Elementi di Matlab

Sommario

- Introduzione
- Variabili
- Manipolazione di elementi
- Creazione di vettori/matrici
- Operazioni elementari
- Funzioni vettorizzate
- Funzioni predefinite e comandi utili
- Gli M-file

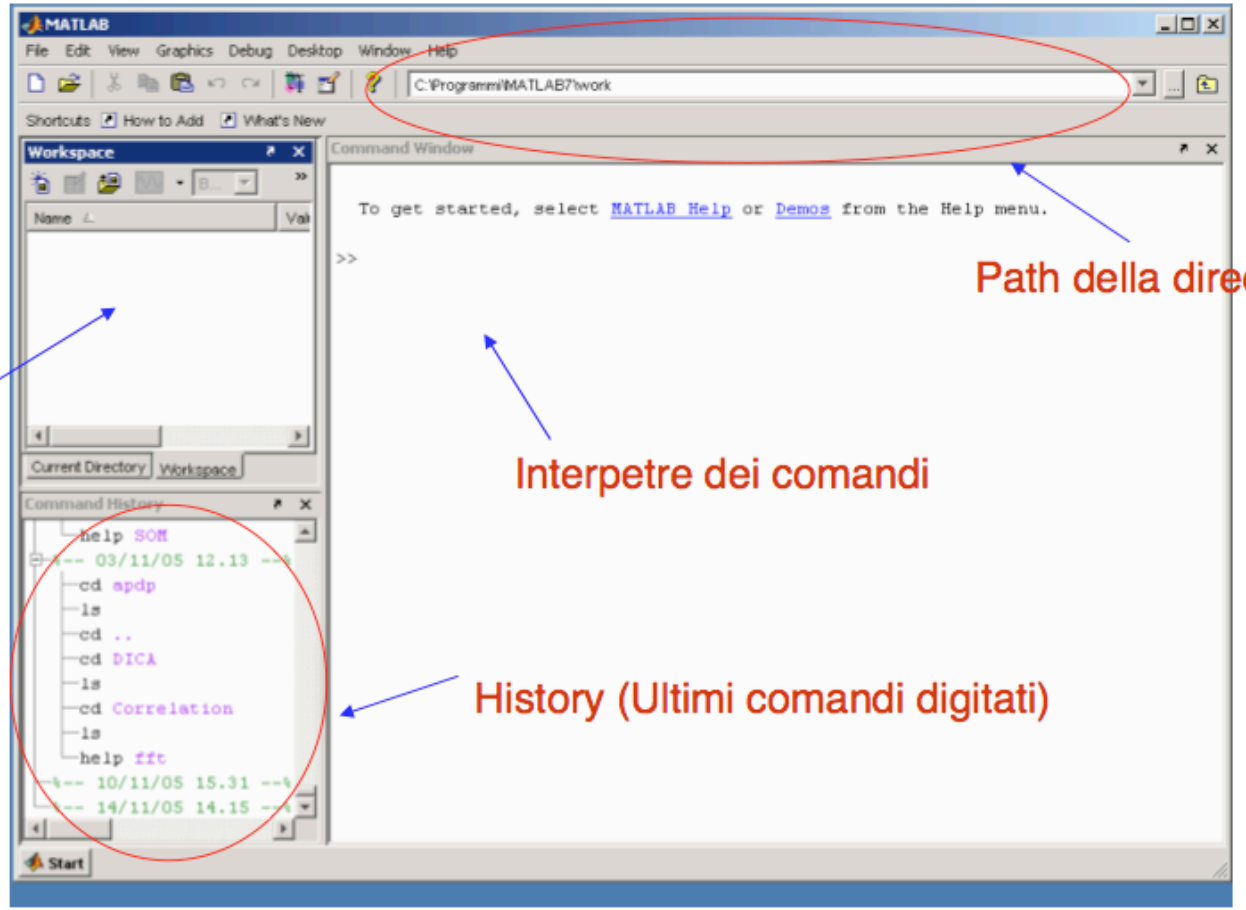
Che cosa è Matlab

- Matlab è un linguaggio ad alto livello per il calcolo scientifico
- Integra un ambiente per il calcolo, la visualizzazione la programmazione
- Matlab è l'acronimo di MatrixLaboratory
- E' un programma che lavora prevalentemente con matrici: qualunque oggetto introdotto è manipolato come se fosse una matrice
- Quindi l'unità fondamentale è il vettore e/o la matrice
- E' un software numerico, non simbolico

Matlab è case sensitive!

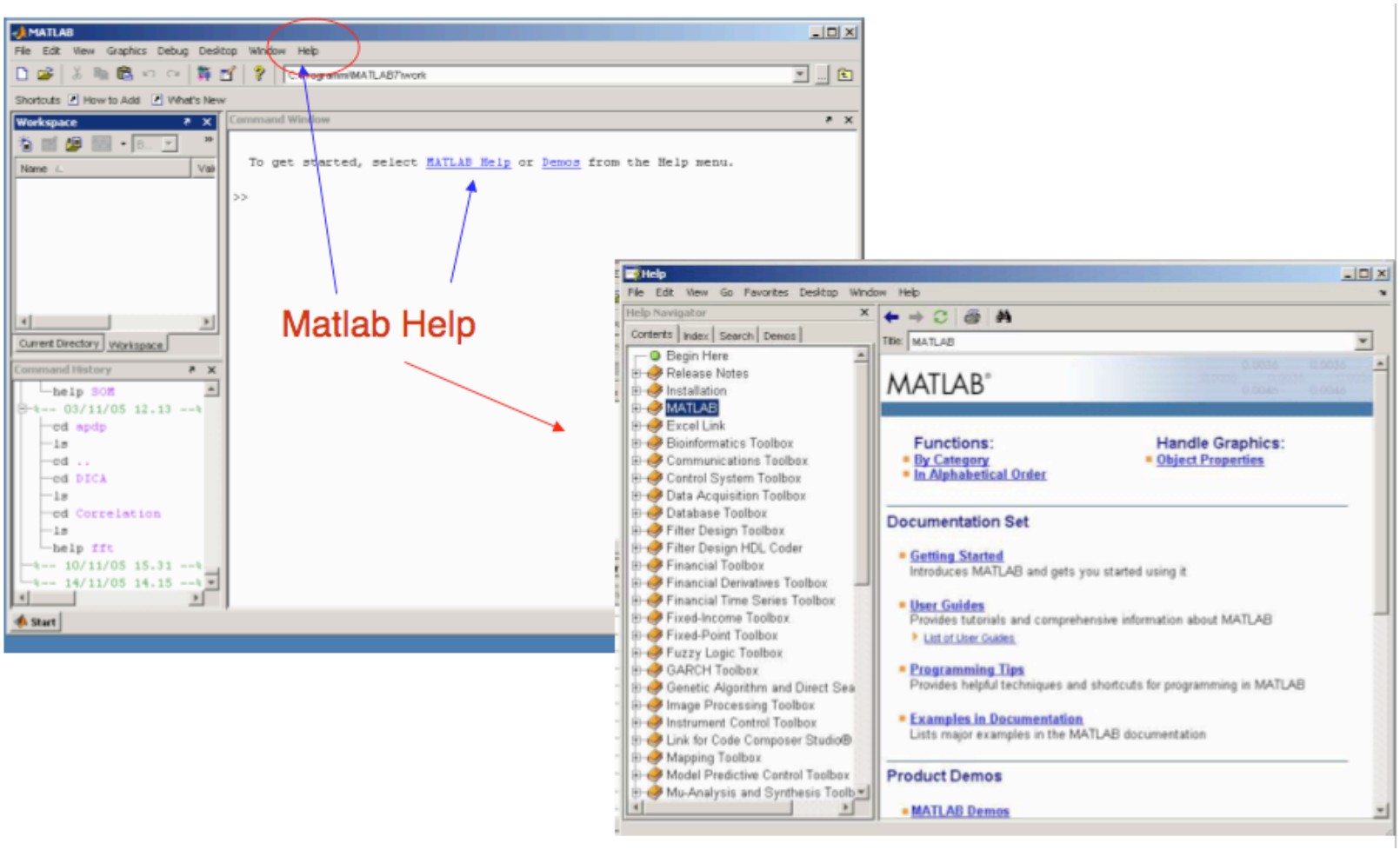
- Contiene numerose funzioni built-in
- Sono disponibili toolbox con raccolte di funzioni aggiuntive

Il layout

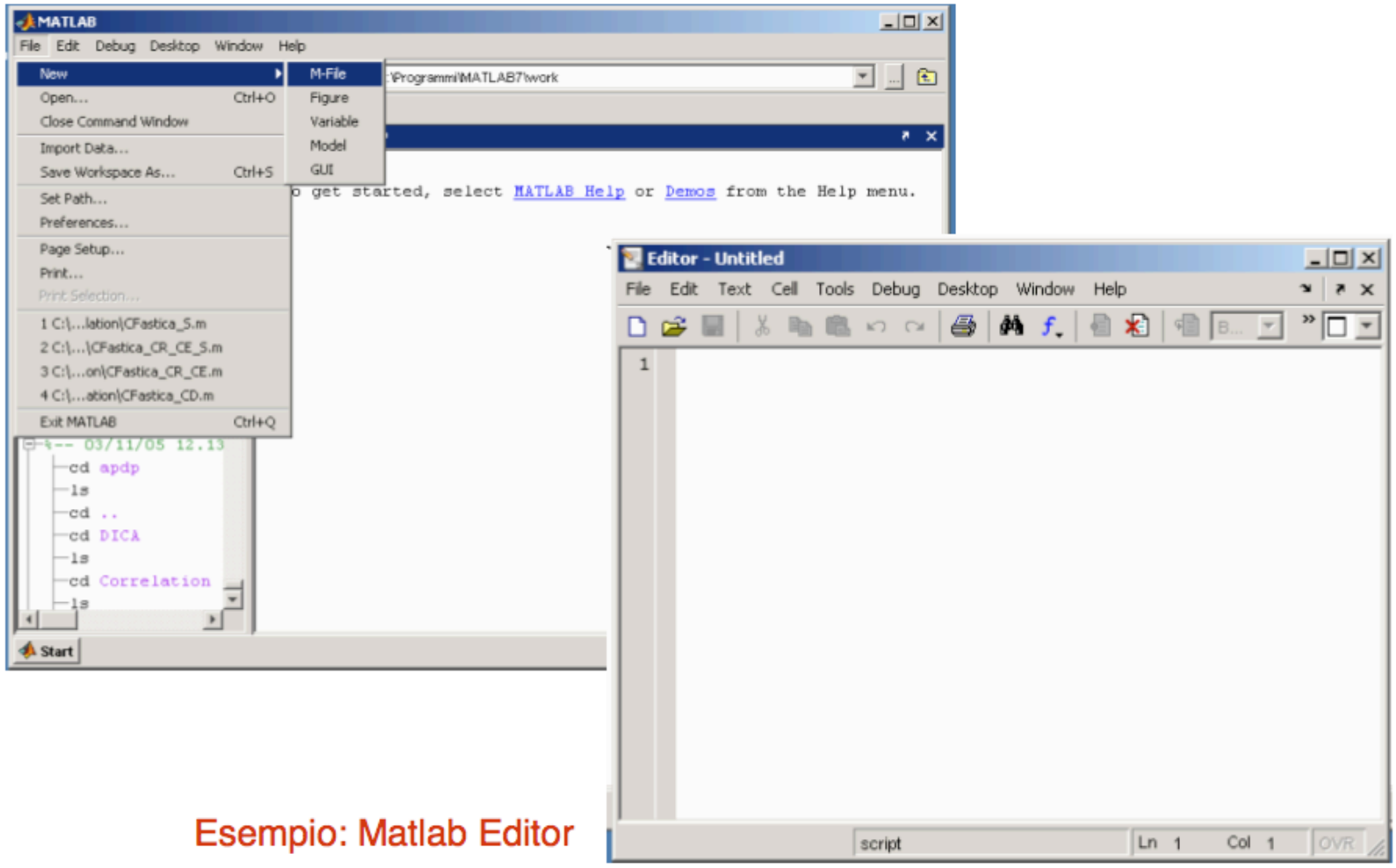


Conten uto del Work- space

Help


Matlab Help
To get started, select MATLAB Help or Demos from the Help menu.
Functions:
By Category
In Alphabetical Order
Handle Graphics:
Object Properties
Documentation Set
Getting Started
Introduces MATLAB and gets you started using it
User Guides
Provides tutorials and comprehensive information about MATLAB
Programming Tips
Provides helpful techniques and shortcuts for programming in MATLAB
Examples in Documentation
Lists major examples in the MATLAB documentation
Product Demos
MATLAB Demos

Creare un nuovo file



Esempio: Matlab Editor

Linea di comando e script

- Matlab è basato su comandi in linea
- I comandi possono anche essere letti da un file di testo
- Gli script sono file con estensione .m
- Matlab lavora con alcuni tipi di dati
  - La matrice n-dimensionale di numeri reali, complessi, caratteri o strutture più complesse
  - I numeri interi e reali sono in doppia precisione
- Variabili
  - Sono case sensitive
  - massimo 19 caratteri
  - devono iniziare con una lettera e possono contenere lettere, numeri e _

Espressioni

- Matlab si basa su espressioni del tipo

  variabile=espressione

- o semplicemente

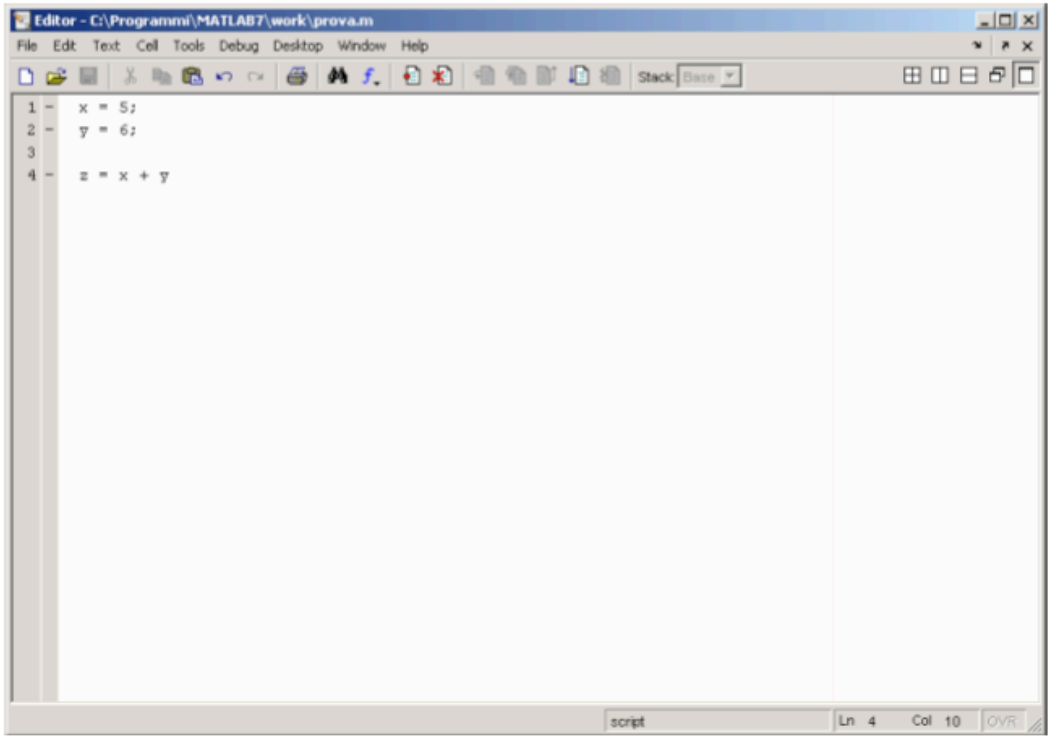
```
>> 100/3
   ans=
         33.3333
```

  – La risposta di un'espressione senza assegnazione viene scritta in una variabile di default chiamata ans

- In Matlab non esistono dichiarazioni di tipo o di dimensioni
- Matlab alloca direttamente la memoria necessaria
- Per avere informazioni si usa il comando whos

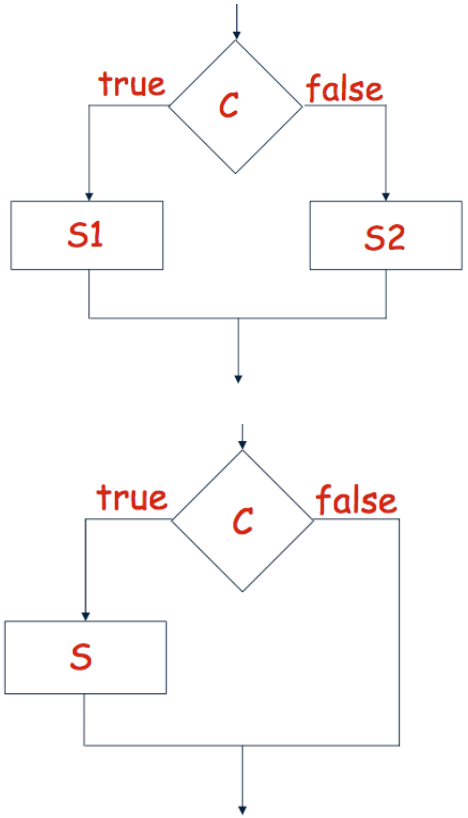
Programmazione

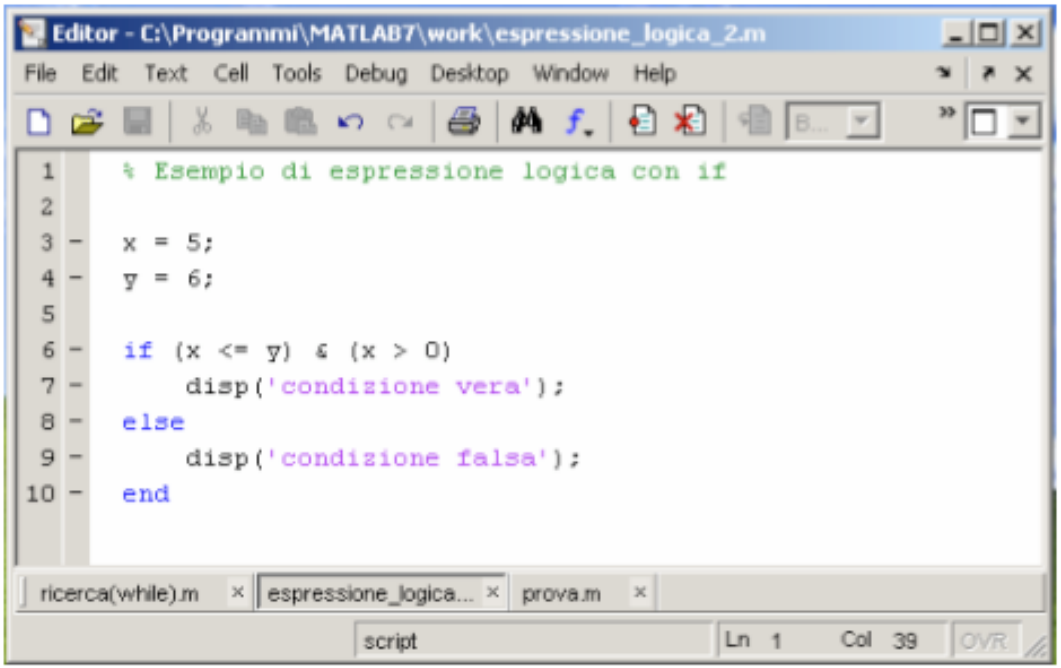
- I file di script vanno salvati in directory contenute nel path
- Un m-file può contenere anche una funzione



Programmazione

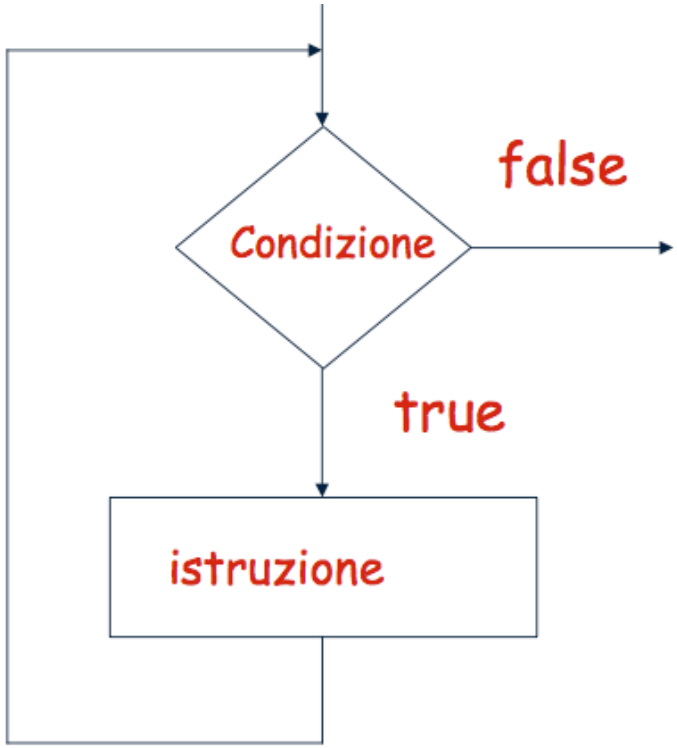
- Il costrutto if-then-else





Programmazione

- Il costrutto while



Editor - C:\Programmi\MATLAB7\work\prova2.m*

File Edit Text Cell Tools Debug Desktop Window Help

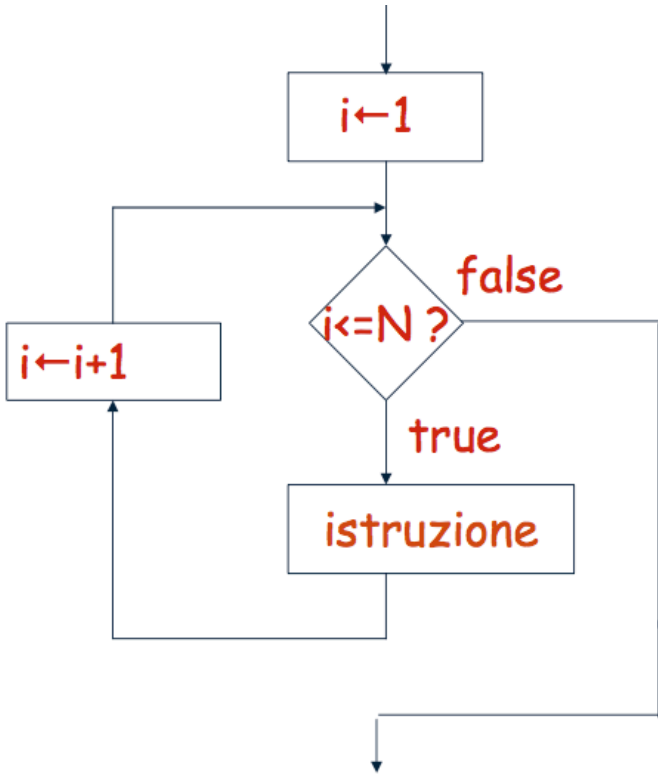
```
1   % Esempio di ciclo while
2
3 - i = 1;
4
5 - while (i <= 10)
6
7 -     disp('questo è il cicolo  ');
8 -     i    % oppure usare disp(i)
9
10 - i = i + 1;
11 - end
12
```

prova.m* × prova2.m* ×

script Ln 7 Col 34 OVR

Programmazione

- Il costrutto for



Editor - C:\Programmi\MATLAB7\work\prova2.m*

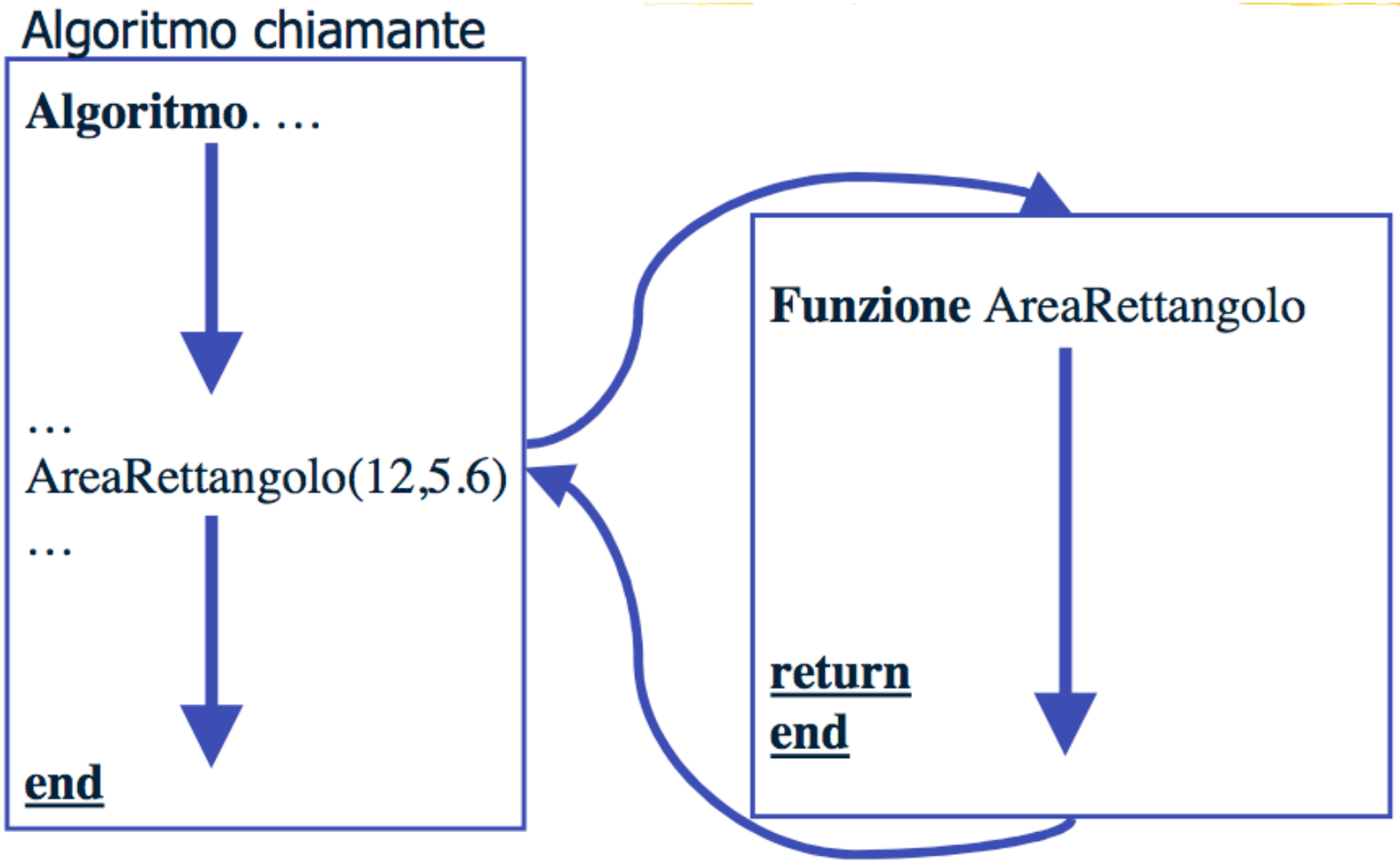
File Edit Text Cell Tools Debug Desktop Window Help

```
1   % Calcolo del fattoriale del numero N (ciclo For)
2
3 - N = input('Dammi il numero N  ');
4
5 - prodotto = 1;
6 - for i = 1 : 1 : N
7 -     prodotto = prodotto * i;
8 - end
9
10 - disp('Il fattoriale è  ');
11 - disp(prodotto);
12
```

prova.m* | prova2.m*

script Ln 4 Col 1 OVR

Sottoprogrammi

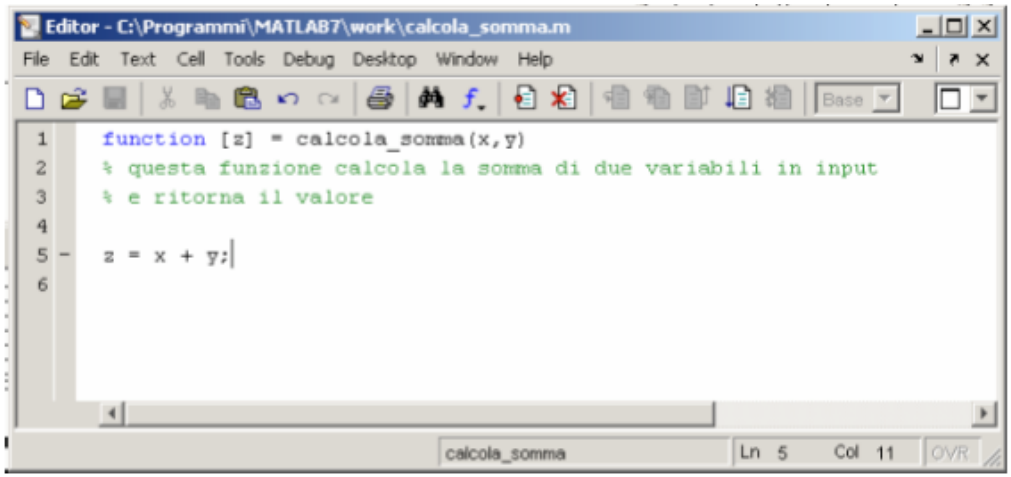


Funzioni

- Una funzione è un sottoprogramma che prende input e restituisce un output.

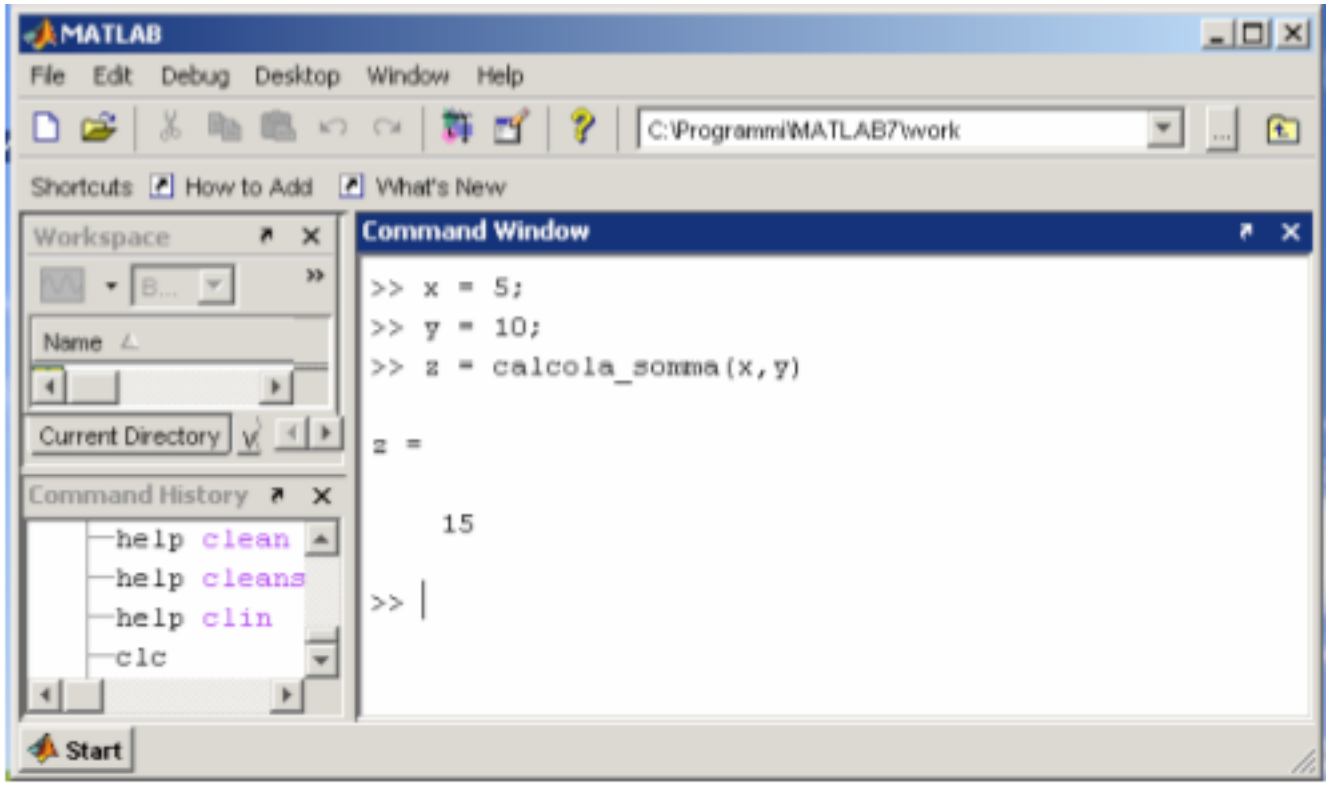
- In Matlab il testo che definisce una funzione ha una riga di intestazione del tipo:

function[z]=nome_funzione(x,y)



Funzioni

- Chiamata di funzione dal workspace



Funzioni

- E' possibile richiamare in una function un'altra function: bisogna però assicurarsi che le function siano nella stessa cartella
- Le variabili interne alle function NON SONO passate nella sessione principale di Matlab
- Ciò può essere fatto dichiarando tali variabili come globali
  - Si utilizza il comando"global", seguito dal nome delle variabili globali
  - E' necessario dichiarare come globali le variabili SIA nella function SIA nella sessione principale di Matlab
  - Esempio: vogliamo che la variabile y della function "media" sia di tipo globale
  - Scriviamo "global y" nella Command Window
  - Si noti che nel Workspace compare la variabile y
  - Questa variabile ancora non èstata definita per cui non sono state assegnate le dimensioni
  - Nel file della funzione nel rigo dopo la definizione della funzione si scrive "global y"

Matrici

- Vettori

  Vettore riga `x = [1 2 3 4 5 6 7 6];`

  Vettore colonna `x = [1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 6 ];`

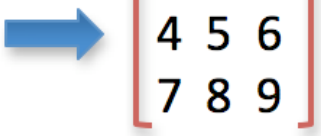
- NB: il ";" alla fine della riga evita la stampa a video del risultato
- Con l'operatore ' (apice) si opera la trasposizione

```
x = [1 2 3 4 5 6 7 6];

y = x' → y = [1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 6];
```

- Matrice

```
matrice = [1 2 3; 4 5 6; 7 8 9];
```



$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Matrici

- Oltre all'introduzione manuale degli elementi in un vettore/matrice, esistono alcuni comandi automatici:
  - Creazione di vettori riga equispaziati con passo unitario:

```
>> v=[3:7]

v =

     3     4     5     6     7

>>
```

  - Creazione di vettori riga equispaziati con passo arbitrario:

```
>> v=[3:0.5:5]

v =

    3.0000    3.5000    4.0000    4.5000    5.0000

>>
```

Matrici

- Creazione di vettori riga con il comando linspace(a, b, n):

```
>> v=linspace(2,7,4)

v =

    2.0000    3.6667    5.3333    7.0000

>>
```

  – Il comando linspace(a, b, n) crea un vettore di n componenti tra a e b, linearmente distruibuite

- Creazione di vettori riga con il comando logspace(a, b, n):

```
>> v=logspace(0,4,3)

v =

           1         100       10000

>>
```

  – Il comando logspace(a, b, n) crea un vettore di n componenti tra a e b, logaritmicamentedistruibuite

Estrazione di elementi

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & 1 \\ 2 & 4 & 3 \end{bmatrix} \qquad A(:) = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & 1 \\ 2 & 4 & 3 \end{bmatrix} \qquad A(:,1) = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} \qquad A(1,:) = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

A(1) A(1,1)	A(4) A(1,2)	A(7) A(1,3)
A(2) A(2,1)	A(5) A(2,2)	A(8) A(2,3)
A(3) A(3,1)	A(6) A(3,2)	A(9) A(3,3)

$A(2,3) = 4 \iff A(6) = 4$

Matrici notevoli

zeros (3) → $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

ones (3) → $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

rand (3) → $\begin{bmatrix} 0.8147 & 0.9134 & 0.2785 \\ 0.9058 & 0.6324 & 0.5469 \\ 0.1270 & 0.0975 & 0.9575 \end{bmatrix}$

distribuzione uniforme [0 1]

randn(3) distribuzione gaussiana a media nulla

```
z = eye(n,m);
```
Matrice identità nxm

Operazioni su matrici

```
>> v1=[1 2 3];
>> v2=[4 5 6];
>> v1+v2

ans =

     5     7     9

>> v1-v2

ans =

    -3    -3    -3

>>
```

"+"

"-"

I vettori/matrici devono avere le stesse dimensioni

```
>> v1=[1 2 3];
>> v2=[4;5;6];
>> v1*v2

ans =

    32

>> v2*v1

ans =

     4     8    12
     5    10    15
     6    12    18

>>
```

"*"

Il prodotto è applicato righe per colonne

Operazioni su matrici

- E' possibile effettuare operazioni elementari sui singoli elementi
- di vettori o matrici:
-

```
>> v1=[1 2 3];
>> v2=[4 5 6];
>> v1.*v2

ans =

     4    10    18

>> v1./v2

ans =

    0.2500    0.4000    0.5000

>>
```

⇨ ".*"

⇨ "./"

```
>> v1=[1 2 3];
>> v1.^2

ans =

     1     4     9

>>
```

⇩ ".^"

".*"

"./"

".^"

Per effettuare operazioni sui singoli elementi di un vettore/matrice è necessario anteporre il "." al simbolo dell'operazione

Funzioni di vettori

- E' possibile creare vettori o matrici usando particolari forme funzionali

- Si supponga di voler costruire un vettore in cui elementi sono le y della seguente funzione per x compreso tra [0, 1]

$$y = \frac{2x^2}{1+x}$$

- Passi necessari:
  - Creazione del vettore x:

```
>> x=[0:0.2:1]

x =

         0    0.2000    0.4000    0.6000    0.8000    1.0000
```

Funzioni di vettori

- Creazione del vettore y:

```
>> y=2*x.^2./(1+x)

y =

         0    0.0667    0.2286    0.4500    0.7111    1.0000

>>
```

- Da notare il "."prima delle operazioni "^"e "/"

- Le funzioni vettorizzate sono importanti per:
  - Diagrammare funzioni
  - Risolvere problemi numerici
  - ecc.

Operazioni su matrici

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & 1 \\ 2 & 4 & 3 \end{bmatrix}$$

size (A) → 3x3

length (A) → 9

min (A) → 1 0 1

max (A) → 2 4 3

max (A(:)) → 4

mean(A(:)), std(A(:)), var(A(:)), ...

sum(A(:)), abs(A(:))

Simboli e funzioni

pi	⇒	pi greco			
i	⇒	unità immaginaria			
sin(x)	⇒	seno	atan(x)	⇒	arcotangente
cos(x)	⇒	coseno	sqrt(x)	⇒	radice quadrata
tan(x)	⇒	tangente	exp(x)	⇒	esponenziale
asin(x)	⇒	arcoseno	log(x)	⇒	log naturale
acos(x)	⇒	arcocoseno	log10(x)	⇒	log in base 10
abs(x)	⇒	valore assoluto di tutti gli elementi del vettore/matrice *x*			
inv(x)	⇒	restituisce l'inversa della matrice *x*			