Introduction to Octave (Part II)
IT WS I - Lecture 19

Saurabh Barjatiya

International Institute Of Information Technology, Hyderabad

04 November, 2009

Contents

1. Statements
   - Condition statements
   - Loop statements
2. Functions
   - Simple functions
   - Function and Script files
   - Octave programs as shell scripts
3. Taking user input
   - Basic user input

Condition statments - 1

'if' statement

if syntax

```
if(condition)
    then-body
elseif(condition)
    elseif-body
else
    else-body
endif
```

Condition statments - 2

'if' example - Checks whether x is even or odd

if example

```
if(rem (x, 2) == 0)
    printf("x is even\n");
else
    printf("x is odd\n");
endif
```

Condition statments - 3

There is no space between ‘else’ and ‘if’ in ‘elseif’

elseif does not contains spaces - Incorrect code

```
if(c1)
    body-1
else if(c2)
    body-2
endif
```

Condition statments - 4

'switch' statement

switch syntax

```
switch(X)
    case 1
        do_something();
    case 2
        do_something_else();
    otherwise
        do_something_completely_different();
endswitch
```

Condition statments - 5

'switch' example

switch example - 1

```
switch(x)
    case 1
        printf("x is 1\n");
    case 2
        printf("x is 2\n");
    otherwise
        printf("x is neither 1 nor 2\n");
endswitch
```

Condition statments - 6

'switch' example

switch example - 2

```
switch(x)
    case "hello"
        printf("x is hello\n");
    case "world"
        printf("x is world\n");
    otherwise
        printf("x is neither hello nor world\n");
endswitch
```

Condition statments - 7

Switch cases dont fall through in octave

Wrong code

```
switch(x)
    case "hello"
    case "world"
        printf("x is either hello or world\n");
    otherwise
        printf("x is neither hello nor world\n");
endswitch
```

Condition statments - 8

If any element of cell array matches then that case statement is executed.

Correct code

```
switch(x)
    case {"hello", "world"}
        printf("x is either hello or world\n");
    otherwise
        printf("x is neither hello nor world\n");
endswitch
```

Loop statments - 1

while statement

while syntax

```
while (condition)
    body
endwhile
```

Loop statments - 2

while example - Calculates first 10 numbers of fibonacci series and stores them in fib

while example

```
fib = ones (1, 10);
i = 3;
while (i <= 10)
    fib (i) = fib (i-1) + fib (i-2);
    i++;
endwhile
fib
```

Loop statments - 3

do-until statement

do-until syntax

```
do
    body
until (condition)
```

Loop statments - 4

do-until example - Calculates first 10 numbers of fibonacci series and stores them in fib

do-until example

```
fib = ones (1, 10);
i = 2;
do
    i++;
    fib (i) = fib (i-1) + fib (i-2);
until (i == 10)
fib
```

Loop statments - 5

for statement

for syntax

```
for var = expression
    body
endfor
```

Loop statments - 6

for example - Calculates first 10 numbers of fibonacci series and stores them in fib

for example - 1

```
fib = ones (1, 10);
for i = 3:10
    fib (i) = fib (i-1) + fib (i-2);
endfor
```

Loop statments - 7

Looping over matrix

for example - 2

```
disp("Loop over a matrix")
for i = [1,3;2,4]
    i
endfor
```

The value of variable will be one column vector at a time

Loop statments - 8

Looping over cell array

for example - 3

```
disp("Loop over a cell array")
for i = {1,"two";"three",4}
    i
endfor
```

The value of variable will be one column vector at a time

Loop statments - 9

Looping over elements of structure

for example - 4

```
x.a = 1
x.b = [1, 2; 3, 4]
x.c = "string"
for [val, key] = x
    key
    val
endfor
```

The values of key and val are set to name of the element and corresponding value in turn.

Loop statments - 10

While looping over elements of structure note that:

- The elements are not accessed in any particular order. If you need to cycle through the list in a particular way, you will have to use the function fieldnames and sort the list yourself.
- The key variable may also be omitted. If it is, the brackets are also optional. This is useful for cycling through the values of all the structure elements when the names of the elements do not need to be known.
- We can use ‘break;’ and ‘continue;’ statements in octave loops.

Contents

1. Statements
   - Condition statements
   - Loop statements
2. Functions
   - Simple functions
   - Function and Script files
   - Octave programs as shell scripts
3. Taking user input
   - Basic user input

Simple functions - 1

function syntax - 1

function syntax - 1

```
function name
    body
endfunction
```

Simple functions - 2

function syntax - 2

function syntax - 2

```
function name (arg-list)
    body
endfunction
```

Simple functions - 3

function syntax - 3

function syntax - 3

```
function ret-var = name (arg-list)
    body
endfunction
```

Simple functions - 4

function syntax - 4

function syntax - 4

```
function [ret-list] = name (arg-list)
      body
endfunction
```

Simple functions - 5

Few points regarding functions

- We can return from functions anytime using return; statement.
- If we do not assign any values to variables that function is suppose to return then error will get generated when function is called
- We can assign default values to arguments of function by using '<variable_name>=<default_value>' syntax while declaring arguments.

Function and Script files - 1

Few points regarding functions

- We can store functions in functions and scripts in files so that we can re-use them.
- In case of function file the file name should be '<function_name>.m' and file should be in path. (Current working directory is included in default path)
- Only one function would be visible from outside of function file, the one whose name matches name of file. It should also be first function in source file.

Function and Script files - 2

- If first source code line in file does not declares a function then file is treated as script file.
- If we do not have any meaningful statements in script file to execute then just use dummy '1;' at top of script file so that first source code line does not defines a function.
- We should 'source()' script file so that we can use functions declared in script file

Octave programs as shell scripts - 1

Shell script that prints arguments passed to it on command line

Example shell script

```
#! /usr/local/bin/octave -qf
printf ("%s", program_name ());
arg_list = argv ();
for counter1 = 1:nargin
    printf (" %s", arg_list{counter1});
endfor
printf ("\n");
```

Octave programs as shell scripts - 2

- The '#!' operator should be in first line starting from first column in shell script
- Look for path of octave using 'which' command to use in shell scripts
- '-qf' is supplied to suppress any warnings printed by octave during initialization
- Shell scripts should be executable. Hence 'chmod +x <filename>' is necessary before we can use shell scripts.

Contents

1. Statements
   - Condition statements
   - Loop statements
2. Functions
   - Simple functions
   - Function and Script files
   - Octave programs as shell scripts
3. Taking user input
   - Basic user input

Basic user input

Taking numeric input

Basic user input - 1

```
a=input("Enter a number : ");
printf("Double of a is %d\n", a*2);
```

Basic user input

Taking string input

Basic user input - 2

```
a=input("Enter name : ", 's');
printf("Hi %s\n", a);
```