CS 1112 Introduction to Computing Using MATLAB

Instructor: Dominic Diaz

Website: https://www.cs.cornell.edu/courses/cs1112/2022fa/
Agenda and announcements

● Previous lecture
  ○ Variables and assignment
  ○ Built-in functions, input and output

● Today’s lecture
  ○ Tips and good practices for writing a program
  ○ Branching (conditional statements)

● Announcements
  ○ Project 1 has been posted [due W 9/7]
    ■ Go to office hours/consultant hours!
    ● Online office hours uses queue me in! Check website for information.
    ■ Fill out CMS poll (by Wed. night) if you would like us to assign you a partner
    ■ Post questions on Ed [link to Ed on front page of website]
  ○ Poll everywhere questions will be graded from Thursday onwards.
    ■ Check out Ed for setup instructions
% Example 1_1: Surface area of a sphere
% r: radius of the sphere [unit]
% A: surface area of the sphere [unit^2]

r = input('Enter the radius: ');
A = 4*pi*r^2;
fprintf('Surface area is %f!\n', A);

A, r

are called variables. Variables are a named memory space to store a value.

= is the assignment operator. It allows us to store values in variables.

Semicolons should end almost all lines of MATLAB code.
- Semicolon → suppresses printing of the result of assignment statement
- No semicolon → prints out the results of assignment statement
- For now, put semicolon at the end of each line of MATLAB code except for comments.
Formatting operators

Inside single or double quotes, % becomes a formatting operator. Formatting operators allow you to convert data stored in a variable to text so it can be printed. %_ allows you to choose the formatting method:

- %f fixed point (or floating point)
- %d decimal (best for integers)
- %e exponential
- %g general—MATLAB chooses a format
- %c character
- %s string

For more on formatting operators, check out: https://www.mathworks.com/help/matlab/matlab_prog/formatting-strings.html

You will need this page for tomorrow’s discussion exercises.
% Example 1_1: Surface area of a sphere
% A: surface area of the sphere
% r: radius of the sphere

r = input('Enter the radius: ');  
A = 4*pi*r^2;  
fprintf('Surface area is %f!\n', A)
Comments

● For readability!
● A comment starts with % and goes to the end of the line
● **Start each program (script) with a concise description of what it does**
● Define each important variable/constant
Tips for writing a program

- Check that you know what things you have as inputs
- Start by writing out the inputs and the outputs then write the steps you need to get from inputs to outputs
- Add comments for readability
- Use variable names that make sense
What’s next?

- So far, all of the statements in our scripts are executed in order.
- We do not have a way to specify that some statements should be executed only under certain conditions.
- We need a new language construct…

IF
Motivating example: strictly increasing quadratic

Consider the quadratic function $q(x) = x^2 + bx + c$ on the interval $[L, R]$. This would be a parabola facing upwards.

Task: Write a code fragment that prints “Increasing” if $q(x)$ is strictly increasing across the interval and “Not increasing” if it does not.

To solve this problem, we need to know what criteria must be met for $q(x)$ to be strictly increasing on $[L, R]$. 
Strictly increasing quadratic

Consider the critical point \( x_c = -\frac{b}{2} \).

Criteria:

\[
\begin{align*}
\text{If } x_c &\leq L, \\
&\quad \text{Print 'Increasing'.} \\
\text{Otherwise,} &\quad \text{Print 'Not increasing'.}
\end{align*}
\]

This way of planning how to write a program is called pseudocode.

Pseudocode: Informal way of writing programs that a human can easily understand.
Strictly increasing quadratic

% Determine if the quadratic function \( q(x) = x^2 + bx + c \) strictly increases over interval \([L, R]\).

\( b = \text{input('Input the coefficient } b: \text{ \n')}; \)
\( c = \text{input('Input the coefficient } c: \text{ \n')}; \)
\( L = \text{input('Input the left endpoint } L: \text{ \n')}; \)
\( R = \text{input('Input the right endpoint } R, L < R: \text{ \n')}; \)

\( xc = -b/2; \)
\( \text{if } \)
\( \quad \text{fprintf('Increasing\n');} \)
\( \text{else} \)
\( \quad \text{fprintf('Not increasing\n');} \)
\( \text{end} \)

Last slide we said we wanted the criteria \( x_c \leq L \)...
Strictly increasing quadratic

% Determine if the quadratic function \( q(x) = x^2 + bx + c \) % strictly increases over interval \([L, R]\).

\[
\begin{align*}
b &= \text{input('Input the coefficient } b: \ \backslash n'); \\
c &= \text{input('Input the coefficient } c: \ \backslash n'); \\
L &= \text{input('Input the left endpoint } L: \ \backslash n'); \\
R &= \text{input('Input the right endpoint } R, L < R: \ \backslash n');
\end{align*}
\]

\[
\begin{align*}
xc &= -b/2; \\
\text{if } xc &\leq L \\
&\quad \text{fprintf('Increasing\n');} \\
\text{else} \\
&\quad \text{fprintf('Not increasing\n');} \\
\text{end}
\end{align*}
\]
The *if* construct

```plaintext
if [boolean expression 1]
    [Statements to be executed if expression 1 evaluated to true]
elseif [boolean expression 2]
    [Statements to be executed if expression 1 evaluates to false
     but expression 2 evaluates to true]
else
    [Statements to be executed if all previous expressions
     evaluate to false]
end
```

```plaintext
if xc <= L
    fprintf('Increasing\n');
else
    fprintf('Not increasing\n');
end
```
Things to know about the if construct

- At most one branch of the statements is executed
- There can be any number of elseif clauses
- There can be at most one else clause
- The else clause must be the last clause in the construct (if there is one)
- The else clause does not have a condition
- NO SEMICOLON after if, elseif, else, and end lines
Example 2 - where is the critical point?

Consider the quadratic function \( q(x) = x^2 + bx + c \) on the interval \([L, R]\). Print “inside” if \( x_c \) is inside the interval, “left” if \( x_c \) is to the left of the interval, or “right” if \( x_c \) is to the right of the interval.
Example 2 - where is the critical point?

Consider the quadratic function $q(x) = x^2 + bx + c$ on the interval $[L, R]$. Print “inside” if $x_c$ is inside the interval, “left” if $x_c$ is to the left of the interval, or “right” if $x_c$ is to the right of the interval.

```matlab
% Determine if the critical point of q(x) = x^2 + bx + c
% is left, right, or inside the interval [L,R].
b = input('Input the coefficient b: \n');
c = input('Input the coefficient c: \n');
L = input('Input the left endpoint L: \n');
R = input('Input the right endpoint R, L < R: \n');

xc = -b/2;
if xc <= R && xc >= L
    fprintf('Inside\n');
elseif xc < L
    fprintf('Left\n');
else
    fprintf('Right\n');
end
```

&& is a logical operator. Here it means that both the $xc \leq R$ and $xc \geq L$ conditions must be true for the computer to print 'Inside'.

Logical operators

&& logical and: Are both conditions true?

Example - “is $L \leq x_c$ and $x_c \leq R$?”
In code - $L \leq x_c \land x_c \leq R$

|| logical or: Is at least one condition true?

Example - “is $x_c \leq L$ or $R \leq x_c$?”
In code - $x_c \leq L \lor R \leq x_c$

~ logical not: negation

Example - “is $x_c$ not outside $[L,R]$?”
In code - $(x_c < L \lor R < x_c)$