Scilab

Programming
&
Functions

By M. D. Bhopatkar

Scilab Workshop

Programming

- Interpreter with it's own syntax
  - Execution of commands
    - Line by line
    - By block

  Set of programming tools
  - Objects-Vectors,Matrices,Polynomials est.
  - Loops
  - Conditionals

for loop

for variable = expression

.

.

.

End

The for loop can iterate on any vector,matrix or lists.

Examples

```
→ x=1; for k=1:4, x = x*k, end
ans  x=24
→ x=1; for k=[-1 3 0], x = x+k, end
ans  x=3
→ l = list(1, [1,2;3,4], 'str')
→ for k=l, disp(k), end
ans  1
        ! 1.  2. !
        ! 3.  4. !
        str
```

while loop

while expr

.

.

.

End

The while loop repeatedly performs a sequence of commands until a condition is satisfied.

LOGICAL OPERATORS

- == equal to
- < smaller than
- > greater than
- <= smaller or equal to
- >= greater or equal to
- ~= or <> not equal to

example

```
→ i=1; x=0
→ while   i<=4
→      x=x+i;
→ end
→ disp (x)
```

ans . x=10

Loop breaks

- Loop can be ended by command break
- In nested loops, break exits from innermost loop

Conditionals

- if then else

  → x=1;

  → if x>0 then y= -x, else y=x, end

  ans y= -1

- select case

  → x= -1;

  → select x, case 1, y=x+5, case -1, y=sqrt(x), end

  ans y= i

- The elseif can be used.It is a keyword recognised by the interpreter.

Functions

- function [y1, ..., yn]=f00(x1, ..., xm)

  .

  .

  .

  endfunction

  <u>Note</u>: function has local environment that communicates with the outside thru input and output arguments

Features of Functions

- Functions can be defined online or offline
- Arguments can be any Scilab objects
- More than one output arguments
- Input/output arguments can be functions
- Functions can be nested

Inline definition

```
-->function [x,y]=f1o(a,b)
→   x=a+b
→   y=a-b
-->endfunction
-->[x,y]=f10(3,2)
 y  =
     1.
 x  =
     5.
```

One-line definition

- y=x^2

```
-->function y=sq(x),y=x^2,endfunction
```

Or

```
-->deff('y=sq(x)','y=x^2')
-->sq(5)
ans =
      25.
```

One-line definition

```
-->deff('y=f01(x)','y=x^3-2*x-5')

-->deff('y=f02(x)','y=3*x^2-2')

-->deff('y=f03(x)','y=x-(f01(x)/f02(x))')

-->f03(2)
 ans = 2.1
 -->f03(ans)
 ans = 2.0945681
```

Functions : Vector/Matrix argument

```
-->function [y]=f00(x)
→     y=x*abs(x)/(1+x^2)
-->endfunction
 -->f00(.5)
 ans = 0.2
-->x=[1.  2.  3.];
-->f00(x)
 !--error 10
```

Inconsistent multiplication.

Use of dot: Vector/Matrix argument

```
-->function [y]=f01(x)
→       y=x .* abs(x) ./ (1+x .^ 2)
-->endfunction

-->x = [1 2 3];


 -->[x ; f01(x)]
 ans =
     1.      2.      3.
    0.5     0.8     0.9
```

Use of dot & feval command

```
-->x=[1  -3 ; 2   -3];
-->y  = f01(x)
 y=
   0.5  - 0.9
   0.8  - 0.9
-->feval(x,f00)  //dot is not used
ans  =
    0.5    0.9
   0.5    0.8
```

Evaluate the expression

- y = x(sin x) / (x^2 + 1)
- (1)Use function with dot operation
- (2)Use function with 'feval' command
- (3)Evaluate directly using the scilab environment.

```
-->x=-1:.4:1;
-->y=x .* sin(x) ./ (x .^2 + 1)
y  =
    0.4207355    0.2491070    0.0382056
0.0382056    0.2491070    0.4207355
-->x=[.5  1 ; -.5  1];
 -->y=x .* sin(x) ./ (x .^2 + 1)
 y  =
     0.1917702    0.4207355
    0.1917702    0.4207355
```

Function written in a file

- Create functions in any editor like Scipad
- Functions are scilab objects and schould not be considered as files.
- Such function should be loaded in the Scilab environment.
- Commands are getf('filename') or exec('filename', -1)
- A file may contain several functions

Function written in scipad

- Y =(x^2 + 2) / (2xsinx)

```
1 function y=fv1(x)
 2   t1=x.^2+2
 3   t 2=2*x.*sin(x)
 4    y=t1./t2
 5  endfunction
```

Example : Vector argument

```
-->exec('d:\mdb\myscilab\fv1.sci')
 -->function y=fv1(x)
-->  t1=x.^2+2
-->  t2=2*x.*sin(x)
-->  y=t1./t2
-->endfunction
 ->x = [1 3 4];
 -->fv1(x)
 ans  =
       1.7825927    12.991307  - 2.9730346
```

Fibonacci Sequence
f(n)=f(n-1)+f(n-2) ; f(1)=f(2)=1

```
-->exec('d:mdb\myscilab\fb.sci')
-->function [x]=fb(n)
-->  x=[1 1];
-->    for i=3:n
-->      c=x($)+x($-1)
-->      x=[x c]
-->  end
-->endfunction
```

```
-->fb(10)
 ans  =

    1.    1.    2.    3.    5.    8.    13.    21.    34.    55.
```

Multiple functions in one file

```
-->exec('d:\mdb\myscilab\fv2.sci')
 -->function [y]=fv2(x)
-->  y=2*x+x^2
-->endfunction
-->function [y]=fv3(x)
-->  y=(2*x+x^2)/(x+5)
-->endfunction
-->function [y]=fv4(x)
-->  y=(2*x+x^2)/(x+5)
-->endfunction
```

Nested functions

```
-->function y=fno(x)
-->a=sin(x)
→  function y=sq(x), y=x^2,endfunction
-->y=sq(a)+1
-->endfunction

-->fno(%pi/4)
 ans  =
    1.5
```

Example : Recursive function

```
function [y]=factorial(x)
    if x==1 then  y=1
    else
        y=x*factorial(x-1)
    end
endfunction
```

Global and Local Variables

```
-->  global z
-->a=5;
-->function [y]=fg(x)
-->  y=x+1;  z=y^2;
-->endfunction

-->x=4;  fg(x)
 ans  =    5.
 -->z
 z  =      25.
```

- Functions can be invoked with less input or output parameters

```
-->b=5
-->function [x,y]=f(a,b)
-->x=a+b,y=a-b
-->endfunction
–>[x,y]=f(2)
y=-3
X=7
```

- Another example :-->plot2d(sin(x))

Multiple defined function

```
-->exec('d:\mdb\myscilab\mf0.sci')
-->function y=f(x)
-->  if x>0 then y=x+1,else y=x-1,end
-->endfunction
-->x=-4:4
 x  =
  - 4.  - 3.  - 2.  - 1.    0.    1.    2.    3.    4.
-->f(x)
  - 5.  - 4.  - 3.  - 2.  - 1.    0.    1.    2.    3.
//incorrect
```

```
-->x=-4:4;
-->[y]=feval(x,f)

-->[x;y]
 ans  =

  - 4.  - 3.  - 2.  - 1.    0.    1.    2.    3.    4.
  - 5.  - 4.  - 3.  - 2.  - 1.    2.    3.    4.    5.
```

Multiple defined functions

Y = x^2 ,1<= x

= sin(2*x) ,-1<x<1

= x / (x^3 + 2) ,x<=-1

Example : Multiple defined function

```
function [y]= mdf1(x)
   if x>=1 then
    y=x^2
    elseif  x>=-1&x<1 then
    y=sin(2*x)
    else
    y=x/(x^3+2)
    end
endfunction
```

Execution

```
-->getf('d:\mdb\myscilab\mdf1.sci')
-->x=[2   -.5   0.1];
 -->[y] = feval(x,mdf1)
 y  =
    4.
   0.0406504
   0.1986693
 -->[ x ;y' ]
ans  =
    2.       - 5.              0.1
    4.         0.0406504    0.1986693
```

Use of logical operators

- y = x^2 ,1<=x

  = x+10 ,-1<= x < 1

  = x ,x < -1

```
-->x=-3:3;

--> y=(1<=x).*(x.^2)+((-1<=x)&(x<1)).*(x+10)+...
-->      (x<-1).*x

-->[x ; y]

 ans  =

    - 3.  - 2.  - 1.    0.    1.    2.    3.

    - 3.  - 2.    9.    10.    1.    4.    9.
```

Derivative of polynomial

```
//derivative of a polynomial
function dp = diff(p)
  cfp=coeff(p)
  l=length(cfp)
  cfdp = cfp(2:l).*[1:l-1]
  dp=poly(cfdp,'x','c')
endfunction
```

Execution-derivative

```
-->getf('d:\mdb\myscilab\diff.sci')
-->p=poly([1 2 3],'x')
 p  =
                    2    3
  - 6 + 11x - 6x + x
 -->dp=diff(p)
 dp  =
                      2
    11 - 12x + 3x
```

Operator Overloading

```
-->1+5*%i< 2
        !--error 144
Undefined operation for the given operands.
check or define function %s_1_s for overloading.
-->function r =%s_1_s(a,b)
-->  r= real(a) < real(b)
-->endfunction
-->1+5*%i< 2
 ans  =
  T
```

More on functions

- If last argument of a function definition is named varargin, then the function can be called with more than N arguments.
- In a function input argument can be a function

  function [y]= regfl(a, b, f, n)
- Introducing a pause command permits debugging of Scilab function

  Execution of function is resumed by 'return' or ' resume' command

Thank you