

ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script

Il tipo più semplice di programma di MATLAB® è denominato *script*. Uno script è un file che contiene numerose righe sequenziali di comandi e richiami di funzioni di MATLAB. È possibile eseguire uno script digitandone il nome dalla riga di comando.

Script

Per creare uno script utilizzare il comando `edit`,

```
edit mysphere
```

Questo comando apre un file vuoto denominato `mysphere.m`. Immettere del codice che crea una sfera unitaria, raddoppia il raggio e crea un grafico dei risultati:

```
[x,y,z] = sphere;  
r = 2;  
surf(x*r,y*r,z*r)  
axis equal
```

Aggiungere quindi del codice che calcola l'area della superficie e il volume di una sfera:

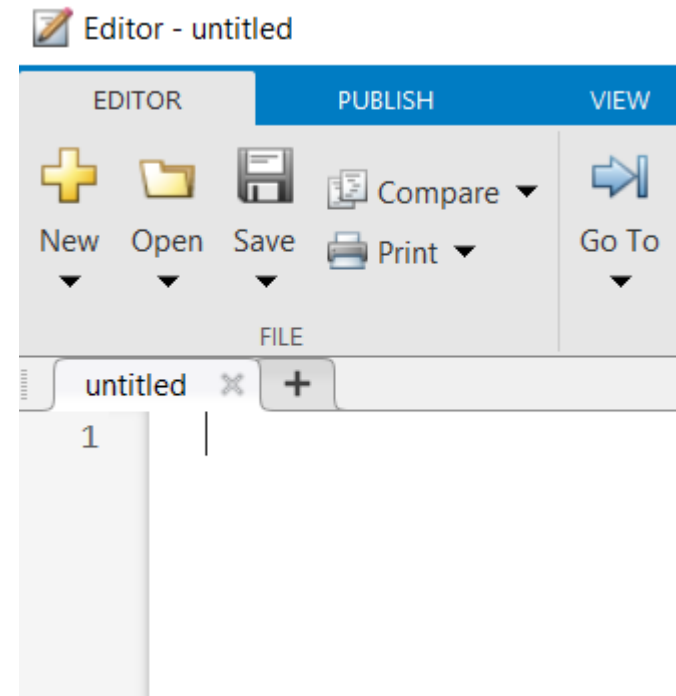
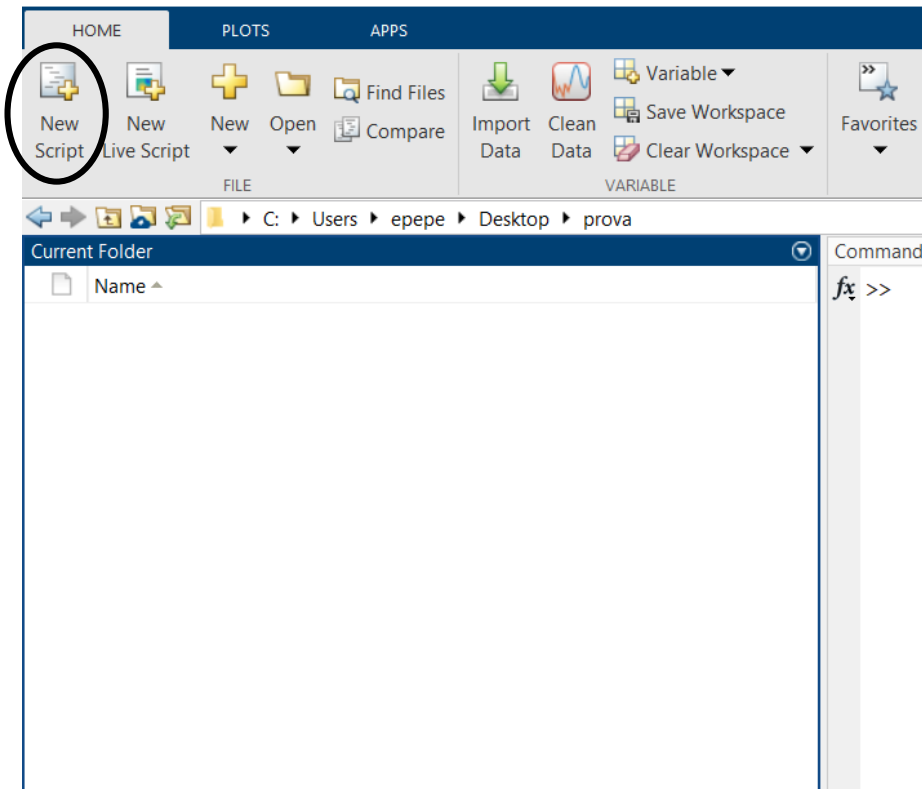
```
A = 4*pi*r^2;  
V = (4/3)*pi*r^3;
```

ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script

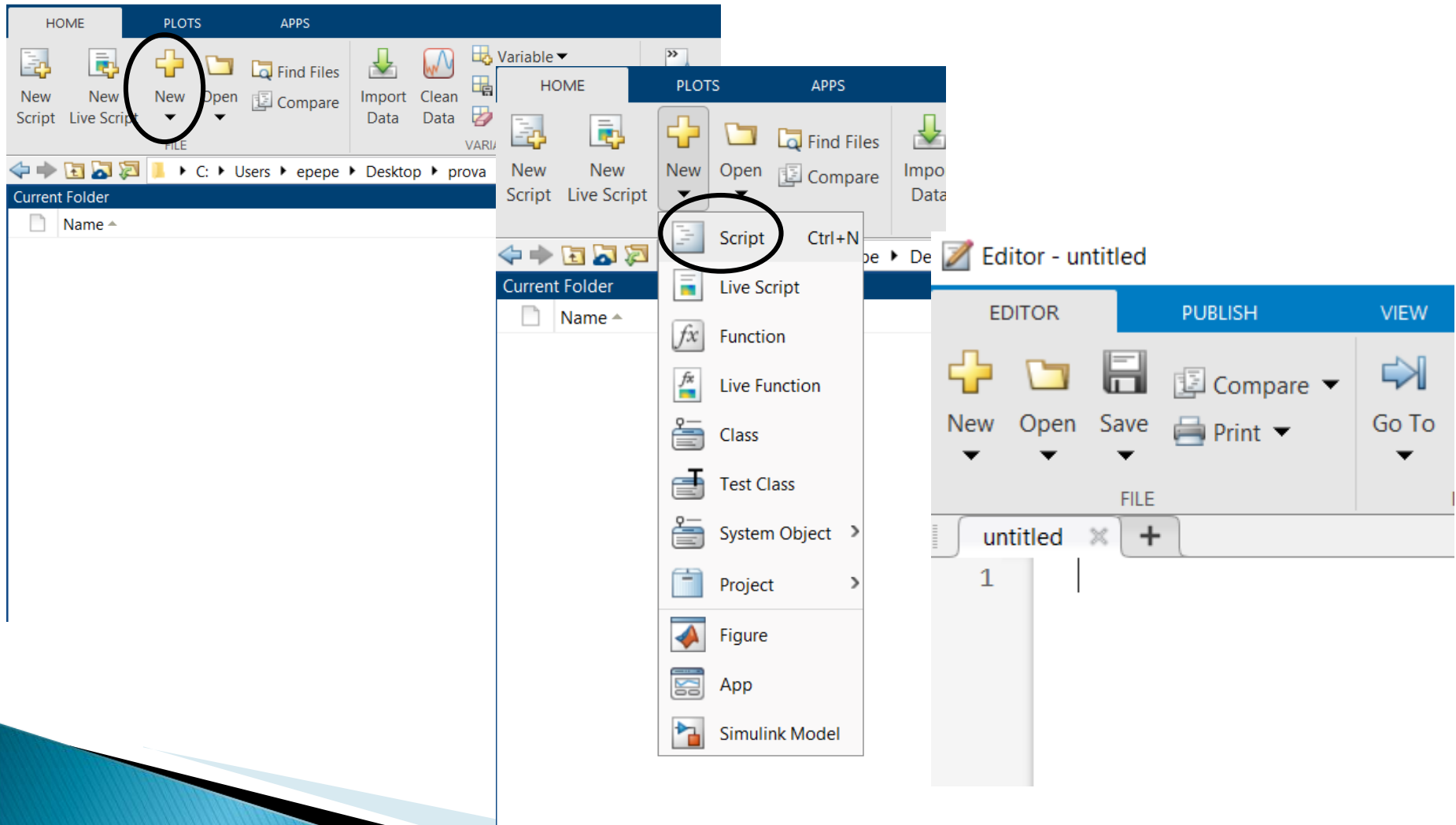


ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script



ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script


Script

Quando si scrive del codice è buona prassi aggiungere dei commenti descrittivi. I commenti consentono ad altri di capire il codice e di fare mente locale quando lo si riesamina in un secondo momento. Per aggiungere i commenti utilizzare il simbolo di percentuale (%).

```
% Create and plot a sphere with radius r.  
[x,y,z] = sphere;      % Create a unit sphere.  
r = 2;  
surf(x*r,y*r,z*r)      % Adjust each dimension and plot.  
axis equal             % Use the same scale for each axis.  
  
% Find the surface area and volume.  
A = 4*pi*r^2;  
V = (4/3)*pi*r^3;
```

Salvare il file nella cartella corrente. Per eseguire uno script, digitarne il nome dalla riga di comando:

```
mysphere
```

È inoltre possibile eseguire script dall'Editor con il pulsante **Run**, .

ESERCITAZIONI

MATLAB – Come iniziare




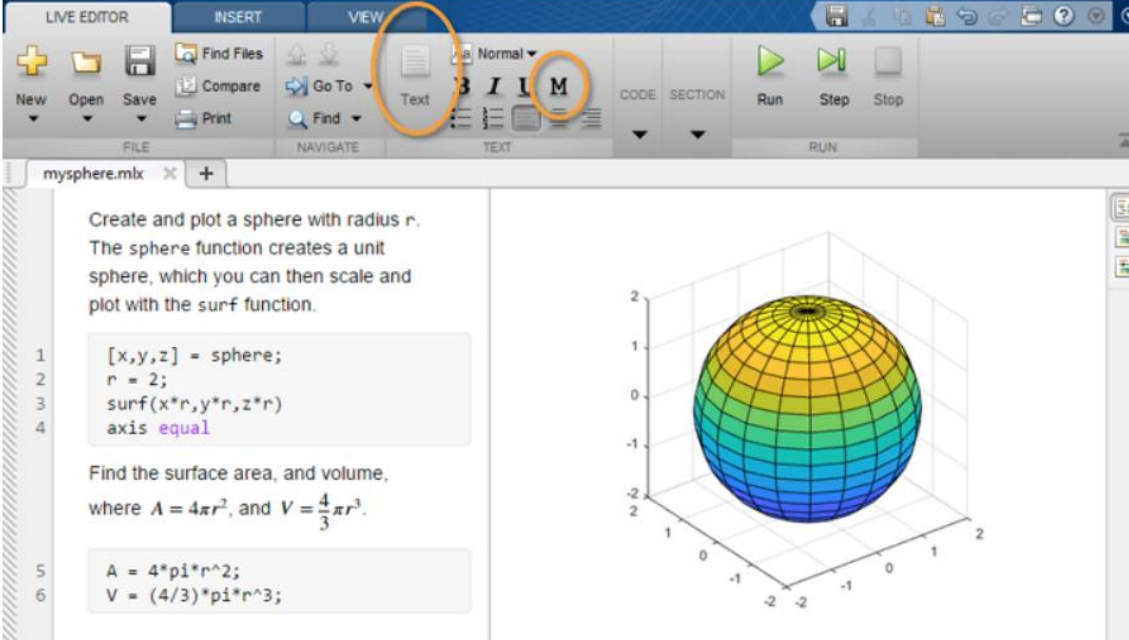
Programmazione e script

Script live

Invece di scrivere codici e commenti in testo normale, è possibile utilizzare le opzioni di formattazione in *script live* per migliorare il codice. Gli script live consentono di visualizzare e di interagire sia con il codice che con l'output e possono includere testo formattato, equazioni e immagini.

Ad esempio, convertire *mysphere* in uno script live selezionando **Save As** e cambiando il tipo di file in un file di codice live di MATLAB (*.mlx). Sostituire quindi i commenti del codice con testo formattato. Ad esempio:

- Converti in testo le righe dei commenti. Selezionare prima tutte le righe che iniziano con il simbolo di percentuale e poi **Text**, . Eliminare i simboli di percentuale.
- Scrivere nuovamente il testo per sostituire i commenti alla fine delle righe di codice. Per applicare un font monospazio ai nomi di funzioni nel testo selezionare **M**. Per aggiungere un'equazione selezionare **Equation** nella scheda **Insert**.



The screenshot shows the MATLAB Live Editor interface. The top toolbar includes tabs for LIVE EDITOR, INSERT, and VIEW. The INSERT tab is active, showing various formatting options. Two orange circles highlight the 'Text' button (a document icon) and the 'M' button (monospace font). The main workspace displays a script for creating a sphere plot. The script includes comments and code for plotting a sphere and calculating its surface area and volume. The output shows a 3D plot of a sphere with a color gradient from blue to yellow.

```
1 [x,y,z] = sphere;  
2 r = 2;  
3 surf(x*r,y*r,z*r)  
4 axis equal  
  
Find the surface area, and volume,  
where  $A = 4\pi r^2$ , and  $V = \frac{4}{3}\pi r^3$ .  
  
5 A = 4*pi*r^2;  
6 V = (4/3)*pi*r^3;
```

Per creare un nuovo script live utilizzando il comando `edit`, includere l'estensione `.mlx` nel nome file:

```
edit newfile.mlx
```

ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script

function

Dichiarare il nome della funzione, gli input e gli output

Sintassi

```
function [y1,...,yN] = myfun(x1,...,xM)
```

Descrizione

`function [y1,...,yN] = myfun(x1,...,xM)` dichiara una funzione denominata `myfun` che accetta input `x1,...,xM` e restituisce output `y1,...,yN`. Questa dichiarazione di istruzione deve essere la prima riga eseguibile della funzione. I nomi validi di una funzione iniziano con un carattere alfabetico e possono contenere lettere, numeri o trattini bassi.

È possibile salvare la funzione:

- In un file di funzione che contiene solo le definizioni della funzione. Il nome del file deve corrispondere al nome della prima funzione del file.
- In un file di script che contiene i comandi e le definizioni della funzione. Le funzioni devono trovarsi alla fine del file. I file di script non possono avere lo stesso nome di una funzione del file. Le funzioni sono supportate negli script dalla release R2016b o successive.

I file possono includere più funzioni locali o annidate. Per una maggiore leggibilità, utilizzare la parola chiave `end` per indicare la fine di ciascuna funzione in un file. La parola chiave `end` è richiesta quando:

- Qualsiasi funzione del file contiene una funzione annidata.
- La funzione è una funzione locale all'interno di un file di funzioni e qualsiasi funzione locale nel file utilizza la parola chiave `end`.
- La funzione è una funzione locale all'interno di un file di script.

ESERCITAZIONI

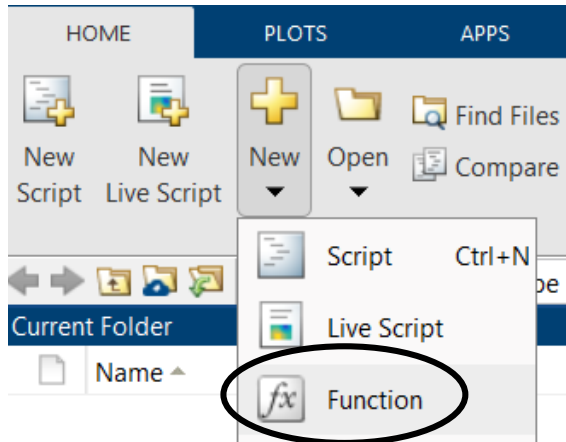
MATLAB – Come iniziare



Programmazione e script

function

Dichiarare il nome della funzione, gli input e gli output



```
untitled * x +
1  function [outputArg1,outputArg2] = untitled(inputArg1,inputArg2)
2  %UNTITLED Summary of this function goes here
3  % Detailed explanation goes here
4  outputArg1 = inputArg1;
5  outputArg2 = inputArg2;
6  end
```

ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script

Esempi

▼ Funzione con un output

Definire una funzione in un file denominato `calculateAverage.m` che accetti un vettore di input, calcoli la media dei valori e restituisca un risultato singolo.

```
function ave = calculateAverage(x)
    ave = sum(x(:))/numel(x);
end
```

Chiamare la funzione dalla riga di comando.

```
z = 1:99;
ave = calculateAverage(z)
```

```
ave =
    50
```

▼ Funzione con più output

Definire una funzione in un file denominato `stat.m` che restituisca la media e la deviazione standard di un vettore di input.

```
function [m,s] = stat(x)
    n = length(x);
    m = sum(x)/n;
    s = sqrt(sum((x-m).^2/n));
end
```

Chiamare la funzione dalla riga di comando.

```
values = [12.7, 45.4, 98.9, 26.6, 53.1];
[ave,stdev] = stat(values)
```

```
ave =
    47.3400
stdev =
    29.4124
```


ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script

▼ Funzione in un file di script

Definire uno script in un file denominato `integrationScript.m` che calcoli il valore dell'integranda in corrispondenza di $2\pi/3$ e l'area sotto la curva da 0 a π . Include una funzione locale che definisce l'integranda $y = \sin(x)^3$.

Nota: per includere funzioni negli script, è necessario disporre della release MATLAB® R2016b o successive.

```
% Compute the value of the integrand at 2*pi/3.
x = 2*pi/3;
y = myIntegrand(x)

% Compute the area under the curve from 0 to pi.
xmin = 0;
xmax = pi;
f = @myIntegrand;
a = integral(f,xmin,xmax)

function y = myIntegrand(x)
    y = sin(x).^3;
end
```

y =

0.6495

a =

1.3333

ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script

▼ Funzioni multiple in un file di funzioni

Definire due funzioni in un file denominato `stat2.m`, dove la prima funzione chiama la seconda.

```
function [m,s] = stat2(x)
    n = length(x);
    m = avg(x,n);
    s = sqrt(sum((x-m).^2/n));
end

function m = avg(x,n)
    m = sum(x)/n;
end
```

La funzione `avg` è una *funzione locale*. Le funzioni locali sono disponibili solo per altre funzioni all'interno dello stesso file.

Chiamare la funzione `stat2` dalla riga di comando.

```
values = [12.7, 45.4, 98.9, 26.6, 53.1];
[ave,stdev] = stat2(values)
```

```
ave =
    47.3400
stdev =
    29.4124
```

ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script

▼ Funzione con convalida dell'argomento

Definire una funzione che limiti l'input a un vettore numerico che non contenga elementi Inf o NaN. Questa funzione utilizza la parola chiave `arguments`, che è valida per le versioni MATLAB® R2019b e successive.

```
function [m,s] = stat3(x)
    arguments
        x (1,:) {mustBeNumeric, mustBeFinite}
    end
    n = length(x);
    m = avg(x,n);
    s = sqrt(sum((x-m).^2/n));
end

function m = avg(x,n)
    m = sum(x)/n;
end
```

Nel blocco di codice `arguments`, `(1,:)` indica che `x` deve essere un vettore. Le funzioni di convalida `{mustBeNumeric, mustBeFinite}` limitano gli elementi in `x` a valori numerici che non siano Inf o NaN. Per maggiori informazioni, vedere [Function Argument Validation](#).

La chiamata della funzione con un vettore che contiene un elemento NaN viola la dichiarazione dell'argomento di input. Questa violazione comporta un errore nella funzione di convalida `mustBeFinite`.

```
values = [12.7, 45.4, 98.9, NaN, 53.1];
[ave,stdev] = stat3(values)
```

```
Invalid input argument at position 1. Value must be finite.
```

ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script

Loop e dichiarazioni condizionali

In qualsiasi script è possibile definire sezioni di codice da ripetere in loop oppure da eseguire in base a determinate condizioni. I loop utilizzano una parola chiave `for` o `while`, mentre le dichiarazioni condizionali utilizzano `if` o `switch`.

I loop sono utili per creare sequenze. Ad esempio, creare uno script denominato `fibseq` che utilizza un loop `for` per calcolare 100 numeri della sequenza di Fibonacci. In questa sequenza i primi due numeri sono 1 e ciascun numero successivo è la somma dei due che lo precedono: $F_n = F_{n-1} + F_{n-2}$.

```
N = 100;  
f(1) = 1;  
f(2) = 1;  
  
for n = 3:N  
    f(n) = f(n-1) + f(n-2);  
end  
f(1:10)
```

Quando si esegue lo script, la dichiarazione `for` definisce un contatore denominato `n` che parte dal 3. Il loop quindi esegue assegnazioni ripetute a `f(n)`, incrementando `n` a ogni esecuzione, fino a raggiungere il numero 100. L'ultimo comando dello script, `f(1:10)`, visualizza i primi 10 elementi di `f`.

```
ans =  
     1     1     2     3     5     8    13    21    34    55
```

Le dichiarazioni condizionali vengono eseguite solo quando determinate espressioni risultano vere. Ad esempio, assegnare un valore a una variabile in base alle dimensioni di un numero casuale: 'low', 'medium' o 'high'. In questo caso il numero casuale è un numero intero compreso tra 1 e 100.

```
num = randi(100)  
if num < 34  
    sz = 'low'  
elseif num < 67  
    sz = 'medium'  
else  
    sz = 'high'  
end
```

La dichiarazione `sz = 'high'` viene eseguita solo quando `num` è uguale o superiore a 67.

ESERCITAZIONI

MATLAB – Come iniziare



Programmazione e script

Loop e dichiarazioni condizionali

In qualsiasi script è possibile definire sezioni di codice da ripetere in loop oppure da eseguire in base a determinate condizioni. I loop utilizzano una parola chiave `for` o `while`, mentre le dichiarazioni condizionali utilizzano `if` o `switch`.

```
n = 10;  
f = n;  
while n > 1  
    n = n-1;  
    f = f*n;  
end  
disp(['n! = ' num2str(f)])
```

n! = 3628800

```
n = input('Enter a number: ');  
  
switch n  
    case -1  
        disp('negative one')  
    case 0  
        disp('zero')  
    case 1  
        disp('positive one')  
    otherwise  
        disp('other value')  
end
```

Sede degli script

MATLAB cerca gli script e altri file in determinate sedi. Per eseguire uno script, il file deve trovarsi nella cartella corrente o in una cartella compresa nel *percorso di ricerca*.

Ad esempio, la cartella di MATLAB creata dal programma di installazione di MATLAB si trova nel percorso di ricerca. Per memorizzare ed eseguire programmi in un'altra cartella, aggiungerla al percorso di ricerca. Selezionare la cartella nel browser Cartella corrente, fare clic con il pulsante destro del mouse e quindi selezionare **Add to Path**.

ESERCITAZIONI

MATLAB – Come iniziare



Guida e documentazione

Tutte le funzioni di MATLAB® sono dotate di una documentazione esplicativa con esempi e descrizione di input, output e sintassi. Vi sono diversi metodi per accedere a queste informazioni dalla riga di comando:

- Aprire la documentazione relativa alla funzione in una finestra a parte con il comando `doc`.


```
doc mean
```

- Per visualizzare suggerimenti sulla funzione (la parte della sintassi della relativa documentazione) nella finestra di comando, effettuare una pausa dopo aver digitato le parentesi aperte per gli argomenti di input della funzione.

```
mean(
```

- Per visualizzare un riepilogo della documentazione sulla funzione nella finestra di comando utilizzare `help`.

```
help mean
```

Per accedere alla documentazione completa sul prodotto fare clic sull'icona della guida .

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

To clear a command from the Command Window without executing it, press the Escape (**Esc**) key.

You can evaluate any statement already in the Command Window. Select the statement, right-click, and then select **Evaluate Selection**.

In the Command Window, you also can execute only a portion of the code currently at the command prompt. To evaluate a portion of the entered code, select the code, and then press **Enter**.

For example, select a portion of the following code:

```
fx >> disp("hello"), disp("world")
```

```
hello
```

Funzioni

ans	Most recent answer
clc	Azzerare la finestra di comando
diary	Log Command Window text to file
format	Set output display format
home	Send cursor home
iskeyword	Determine if input is MATLAB keyword
more	Control paged output in Command Window
commandwindow	Select the Command Window
commandhistory	Open Command History window

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

▼ Log Statements and Output

Create a diary file and record several statements and their output.

Enable logging and save the resulting log to myDiaryFile.

```
diary myDiaryFile
```

Perform a calculation, and create and display a matrix of ones in the Command Window.

```
a = 1;  
b = sin(a);  
  
x = ones(4)
```

x =

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

Disable logging and display the log file in the Command Window.

```
diary off  
type myDiaryFile
```

```
a = 1;  
b = sin(a);  
x = ones(4)
```

```
x =
```

```
1     1     1     1  
1     1     1     1  
1     1     1     1  
1     1     1     1
```

```
diary off
```

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

```
>> fmt=format
```

```
fmt =
```

DisplayFormatOptions with properties:

NumericFormat: "short"

LineSpacing: "loose"

```
format loose
```

```
format compact
```

By default, MATLAB displays blank lines in Command Window output.

You can select one of two line spacing options in MATLAB.

- `loose` — Keeps the display of blank lines (default).

```
>> x = [4/3 1.2345e-6]
```

```
x =
```

```
1.3333    0.0000
```

- `compact` — Suppresses the display of blank lines.

```
>> x = [4/3 1.2345e-6]
```

```
x =
```

```
1.3333    0.0000
```

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

▼ Move Cursor to Home Position

Use the `magic` function to create and display a 5-by-5 integer matrix in the Command Window.

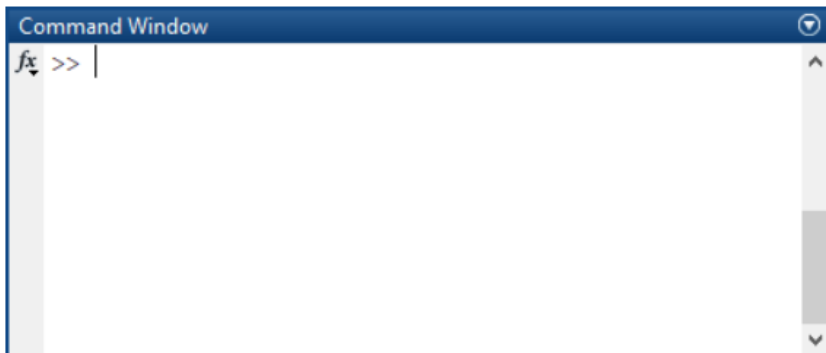
```
magic(5)
```

```
ans = 5x5
```

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

Run the `home` function to scroll the displayed matrix and any other visible text out of view and to move the cursor to the upper-left corner of the Command Window.

```
home
```



Use the Command Window scroll bar to scroll back up and see the hidden text.

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

To test if the word `while` is a MATLAB keyword,

```
iskeyword('while')  
ans =  
    1
```

To obtain a list of all MATLAB keywords,

```
iskeyword  
'break'  
'case'  
'catch'  
'classdef'  
'continue'  
'else'  
'elseif'  
'end'  
'for'  
'function'  
'global'  
'if'  
'otherwise'  
'parfor'  
'persistent'  
'return'  
'spmd'  
'switch'  
'try'  
'while'
```

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

▼ View Help One Page at a Time

Use the `more` function to control the number of help lines displayed in the command window.

Enable paging in the Command Window and then get help on the `plot` function.

```
more on  
help plot
```

`plot` Linear plot.

`plot(X,Y)` plots vector `Y` versus vector `X`. If `X` or `Y` is a matrix, then the vector is plotted versus the rows or columns of the matrix, whichever line up. If `X` is a scalar and `Y` is a vector, disconnected line objects are created and plotted as discrete points vertically at `X`.

`plot(Y)` plots the columns of `Y` versus their index.

If `Y` is complex, `plot(Y)` is equivalent to `plot(real(Y),imag(Y))`.

--more--

Press the **Space** key to view the help page by page until the last page is displayed. Call `more off` to disable paging.

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

▼ Select the Command Window After Creating a Plot

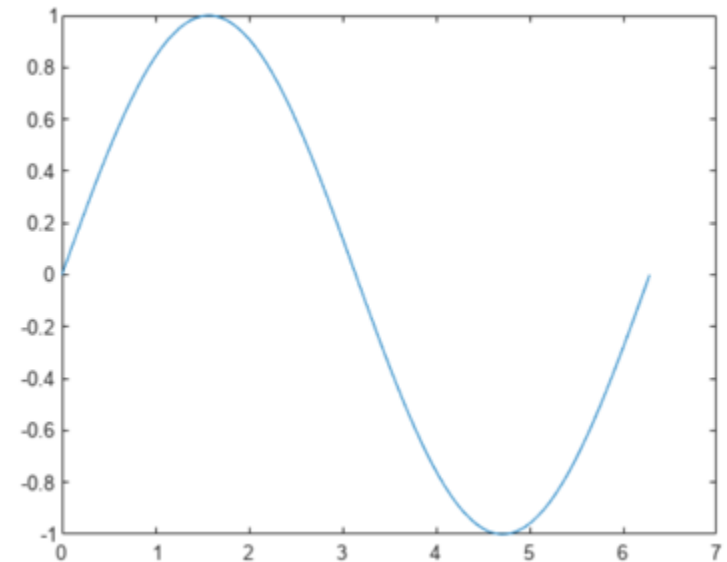
Create a line plot and then bring focus back to the Command Window.

Create a line plot by typing these statements at the command line. MATLAB creates a figure containing the line plot. Focus moves to the figure.

```
x = 0:pi/100:2*pi;  
y = sin(x);  
plot(x,y)
```

Bring focus back to the Command Window.

```
commandwindow
```



ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

The image shows the MATLAB interface with three main panels:

- Command Window:** Contains the text "New to MATLAB? See resources for [Getting Started.](#)" and the command `>> commandhistory` followed by a prompt `fx >>`.
- Workspace:** A table with two columns: "Name" and "Value". It is currently empty.
- Command History:** A list of previously executed commands, including `more off`, `clear all`, `clc`, `doc commandwindow`, `commandwindow`, `commandhistory`, `format`, `fmt=format`, `doc more`, and several instances of `clc` and `commandhistory`.

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

Estensione di dichiarazioni lunghe su più righe

Questo esempio mostra come proseguire una dichiarazione alla riga successiva utilizzando l'ellissi (...).

```
s = 1 - 1/2 + 1/3 - 1/4 + 1/5 ...  
    - 1/6 + 1/7 - 1/8 + 1/9;
```

Costruire un vettore di caratteri lungo concatenando vettori più corti:

```
mytext = ['Accelerating the pace of ' ...  
          'engineering and science'];
```

Le virgolette iniziali e finali di un vettore di caratteri devono comparire sulla stessa riga. Ad esempio, questo codice restituisce un errore poiché ogni riga contiene solo una virgoletta:

```
mytext = 'Accelerating the pace of ...  
          engineering and science'
```

Un'ellissi al di fuori di un testo citato equivale a uno spazio. Ad esempio,

```
x = [1.23...  
     4.56];
```

equivale a

```
x = [1.23 4.56];
```


ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

Variable Names

Valid Names

A valid variable name starts with a letter, followed by letters, digits, or underscores. MATLAB® is case sensitive, so A and a are *not* the same variable. The maximum length of a variable name is the value that the `namelengthmax` command returns.

You cannot define variables with the same names as MATLAB keywords, such as `if` or `end`. For a complete list, run the `iskeyword` command.

Examples of valid names:	Examples of invalid names:
x6	6x
lastValue	end
n_factorial	n!

Conflicts with Function Names

Avoid creating variables with the same name as a function (such as `i`, `j`, `mode`, `char`, `size`, and `path`). In general, variable names take precedence over function names. If you create a variable that uses the name of a function, you sometimes get unexpected results.

Check whether a proposed name is already in use with the `exist` or `which` function. `exist` returns 0 if there are no existing variables, functions, or other artifacts with the proposed name. For example:

```
exist checkname
```

```
ans =  
    0
```

If you inadvertently create a variable with a name conflict, remove the variable from memory with the `clear` function.

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

Check Syntax as You Type

Syntax Highlighting

To help you identify MATLAB® elements, some entries appear in different colors in the Command Window, the Editor, and the Live Editor. This color display is known as *syntax highlighting*. By default:

- Keywords are blue.
- Character vectors and strings are purple.
- Unterminated character vectors are maroon.
- Comments are green.

```
% check to see if A is greater than B
if A > B
    "greater"
elseif A < B
    "less"
end
```

Except for errors, output in the Command Window does *not* appear with syntax highlighting.

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

Case and Space Sensitivity

MATLAB® code is sensitive to casing, and insensitive to blank spaces except when defining arrays.

Uppercase and Lowercase

In MATLAB code, use an exact match with regard to case for variables, files, and functions. For example, if you have a variable, `a`, you cannot refer to that variable as `A`. It is a best practice to use lowercase only when naming functions. This is especially useful when you use both Microsoft® Windows® and UNIX® platforms because their file systems behave differently with regard to case.

When you use the `help` function, the help displays some function names in all uppercase, for example, `PLOT`, solely to distinguish the function name from the rest of the text. Some functions for interfacing to Oracle® Java® software do use mixed case and the command-line help and the documentation accurately reflect that.

Spaces

Blank spaces around operators such as `-`, `:`, and `()`, are optional, but they can improve readability. For example, MATLAB interprets the following statements the same way.

```
y = sin (3 * pi) / 2
y=sin(3*pi)/2
```

However, blank spaces act as delimiters in horizontal concatenation. When defining row vectors, you can use spaces and commas interchangeably to separate elements:

```
A = [1, 0 2, 3 3]
```

```
A =
```

```
1      0      2      3      3
```

Because of this flexibility, check to ensure that MATLAB stores the correct values. For example, the statement `[1 sin (pi) 3]` produces a much different result than `[1 sin(pi) 3]` does.

```
[1 sin (pi) 3]
```

```
Error using sin
Not enough input arguments.
```

```
[1 sin(pi) 3]
```

```
ans =
```

```
1.0000    0.0000    3.0000
```

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

Name=Value in Function Calls

Since R2021a

MATLAB® supports two syntaxes for passing name-value arguments.

`plot(x,y,LineWidth=2)` **name=value syntax**

`plot(x,y,"LineWidth",2)` **comma-separated syntax**

Use the name=value syntax to help identify name-value arguments for functions and to clearly distinguish names from values in lists of name-value arguments.

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

Ignore Function Outputs

This example shows how to ignore specific outputs from a function using the tilde (~) operator.

Request all three possible outputs from the `fileparts` function.

```
helpFile = which('help');  
[helpPath,name,ext] = fileparts(helpFile);
```

The current workspace now contains three variables from `fileparts`: `helpPath`, `name`, and `ext`. In this case, the variables are small. However, some functions return results that use much more memory. If you do not need those variables, they waste space on your system.

If you do not use the tilde operator, you can request only the first N outputs of a function (where N is less than or equal to the number of possible outputs) and ignore any remaining outputs. For example, request only the first output, ignoring the second and third.

```
helpPath = fileparts(helpFile);
```

If you request more than one output, enclose the variable names in square brackets, `[]`. The following code ignores the output argument `ext`.

```
[helpPath,name] = fileparts(helpFile);
```

To ignore function outputs in any position in the argument list, use the tilde operator. For example, ignore the first output using a tilde.

```
[~,name,ext] = fileparts(helpFile);
```

You can ignore any number of function outputs using the tilde operator. Separate consecutive tildes with a comma. For example, this code ignores the first two output arguments.

```
[~,~,ext] = fileparts(helpFile);
```

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

Interruzione dell'esecuzione

Per interrompere l'esecuzione di un comando di MATLAB®, premere **CTRL+C** o **CTRL+INTERR**.

Sulle piattaforme Apple Macintosh, è inoltre possibile utilizzare **Command+**. (tasto Command e tasto punto).

CTRL+C non sempre interrompe l'esecuzione di file che richiedono molto tempo per essere eseguiti o che richiamano funzioni integrate, o di file MEX che richiedono molto tempo per essere eseguiti. Se si verifica questo problema, includere una funzione `drawnow`, `pause` o `getframe` nel file, ad esempio all'interno di un loop di grandi dimensioni. Inoltre, **CTRL+C** potrebbe essere meno reattivo se si avvia MATLAB con l'opzione `-nodesktop`.



Nota

Per alcune operazioni, l'arresto del programma potrebbe generare errori nella Finestra di comando.

Per interrompere l'esecuzione di una funzione o di uno script in modo programmatico prima di raggiungere la fine, utilizzare la funzione `return`. MATLAB restituisce il controllo alla finestra di comando o alla funzione che la invoca.



Return Control to Keyboard

In your current working folder, create a function, `findSqrRootIndex`, to find the index of the first occurrence of the square root of a value within an array. If the square root is not found, the function returns NaN.

```
function idx = findSqrRootIndex(target,arrayToSearch)

idx = NaN;
if target < 0
    return
end

for idx = 1:length(arrayToSearch)
    if arrayToSearch(idx) == sqrt(target)
        return
    end
end
```

ESERCITAZIONI

MATLAB – Nozioni fondamentali



Inserimento dei comandi

At the command prompt, call the function.

```
A = [3 7 28 14 42 9 0];  
b = 81;  
findSqrRootIndex(b,A)
```

```
ans =
```

```
6
```

When MATLAB encounters the return statement, it returns control to the keyboard because there is no invoking script or function.

Riferimenti Bibliografici

[1] <https://it.mathworks.com>