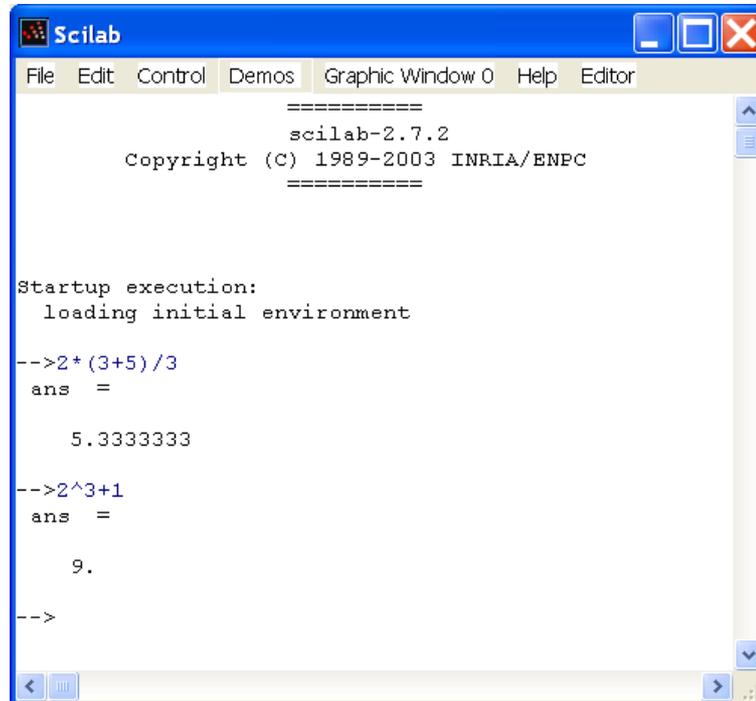


## Introducción a Scilab

1. Scilab es básicamente una calculadora programable que podemos utilizar para realizar cálculos sencillos. Las operaciones básicas realizables son: suma (+), resta (-), producto (\*), división (/) y potenciación (^):



```
Scilab
File Edit Control Demos Graphic Window 0 Help Editor
=====
scilab-2.7.2
Copyright (C) 1989-2003 INRIA/ENPC
=====

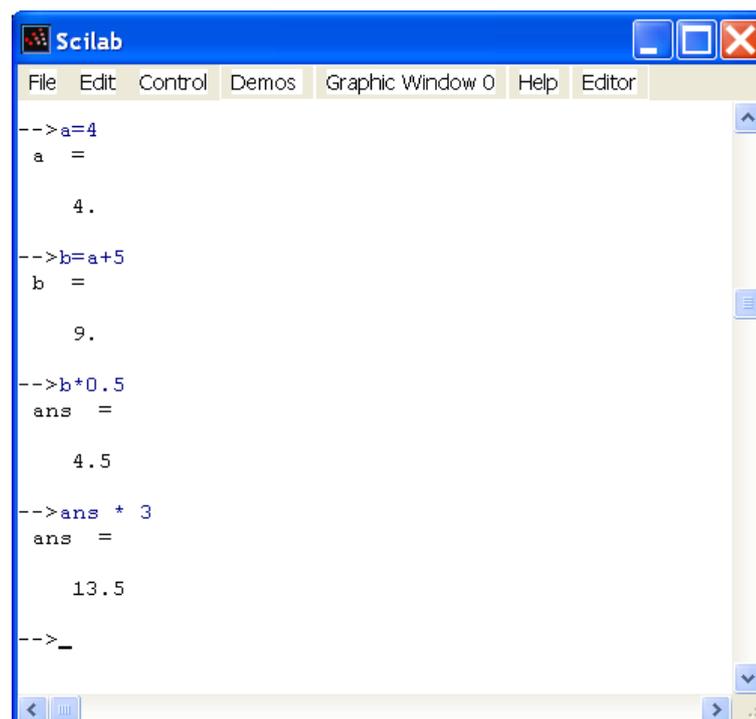
Startup execution:
loading initial environment

-->2*(3+5)/3
ans =
    5.3333333

-->2^3+1
ans =
    9.

-->
```

2. Podemos hacer uso de las variables. La asignación de valores a variables se realiza mediante el símbolo “=”. Una vez se le haya asignado un valor a una variable, podemos utilizarla en expresiones posteriores. La variable *ans* contiene el valor de la última expresión evaluada:



```
Scilab
File Edit Control Demos Graphic Window 0 Help Editor

-->a=4
a =
    4.

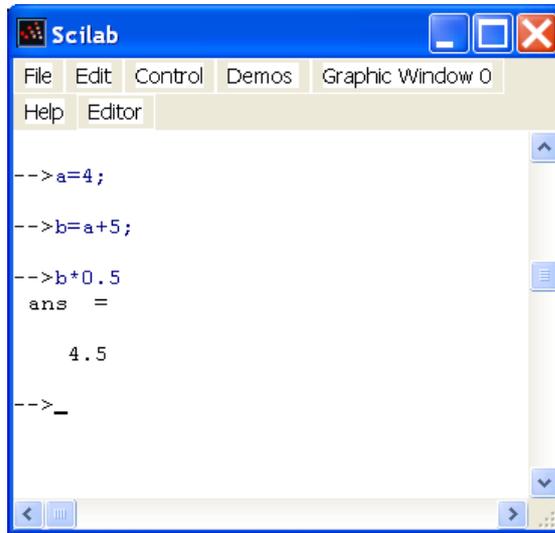
-->b=a+5
b =
    9.

-->b*0.5
ans =
    4.5

-->ans + 3
ans =
    13.5

-->_
```

3. La evaluación de una expresión genera un *eco*. Dicho eco puede eliminarse si la expresión finaliza con “;”. Cuando programemos los algoritmos para Scilab, de no eliminar dicho eco, aparecerán por pantalla todos los cálculos intermedios que realiza nuestro algoritmo.



```
Scilab
File Edit Control Demos Graphic Window 0
Help Editor

-->a=4;

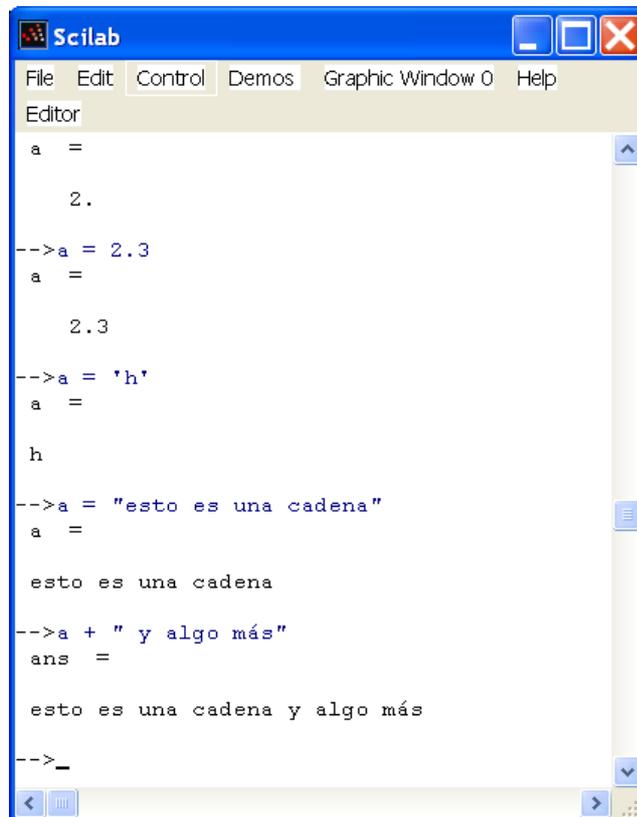
-->b=a+5;

-->b*0.5
ans =

    4.5

-->_
```

4. Las variables de Scilab no llevan asociado un único *tipo de dato*; pueden representar enteros, reales, complejos, caracteres y hasta cadenas (para el caso de cadenas las operaciones no tienen sentido, salvo la suma, que representa la concatenación):



```
Scilab
File Edit Control Demos Graphic Window 0 Help
Editor

a =

    2.

-->a = 2.3
a =

    2.3

-->a = 'h'
a =

    h

-->a = "esto es una cadena"
a =

    esto es una cadena

-->a + " y algo más"
ans =

    esto es una cadena y algo más

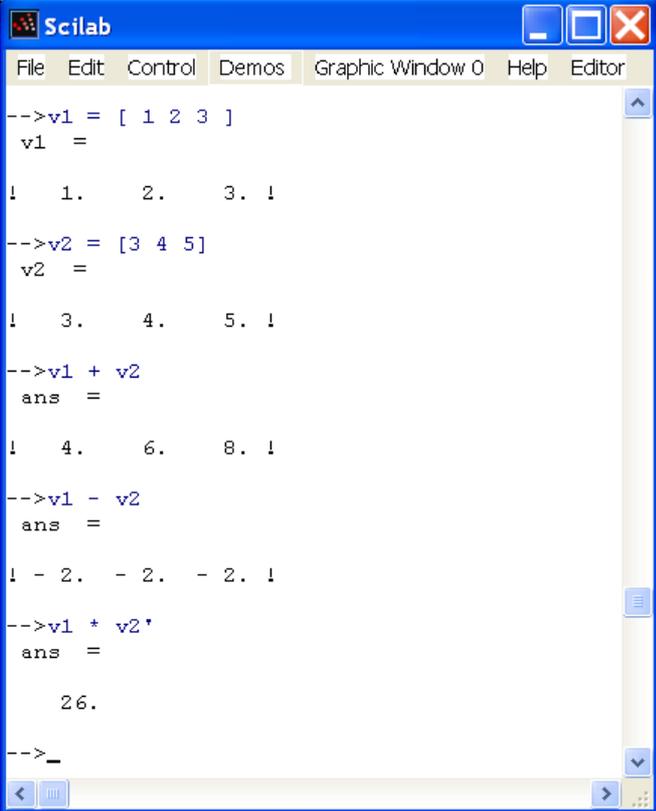
-->_
```

5. A pesar de que hasta ahora las variables han representado valores escalares, Scilab trata a las variables como matrices de dimensiones  $n \times m$ . Las variables

hasta ahora utilizadas equivalen a matrices de dimensión  $l \times l$ . Una matriz se representa mediante la expresión:

$$[ a_{11} a_{12} a_{13} \dots a_{1m} ; a_{21} a_{22} a_{23} \dots a_{2m} ; \dots ; a_{n1} a_{n2} a_{n3} \dots a_{nm} ]$$

Todas las operaciones anteriormente nombradas son aplicables a las matrices, eso si, teniendo en cuenta que estamos hablando de operaciones matriciales<sup>1</sup>. Además de los operadores previamente mencionados, existe el operador transposición ('):

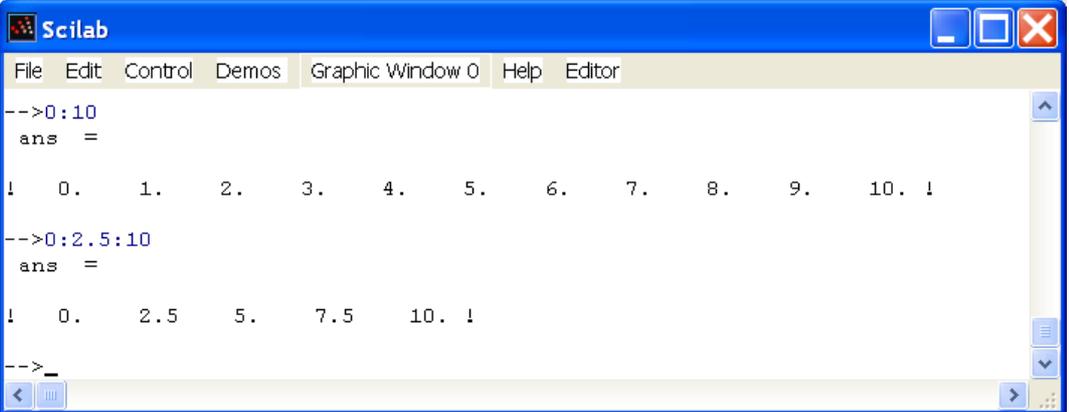


```

Scilab
File Edit Control Demos Graphic Window 0 Help Editor
-->v1 = [ 1 2 3 ]
v1 =
! 1. 2. 3. !
-->v2 = [3 4 5]
v2 =
! 3. 4. 5. !
-->v1 + v2
ans =
! 4. 6. 8. !
-->v1 - v2
ans =
! -2. -2. -2. !
-->v1 * v2'
ans =
26.
-->_

```

6. El operador ":" permite generar vectores cuyos valores forman una secuencia ascendente o descendente. La expresión *inicio:final* genera los números de *inicio* a *final* con intervalo 1. Es posible también generar una secuencia con un intervalo diferente mediante la expresión *inicio:intervalo:final*.



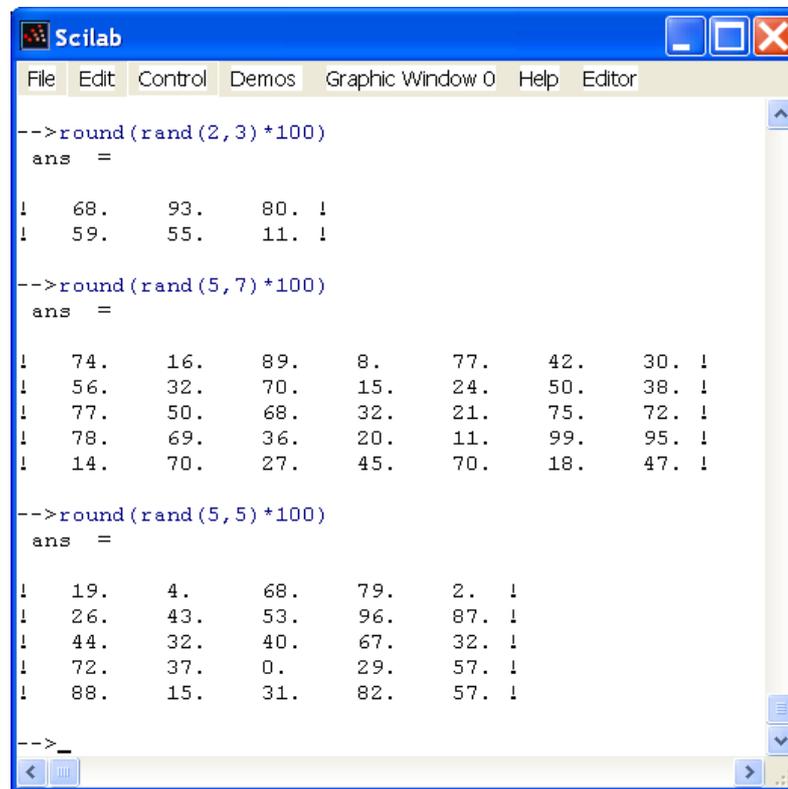
```

Scilab
File Edit Control Demos Graphic Window 0 Help Editor
-->0:10
ans =
! 0. 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. !
-->0:2.5:10
ans =
! 0. 2.5 5. 7.5 10. !
-->_

```

<sup>1</sup> Existen tres operadores adicionales: ".\*" "./" y ".^". Estos operadores realizan la operación "elemento a elemento" en vez de de forma matricial.

7. Dado que en adelante trataremos operaciones sobre matrices, y para no perder mucho tiempo en la generación de las mismas, veremos una curiosa forma de obtener matrices con valores enteros aleatorios. La función  $rand(n,m)$  genera matrices aleatorias de dimensión  $n \times m$  con valores aleatorios en el rango  $[0 1]$ , mientras que la función  $round(x)$  realiza el redondeo al valor entero más cercano. Podemos ya generar matrices de enteros automáticamente de la siguiente forma:



```
Scilab
File Edit Control Demos Graphic Window 0 Help Editor

-->round(rand(2,3)*100)
ans =

! 68. 93. 80. !
! 59. 55. 11. !

-->round(rand(5,7)*100)
ans =

! 74. 16. 89. 8. 77. 42. 30. !
! 56. 32. 70. 15. 24. 50. 38. !
! 77. 50. 68. 32. 21. 75. 72. !
! 78. 69. 36. 20. 11. 99. 95. !
! 14. 70. 27. 45. 70. 18. 47. !

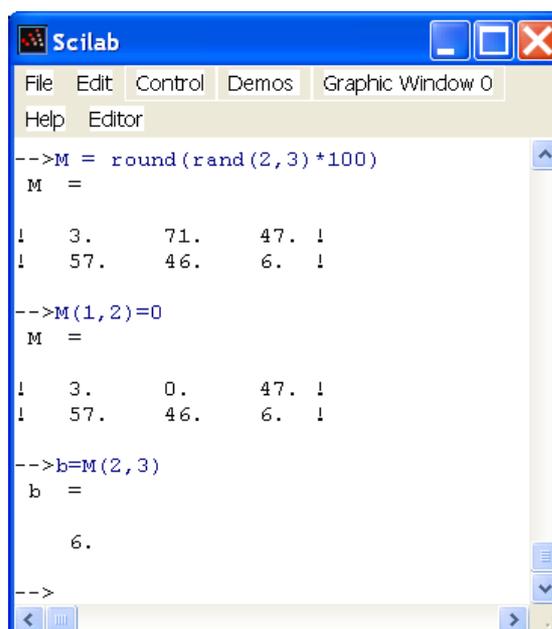
-->round(rand(5,5)*100)
ans =

! 19. 4. 68. 79. 2. !
! 26. 43. 53. 96. 87. !
! 44. 32. 40. 67. 32. !
! 72. 37. 0. 29. 57. !
! 88. 15. 31. 82. 57. !

-->
```

8. Es posible acceder a los elementos de una matriz tanto para usarlos como para modificarlos, para lo cual usaremos la expresión:

*Nombre\_vector(i) o Nombre\_matriz(i,j)*



```
Scilab
File Edit Control Demos Graphic Window 0
Help Editor

-->M = round(rand(2,3)*100)
M =

! 3. 71. 47. !
! 57. 46. 6. !

-->M(1,2)=0
M =

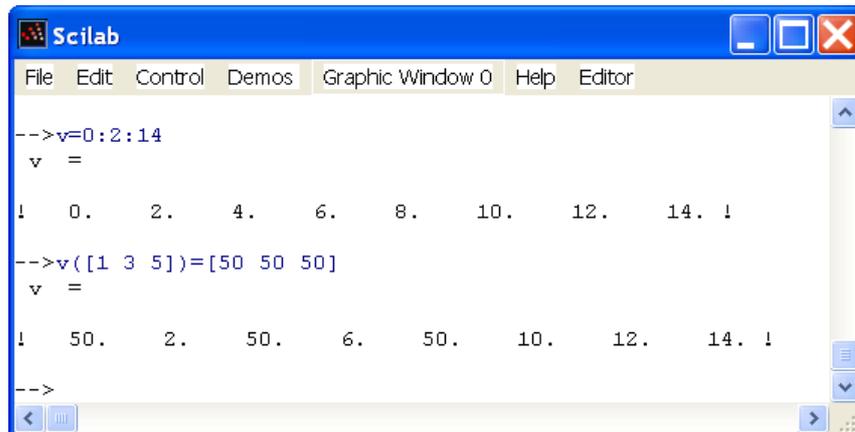
! 3. 0. 47. !
! 57. 46. 6. !

-->b=M(2,3)
b =

6.

-->
```

9. No estamos limitados a acceder a un solo elemento de una matriz; si reemplazamos los índices por vectores de índices, obtendremos un conjunto de elementos de la matriz:

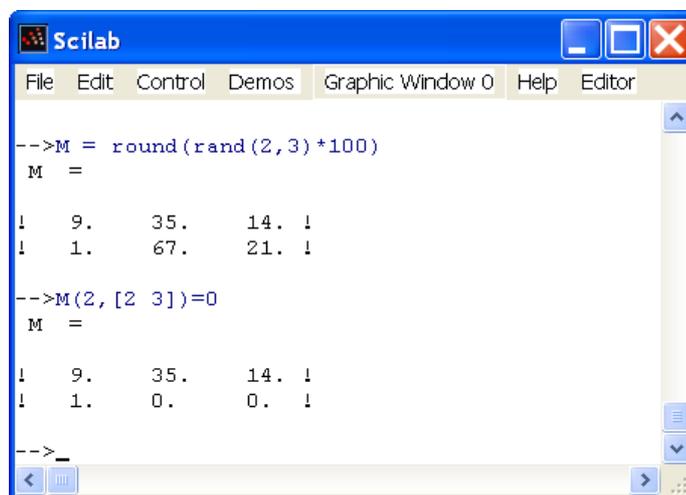


```
Scilab
File Edit Control Demos Graphic Window 0 Help Editor

-->v=0:2:14
v =
! 0. 2. 4. 6. 8. 10. 12. 14. !

-->v([1 3 5])=[50 50 50]
v =
! 50. 2. 50. 6. 50. 10. 12. 14. !

-->
```



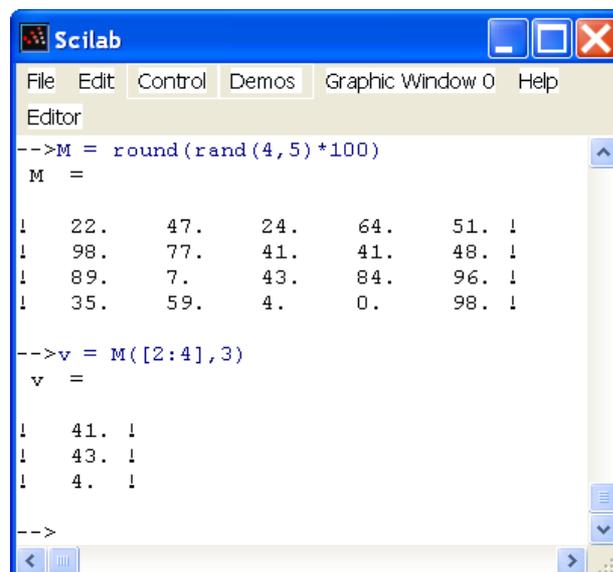
```
Scilab
File Edit Control Demos Graphic Window 0 Help Editor

-->M = round(rand(2,3)*100)
M =
! 9. 35. 14. !
! 1. 67. 21. !

-->M(2,[2 3])=0
M =
! 9. 35. 14. !
! 1. 0. 0. !

-->
```

10. El uso del operador “:” es muy adecuado cuando se desea acceder a valores contiguos de una matriz, ya que no será necesario indicar todos los índices. De hecho, cuando no se indica *inicio* y *final*, dicho operador genera todo el rango posible de índices:



```
Scilab
File Edit Control Demos Graphic Window 0 Help
Editor

-->M = round(rand(4,5)*100)
M =
! 22. 47. 24. 64. 51. !
! 98. 77. 41. 41. 48. !
! 89. 7. 43. 84. 96. !
! 35. 59. 4. 0. 98. !

-->v = M([2:4],3)
v =
! 41. !
! 43. !
! 4. !

-->
```

```

Scilab
File Edit Control Demos Graphic Window 0 Help Editor
-->M = round(rand(5,6)*100)
M =
! 79. 93. 31. 38. 78. 96. !
! 86. 41. 94. 9. 22. 45. !
! 20. 60. 24. 7. 34. 0. !
! 71. 55. 88. 73. 43. 1. !
! 52. 86. 49. 78. 63. 97. !

-->M(4,:) = 0
M =
! 79. 93. 31. 38. 78. 96. !
! 86. 41. 94. 9. 22. 45. !
! 20. 60. 24. 7. 34. 0. !
! 0. 0. 0. 0. 0. 0. !
! 52. 86. 49. 78. 63. 97. !

-->

```

11. Existe también el *conjunto vacío*, representado por el símbolo  $[\ ]$ . Cuando a un valor (o conjunto de valores) de un vector se le asigna el *conjunto vacío*, dicho elemento es eliminado del vector. De forma similar, cuando a una fila o columna (o conjunto de filas o columnas) de una matriz se le asigna el *conjunto vacío*, la fila o columna es eliminada:

```

Scilab
File Edit Control Demos Graphic Window 0 Help Editor
-->v = round(rand(1,10)*100)
v =
! 74. 29. 40. 98. 1. 70. 89. 65. 41. 86. !

-->v = round(rand(1,7)*100)
v =
! 24. 48. 98. 45. 23. 22. 80. !

-->v(2) = []
v =
! 24. 98. 45. 23. 22. 80. !

-->v(4:6) = []
v =
! 24. 98. 45. !

-->

```

```

Scilab
File Edit Control Demos Graphic Window 0 Help Editor
-->M = round(rand(5,6)*100)
M =
| 78.  90.  53.  78.  59.  78. |
| 48.  95.  29.  64.  35.  28. |
| 78.  70.  70.  7.  44.  69. |
| 15.  96.  94.  82.  88.  72. |
| 77.  5.  11.  35.  76.  37. |

-->M(2,:)=[]
M =
| 78.  90.  53.  78.  59.  78. |
| 78.  70.  70.  7.  44.  69. |
| 15.  96.  94.  82.  88.  72. |
| 77.  5.  11.  35.  76.  37. |

-->M(:,[4 5])=[]
M =
| 78.  90.  53.  78. |
| 78.  70.  70.  69. |
| 15.  96.  94.  72. |
| 77.  5.  11.  37. |

-->

```

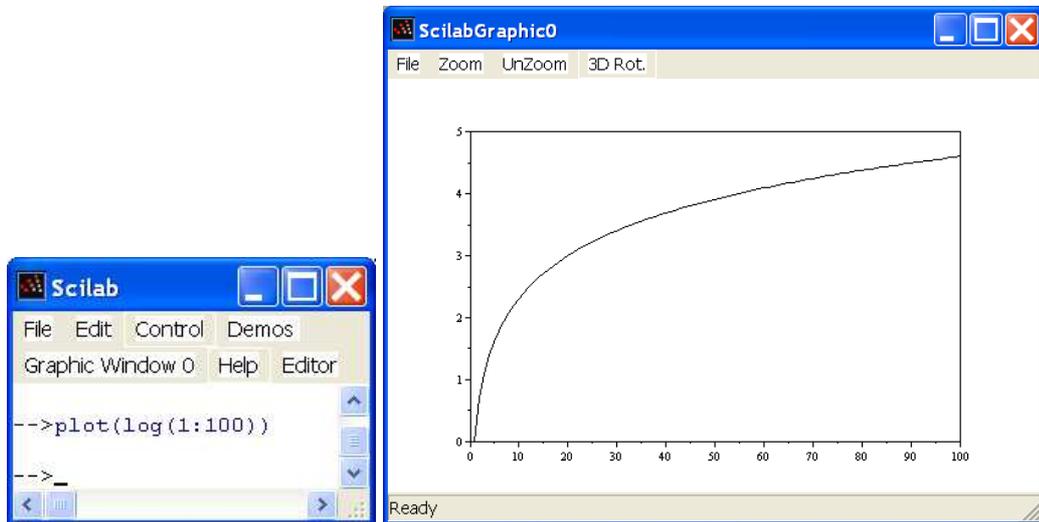
12. Existen infinidad de funciones matemáticas que podemos usar en nuestras expresiones. Podemos obtener un listado de las mismas pinchando sobre el botón de menú *help* y seleccionando *Elementary Functions*:

```

Elementary Functions
Previous Next Chapters Reload Options... Copy Quit
abs - absolute value, magnitude
acos - element wise cosine inverse
acosh - hyperbolic cosine inverse
acoshm - matrix hyperbolic inverse cosine
acosm - matrix wise cosine inverse
addf - symbolic addition
adj2sp - converts adjacency form into sparse matrix.
amell - Jacobi's am function
and (&) - logical and
asin - sine inverse
asinh - hyperbolic sine inverse
asinhm - matrix hyperbolic inverse sine
asinm - matrix wise sine inverse
atan - 2-quadrant and 4-quadrant inverse tangent
atanh - hyperbolic tangent inverse
atanhm - matrix hyperbolic tangent inverse
atanm - square matrix tangent inverse
bessefi - Modified I sub ALPHA Bessel functions of the first kind.
bessej - Modified J sub ALPHA Bessel functions of the first kind.
bessek - Modified K sub ALPHA Bessel functions of the second kind.
bessely - Modified Y sub ALPHA Bessel functions of the second kind.
binomial - binomial distribution probabilities
bloc2exp - block-diagram to symbolic expression
bloc2ss - block-diagram to state-space conversion
calerf - computes error functions.
ceil - rounding un

```

13. Para finalizar, estudiaremos brevemente una de las funciones de la librería gráfica, la función `plot()`. Esta función nos permite representar vectores y matrices de una forma gráfica. `plot()` no puede representar de una forma simbólica funciones, no obstante podemos calcular un vector que sea representativo del rango de la función que deseemos representar. Si la función es llamada con un solo argumento, dicho argumento es dibujado en el *eje y*, mientras que si es llamada con dos argumentos, toma el primero como *eje x* y el segundo como *eje y*:



## Como práctica

1. Escribir el código en Scilab necesario para generar una gráfica de la función  $\text{seno}(x)$  en el intervalo  $[0, 2\pi]$  (el valor  $\pi$  puede obtenerse mediante la expresión `%pi`). Asegurarse de que la escala del *eje x* es la correcta.

