WHAT WE WILL LEARN TODAY

Variables
Operations
Conditions
Loops
There are two ways to look at it:
- data is given a name
- A variable “contains” data
NAMING RULE FOR VARIABLES

- Must begin with a letter
- Only first 63 characters are significant
- Make sure uniqueness
- Give your variables descriptive and easy to remember names
- Create a data dictionary
- Case sensitive
var = 40i;
var2 = var/5;
x=1;
array = [1, 2, 3, 4];
array2 = [1; 2; 3; 4]
array3 = [1, 2, 3; 4, 5, 6]
a = [0 1+7]
b = [a(2) 7 a]
INITIALIZING VARIABLES

With shortcut expressions
- first:incr:last
- \( x = 1:2:10 \)
- \( y = 1:10 \)

With built-in functions
- \( a = \text{zeros}(2) \)
- \( b = \text{zeros}(2, 3) \)
- \( c = \text{ones}(3, 4) \)

With keyboard input
- \( \text{in1} = \text{input('Enter data: '}\) \)
- \( \text{in2} = \text{input('Enter data: '}\ , \text{'s'}\); //store the character string \)

\( \text{In1} = 1 \quad \text{ln2} = '1' \)
The fundamental data unit is the array

Arrays are classified as either vectors or matrices

\[
a = \begin{bmatrix}
1 & 2 \\
3 & 4 \\
5 & 6
\end{bmatrix}
\text{This is a } 3 \times 2 \text{ matrix, containing 6 elements.}
\]

\[
b = [1 \hspace{2pt} 2 \hspace{2pt} 3 \hspace{2pt} 4]
\text{This is a } 1 \times 4 \text{ array containing 4 elements, known as a row vector.}
\]

\[
c = \begin{bmatrix}
1 \\
2 \\
3
\end{bmatrix}
\text{This is a } 3 \times 1 \text{ array containing 3 elements, known as a column vector.}
\]

\[
a = [1 \hspace{2pt} 2; \hspace{2pt} 3 \hspace{2pt} 4; \hspace{2pt} 5 \hspace{2pt} 6]
\]

\[
b = [1 \hspace{2pt} 2 \hspace{2pt} 3 \hspace{2pt} 4]
\]

\[
c = [1; \hspace{2pt} 2; \hspace{2pt} 3]
\]
2x3x2 arrays
- \( c(:, :, 1) = [1 \ 2 \ 3; \ 4 \ 5 \ 6] \)
- \( c(:, :, 2) = [7 \ 8 \ 9; \ 10 \ 11 \ 12] \)

Storing arrays in memory
- Column major order

Accessing multidimensional arrays with one dimension
- \( a = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9; \ 10 \ 11 \ 12] \)
- \( a(5), a(1,2) \)
Limiting the range of an array

- \( a = [1 \ 2 \ 3 \ 4; \ 5 \ 6 \ 7 \ 8; \ 9 \ 10 \ 11 \ 12] \)
- \( a(1,:) \rightarrow [1, 2, 3, 4] \)
- \( b = a(1:2, 1:2) \rightarrow [1, 2; 5, 6] \)

**end** keyword

- \( c = a(2:end, 2:end) \rightarrow [6, 7, 8; 10, 11, 12] \)

\( a(1:2, 1:2) = 1 \)
### SPECIAL VALUES

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pi</td>
<td>contains $\pi$ to 15 significant digits</td>
</tr>
<tr>
<td>i, j</td>
<td>contains the value i (imaginary number)</td>
</tr>
<tr>
<td>inf</td>
<td>division by zero</td>
</tr>
<tr>
<td>nan</td>
<td>Not-a-Number, the result of an undefined mathematical operation</td>
</tr>
<tr>
<td>clock</td>
<td>the current date and time in the form of row vector</td>
</tr>
<tr>
<td>date</td>
<td>the current date in a character string format</td>
</tr>
<tr>
<td>eps</td>
<td>epsilon</td>
</tr>
<tr>
<td>ans</td>
<td>used to store the result of an expression that is not assigned to some other variable</td>
</tr>
</tbody>
</table>
DISPLAYING DATA IN THE VARIABLE
DISPLAYING DATA CONTAINED IN VARIABLES

\[ x = 2 * (1 - 2*i)^3; \]
\[ \text{str} = [' \text{disp: x = ' num2str(x)]} \]
\[ \text{disp(str)} \]
\[ \text{fprintf(' fprintf : x = %8.4f\n', x) } \rightarrow \text{ Formatted output} \]

\[ \text{disp: x = -22+4i} \]
\[ \text{fprintf x = -22.0000} \]
DATA FILES

save filename var1 var2 var3 (default type : -mat)
- $x = \begin{bmatrix} 1.23 & 3.14 & 6.28; & -5.1 & 7.00 & 0 \end{bmatrix}$
- save -ascii x.dat x
- x.dat

| 1.2300000e+00 | 3.1400000e+00 | 6.2800000e+00 | -5.1000000e+00 | 7.0000000e+00 | 0.0000000e+00 |

load filename
- load x.dat
OPERATION
OPERATIONS

Arithmetic operation
- $a + b$, $a - b$, $a \cdot b$, $a / b$, $a^b$

Array and matrix operation
- $a \cdot b$, $a / b$, $a \backslash b$
- $a + b$, $a - b$, $a \cdot * b$, $a ./ b$, $a \backslash b$, $a ^ ^ b$

Precedence
- Parentheses (inner most to outward)
- Exponential (left to right)
- Multiplication and division (left to right)
- Addition and subtraction (left to right)
LOGICAL DATA TYPE

true or false

\[ a1 = true; \]
\[ a2 = false; \]

non-zero for true, 0 for false
RELATIONAL OPERATORS

Form
- $a_1 \text{ op } a_2$

Operators and examples

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Equal to</td>
<td>3 &lt; 4</td>
<td>true (1)</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
<td>3 &lt;= 4</td>
<td>true (1)</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>3 == 4</td>
<td>false (0)</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td>3 &gt; 4</td>
<td>false (0)</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>4 &lt;= 4</td>
<td>true (1)</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
<td>‘A’ &lt; ‘B’</td>
<td>true (1)</td>
</tr>
</tbody>
</table>

$a = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}, b = 0 \quad \Rightarrow \quad a > b ? \begin{bmatrix} true & false \\ false & true \end{bmatrix}$
LOGIC OPERATORS

Form
• \( l_1 \ op \ l_2 \)
• \( \ op \ l_1 \)

Operators and truth table

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>Logical AND</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Logical AND with shortcut evaluation</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>xor</td>
<td>Logical Exclusive OR</td>
</tr>
<tr>
<td>~</td>
<td>Logical NOT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inputs</th>
<th>and</th>
<th>or</th>
<th>xor((l_1, l_2))</th>
<th>not</th>
</tr>
</thead>
<tbody>
<tr>
<td>(l_1)</td>
<td>(l_2)</td>
<td>(l_1 &amp; l_2)</td>
<td>(l_1 &amp;&amp; l_2)</td>
<td>(l_1 \mid\mid l_2)</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

\[ a = \begin{bmatrix} true & false \\ false & true \end{bmatrix}, \ b = false \implies a \& b \ ? \begin{bmatrix} false & false \\ false & false \end{bmatrix} \] a \&\& b \ ? Illegal
HIERARCHY OF OPERATION

Precedence
- Arithmetic operator
- Relational operator
- \sim{} operation
- \& and && operator
- |, ||, and xor operator
Think of the program that returns the absolute value of the input number

```
if a < 0
    result = -a
else
    result = a
end
```
BRANCHES

if control_expr_1
    statement1
    statement2
    ...
else
    statement1
    statement2
    ...
end

if control_expr_1
    ...
    if control_expr_2
        ...
        if control_expr_3
            ...
            end
            ...
        end
        end
    end
    ...
end
BRANCHES

Example of if

\[ ax^2 + bx + c = 0 \quad x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

if \( b^2 - 4ac < 0 \)
    disp('This equation has two complex roots.');
elseif \( b^2 - 4ac = 0 \)
    disp('This equation has two identical real roots.');
else
    disp('This equation has two distinct real roots.');
end
BRANCHES

if control_expr_1
    statement1
    statement2
    ...
elseif control_expr_2
    statement1
    statement2
    ...
else
    statement1
    statement2
    ...
end
**SWITCH**

```
switch switch_expr
  case case_expr_1
    statement1
    statement2
    ...
  case case_expr_2
    statement1
    statement2
    ...
  otherwise
    statement1
    statement2
    ...
end
```

```
switch switch_expr
  case {case_expr_1, case_expr_2, case_expr_3}
    statement1
    statement2
    ...
  otherwise
    statement1
    statement2
    ...
end
```
Example of `switch`

```plaintext
switch (value)
case {1,3,5,7,9}
    disp('The value is odd.');
case {2,4,6,8,10}
    disp('The value is even.');
otherwise
    disp('The value is out of range.');
end
```
Think of the program that adds the numbers from 1 to 100

result = 1+2+3+4+…. ?

If the addition should be done up to 1000000?

Think of the program that finds out multiples of seven within the rage of [1, 10000]

First of all, you have to access all the numbers from 1 to 10000.

Then, check whether each number is multiples of seven or not
LOOPS

if \( \text{var} \) is a multiple of seven, then display \( \text{var} \)
\( \text{var} \) is increased by 1

check whether \( \text{var} \) is less than or equal to 10000

Repetition of the same work 10000 times
WHILE LOOP

while expression
statement1
statement2
...
end

var = 1;
while var <= 10000
    if mod(var, 7) == 0
        fprintf('%d is a multiple of seven \n', var)
    end
    var = var+1
end

while true
    disp("ECP is fun\n");
FOR LOOP

```plaintext
for index = expr
    statement1
    statement2
    ...
end
```

```plaintext
for ii = 1:2:10
    Statement 1
    ...
    Statement n
end
```

```plaintext
for ii = [1 2 3;4 5 6]
    Statement 1
    ...
    Statement n
end
```

→ loop index can be a vector
→ [1;4] at the first turn,
→ [2;5] at the second,
→ [3;6] at the third
BREAK AND CONTINUE

for ii = 1:5
  if ii == 3;
    break;
  end
  fprintf('ii = %d\n',ii);
end
disp(['End of loop!']);

for ii = 1:5
  if ii == 3;
    continue;
  end
  fprintf('ii = %d\n',ii);
end
disp(['End of loop!']);
BREAK AND CONTINUE

```matlab
for ii = 1:5
    if ii == 3;
        break;
    end
    fprintf('ii = %d\n',ii);
end
disp(['End of loop!']);
```

```matlab
for ii = 1:5
    if ii == 3;
        continue;
    end
    fprintf('ii = %d\n',ii);
end
disp(['End of loop!']);
```
Write down the program that displays only the prime number. The input value is from 1 to 10000.