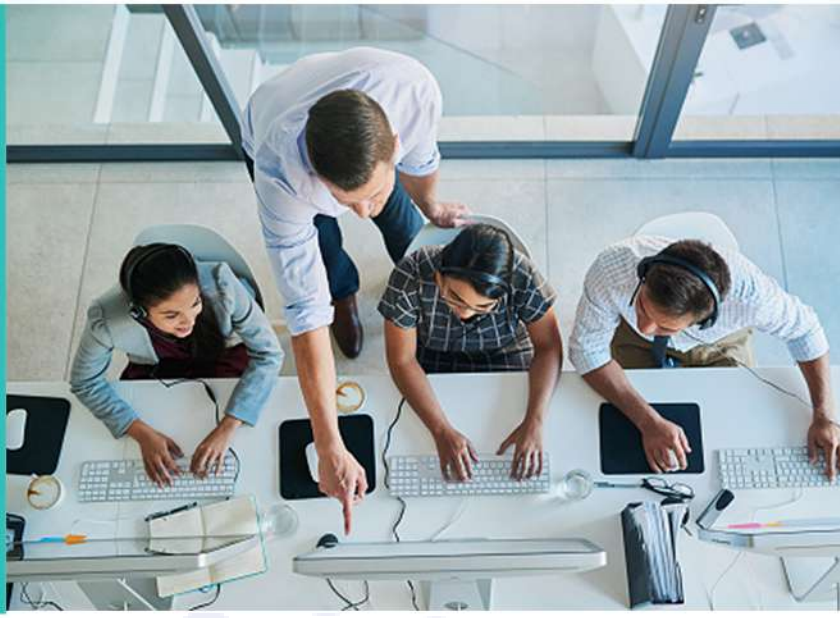
# Andhra Pradesh State Skill Development Corporation

Skill AP
APSSDC





# SciLab

## Programming in Scilab
## Iterations

# Programming in Scilab

## Iterations

The action or a process of iterating or repeating: such as a procedure in which repetition of a sequence of operations yields results successively closer to a desired results.

This is a continuous process

syntax:

variable name=initial value : final value

here colon(:) indicates the continous meaning.

or

variable name=intial value : difference : final value

examples:

```
    --> i=1:5
 i  =
   1.   2.   3.   4.   5.
--> u=1:2:10
 u  =
   1.   3.   5.   7.   9.
--> m=-2:3:15
 m  =
  -2.   1.   4.   7.   10.   13.
```

Examples of Iterations using conditional branching:

Exam1:

```
for i=1:2:10
   disp(i)
end
```

output:

```
--> exec('C:\Users\DELL\w1.sci', -1)
```

1.

3.

5.

7.

9.

Exam2:

```
for i=1:6
   disp(i)
end
```

output:

```
--> exec('C:\Users\DELL\w2.sci', -1)
```

1.

2.

3.

4.

5.

6.

## for loops

A loop is a facility to repeat a set of statements as per specified conditions. Scilab gives you a choice of two loops, viz, for and while. The for loop is frequently used, usually where the loop is to be repeated for a fixed number of times. The while loop keeps repeating an action until an associated test becomes false. This is useful when the programmer does not know in advance how many times the loop will be repeated.

A for loop statement has the following structure:

for(starting value; stepping value; ending value)

Statements to be repeated;---->? Body of loop

end

Here is an example: The following loop will print the numbers from 1 to 10:

```
for i=1:10   // i is known as loop index
disp( i )          // This is repeated 10 times taking i as 1,2,3…….10
end
```

The for loop consists of the keyword for followed by ( ) followed by any number of statements ending with “end”. After the keyword “for”, there are three parts separated by colons. The first part defines the starting value. The second part is this stepping value and the last part is the ending value.

The stepping value can be skipped, If it is one. The above loop can also be written as for i=1:50

Examples below indicate the different possible uses of for loops through printing of selected sequences of numbers.

**Numbers from 1 to 50:**

```
printf(“numbers from 1 to 50 are:\n”);
For i=1:50;printf(“%d\n”,i);end
```

**Numbers from -5 to 15**

```
printf(‘numbers -5 to 15 is\n”);
For i=5:15;printf(“%d\n”,i);end
```

**Odd numbers from 1 to 50**

```
printf(“odd numbers 1 to 50 are:\n”)
For i=1:50;
m=modulo(i,2);
if(m<>0)then printf(“%d\n”,i);
```

end

**Even numbers from 1 to 50**

printf("Even numbers from 1 to 50 are:\n");

for i=1:50;

m=modulo(i,2);if(m==0)then ; printf("%d\n",i); end

**Multiple of 3, from 1 to 25**

printf("Multiple of 3, from 1 to 25 are:\n");

For i=1:25;

m=modulo(i,3);

if(m==0)then ; printf("%d\n",i); end

end

**Multiple of 3 and also 5, from 1 to 25**

printf("Multiple of 3 and 5 from 1 to 25 are:\n");

for i=1:25;

m=modulo(i,3);

n=modulo(i,5);

if((m==0) & (n==0)) then ; printf("%d\n",i);

end