CFCS1

Matlab Programming

Miles Osborne

School of Informatics
University of Edinburgh
miles@inf.ed.ac.uk

January 14, 2010

Data Types

- Literals: 10, 10.5, -3 etc.
- Booleans: 0 or 1
- Vectors: (see class).
- Matrices: (see class).
- Strings: 'hello I am a string'

Expressions

Here are some example expressions:

```
octave-2.9.18:1> 1
ans =  1
octave-2.9.18:2> 1 + 1
ans =  2
octave-2.9.18:3> a = 1
a =  1
octave-2.9.18:4> a = a + 1
a =  2
octave-2.9.18:5> a
a =  2
```

Expressions

More example expressions:

```
octave-2.9.18:9> a = 10 ; b = 11
b =  11
octave-2.9.18:10> % A comment
octave-2.9.18:10>
octave-2.9.18:10> exp(a)
ans =  2.2026e+04
octave-2.9.18:11> b = (a + 10) * 12;
octave-2.9.18:12>
```


Comparisons

EXP COMPARISON *EXP*

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

$a > 10$

Logical Operators

&	and
|	or
~	not

$(a == 10) \mid (b < 0)$

IF Statements

```
if CONDITION
   EXP
   EXP
end
```

- A *CONDITION* is a test which evaluates to *true* or *false*.
- The reservered word *end* terminates the set of statements.

```
if (a > b)
   disp('a is greater than b');
   a = 1;
end;
```

IF Statements

```
if CONDITION
   EXP
else
   EXP
end
```

- The reserved word *else* specifies the expressions that are evaluated if the test is false.

```
if (a > b)
   disp('a is greater than b')
else
   disp('b is greater than a')
end;
```


IF Statements

```
if CONDITION
   EXP
elseif CONDITION
   EXP
else
   EXP
end
```

- The reserved word *elseif* specifies another test that is evaluated if the previous test is false.

IF Statements

```
if (a > b)
   disp('a is greater than b')
elseif (a == b)
   disp('b is a')
else
   disp('b is greater than a)
end;
```

- The *elseif* statement allows for *if* statements to be chained together.

FOR-loop

```
for INDEX = EXP:  FINISH
   EXP
   EXP
end
```

- *FOR* loops execute a block of code a fixed number of times.
- *FINISH* is a test for when we stop.
- *FOR* loops (and loops in general) can be nested.

FOR-loop

```
for a = 0: 5
   disp('hello')
end;
```


WHILE-loop

```
while CONDITION
   EXP
   EXP
end
```

- *WHILE loops* execute a block of code a variable number of times.
- A *while* loop is a generalised *for* loop.
- To break out of a loop mid-way, use the *break* statement.

WHILE-loop

```
a = 0;
while a < 10
   b = 1;
   a = a + 1;
end;
```


Functions

Typically, we want to specify a repeated operation:

- A MATLAB function is stored in a file ending with a *.m* extension.
- The function name must be the same as the file name (less extension).
- MATLAB functions have two parameter lists:
  - A list of arguments.
  - A list of results.
- Arguments can be changed, but that is bad pracice.
- Arguments are copied when a function is invoked.

Functions

```
function [output_list] = function_name(input_list)
```

- The first word must be *function*.
- Optional arguments are enclosed in square brackets.
- (If there are no arguments, then the brackets are dropped)
- Arguments are separated using commas.

```
function addtwo(x,y)
% add x and y

x + y
```


Functions

```
function [result] = addtwo(x,y)
% add x and y

result = x + y
```

- Here we have returned the result of adding x and y.
- Comments after the function are printed eg:

  `help addtwo`
- All variables in a function are local (unless global):

  `global b;`

Writing Output

- The *disp* function can write simple messages:

  `disp(a)`
- The c-like *printf* function can write more complex output:

  `printf('%d %d\n',a,b);`

Reading Input

- The *input* function can read simple input:

  `b = input('type a number:')`
- The c-like *sscanf* function can read more complex output:

  `s = '2.71 3.14';`
  `a = sscanf(s,'%f')`
- This creates a two-element vector from the string representation.

Files

- Files have *names*: the actual name you see.
- Files are manipulated using *file handles*.
- A file handle indicates the position within a file.
- Files have *modes*: append to end, write from scratch etc.

Files: Writing

(Example taken fom Web)

```
output = fopen('myfile.txt','wt'); %'wt' means write te
if (output < 0)
   error('failed to open myfile.txt');
end;
a = 10;
fprintf(output, 'A line of text %d\n',a);
fclose(output);
```


Files: Reading

- To read from a file, use *fscanf*

```
input = fopen('myfile.txt','rt');  % 'rt' means read text
if (input < 0)
   error('failed to open myfile.txt');
end;
a = fscanf(input, '%d\n');
disp(a);
fclose(input);
```


Summary

- MATLAB is a fairly standard programming language.
- There is a lot of online help.
- MATLAB is quite quirky.