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

Recap

All lines of code comprise a script or program.

Any % not in single or double quotes denotes a comment.

```
% 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

You will need this page for tomorrow's discussion exercises.

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

```
% 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)
```

Symbol to indicate that the rest of the line is a comment–not to be executed as code

%f is replaced by the value stored in A

Inside single quotes, it becomes a formatting operator.

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?

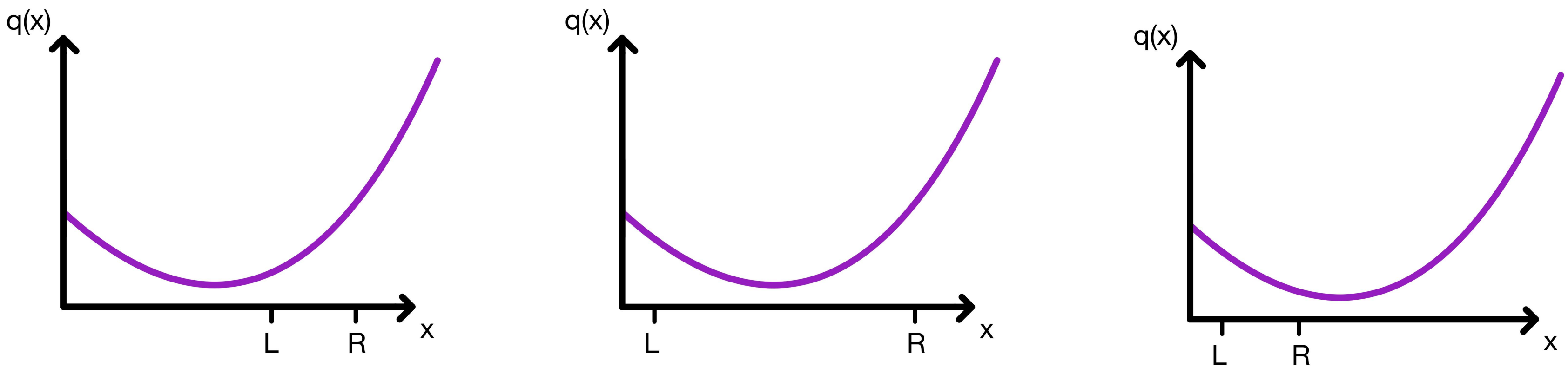
- So far, all of the statements in our scripts are executed in order
- We do not have a way to specify that some statements should be executed only under certain conditions
- We need a new language construct…

IF

Motivating example: strictly increasing quadratic

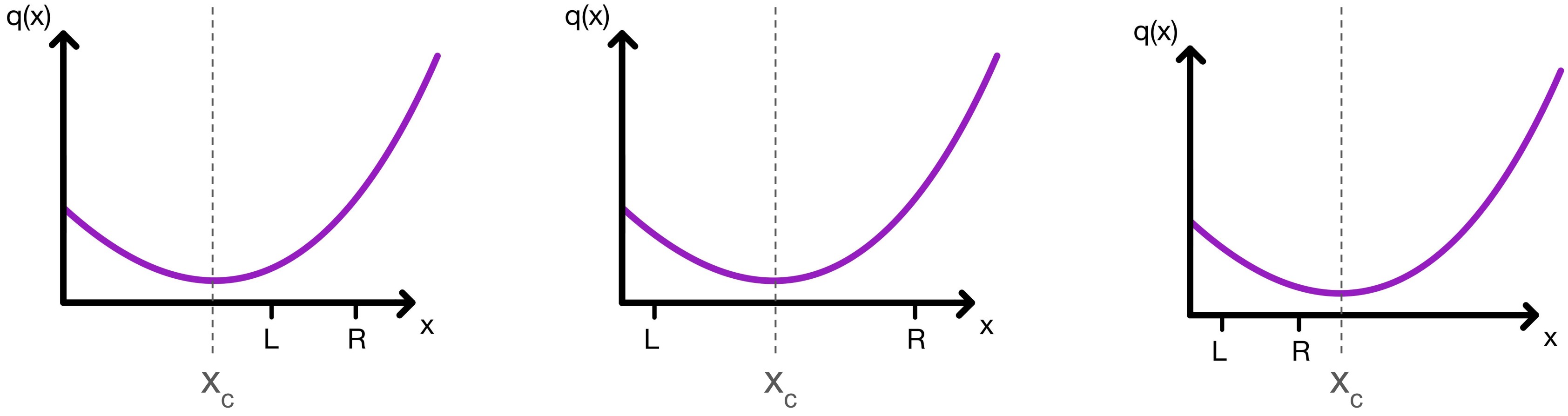
Consider the quadratic function $q(x) = x^2 + bx + c$ on the interval [L, R]. This would be a parabola facing upwards.

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 = -b/2.

Criteria:

If $x_c \leq$ L,
    Print 'Increasing'.
Otherwise,
    Print 'Not increasing'.

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 = 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 ____________
    fprintf('Increasing\n');
else
    fprintf('Not increasing\n');
end
```

Relational Operators

`<`	Less than
`>`	Greater than
`<=`	Less than or equal to
`>=`	Greater than or equal to
`==`	Equal to
`~=`	Not equal to

Last slide we said we wanted the criteria $x_c \le L$…

Strictly increasing quadratic

```
% Determine if the quadratic function q(x) = x^2 + bx + c
% strictly increases over 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 <= L
    fprintf('Increasing\n');
else
    fprintf('Not increasing\n');
end
```

Relational Operators

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
~=	Not equal to

The if construct

```
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
```

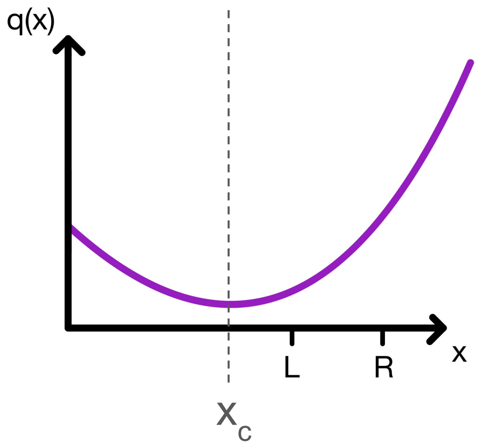
```
if xc <= L
    fprintf('Increasing\n');
else
    fprintf('Not increasing\n');
end
```

Things to know about the `if` construct

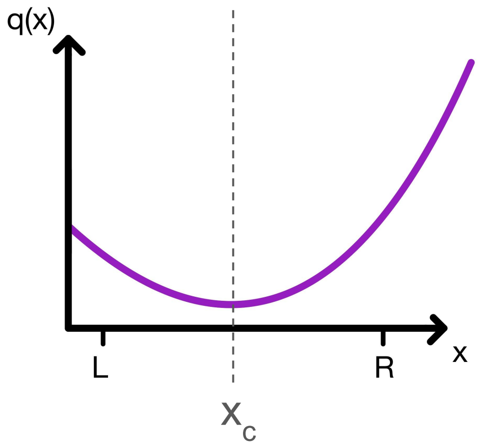
- At most one branch of the statements is executed
- There can be any number of `elseif` clauses
- There can be at most one `else` clause
- The `else` clause must be the last clause in the construct (if there is one)
- The `else` clause does not have a condition
- NO SEMICOLON after `if`, `elseif`, `else`,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

Example 2 - where is the critical point?

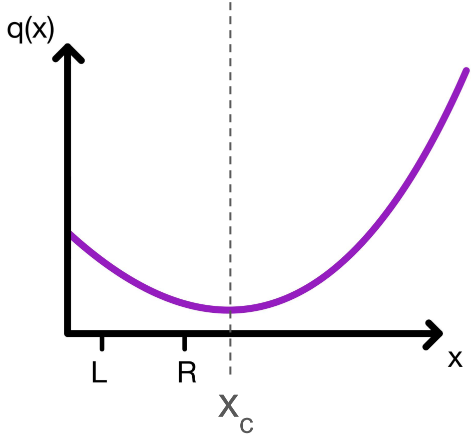
Consider the quadratic function $q(x) = x^2 + bx + c$ on the interval [L, R]. Print “inside” if x_c is inside the interval, “left” if x_c is to the left of the interval, or “right” if x_c is to the right of the interval.



Print “left”



Print “inside”



Print “right”

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.

```
% 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 <= R` and `xc >= L` conditions must be true for the computer to print 'Inside'.

Logical operators

&& logical <u>and</u>: Are both conditions true?

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

In code - `L <= xc && xc <= R`

|| logical <u>or</u>: Is at least one condition true?

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

In code - `xc <= L || R <= xc`

~ logical <u>not</u>: negation

Example - “is x_c not outside [L,R]?”

In code - `~(xc < L || R < xc)`