```
%esegue un’operazione
4*5/sqrt(3)
ans =
   11.5470
%definisce una variabile
a=2
a =
     2
3*a-4
ans =
     2
%elenca le variabili nel WorkSpace (nome, tipo, dimensione,...)
whos
  Name      Size             Bytes  Class     Attributes

  a         1x1                  8  double
  ans       1x1                  8  double

%definisce un vettore (array)
v = [3 ;5]
v =
     3
     5
%definisce una matrice (doppio array!)
A = [ 1 , 2 ; 5 6 ]
A =
     1     2
     5     6
whos
  Name      Size             Bytes  Class     Attributes

  A         2x2                 32  double
  a         1x1                  8  double
  ans       1x1                  8  double
  v         2x1                 16  double

% * esegue il prodotto righe x colonne
A*v
ans =
    13
    45
w = A*v
w =
    13
    45
% La dimensione di w Ã¨ stata scelta automaticamente
whos
  Name      Size             Bytes  Class     Attributes

  A         2x2                 32  double
  a         1x1                  8  double
  ans       2x1                 16  double
  v         2x1                 16  double
  w         2x1                 16  double

% fare riferimento ad un elemento di un vettore
w(2)
ans =
    45
w(3) = 2
w =
    13
    45
     2
%MatLab ha allungato il vettore
w(10)=1
```

```
w =
    13
    45
     2
     0
     0
     0
     0
     0
     0
     1
% se necessario, riempiendo con 0
%Creare matrici e vettori di 0 o 1 (utile anche per preallocare la memoria)
z = zeros(3,4)
z =
     0     0     0     0
     0     0     0     0
     0     0     0     0
z = ones(3,1)
z =
     1
     1
     1
%ottenere help (nella Command Window)
help zeros
 ZEROS  Zeros array.
    ZEROS(N) is an N-by-N matrix of zeros.

    ZEROS(M,N) or ZEROS([M,N]) is an M-by-N matrix of zeros.

    ZEROS(M,N,P,...) or ZEROS([M N P ...]) is an M-by-N-by-P-by-... array of
    zeros.

    ZEROS(SIZE(A)) is the same size as A and all zeros.

    ZEROS with no arguments is the scalar 0.

    ZEROS(M,N,...,CLASSNAME) or ZEROS([M,N,...],CLASSNAME) is an
    M-by-N-by-... array of zeros of class CLASSNAME.

    Note: The size inputs M, N, and P... should be nonnegative integers.
    Negative integers are treated as 0.

    Example:
       x = zeros(2,3,'int8');

    See also <a href="matlab:help eye">eye</a>, <a href="matlab:help ones">ones</a>.

    Reference page in Help browser
       <a href="matlab:doc zeros">doc zeros</a>

%ottenere help (in una finestra separata, con piÃ¹ dettagli)
doc plot
%disegnare un grafico
plot(w)
% w sono le "ordinate"; ascisse automaticamente impostate a 1:length(w)
x=[1,3,4,5,6,8,9,10,15,17]
x =
  Columns 1 through 6
     1     3     4     5     6     8
  Columns 7 through 10
     9    10    15    17
plot(x,w)
%specificare sia ascisse che ordinate
%cambiare formato linea
plot(x,w,'r')
```

```
plot(x,w,'r*')
plot(x,w,'r*--')
doc LineSpec
%creare un vettore con elementi equispaziati (1)
% indicando (inizio, fine, numelementi)
x = linspace(0 , 20 , 11)
x =
  Columns 1 through 6
     0     2     4     6     8    10
  Columns 7 through 11
    12    14    16    18    20
%applicare una funzione ad un vettore
y=sin(x)
y =
  Columns 1 through 3
         0    0.9093   -0.7568
  Columns 4 through 6
   -0.2794    0.9894   -0.5440
  Columns 7 through 9
   -0.5366    0.9906   -0.2879
  Columns 10 through 11
   -0.7510    0.9129
plot(x,y)
% i (pochi) punti sono stati uniti con rette
x=linspace(0,2*pi,50);
y=sin(x);
plot(x,y)
% grafico con maggiore risoluzione
%creare un vettore con elementi equispaziati (2)
% scegliendo inizio:passo:fine
x = 0 : 0.1 : 2*pi;
whos x
  Name      Size            Bytes  Class     Attributes

  x         1x63              504  double

%se il passo Ã¨ 1, posso sottintenderlo
1:5
ans =
     1     2     3     4     5
3:7
ans =
     3     4     5     6     7
%fare riferimento ad un sotto-array
v=linspace(0,10,5)
v =
  Columns 1 through 3
         0    2.5000    5.0000
  Columns 4 through 5
    7.5000   10.0000
v(3)
ans =
     5
v([3,5])
ans =
     5    10
v([3,5]) = [3,7]
v =
  Columns 1 through 3
         0    2.5000    3.0000
  Columns 4 through 5
    7.5000    7.0000
v([2,4]) = 1
v =
     0     1     3     1     7
%Sovrapporre grafici in una stessa finestra
```

```
x=linspace(0,pi,100);
plot(x,cos(x),'b')
hold on
plot(x,sin(x),'r')
plot(x,exp(-x),'r')
plot(x,exp(-x),'g')
legend('cos(x)','sin(x)','exp(-x)')
hold off
plot(x,2*x,'k')
%attivare la griglia di riferimento degli assi
grid on
grid off
%Eseguire operazioni aritmetiche "element-wise"
y = x^2
??? Error using ==> mpower
Matrix must be square.

y = x * x
??? Error using ==> mtimes
Inner matrix dimensions must agree.

%Errore perchÃ© di default esegue prodotto riga X colonna
x=-1:3
x =
    -1     0     1     2     3
x.*x
ans =
     1     0     1     4     9
x.^ 2
ans =
     1     0     1     4     9
x.^ x
ans =
    -1     1     1     4    27
x ./ x
ans =
     1   NaN     1     1     1
% 0/0 restituisce NaN
2/0
ans =
   Inf
(-3)/0
ans =
  -Inf
4/ans
ans =
     0
x=1e308
x =
  1.0000e+308
200*x
ans =
   Inf
%overflow appena supero realmax
realmax
ans =
  1.7977e+308
s=realmin
s =
  2.2251e-308
%sotto realmin, modifica interna della rappresentazione per poter
% scendere un po' di piÃ¹ verso lo zero (con minori cifre significative)
s/200
ans =
  1.1125e-310
%ancora un po'...
```

```
s/20000
ans =
  1.1125e-312
% ma non troppo!
s/1e16
ans =
     0
%underflow!
x=linspace(-1,1,50);
plot(x , x .* x)
%Rappresentazione di un polinomio mediante l' "elenco" dei suoi coeff.
doc polyval

P = [ 2 , 0 , 1]
P =
     2     0     1
plot(x, polyval(P,x))
clear all
%Creiamo il file prova.m
edit prova.m

%===== prova.m =====
%sotto realmin, modifica interna della rappresentazione per poter
% scendere un po' di piÃ¹ verso lo zero (con minori cifre significative)
s/200
ans =
  1.1125e-310
%ancora un po'...
s/20000
ans =
  1.1125e-312
% ma non troppo!
s/1e16
ans =
     0
%underflow!
x=linspace(-1,1,50);
plot(x , x .* x)
%Rappresentazione di un polinomio mediante l' "elenco" dei suoi coeff.
doc polyval

P = [ 2 , 0 , 1]
P =
     2     0     1
plot(x, polyval(P,x))
clear all
%Creiamo il file prova.m
edit prova.m

%===== prova.m =====
a=4;
b=3;
a-b
%====================

%Cancelliamo tutte le variabili nel Workspace
clear all
whos
prova
ans =
     1
whos
  Name      Size            Bytes  Class     Attributes

  a         1x1                 8  double
  ans       1x1                 8  double
```

```
  b         1x1                 8  double

%il programma ha creato le variabili nel WorkSpace!
% creiamo una function provafun.m

%===== provafun.m =====
function provafun(a);
b=3;
a-b
%======================

clear all
provafun(2)
ans =
    -1
whos
%la function usa uno spazio di memoria "privato"
%Non abbiamo a disposizione i risultati
% a meno che non specifichiamo variabili in uscita

%===== provafun2.m =====
function [r,s]=provafun2(a,b);
r=a-b;
s=a+b;
%=======================

clear all
provafun2(2,1)
ans =
     1
whos
  Name      Size            Bytes  Class     Attributes

  ans       1x1                 8  double

clear all
x=provafun2(2,1)
x =
     1
whos
  Name      Size            Bytes  Class     Attributes

  x         1x1                 8  double

%Per salvare tutte la variabili in uscita
% assegno output a un "vettore di variabili"
[x,y]=provafun2(2,1)
x =
     1
y =
     3

% Calcoliamo la funzione sqrt(x^2+1)-x
%===== cancellazione.m =====
xx = logspace(0, 10,20);
err = zeros(1,length(xx)); %length = lunghezza vettore
for k = 1:length(xx) %sintassi: for variabile=vettore dei valori
    x = xx(k);
    E = 1/(sqrt(x^2+1)+x);
    F = sqrt(x^2+1)-x;
    err(k) = abs((E-F)/E);
end
loglog(xx,err,'ro-')
%============================
% La formula "F" soffre per errori di perdita di cifre significative
```

```
% Ottimizziamo eliminando il ciclo for
%===== cancellazione2.m =====
xx = logspace(0, 10,20);
% Sostituisco il ciclo for con le operazioni "element-wise"
% non serve piÃ¹ preallocare err
E = 1./(sqrt(xx.^2+1)+xx);
F = sqrt(xx.^2+1)-xx;
err = abs((E-F)./E);
loglog(xx,err,'ro-')
%============================

% e ancora evitando di ripetere la chiamata a sqrt
%===== cancellazione2.m =====
xx = logspace(0, 10,20);
S = sqrt(xx.^2+1);
E = 1./(S+xx);
F = S-xx;
err = abs((E-F)./E);
loglog(xx,err,'ro-')
%============================

%Confronto di due modi di calcolare 2*pi con il metodo di Archimede
% Solo evitando gli errori di cancellazione si ottengono risultati
% significativi ed in questo caso si raggiunge anche la precisione
% macchina

%===== archimede.m =====
% calcola 30 elementi della successione p(k)
%  dei perimetri dei poligoni inscritti
%  con 2^k lati
% 4 lati, perimetro 4*sqrt(2)
k = 2; % N lati = 2^k
p(2) = 4*sqrt(2);
P(2) = 4*sqrt(2);

k_max = 30;
K=k:k_max;

for k=K(2:end)
    l = p(k-1) / 2^(k-1); %ricava il lato dal perimetro
    l1 = sqrt(2-sqrt(4-l^2)); %lato per il poligono successivo
    p(k) = 2^k * l1; %perimetro poligono successivo

    l = P(k-1) / 2^(k-1);
    l1 = l / sqrt(2+sqrt(4-l^2)); %lato per il poligono successivo
    P(k) = 2^k * l1; %perimetro poligono successivo
end
figure(1)
plot(K,p(K) , 'bo-' , K , P(K), 'rv-')
title 'p(k) dovrebbe convergere a 2\pi dal basso'
figure(2)
semilogy(K , abs(2*pi - p(K)) , 'o-' , K , abs(2*pi-P(K)), 'rv-')
title 'La precisione si arresta a 10^{-8} con la prima formula'
%======================

ESERCIZI:
* come devo inizializzare p(1) nel codice precedente per poter
iniziare il calcolo con k=1?
```