CS100M/CIS121/EAS121
Introduction to Computer Programming

Spring 2004

Lecture 4
MATLAB Statements

Announcements

- Lab debacle

Tokens: arrays

- Everything in MATLAB is actually an array
- Even scalars are arrays
- examples:

```
[1 2 3 4]

[1 2 3 ; 4 5 6]

1:4

0:2:10

10:-2:0

[]
```


Objectives

- What do you know? MATLAB alphabet, words
- Language needs sentences!
- Assemble tokens into statements
- Statements have classifications for purpose
- Statements have terminal punctuation
- Kinds of statements? expression, function, I/O, selection, repetition, empty,...

Empty Statement

- Simplest! Do nothing
- eg) press return or use semi-colon
  - eg) >> `;`
- Why?
  - **punctuation**! need way of ending other statements
  - adding white space between lines
  - *sometimes* good for logic:
    - eg) if find wrong value, do nothing
    - eg) `for ii=1:1e10; end % what?`

Statement Punctuation

- semicolon:
  - separate 2 statements on same line
  - suppress output of one statement
- comma:
  - separate 2 statements on same line
- return:
  - end statement
  - add whitespace between lines (can be good style!)

Expression Statements

- ***Expression*** reminder:
  - operands, operators
  - combine to produce value
- Examples
  - eg) `1 + 1`
  - eg) `1 + sqrt(4)`
  - eg) `1 || 0`
- Why?
  - algorithms require math
  - look for values to solve problems
  - functions produce results

Expression Statements

- Expressions evaluate to a value
- Difference between expression and expression statement?
  - Expression written as a *command*
  - tell MATLAB to evaluate the expression and store result in `ans`

    eg) >> `1 + 1` ↵

    eg) >> `sqrt(4)` ↵
  - note how return/enter (↵) acts as punctuation

Operator Precedence

- Example) 1 + 2 / 3 $\frac{(1+2)}{3}$
  - need parens? why?
  - some operators more important than others
- Operator precedence
  - languages define importance
  - use parens to force an operation (also good style!)
  - MATLAB? pg 49, 89 (and more) Chapman
  - search **Operator Precedence** in MATLAB Help! (copied on next panel)

MATLAB Operator Precedence

Directly from MATLAB's Help:

Parentheses ( )

Transpose (. '), power (. ^), complex conjugate transpose ('), matrix power(^)

Unary plus (+), unary minus, (-), logical negation (~)

Multiplication (. *), right division (. /), left division(. \ ), matrix multiplication (*), matrix right division (/), matrix left division (\ )

Addition (+), subtraction (–)

Colon operator (:)

Less than (<) less than or equal to (<=) greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)

Element-wise AND (&)

Element-wise OR (|)

Short-circuit AND (&&)

Short-circuit OR (| |)

Operator Associativity

- Example) 1 – 2 – 3
- What's the answer? why?
- ***Associativity***:
  - "direction" operators work
  - most operators work from left to right

Declaration

- Declare (state) to language the type of value that a variable can store
- MATLAB? ***everything is an array***
- So...
  - MATLAB is ***weakly typed***
  - MATLAB does have a type (array), but it subsumes all other types!
  - eg) `x = 'a' ; x = 1 ; x = 1:4 ;` (all different expression types, but same variable type!)

Assignment

- Want ways to store information!
- Syntax: ***name = expression***
- eg) **`x = 3`**
- pseudocode:
  - variable *gets* value of expression
  - *name* ← *value*
    (someone please ask why a right arrow)

Some Assignment Rules

- Variables must be assignable (only legal names!)
- Variables MUST have initial value before being used
- Variables do not receive default values
- Variables retain values until reassigned, cleared, or function ends (more on this later)
- Avoid using common/built-in function names

Handy MATLAB Functions for Assignments

- **`clear`**: clear all variables
- **`clear var`**: clear just ***`var`***
- **`clear v1 v2 v3`**: clear ***`v1 v2 v3`***
- **`who`**: list all current variables
  (see also menu View → Workspace)
- **`isvarname(str)`**: check if string ***`str`*** can be a variable name

Function Calls

- provide collection of statements
  - resembles script
  - differences?
    - can evaluated as part of an expression
    - can take 0 to many input values and operate on them
    - eg) `1 + sqrt(4)`
    - eg) `rand`
- syntax?

Function Syntax

- calling syntax: ***name*(*arguments*)**
  - ***name*** is the name of the function
  - ***arguments*** are values that you pass to the function
  - sometimes functions have no arguments (and thus do not need the parentheses)
  - eg) `plot(1:4,[1,2,4,16])`
- defining syntax?
  - will see later
  - hint: see `help function`

I/O

- I/O
  - input
  - output
- transfer information to and from computer
- CS100 focus:
  - user: prompt for text, display results
  - file: get text data, put text data (later)
  - plotting: graph of data (later)

User Output

- user output:
  - `disp(stringarray)`
  - eg) `disp('hi') % single string OK`
  - eg) `disp(['a','b','c']) % many strings`
  - eg) `disp(['The answer: ',num2str(13)])`
- see also `sprintf`, `diary`

User Input

- `input(string)`:
  - eg) `size = input('Your waist? ')`
  - After you press enter, MATLAB waits for you to enter an expression
  - MATLAB evaluates the expression and stores it the variable you provide (or `ans` if you don't)
  - If you press enter without giving an expression, MATLAB uses the empty array `[]`
- see also `input(string,'s')`

More Statements!

- Selection
  - want to choose tasks

  eg) `if x>5, disp('a'), else disp('b'), end`
- Repetition
  - want to repeat tasks

  eg) `x = 1, while x < 5, x=x+1, end;`

  eg) `for ii=1:4, ii, end;`
- Object Creation
  - create "things" with data and action
  - wait for Java, though MATLAB can do this!

Where are we?

- Problems and Solutions
- Need problem solving approach
- Need general way to describe solutions (algorithms)
- Need to convert algorithm to computer language
  - syntax, semantics
  - character set, tokens, statements
- Learn more branching statements
  - selection
  - repetition