Topics: Anatomy of a program, variable & assignment, input & output, branching
Reading: CFile sec 1.2, 1.3

Example: surface area of a sphere

```
% Example 1_1:  Compute surface area of a sphere
% A: surface area of the sphere
% r: radius of the sphere

r= input('Enter the radius: ');

A= 4*3.14159*r*r;

fprintf('Surface area is %f.\n', A);
```

Anatomy of a program

- input
- calculation
- output
- comments—documentation and explanation

Definitions

- **Algorithm**: a set of procedures for solving a problem
- **Program**: an algorithm implemented in some language
- **Variable**: a named memory space for storing a value
- **Assignment**: the action of putting a value into a variable
- **Expression**: a combination of operators and operands (variables, constants) that evaluate to a value

Variables & assignment

A *variable* is a named memory space for storing a value. Think about it as a box to hold an item. Valid variable names begin with a letter and can contain digits. Always use *meaningful* variable names!

Assignment is the action of putting a value into a variable. The assignment operator is the symbol = but do *not* read this as "equal." Read the statement `x= 3` as "x *gets* three." Assignment is an action; it is not an expression of equality. Some example assignment statements are

```
x= 2*3.1416
y= 1+x
z= 4^2 - cos(y)
```

In an assignment, the expression on the right hand side (rhs) is evaluated *before* the assignment operation. Therefore, any variable on the rhs *must be initialized.*

Statements are executed in sequence:

```
x= 2*3.14
y= 1+x
x= 5
% What is y now?
```

Calling a Matlab Built-in function

An expression may include a call to a function. Matlab has numerous built-in, or predefined, functions related to mathematics, text handling, graphics, . . . , etc. For example, the statement

```
  z= 4^2 - cos(y)
```

involves a *call* to the built-in cosine function and the value in variable `y` is passed to the function. So `y` is the *argument* to the function. The function *returns* the result of `cos(y)`, a value, which is then used in the subtration operation in the above statement.

Input & output statements

Input:	*variable* = `input('`*prompt*`')`
Output:	`disp('`*words to be displayed*`')`
	`fprintf('Value of x is %f, not %d!\n', x, y)`

Some substitution sequences	
%f	fixed point (or floating point)
%d	decimal—whole number
%e	exponential
%g	general—Matlab chooses
%c	character
%s	string

Comments

- Use comments for readability!
- A comment starts with the "%" symbol and goes to the end of the line
- Start each program with a *concise* description of what it does
- Define each important variable/constant
- Top a block of code for a specific task with a *concise* comment

Example: expanding sphere

Modify the previous program to calculate the increase in surface area given an increase in the radius of a sphere.

```
% Example 1_2:  Explore how the surface area of a sphere
% changes with an increase in the radius.

r= input('Enter radius r in miles: ');
delta= input('Enter delta r in inches: ');









fprintf('Increase in area (mile^2) is %f.\n', incr);
```

Branching

Consider the quadratic function $q(x) = x^2 + bx + c$ on the interval $[L, R]$. Which is smaller, $q(L)$ or $q(R)$?

```
% Fragment 1
  qL= L^2 + b*L + c;  % q(L)
  qR= R^2 + b*R + c;  % q(R)

  --------------------
     fprintf('qL less than qR\n');

  --------
     fprintf('qR less than or equal to qL\n');

  --------
```

Relational Operators

Operator	Meaning
>	greater than
>=	greater than or equal to
==	equal to
~=	not equal to
<=	less than or equal to
<	less than

Do the following code fragments do the same thing?

```
  % Fragment 1
    if (x>y)
       disp('hey')
    else
       disp('ho')
    end

  % Fragment 2
    if (y>x)
       disp('ho')
    else
       disp('hey')
    end
```