Storing Values for Safe Keeping

Variables and their Values

MATLAB programming environment

CS12 Scientific Computation
Department of Computer Science
Wellesley College
Storing values in variables

Assignment statements

- General format of an assignment statement:
  
  \[ \text{variable-name} = \text{expression} \]

- Right hand side can be any expression that evaluates to a value:

  \[ \text{>> hypotenuse} = \text{sqrt}(3^2 + 4^2) \]

- Executing an assignment statement:
  1) evaluate the expression on the right side
  2) assign value of expression to variable named on the left side
Using variables in expressions

Create a variable to store the circumference of a circle with the stored value of radius

Hunting for ‘s

```matlab
>> x = 17
>> x + 1 = 10
>> x = x / x + x
>> diff = (x - y) / 2
>> end = x - 18
>> y = x = 2
```
Assigning variables in a script

- Statements in script are executed as if they were typed directly into the Command Window

- Variables created in CW can be changed by assignments in scripts and vice versa!

```
% assignVars.m
% assigns variables
% to values

a = 1
b = 2
c = 3
```

Speaking of ‘s

- What would happen if the script contained a syntax error? For example:

```
% buggyScript.m
a = 10
b = 20 +
c = 30
```
What’s in a name?

- **Variable names** must start with a letter and may contain any number of letters, digits and underscore characters (‘_’)
  - C3PO  my_monthly_pay  yourTurn
- **Case matters!**
  - stella ≠ Stella
- MATLAB reserves some words for special purposes:
  - break  case  catch  continue  else  elseif  end  for  function  global  if  otherwise  persistent  return  switch  try  while

Choose concise, meaningful names

- The good, the bad, & the ugly
  - maxArea
  - xxyyyzz_3b
  - totalMonthlyHedgeFunds
  - 2pi
  - tf
  - result
- **Case matters!**
  - stella ≠ Stella
Time-out exercise

- Write a sequence of assignment statements that exchange the values of thing1 & thing2

- Before
  
  \[
  \begin{array}{c}
  \text{thing1} \\
  \text{thing2}
  \end{array}
  \]
  
  \[
  \begin{array}{c}
  2 \\
  5
  \end{array}
  \]

- After

  \[
  \begin{array}{c}
  \text{thing1} \\
  \text{thing2}
  \end{array}
  \]
  
  \[
  \begin{array}{c}
  5 \\
  2
  \end{array}
  \]

Floating point numbers

- Decimal numbers are represented as a type called double
  
  \[
  \text{>> number} = 8.43
  \]
  
  number =
  
  8.4300

- MATLAB normally prints 4 digits after the decimal point and rounds numbers up or down to fit*

* Compact notation is used only for printing - try typing \text{format long}
Really big & really little numbers

- MATLAB uses scientific notation to print very large or very small numbers
  
  ```
  >> number = 1230000000
  number = 1.2300e+09
  >> number = 0.000789
  number = 7.8900e-04
  ```

- MATLAB has a special representation for really big numbers
  
  ```
  >> number = 1.2e+9999
  number = Inf
  ```

* What do get when you type: 1/0 or 1/inf or 0*inf?

Strings

- **Strings** consist of letters, digits, symbols, and white space, surrounded by single quotes:
  
  ```
  myName = 'Sam I am'
  eec = ' )&it:s;elf,'
  thirteen = '6 + 7'
  ```

- Common `s
  
  ```
  callMe = Ishmael
  reply = 'Madam, I’m Adam'
  ```
Unfriendly programs

- The following program converts pounds to stones:
  
  ```
  % weights.m
  pounds = input('');
  stones = 0.0714 * pounds
  ```

- If we run it by typing
  
  ```
  >> weights
  ```
  
  it would just sit there and look at us

Friendly programs

- Tell the user what's needed and what's printed:
  
  ```
  % weights.m
  pounds = input('Enter your weight in pounds: ');
  disp('Your weight in stones is: ');
  stones = 0.0714 * pounds
  ```

- When run...
  
  ```
  >> weights
  Enter your weight in pounds: 120
  Your weight in stones is:
  stones =
  8.5680
  ```