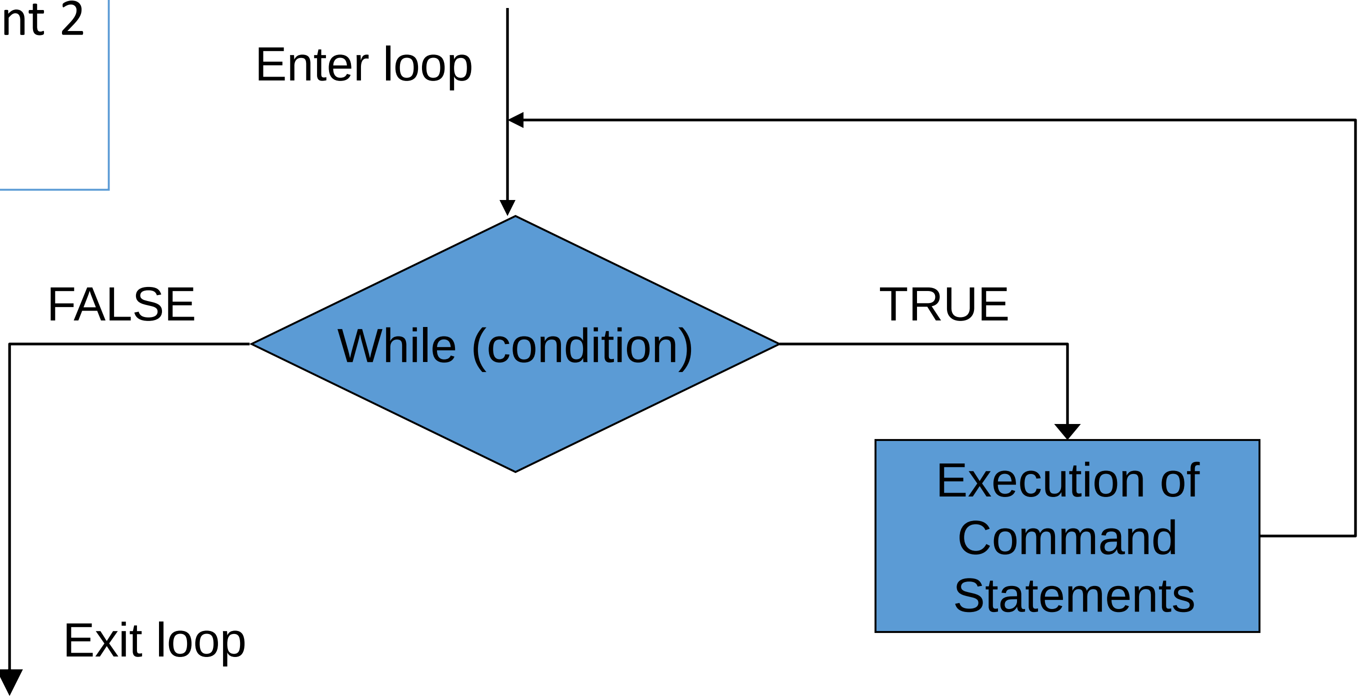
```
for x = [1 2 3 4 5]
    statement 1
    statement 2
    …
end
```

From slides by N. Vidic

Slide prepared by Irene Mena

The while loop will run as long as the condition is true

```
while x < 5
    statement 1
    statement 2
    …
end
```



Enter loop

FALSE

While (condition)

TRUE

Execution of Command Statements

Exit loop

Make sure you define variables and have some way to change the condition, otherwise you could get stuck in an "infinite" loop!

From slides by N. Vidic

Slide prepared by Irene Mena

Let’s try this! Will these all produce the same results?

```
for x = [1 2 3 4 5]
    disp(x)
end
```

```
for x = [1:5]
    disp(x)
end
```

```
x=1;
while x <= 5
    disp(x)
    x=x+1;
end
```

Why is this important here?

From slides by N. Vidic

Slide prepared by Irene Mena

Other similar examples – try them!

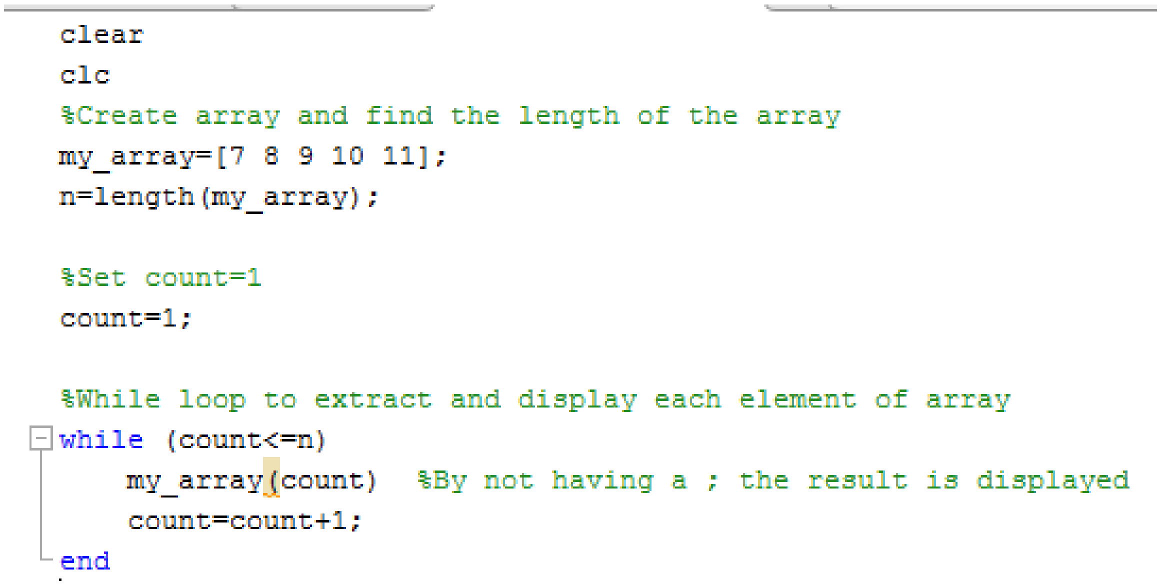
```
for i=1:1:6
    a=i;
    disp(a)
end
```

```
i = 10;
while i<20
    a=i;
    disp(a)
    i=i+1;
end
```

Note that the value of “i” in the while loop had to be initialized and incremented. In the for loop this was done with the colon notation.

We can use the for and while loops to extract elements from arrays

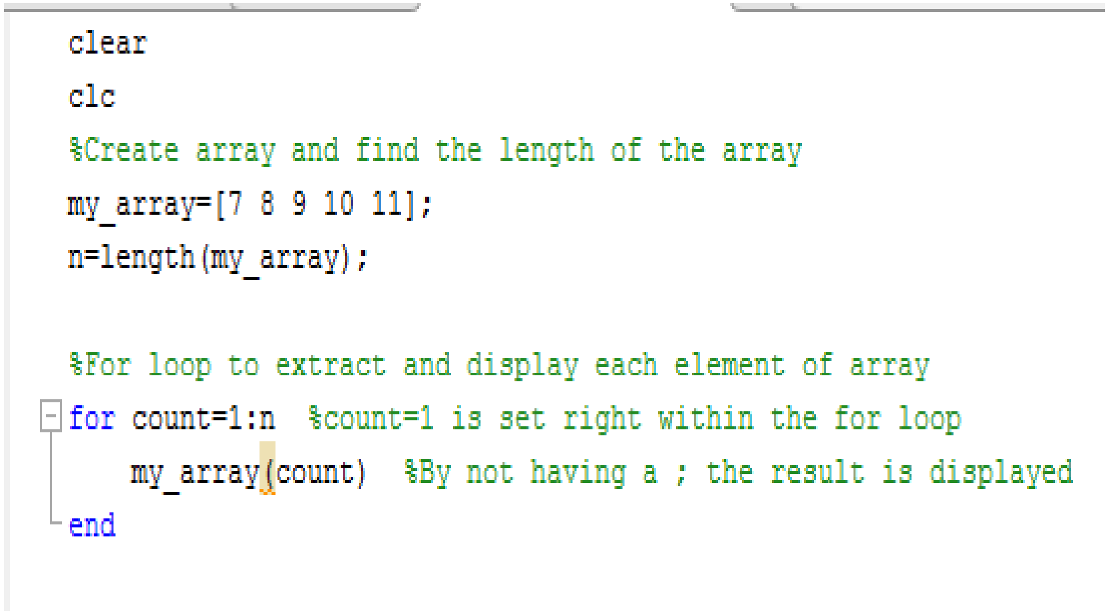
- while loop
  - Initialize count variable, set condition, increment count variable



```
clear
clc
%Create array and find the length of the array
my_array=[7 8 9 10 11];
n=length(my_array);

%Set count=1
count=1;

%While loop to extract and display each element of array
while (count<=n)
    my_array(count)  %By not having a ; the result is displayed
    count=count+1;
end
```

We can use the for and while loops to extract elements from arrays

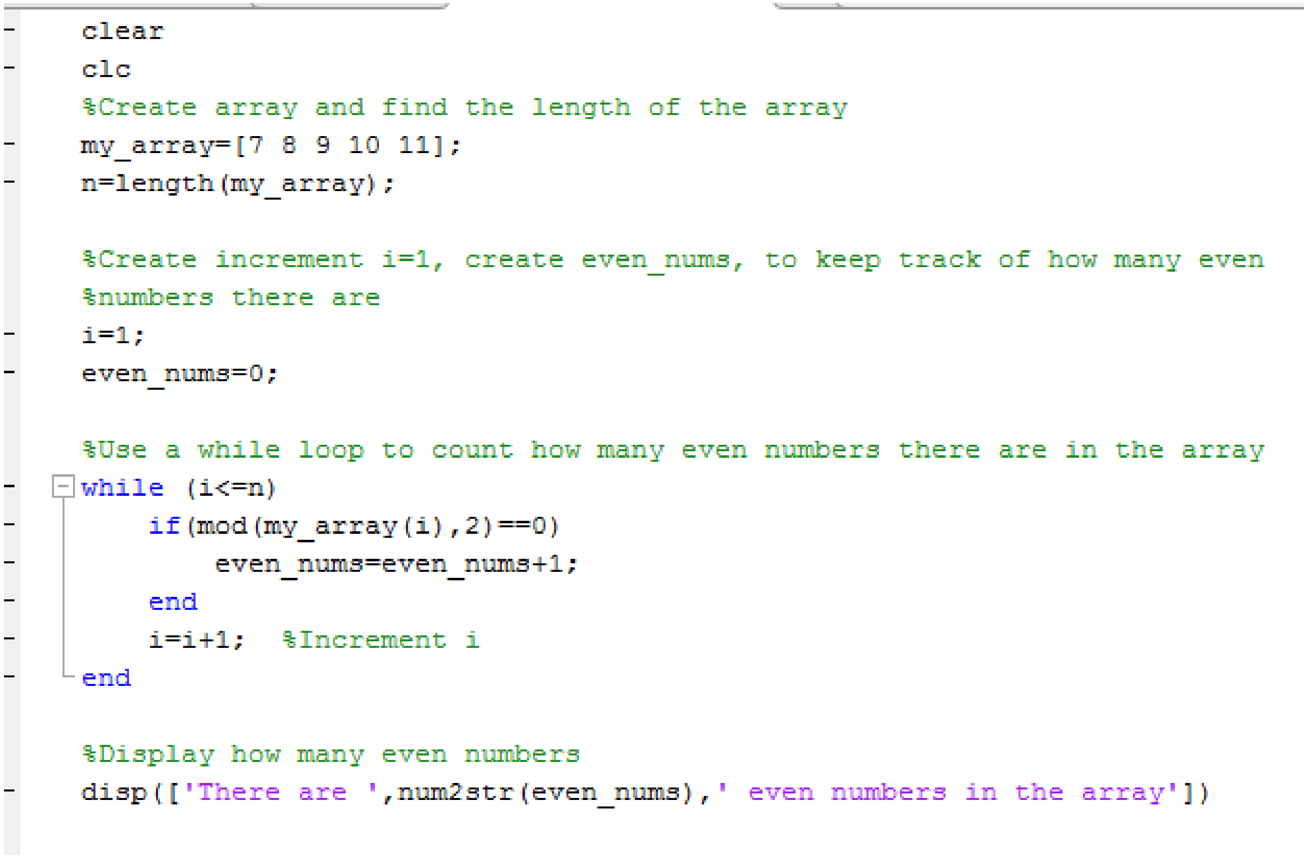
- for loop
  - Set limits (lower to upper, or upper to lower) and increments (using colon notation)



```
clear
clc
%Create array and find the length of the array
my_array=[7 8 9 10 11];
n=length(my_array);

%For loop to extract and display each element of array
for count=1:n  %count=1 is set right within the for loop
    my_array(count)  %By not having a ; the result is displayed
end
```

We can use the for and while loops to count

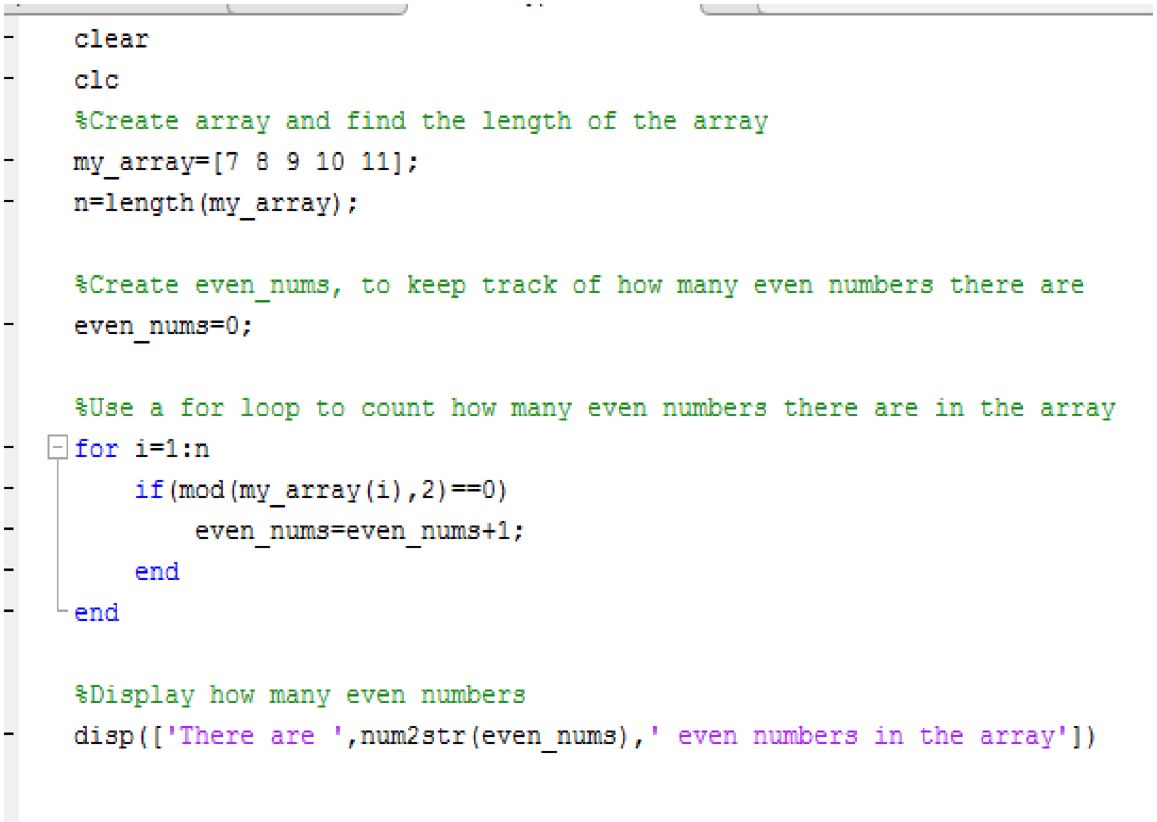
- while loop



```
clear
clc
%Create array and find the length of the array
my_array=[7 8 9 10 11];
n=length(my_array);

%Create increment i=1, create even_nums, to keep track of how many even
%numbers there are
i=1;
even_nums=0;

%Use a while loop to count how many even numbers there are in the array
while (i<=n)
    if(mod(my_array(i),2)==0)
        even_nums=even_nums+1;
    end
    i=i+1;  %Increment i
end

%Display how many even numbers
disp(['There are ',num2str(even_nums),' even numbers in the array'])
```

We can use the for and while loops to count

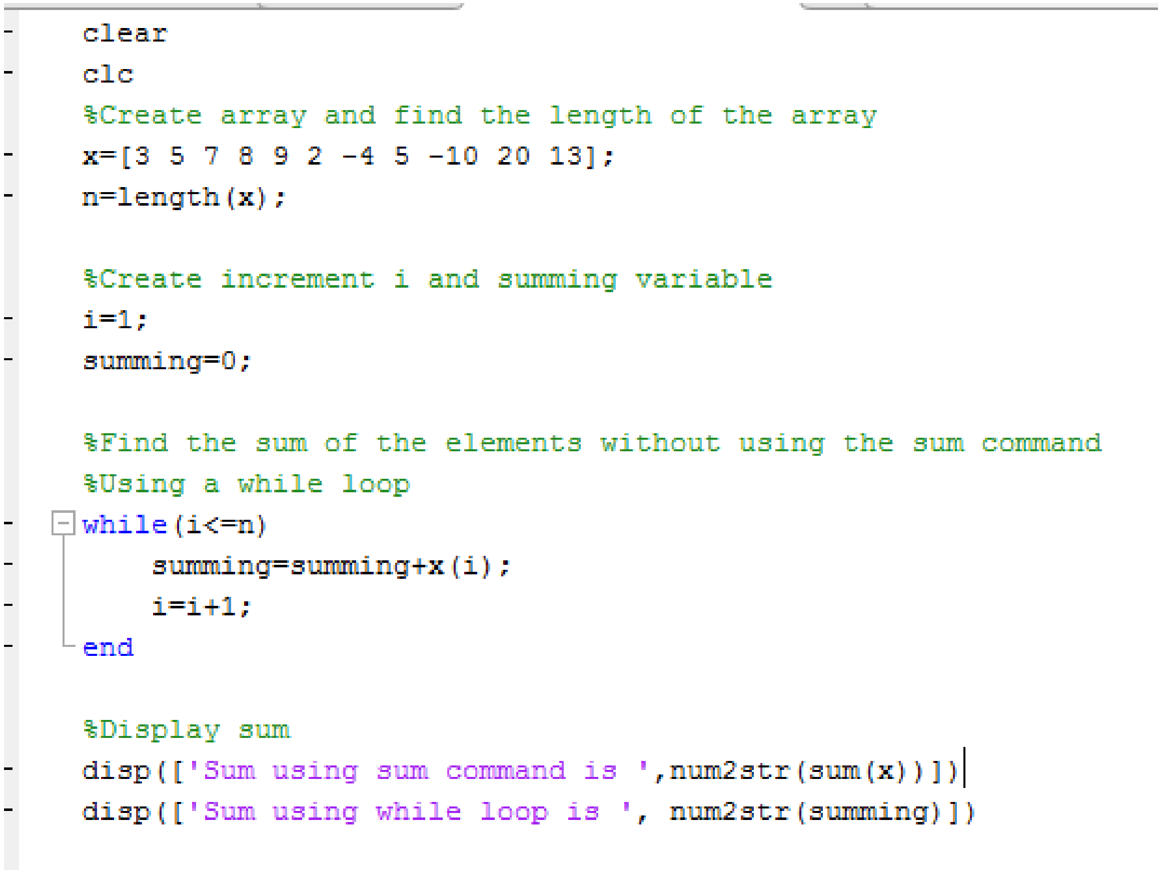
- for loop



```
clear
clc
%Create array and find the length of the array
my_array=[7 8 9 10 11];
n=length(my_array);

%Create even_nums, to keep track of how many even numbers there are
even_nums=0;

%Use a for loop to count how many even numbers there are in the array
for i=1:n
    if(mod(my_array(i),2)==0)
        even_nums=even_nums+1;
    end
end

%Display how many even numbers
disp(['There are ',num2str(even_nums),' even numbers in the array'])
```

You can use loops to perform operations, in place of using existing Matlab commands

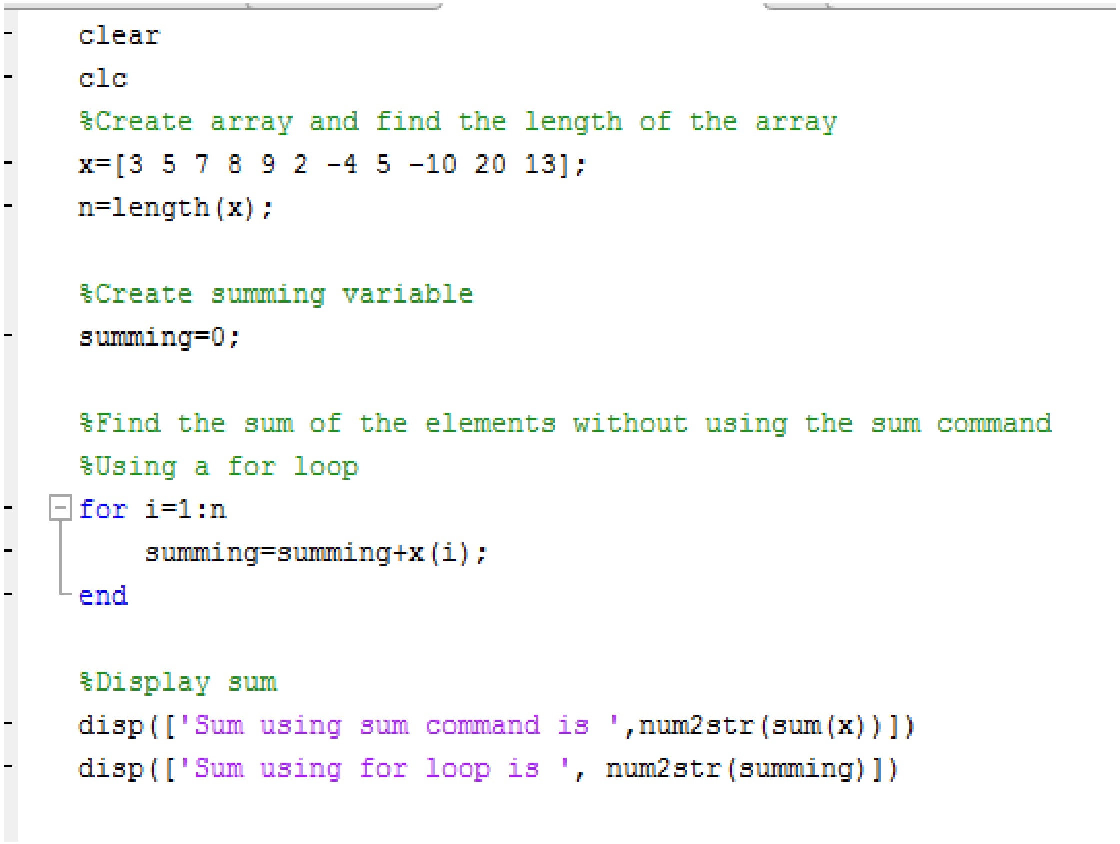
Finding sum without using sum command – using a while loop



```
clear
clc
%Create array and find the length of the array
x=[3 5 7 8 9 2 -4 5 -10 20 13];
n=length(x);

%Create increment i and summing variable
i=1;
summing=0;

%Find the sum of the elements without using the sum command
%Using a while loop
while(i<=n)
    summing=summing+x(i);
    i=i+1;
end

%Display sum
disp(['Sum using sum command is ',num2str(sum(x))])
disp(['Sum using while loop is ', num2str(summing)])
```

You can use loops to perform operations, in place of using existing Matlab commands

Finding sum without using sum command – using a for loop

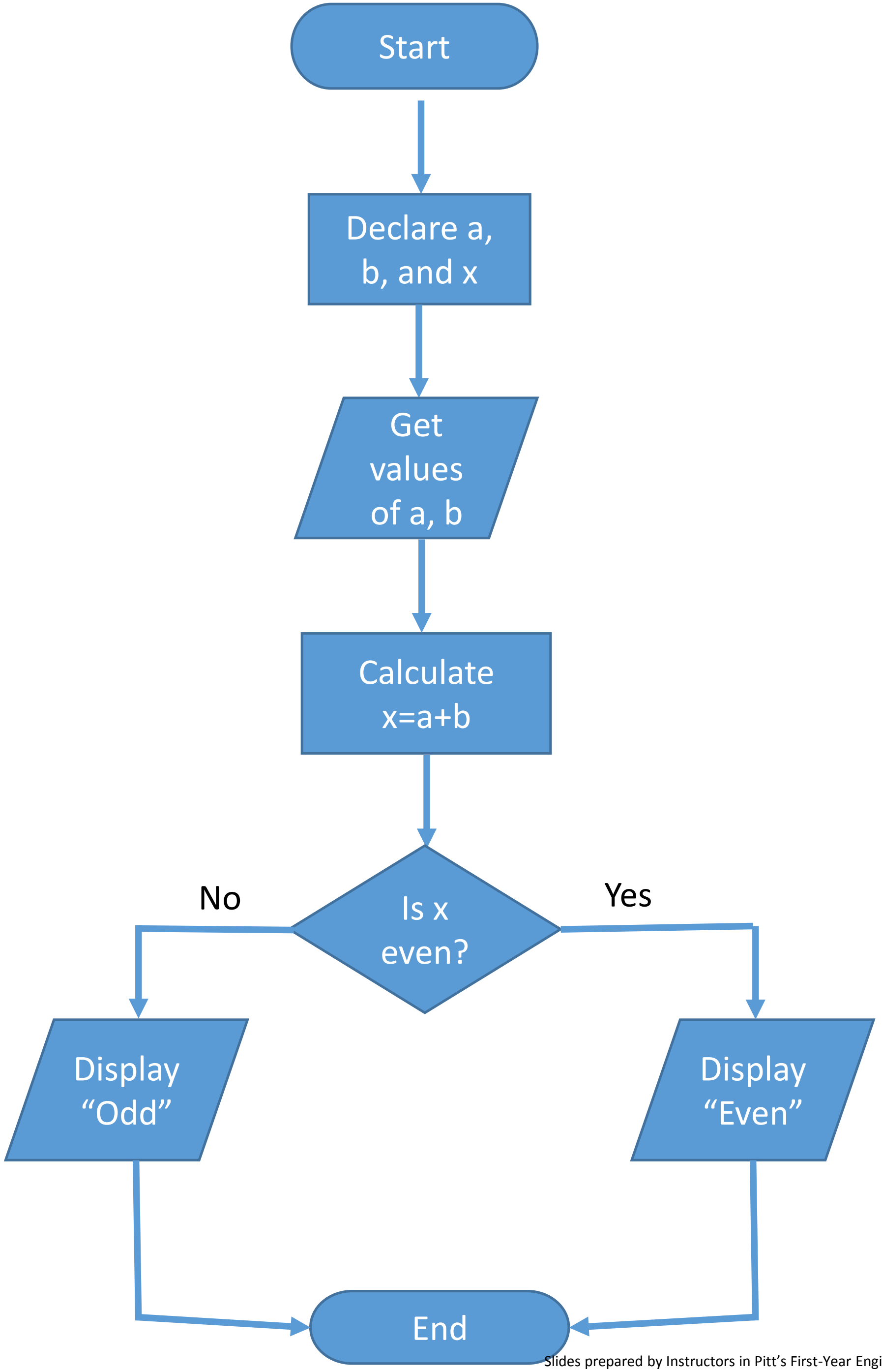


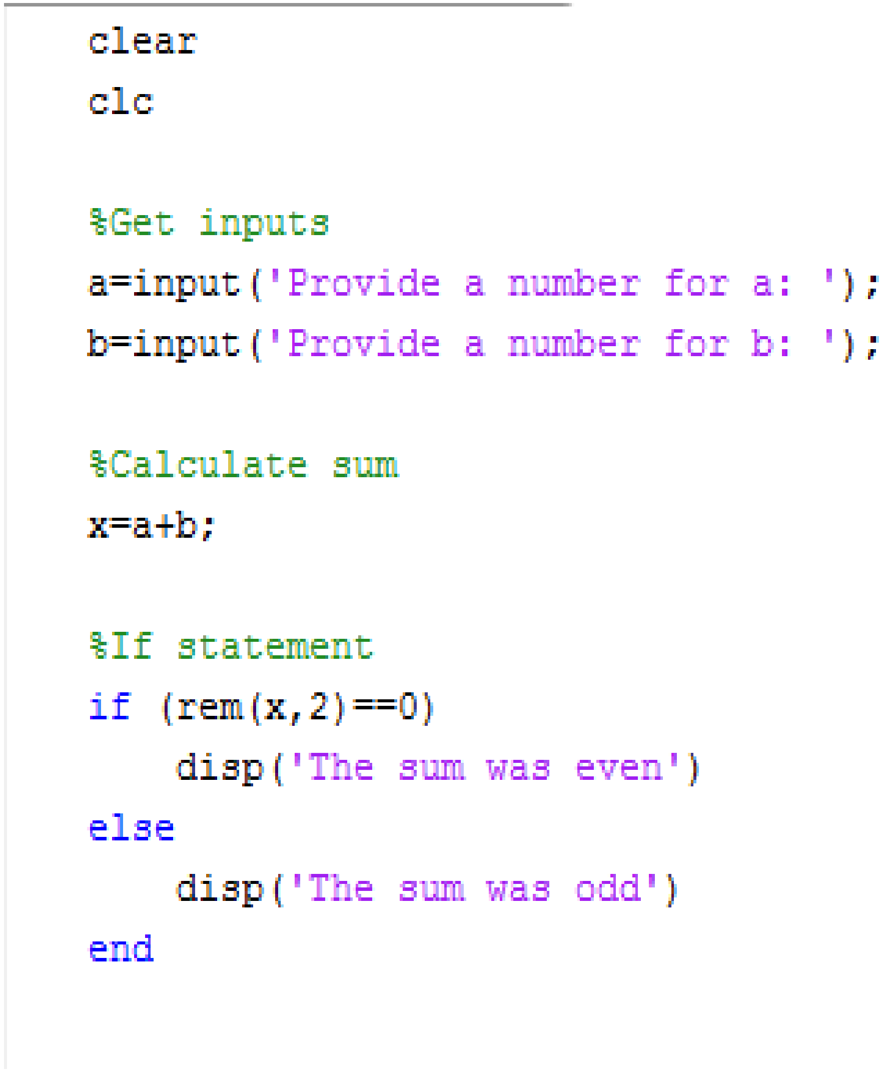
```
clear
clc
%Create array and find the length of the array
x=[3 5 7 8 9 2 -4 5 -10 20 13];
n=length(x);

%Create summing variable
summing=0;

%Find the sum of the elements without using the sum command
%Using a for loop
for i=1:n
    summing=summing+x(i);
end

%Display sum
disp(['Sum using sum command is ',num2str(sum(x))])
disp(['Sum using for loop is ', num2str(summing)])
```

Let’s look at some flowcharts we have created, and write a Matlab code

Example 1

- We need to determine if the sum is even or odd, and display a message accordingly

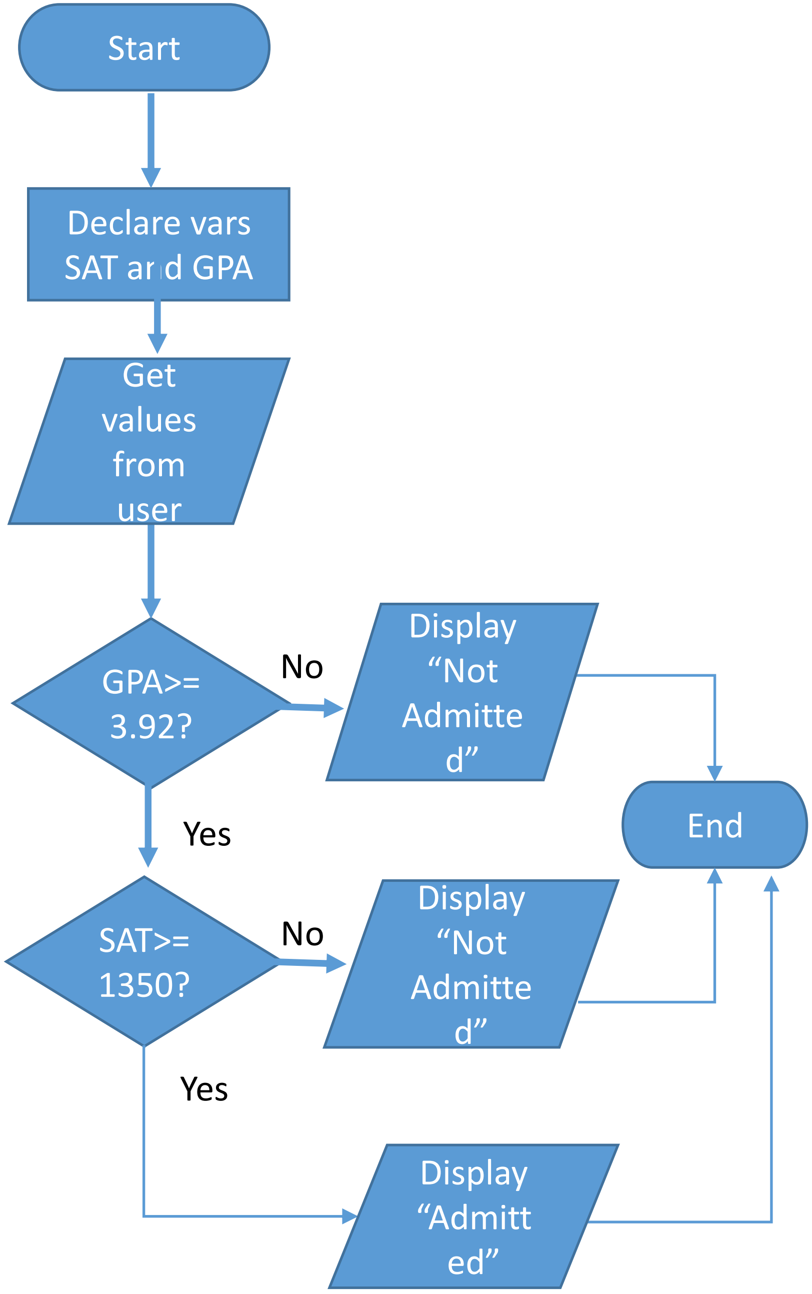


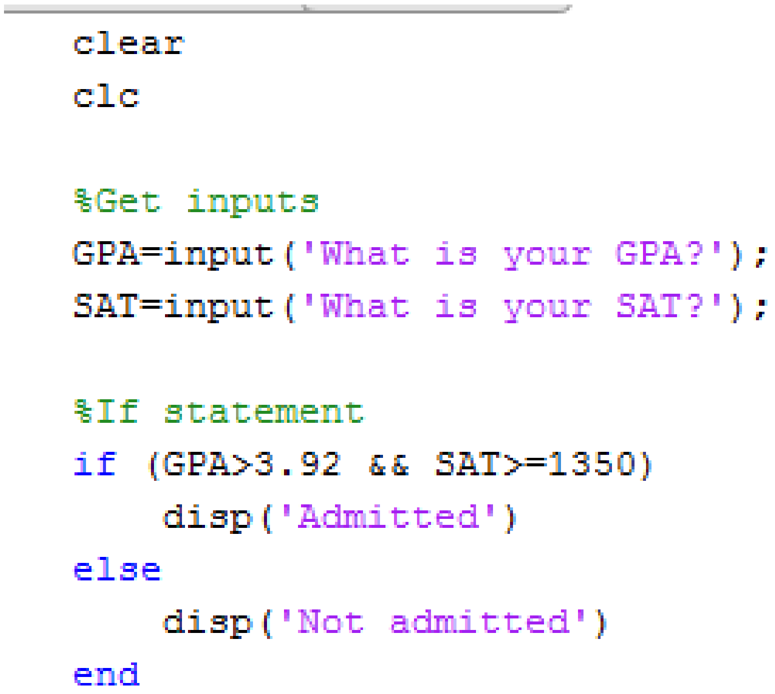


```
clear
clc

%Get inputs
a=input('Provide a number for a: ');
b=input('Provide a number for b: ');

%Calculate sum
x=a+b;

%If statement
if (rem(x,2)==0)
    disp('The sum was even')
else
    disp('The sum was odd')
end
```

Example 2

- Declare two variables (one for GPA, one for SAT)
- Ask the user to provide the values for each
- Determine if the user will be admitted to a certain college. To be admitted, the user needs GPA>3.92, SAT>=1350
- Display whether the user is admitted or not admitted, based on the numbers they provided

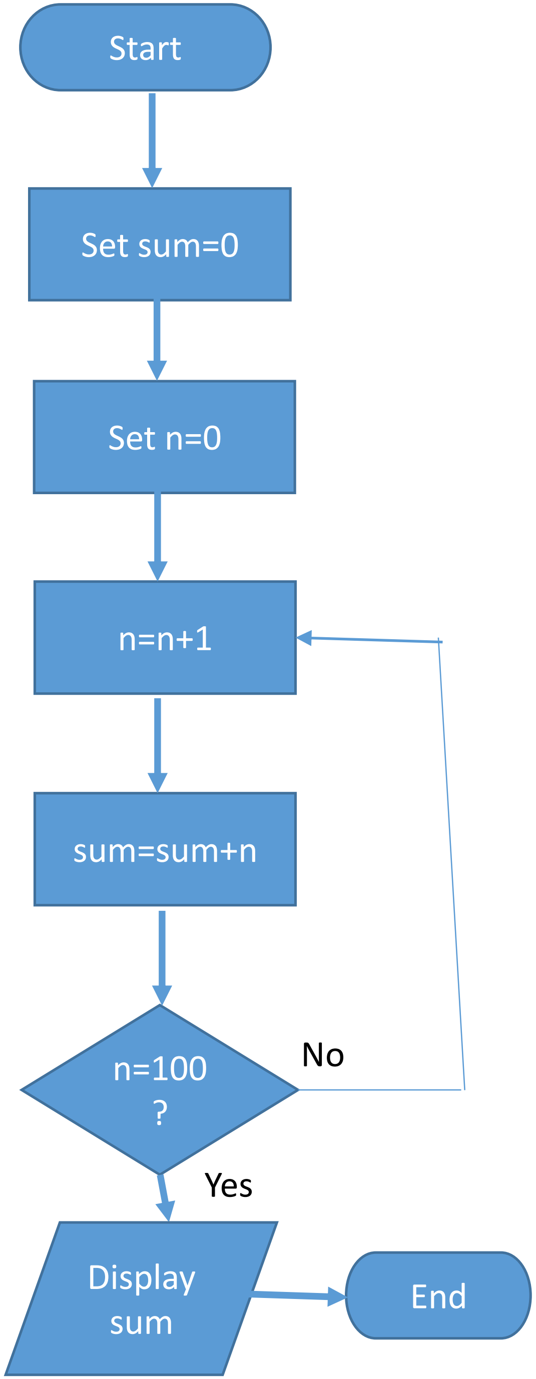




```
clear
clc

%Get inputs
GPA=input('What is your GPA?');
SAT=input('What is your SAT?');

%If statement
if (GPA>3.92 && SAT>=1350)
    disp('Admitted')
else
    disp('Not admitted')
end
```

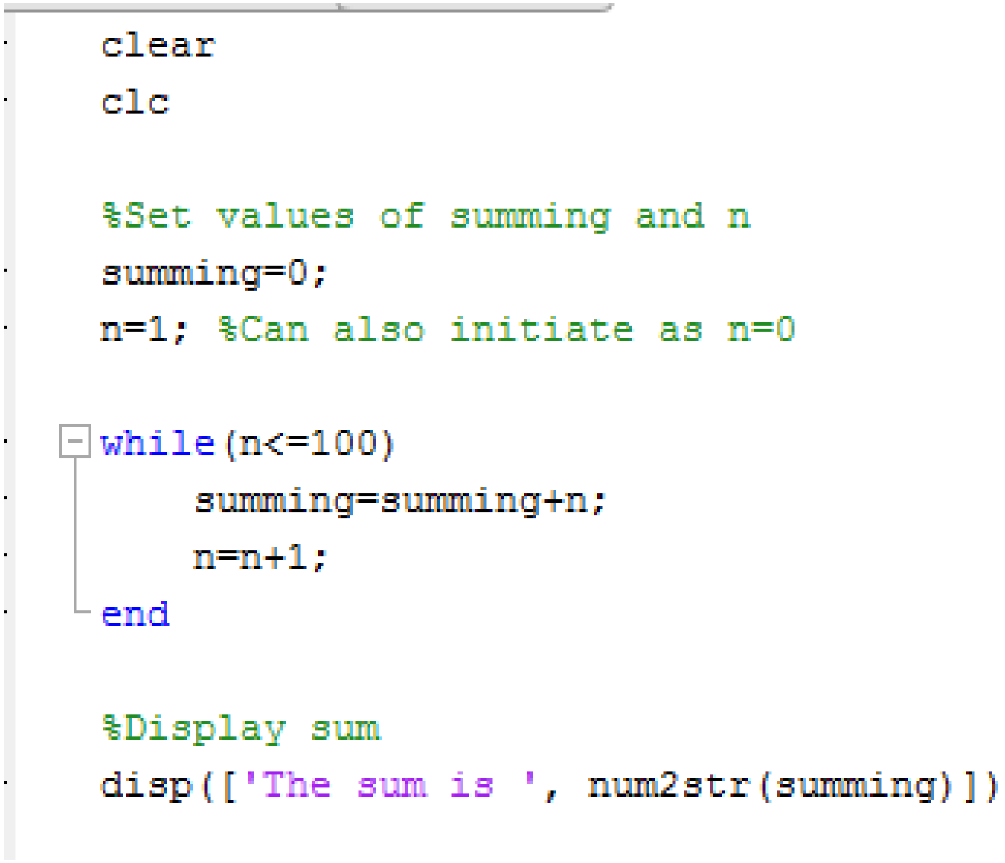
Example 3

- You need to add numbers 1-100, then display final sum



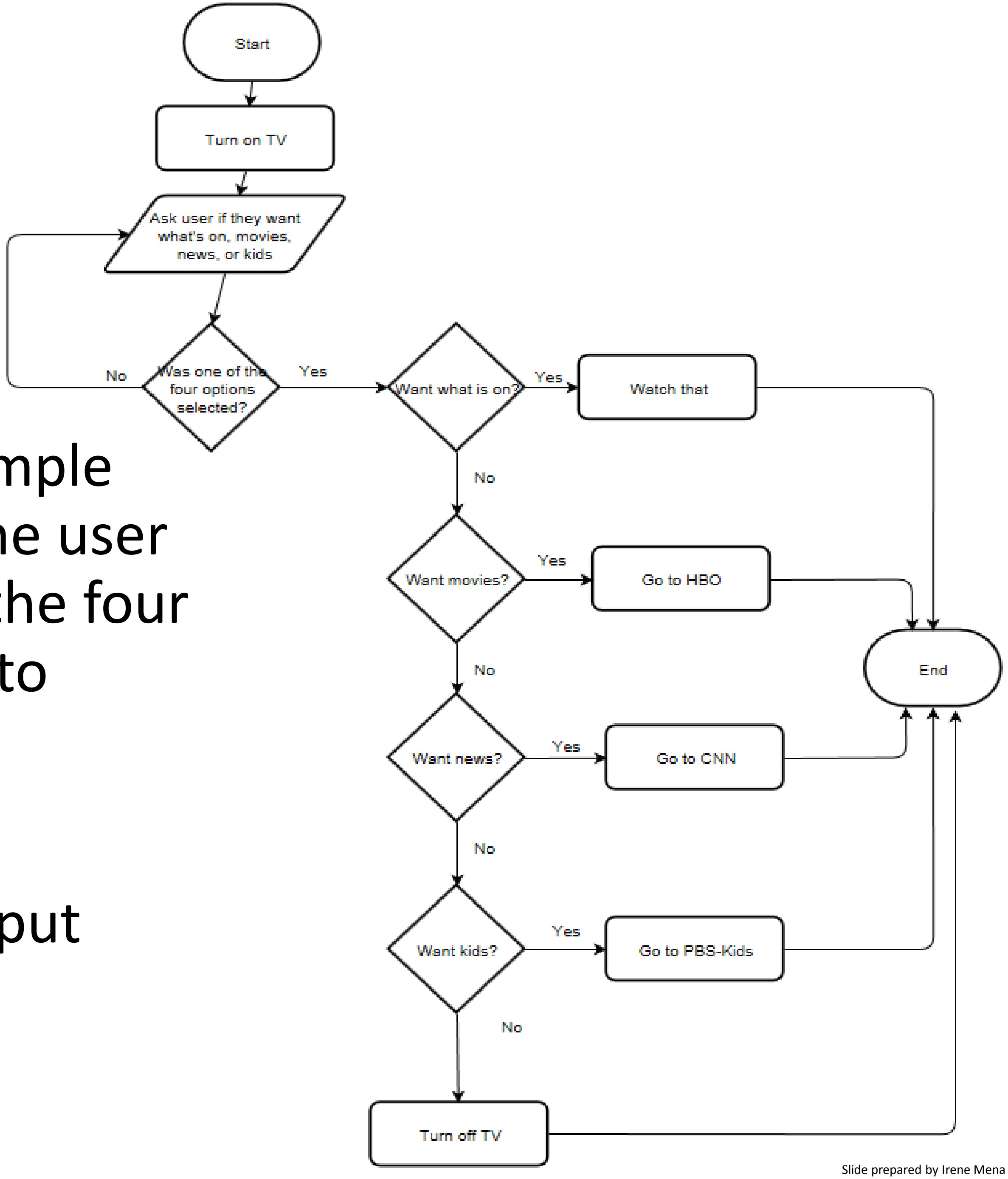
Example from [1]

Slide prepared by Irene Mena
Please do not copy or distribute



```
clear
clc

%Set values of summing and n
summing=0;
n=1; %Can also initiate as n=0

while(n<=100)
    summing=summing+n;
    n=n+1;
end

%Display sum
disp(['The sum is ', num2str(summing)])
```

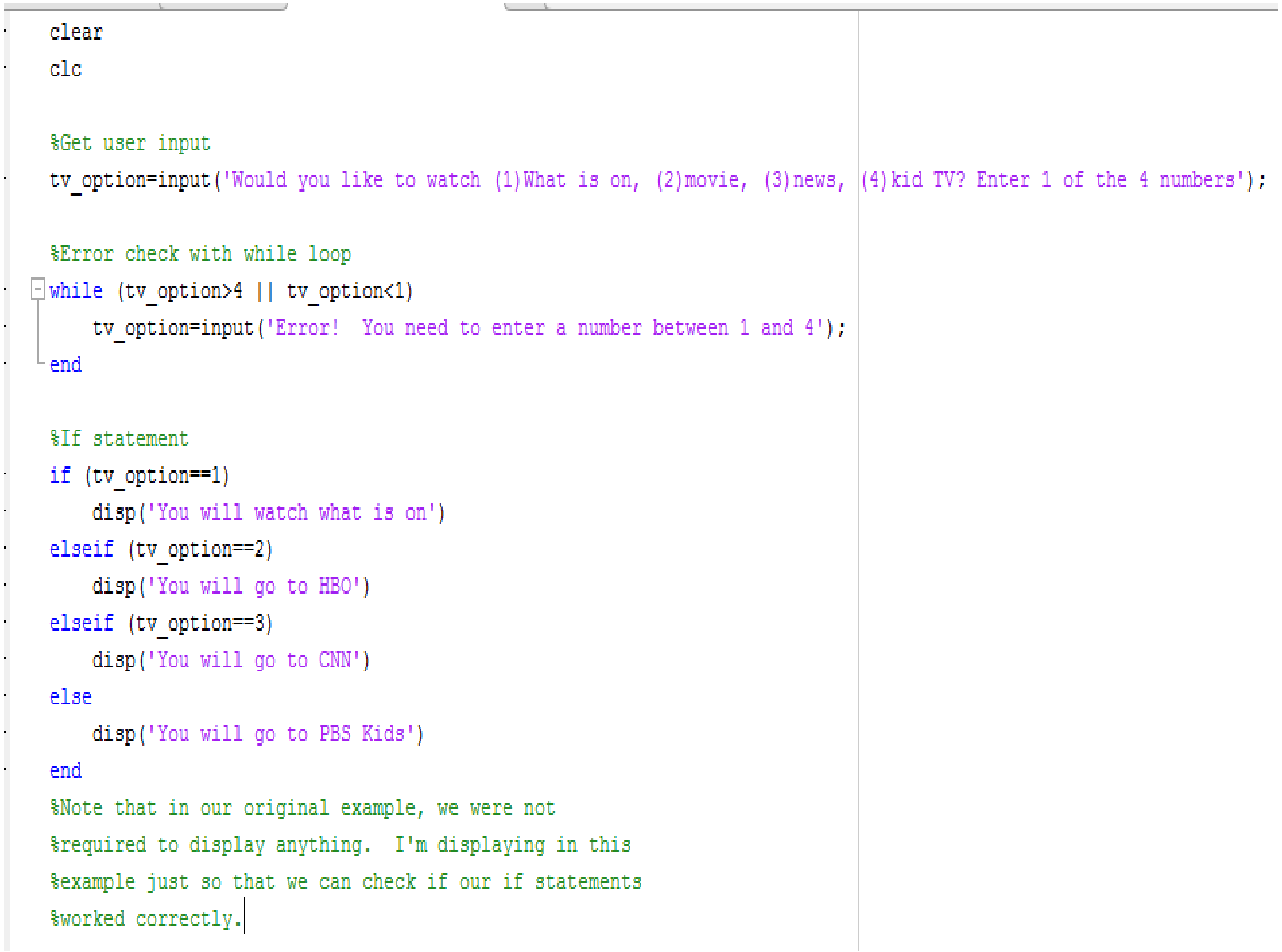
Example 4



- Modify our TV example such that we ask the user to select which of the four options they want to watch

- Error check user input

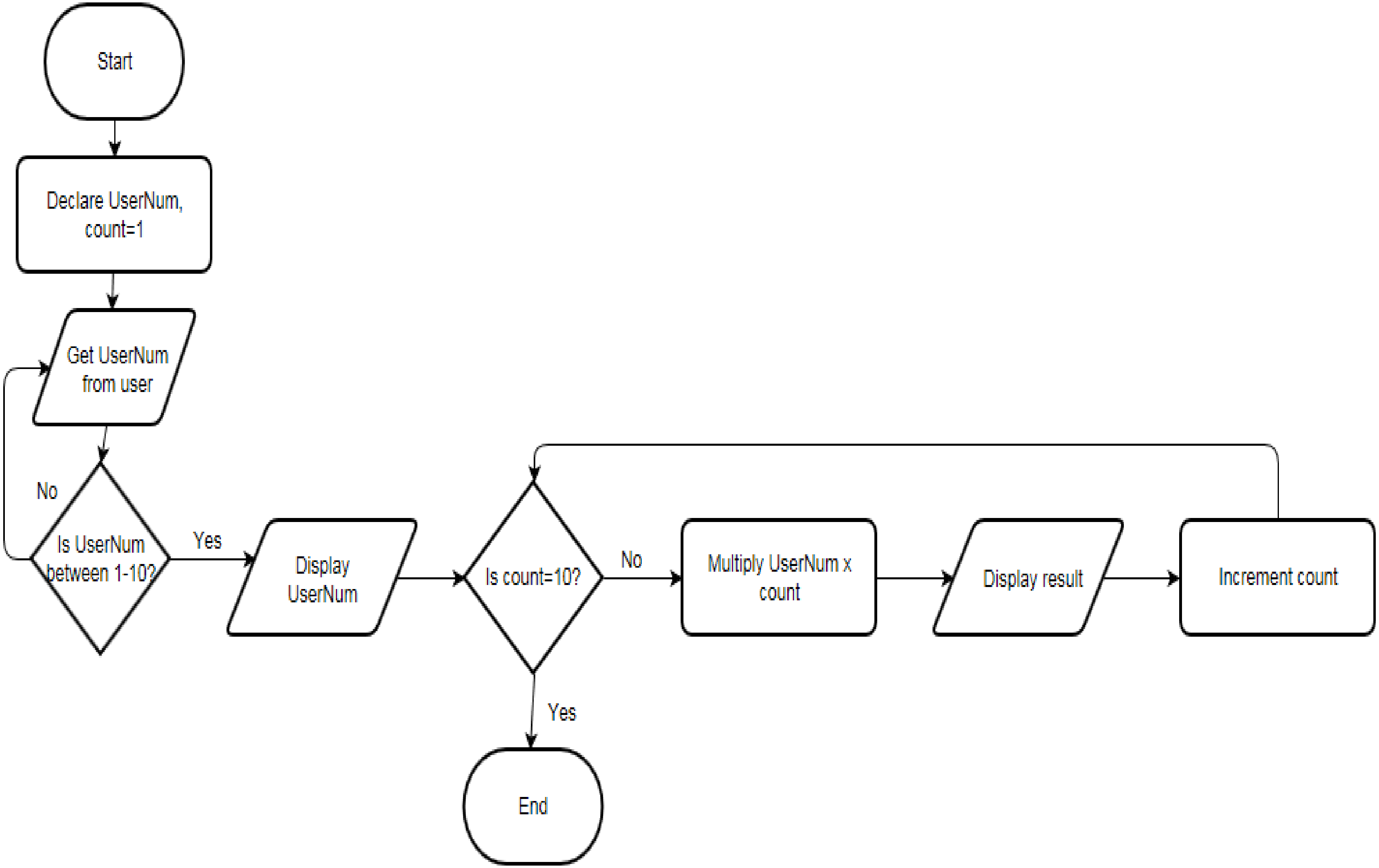
Slide prepared by Irene Mena
Please do not copy or distribute



```
clear
clc

%Get user input
tv_option=input('Would you like to watch (1)What is on, (2)movie, (3)news, (4)kid TV? Enter 1 of the 4 numbers');

%Error check with while loop
while (tv_option>4 || tv_option<1)
    tv_option=input('Error!  You need to enter a number between 1 and 4');
end

%If statement
if (tv_option==1)
    disp('You will watch what is on')
elseif (tv_option==2)
    disp('You will go to HBO')
elseif (tv_option==3)
    disp('You will go to CNN')
else
    disp('You will go to PBS Kids')
end
%Note that in our original example, we were not
%required to display anything.  I'm displaying in this
%example just so that we can check if our if statements
%worked correctly.
```

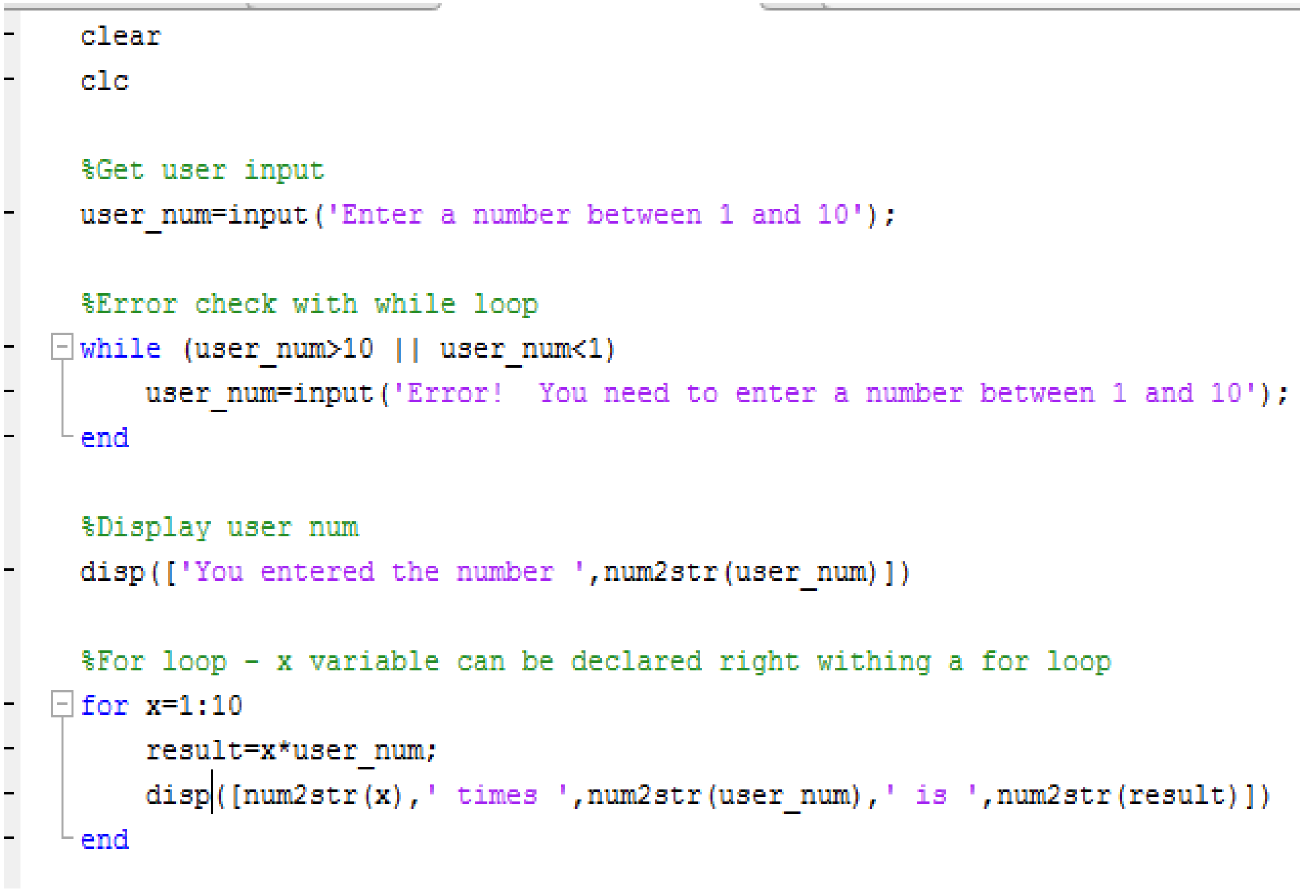
Your turn!

- Create a program that displays the multiplication table for a number provided by the user
    - Save the user-provided number in a previously declared variable
    - Users can only provide numbers between 1 and 10 (make sure you error check)
    - The program displays the number given by the user once, then displays the result of the number times numbers 1-10
    - Use loops to help you create a more efficient program



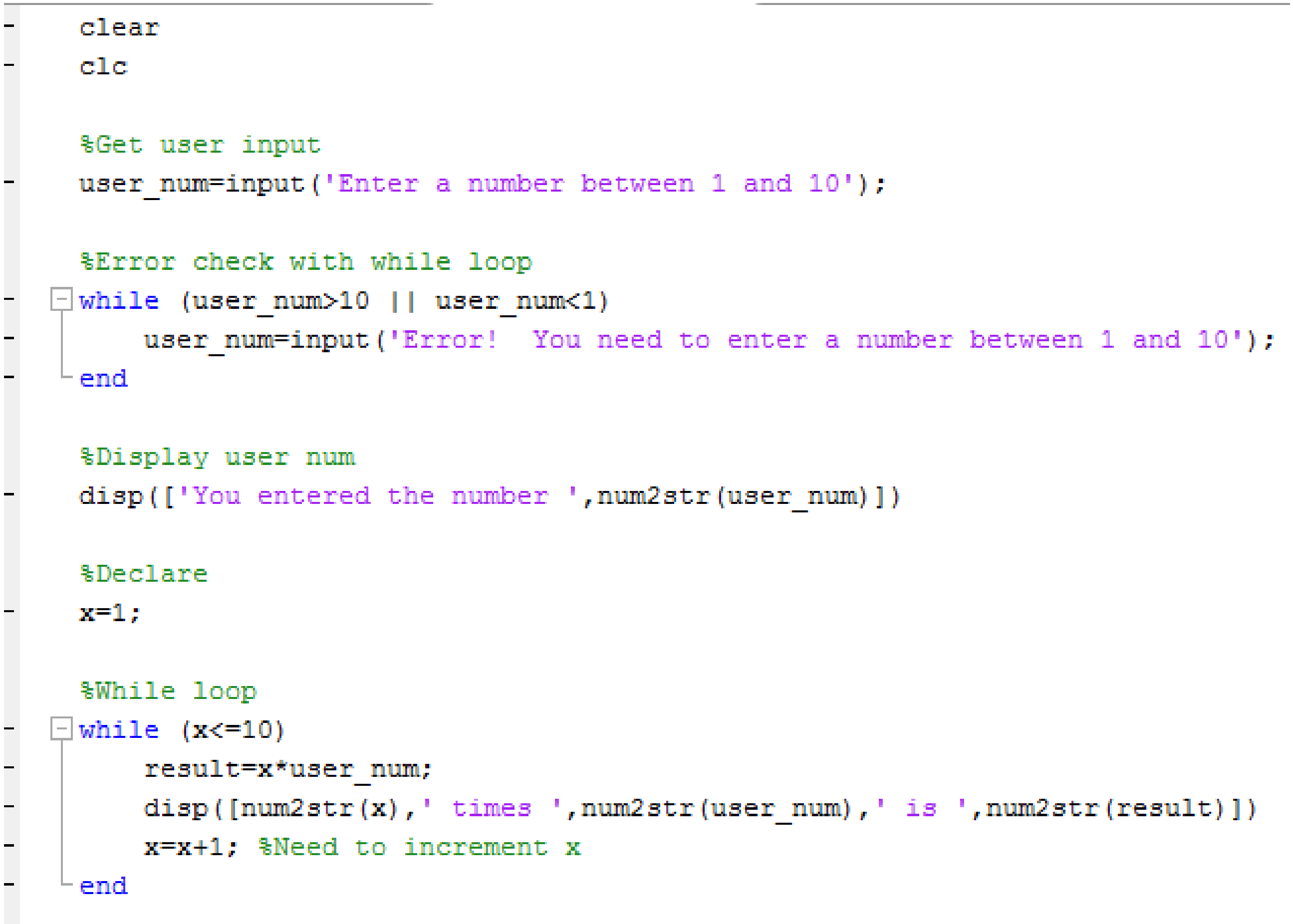
Slide prepared by Irene Mena
Please do not copy or distribute

Using a for loop (using a while loop in next slide)



```
clear
clc

%Get user input
user_num=input('Enter a number between 1 and 10');

%Error check with while loop
while (user_num>10 || user_num<1)
    user_num=input('Error!  You need to enter a number between 1 and 10');
end

%Display user num
disp(['You entered the number ',num2str(user_num)])

%For loop - x variable can be declared right withing a for loop
for x=1:10
    result=x*user_num;
    disp([num2str(x),' times ',num2str(user_num),' is ',num2str(result)])
end
```

Using a while loop



```
clear
clc

%Get user input
user_num=input('Enter a number between 1 and 10');

%Error check with while loop
while (user_num>10 || user_num<1)
    user_num=input('Error!  You need to enter a number between 1 and 10');
end

%Display user num
disp(['You entered the number ',num2str(user_num)])

%Declare
x=1;

%While loop
while (x<=10)
    result=x*user_num;
    disp([num2str(x),' times ',num2str(user_num),' is ',num2str(result)])
    x=x+1; %Need to increment x
end
```