Welcome to the spoken tutorial on iterative calculations using Scilab.

I am using scilab version 5.2 in Mac operating system , but these calculations should work in other versions and also in Scilab that runs in linux and windows.

I will use the code available in the file iteration.sce. I have opened this file using the Scilab editor, which I plan to use only as an editor.

Let us create a vector using the colon operator.

```
i = 1:5
```

creates a vector from 1 to 5, in increments of 1.

In this command,

```
i = 1:2:5,
```

we see that the middle argument of 2 indicates the increment. 1 is the first argument where the vector starts. i cannot go beyond 5. It can be equal to 5, however.

Note that if the ending argument changes to 6 the result remains the same.

```
i = 1:2:6
```

It is not difficult to explain this behaviour. Can you think for a moment why this happens?

We will now demonstrate the use of the for statement to perform iterative calculations.

```
for i = 1:2:7
  disp(i)
end
```

This code prints out i, as we go through the loop. The display is due to the command disp - the passed argument is displayed. Recall that the for loop is used for integer values. In this case, four integer values, namely, 1, 3, 5 and 7 are displayed. The number of times the iterations take place is known as priori in for loops.

In the rest of this tutorial, we will stick to the default increment of 1. Let us begin with the loop that displays i equal to 1 to 5.

```
for i = 1:5
  disp(i)
end
```

We will modify this code by introducing the break statement.

```
for i = 1:5
  disp(i)
  if (i==2),
    break
  end
end
```

Note that i is displayed only up to 2. The iteration is not carried out till the last value of i, namely, 5.

When i is equal to 2, the if block is executed for the first time. The break command, however, terminates the loop. If we want to get out of a loop when some intermediate condition is satisfied, we can use the break statement.

Note that "i is equal to 2" statement uses the "equal to" sign twice. This is the standard way to compare the equality in programming languages. The result of this comparison statement is a boolean: true or false.

We will introduce the continue statement.

```
for i = 1:5
  if (i<=3) then
    continue
  else
    disp(i)
  end
end
```

This results in i getting displayed only for 4 and 5. For i less than or equal to 3, as given by the i<=3 statement, nothing happens.

The continue statement makes the program skip the rest of the loop. Unlike the break statement, however, it does not exit the loop. The parameter i is incremented and all the calculations of the loop are executed for this new i.

We take a small break and show how to get help for operators of the type <=. Let us type

```
help <=
```

This opens the scilab help browser. We see that the help is available under the option less. So now we type

```
help less
```

We see the required help instructions here.

The for statement in Scilab is more powerful than in programming languages. For example, let us perform a for loop over a vector:

```
v = [1 5 3];
for x = v
  disp(x)
end
```

This script displays all values of v. Until now we have been displaying only the variables. We can indeed display the result of a calculation as well. The following code displays the square of the numbers.

```
v = [1 5 3];
for x = v
  disp(x^2)
end
```

We have spent quite a bit of time explaining the for loop. Let us now move on to the while loops.

The while statement allows us to perform a loop when a boolean expression is true. At the beginning of the loop, if the expression is true, the statements in the body of the while loop are executed. If the program is written well, the expression becomes false and the loop is ended.

Now let us see an example for the while loop:

```
i = 0;
while(i<=5)
  i = i+1;
  disp(i)
end
```

The values of i, from 1 to 6 are displayed.

Break and continue statements inside the while loop work exactly as they did in the for loop, as we demonstrate using break:

```
i = 0;
while(i<=5)
  i = i+1;
  disp(i)
  if(i==3) then
    break
```

```
    end
  end
```

We can see that the moment i becomes equal to 3, the program exits the loop, thanks to the break statement.

You can also try the example for continue statement in while loop.

This brings us to the end of this spoken tutorial on iterative calculations using Scilab.

Spoken Tutorials are part of the Talk to a Teacher project, supported by the National Mission on Education through ICT. More information on the same is available at http://spoken-tutorial.org/NMEICT-Intro.

Thanks for joining. Goodbye.