



Matlab: Script e Funzioni

Informatica B

Script

Cos'è uno script (m-file)

- ❑ Uno script è un file di testo contenente una sequenza di comandi MATLAB
 - ▶ non deve contenere caratteri di formattazione (solo testo puro)
 - ▶ viene salvato con estensione .m
- ❑ I comandi all'interno di uno script sono eseguiti sequenzialmente, come se fossero scritti nella finestra dei comandi
 - ▶ Per eseguire il file si digita il suo nome (senza .m)
 - ▶ I risultati appaiono nella finestra dei comandi (se non usiamo il ;)

- ❑ Può essere creato utilizzando un qualsiasi editor di testo
 - ▶ Ricordarsi di salvare il file come “solo testo” e di dare l'estensione .m
 - ▶ Il file di script deve essere presente nella directory corrente o il cammino (path) che identifica la directory in cui si trova lo script deve essere salvato tra i path di Matlab
- ❑ MATLAB include un editor dove creare o modificare script
- ❑ Il nome del file (e dello script) segue le stesse regole dei nomi di variabile:
 - ▶ deve cominciare con una lettera e può contenere cifre e il carattere underscore, fino a 31 caratteri
 - ▶ è opportuno non usare un nome già in uso (il comando **exist** permette di verificare se un nome è già utilizzato)

A cosa serve uno script?

- ❑ Uno script ci permette di progettare la soluzione (Matlab) ad un problema e memorizzarla in maniera permanente (nel file script)
- ❑ Infatti, uno script può
 - ▶ essere facilmente ri-eseguito
 - ▶ essere facilmente modificato (sviluppo incrementale)
 - ▶ essere spedito a qualcuno
 - ▶ ...

- ❑ Quando l'interprete incontra il carattere % ignora tutto ciò che lo segue su quella riga (che viene detto **commento**)
- ❑ I commenti
 - ▶ servono solo a chiarire il funzionamento del programma
 - ▶ possono partire dall'inizio di una riga o dalla metà
 - ▶ i commenti posti all'inizio di uno script vengono utilizzati dal comando **help** come descrizione del programma
- ❑ Esempio

```
% Questo script converte una velocità da km/h a m/s  
v = input('Inserire velocità in km/h')  
v = v * (1000/3600) % converto v da km/h a m/s
```

1. Sezione dei commenti:
 - ▶ Il nome del programma e le parole chiave, nella prima riga
 - ▶ La data di creazione e i nomi degli autori nella seconda riga
 - ▶ La definizione dei nomi delle variabili per ogni variabile di input e di output
 - ▶ Il nome di ogni funzione creata dall'utente che viene usata nel programma
 - ▶ Il comando help visualizza tutta la sezione dei commenti all'inizio dello script
2. Sezione di Input: inserimento dei dati in input e/o uso di funzioni di input
3. Sezione di calcolo
4. Sezione di output: uso si funzioni per visualizzare i risultati del programma

- ❑ Gli script non permettono di definire esplicitamente variabili di input o variabili di output
- ❑ Le variabili utilizzate in uno script possono essere:
 - ▶ variabili esistenti nella memoria di lavoro (**workspace**)
 - ▶ variabili create nello script (tramite assegnamento oppure acquisizione da tastiera o file)
- ❑ Le variabili create in uno script restano nella memoria di lavoro (**workspace**) anche al termine della sua esecuzione
- ❑ Il valore assegnato alle variabili durante l'esecuzione di uno script permane anche alla fine della sua esecuzione

Funzioni

A cosa servono le funzioni?

```
x = input('inserisci velocità in km/h: ');  
y = x * (1000/3600);  
d = input ('inserire incremento di velocità: ');  
x2 = x + d;  
y2 = x2 * (1000/3600);
```

A cosa servono le funzioni?

```
x = input('inserisci velocità in km/h: ');
```

```
y = x * (1000/3600);
```

```
d = input ('inserire incremento di velocità: ');
```

```
x2 = x + d;
```

```
y2 = x2 * (1000/3600);
```

A cosa servono le funzioni?

- ❑ Riusabilità
 - ▶ Scrivo una sola volta codice utilizzato spesso
 - ▶ Modifiche e correzioni sono gestibili facilmente
- ❑ Leggibilità
 - ▶ Incapsulo porzioni di codice complesso
 - ▶ Aumento il livello di astrazione dei miei programmi
- ❑ Flessibilità
 - ▶ Posso aggiungere funzionalità non presenti nelle funzioni di libreria

```
function y=converti (x)
    y = x* (1000/3600); } corpo
```

testata

x è l'argomento della funzione (serve a fornire l'input)

y è il valore di ritorno della funzione (serve a fornire l'output)

- ❑ La testata inizia con la parola chiave **function** e definisce:
 - ▶ nome della funzione
 - ▶ argomenti (input)
 - ▶ valore di ritorno (output)
- ❑ Il corpo definisce le istruzioni da eseguire quando la funzione viene chiamata
 - ▶ Utilizza gli argomenti e assegna il valore di ritorno

Le funzioni (2)

- ❑ Una funzione può avere più argomenti separati da virgola:

`function f(x,y)`

- ❑ Nel caso sia necessario ritornare più valori, possiamo usare un array:

`function [v1,v2,...] = f(x,y)`

- ❑ Esempio:

```
function [minore, maggiore] = minmax(a,b,c)
    minore = min ([a,b,c]);
    maggiore = max ([a,b,c]);
```

- ❑ Una funzione può essere invocata in un programma attraverso il suo nome, seguito dagli argomenti fra parentesi rotonde
- ❑ La funzione viene quindi eseguita e il suo valore di ritorno viene calcolato.
- ❑ Esempio

```
x = input('inserisci velocità in km/h: ');  
y = converti(x);  
d = input('inserire incremento di velocità: ');  
y2 = converti(x + d);
```

I parametri

□ Definizioni:

- ▶ I **parametri formali** sono le variabili usate come argomenti e valore di ritorno **nella definizione** della funzione
- ▶ I **parametri attuali** sono i valori (o le variabili) usati come argomenti/valore di ritorno **nella invocazione** della funzione

□ Esempio:

```
function y = converti(x)
```

```
    y = x*(1000/3600);
```

```
>> v = converti(50);
```

y ed x sono parametri formali
v e 50 sono parametri attuali

I parametri (2)

- ❑ Qualsiasi tipo di parametri è ammesso (scalari, vettori, matrici, strutture, ecc.)
- ❑ I parametri attuali vengono associati a quelli formali in base alla posizione: il primo parametro attuale viene associato al primo formale, il secondo parametro attuale al secondo parametro formale, ecc.
- ❑ Un invocazione di funzione deve contenere un numero di parametri attuali identico al numero di parametri formali
- ❑ Esempio

>> [x,y]=sumProd(4,5)

```
function [s,p]=sumProd(a,b)
s=a+b;
p=a*b;
```

- ❑ Quando una funzione viene eseguita, viene creato un **workspace "locale"** in cui vengono memorizzate tutte le variabili usate nella funzioni (**inclusi i parametri formali**)
 - ▶ All'interno delle funzioni **non si può accedere al workspace "principale"** (nessun conflitto di nomi)
 - ▶ Quando la funzione viene eseguita, **il workspace "locale" viene distrutto!**
- ❑ Quando viene invocata una funzione:
 - ▶ Vengono calcolati i valori dei parametri attuali di ingresso
 - ▶ Viene creato un workspace "locale" per la funzione
 - ▶ I valori dei parametri attuali di ingresso vengono copiati nei parametri formali all'interno del workspace "locale"
 - ▶ Viene eseguita la funzione
 - ▶ Vengono copiati i valori di ritorno dal workspace "locale" a quello "principale" (nei corrispondenti parametri attuali)
 - ▶ Il workspace "locale" viene distrutto

Esecuzione di una funzione: esempio

```
(1) >> x=3;  
(2) >> w=2;  
(3) >> r = funz(4);
```

```
function y = funz(x)  
y = 2*x;      %(1')  
x = 0;        %(2')  
z=4;          %(3')  
x=w+1;        %(4')
```

W "principale" dopo (2)

```
x=3  
w=2
```

W "principale" dopo (3)

```
x=3  
w=2  
r= 8
```

~~W "locale" dopo (3)~~

W "locale" dopo(1')

```
x=4  
y=8
```

W "locale" dopo(3')

```
x=0  
y=8  
z=4
```

W "locale" (4')

```
x=0  
y=8  
z=4  
w=? → errore
```

- ❑ Come nel caso degli script le funzioni possono essere scritte in file di testo sorgenti
 - ▶ Devono avere estensione .m
 - ▶ Devono avere lo stesso nome della funzione
 - ▶ Devono iniziare con la parola chiave **function**
- ❑ Attenzione a non “ridefinire” funzioni esistenti
 - ▶ `exist('nomeFunzione')` → 0 se la funzione non esiste