# Laboratorio di Informatica T

Introduzione a Matlab ed Espressioni

# Cos'è Matlab

**Matlab è un sistema software per calcolo numerico**

Sistema = collezione di componenti SW (pensiamoli come "programmi")

- Ampiamente utilizzato per applicazioni di ingegneria
- Si interagisce con il SW mediante un linguaggio di programmazione

Di seguito con "Matlab" ci riferimento sia al SW che al linguaggio

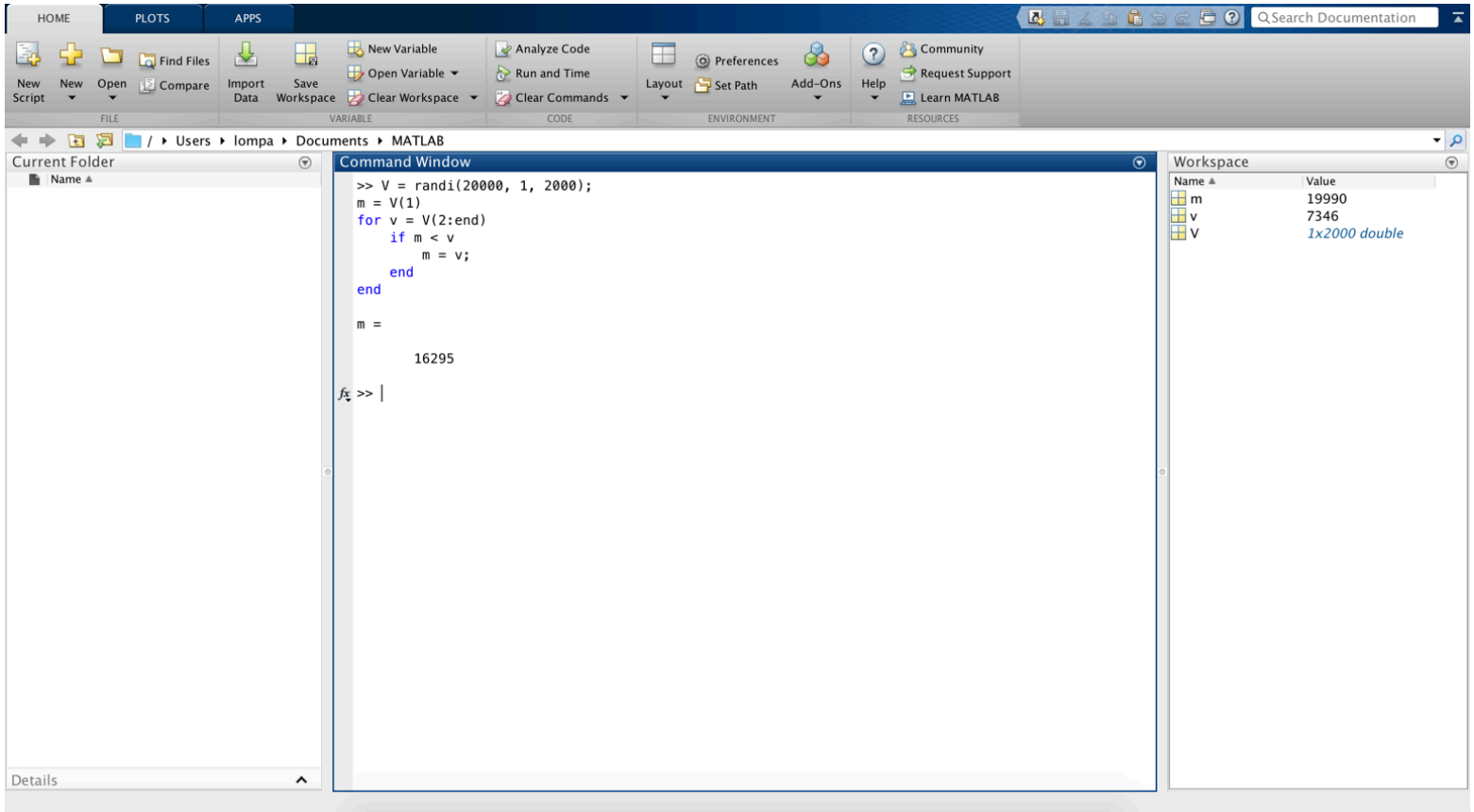
**Come accedere all'utilizzo di Matlab**

- Matlab è installato sulle macchine del laboratorio
- Matlab è scaricabile gratuitamente degli studenti UNIBO

È importantissimo scaricalo per esercitarsi a casa

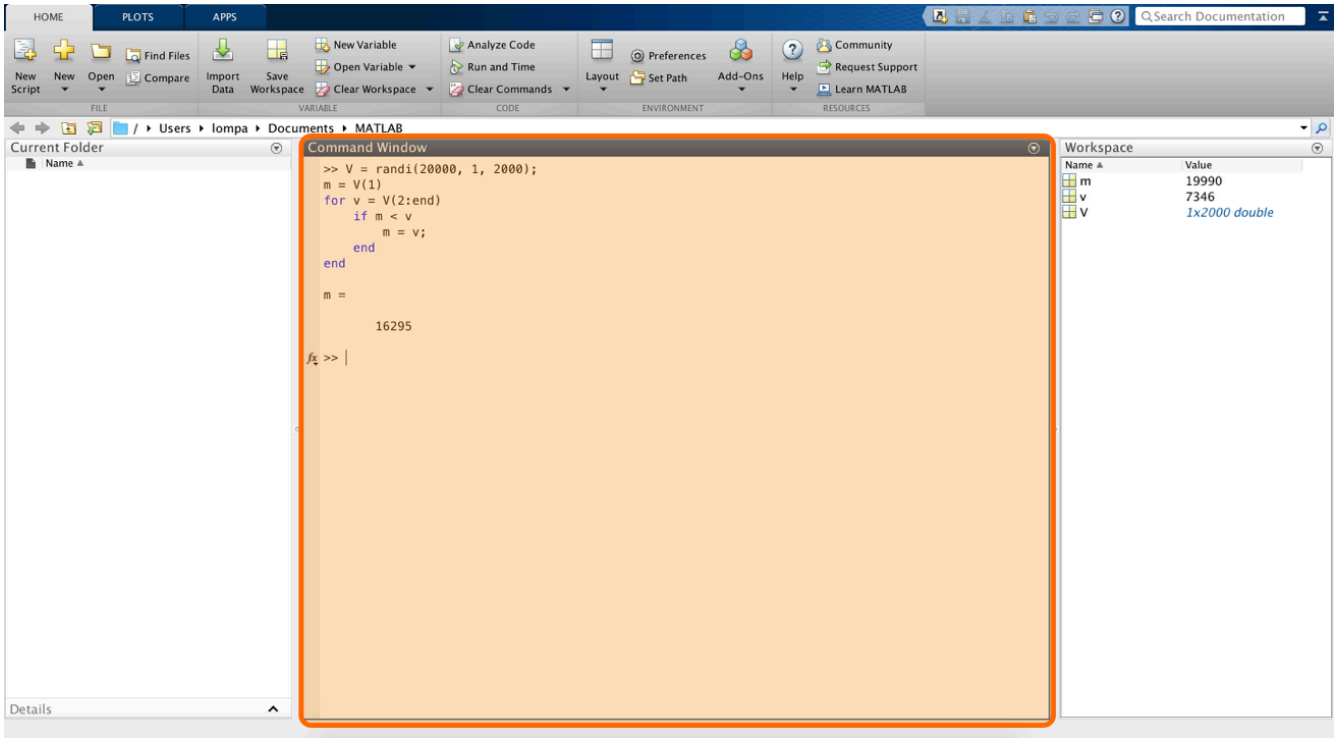
# Interfaccia Utente di Matlab

Matlab ha una interfaccia grafica (Graphical User Interface)



# Componenti della GUI di Matlab

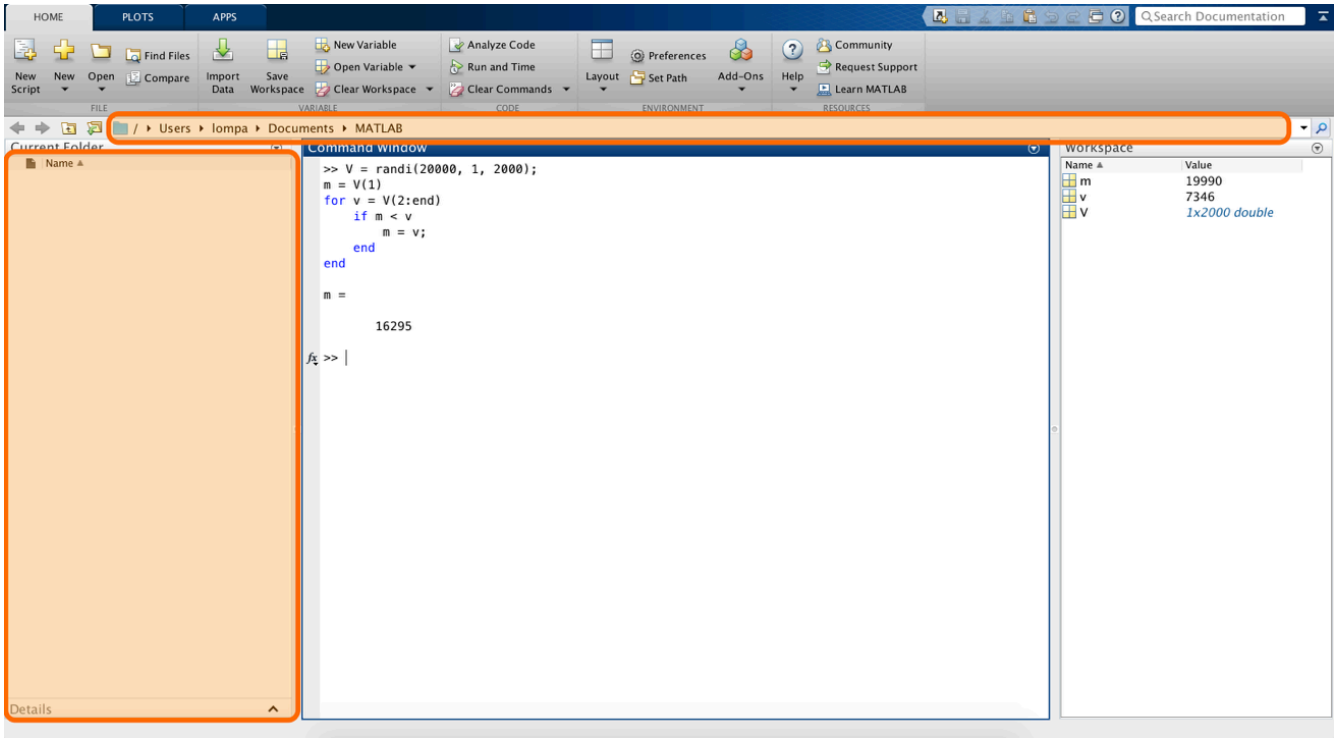
L'elemento più importante della GUI è la finestra dei comandi




Il simbolo ">>" si chiama prompt ed indica dove potete scrivere

# Componenti della GUI di Matlab

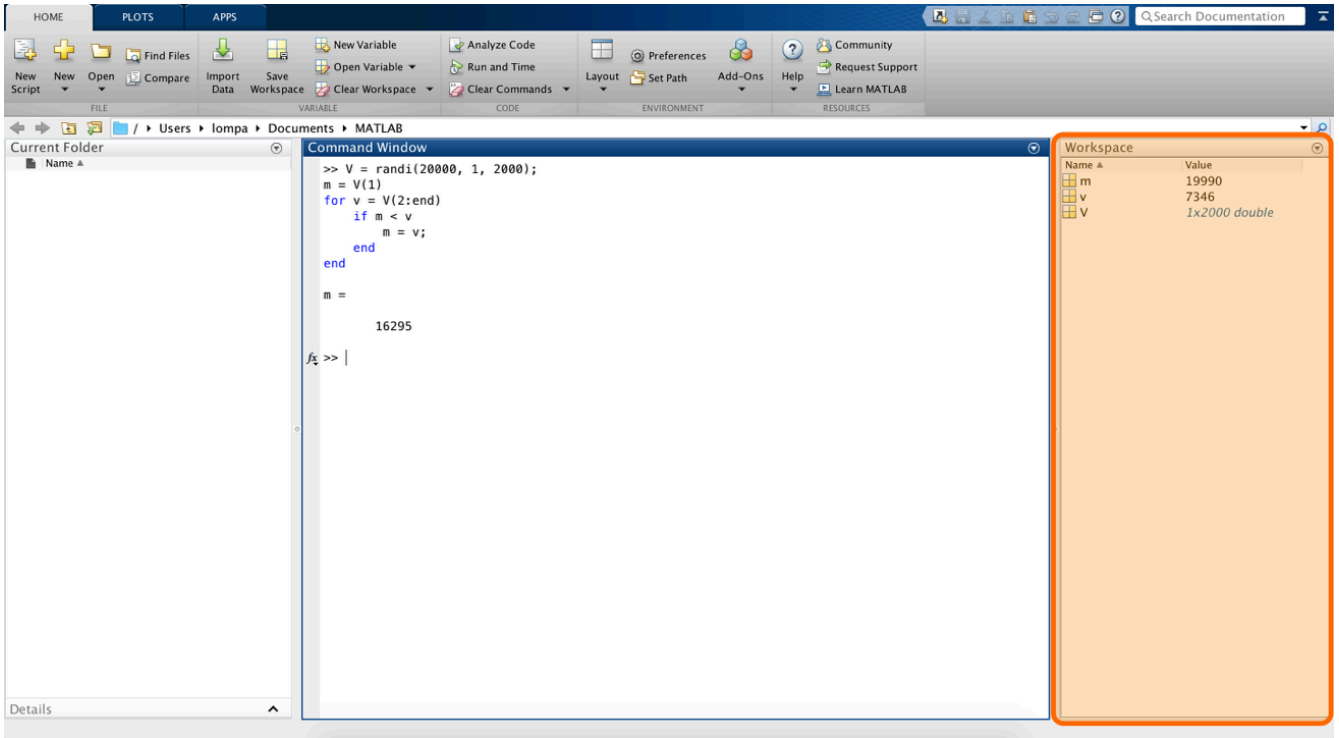
Nella colonna di SX trovate un file browser




Permette di esplorare la cartella corrente ed il suo contenuto

# Componenti della GUI di Matlab

Nella colonna di DX trovate la finestra di workspace




Permette di visualizzare le variabili che sono state definite

# Il "Linguaggio Matlab"

Il "linguaggio Matlab" si basa su pochi concetti fondamentali

- Tipi di dato elementari
- Tipi di dato composti
- Variabili
- Espressioni e funzioni
- Istruzioni di controllo del flusso

E diversi concetti più complessi, ma meno essenziali:

- Range, Funzioni anonime, ...

Oggi cominciamo a vederli insieme

# Tipi di Dato Elementari: Numeri Reali

**Il tipo di dato più importante in Matlab sono i numeri "reali"**

Sintassi (notazione "normale):

```
3.14159     % Prime cifre di pi-greco
1000        % Il testo dopo "%" viene ignorato!
```

Sintassi (notazione scientifica):

```
314159e-5   % e-5 sta per "* 10^-5"
1e3         % e3 sta per "* 10^3"
```

Semantica: un numero in $\mathbb{R}$

**Provate a digitare un numero e battere [INVIO]!**

# Tipi di Dato Elementari: Numeri Complessi

**Matlab permette di trattare anche i numeri complessi**

Sintassi:

```
3 + 2i
1 + 4j
```

- La lettera `i` indica la parte immaginaria
- La lettera `j` si può utilizzare in alternativa ad `i`
  - In elettrotecnica e teoria dei segnali si use $j$ invece di $i$

Semantica: un numero in $\mathbb{C}$

- Nota: i complessi ed i reali sono rappresentati in modo approssimato
- Vedrete in Analisi Numerica in che modo e cosa comporta

# Tipi di Dato Elementari: Valori Logici

**Matlab permette di trattare valori logici: "vero" o "falso"**

```
true     % Corrisponde a "vero"
false    % Corrisponde a "falso"
```

I due valori sono assimilabili a numeri:

- `true` è assimilabile al numero `1`
- `false` è assimilabile al numero `0`

Matlab però tiene traccia del fatto che si tratta di valori logici:

```
true  % Risp.: ans = logical 1
```

- Viene evidenziato che si tratta di un valore logico

# Variabili ed Assegnamento

Per memorizzare dati, Matlab utilizza il concetto di variabile:

**Una variabile è una astrazione per un dato in memoria**

Potete pensarla come un contenitore con un nome

- Una variabile viene definita assegnandovi un valore
- Per farlo si utilizza l'operatore di assegnamento

Sintassi:

```
<variabile> = <dato>
```

# Variabili ed Assegnamento

**Vediamo un esempio:**

```
a = 5
```

- Quando premete [INVIO], viene definita la variabile `a` con valore `5`
- Controllate la finestra di workspace!

**Il contenuto di una variabile può essere cambiato:**

```
a = 5
a = 3
```

- Viene definita la variabile `a` con valore `5`
- Al passo successivo ad `a` viene assegnato il valore `3`

Una variabile è un contenitore, ricordate?

# Variabili ed Assegnamento

## Una variabile può contenere qualunque tipo di dato:

```
x = 1
x = 2 + i
x = false
```

- `x` contiene prima un reale, poi un complesso, poi un valore logico

## Il nome di una variabile:

- Deve iniziare con una lettera o con "_"
- I caratteri successivi possono essere lettere, "_", o numeri

```
r2d2 = 0          % È un nome valido
1var = 5          % Non valido
```

# Visualizzare il Contenuto di una Variabile

Per visualizzare il contenuto di una variabile basta scriverne il nome:

```
x       % Malab "risponde" con x = logical 0
r2d2    % Malab "risponde" con r2d2 = 0
```

- Funziona solo con le variabili che sono state definite
- Variabile non definita → nessuna risposta o errore

**Completamento mediante [TAB]:**

- Se iniziate a scrivere un nome di variabile...
- ...Potete cercare di completarlo automaticamente usando [TAB]

```
r2d2 [TAB]     % Viene completato in r2d2
```

# Variabile `ans`

Se digitate un numero e premete [invio]:

```
42
```

Matlab "risponde" con:

```
ans = 42
```

- Vuol dire che il numero **`42`** è stato inserito nella variabile **`ans`**
- **`ans`** è una variabile (quasi) come tutte le altre
- In questo modo il risultato non viene perduto

Provate a visualizzarne il contenuto!

# Altre Variabili Speciali

**Alcune variabili sono automaticamente disponibili:**

E.g.: pi-greco "**pi**"

```
pi % Risposta: pi = 3.1416 (solo alcune cifre)
```

Cosa succede se ridefinite una variabile speciale?

```
pi = 42
```

- La vecchia **pi** diventa non accessibile
- Potete riportare tutto come prima con:

```
clear all  % Elimina tutte la variabili definite
```

# Espressioni (o "Cosa Fare con i Dati")

**Una espressione è una notazione**
**che restituisce un valore quando viene eseguita**

Il processo per cui questo avviene si chiama anche valutazione:

- Quando scrivete una espressione e premete [INVIO]...
- ...Matlab la valuta e restituisce (o denota) un valore

Un modo per pensarla: il valore restituito "rimpiazza" l'espressione

**Si può usare una espressione ovunque sia richiesto un dato**

# Espressioni: Notazione per un Dato

**La notazione per un tipo di dato è un esempio di espressione:**

- Quando scrivete:

```
10
```

- quello che avete fatto è scrivere del testo
- Nel momento in cui premete [INVIO]
- Matlab valuta l'espressione ed ottiene il valore `10`

Il risultato dell'espressione viene memorizzato nella variabile **`ans`**

# Espressioni: Nome di Variabile

**Il nome di una variabile è una espressione semplice**

```
x
```

- Quando premete [INVIO] dopo il secondo passo…
- …Matlab controlla se la variabile `x` sia definita…
- …Se lo è, restituisce il valore corrente

Se la variabile non è definita viene riportato un errore

**Eccezione:** Il nome di una variabile non è una espressione se compare a sinistra dell'operatore di assegnamento "="

```
x = 10 % x in questo caso non è una espressione
```

# Espressioni: Chiamata a Funzione

Matlab fornisce un costrutto fondamentale per comporre espressioni:

**Si chiama chiamata a funzione una notazione che permette di eseguire un sotto-programma precedentemente definito**

Qualche esempio:

```
plus(2, 5) % Esegue una somma
minus(10, 3) % Esegue una sottrazione
times(2, 3) % Esegue una moltiplicazione
```

Chiamando una funzione si esegue il sotto-programma corrispondente

# Sintassi di una Chiamata a Funzione

**La sintassi per una chiamata a funzione è:**

```
<nome della funzione>(<dato>, <dato>, ...)
```

I dati tra parentesi rappresentano l'input del sotto-programma

- Si chiamano parametri

Tipicamente un funzione incapsula un algoritmo

- Quindi la funzione restituisce un risultato

Si chiama "funzione" per analogia con le funzioni matematiche:

- Accetta dati di ingresso e restituisce un risultato
- Ha addirittura la stessa notazione di una funzione matematica

# Qualche Esempio di Chiamata a Funzione

Vediamo qualche altro esempio di chiamata a funzione:

```
sin(3.14156)   % Seno (funzione trigonometrica)
cos(pi)        % Coseno
atan(1)        % Arcotangente
abs(-3)        % Valore assoluto
power(2, 3)    % potenza: 2^3
sqrt(4)        % Radice quadrata
exp(2)         % esponenziale: e^2
log(2.7183)    % logaritmo naturale
log10(100)     % Logaritmo in base 10
real(2 + i)    % Parte reale
imag(2 + i)    % Parte immaginaria
```

# Semantica di una Chiamata a Funzione

**Una chiamata a funzione viene valutata come segue:**

- Matlab valuta i parametri
- Matlab esegue il sotto-programma
- Quando il sotto-programma termina, Matlab recupera il risultato

**I parametri possono essere espressioni di qualunque tipo**

In particolare, possono essere delle altre chiamate a funzione:

```
plus(5, times(2, minus(7, 5)))
```

- In questo modo è possibile comporre espressioni
- Al momento di valutare i parametri, il processo viene ripetuto

# Semantica di una Chiamata a Funzione

La nostra espressione di partenza:

```
plus(5, times(2, minus(7, 5)))
```

Per valutare **`plus`** dobbiamo valutare i parametri:

- Il primo parametro è `5` (immediato da valutare)
- Il secondo parametro è una invocazione di **`times`**
- Per valutare **`times`** dobbiamo valutare i parametri:
  - Il primo parametro è `2`
  - Il secondo parametro è una invocazione di **`minus`**
  - Per valutare **`minus`** dobbiamo valutare i parametri:
    - Il primo parametro è `7`
    - Il secondo parametro è `5`

# Semantica di una Chiamata a Funzione

La nostra espressione di partenza:

```
plus(5, times(2, minus(7, 5)))
```

Per valutare **`plus`** dobbiamo valutare i parametri:

- Il primo parametro è `5` (immediato da valutare)
- Il secondo parametro è una invocazione di **`times`**
- Per valutare **`times`** dobbiamo valutare i parametri:
  - Il primo parametro è `2`
  - Il secondo parametro è quindi `2`

```
plus(5, times(2, 2))
```

# Semantica di una Chiamata a Funzione

La nostra espressione di partenza:

```
plus(5, times(2, minus(7, 5)))
```

Per valutare **`plus`** dobbiamo valutare i parametri:

- Il primo parametro è `5` (immediato da valutare)
- Il secondo parametro è quindi `4`

```
plus(5, 4)
```

# Semantica di una Chiamata a Funzione

La nostra espressione di partenza:

```
plus(5, times(2, minus(7, 5)))
```

L'intera espressione denota il valore `9`

```
9
```

# Funzioni con Sintassi Speciale: Operatori

**Le funzioni aritmetiche hanno anche una sintassi speciale**

- E.g. invece di scrivere `plus(2, 3)`...
- Possiamo usare l'operatore di somma "`+`" e scrivere `2 + 3`

Vediamo i principali operatori aritmetici_

```
A + B   % somma, e.g. 2 + 3
A - B   % sottrazione, e.g. 2 - 3
- A     % cambiamento di segno, e.g. -2
A * B   % prodotto, e.g.  2 * 3
A / B   % divisione, e.g.  2 / 3
A^B     % elevamento a potenza, e.g. 2^3
```

- `A` e `B` sono due espressioni che denotano un valore numerico
- Il risultato è un valore numerico

# Funzioni con Sintassi Speciale: Operatori

**Hanno sintassi speciale anche per gli operatori di confronto:**

```
A == B   % "vero" se uguali
A ~= B   % "vero" se diversi
A <  B   % "vero" se minore
A <= B   % ...
A >  B   % ...
A >= B   % ...
```

- `A` e `B` sono due espressioni che denotano un valore numerico
- Il risultato è un valore logico

Per esempio:

```
1 < 10    % Risp.: ans = logical 1
```

# Funzioni con Sintassi Speciale: Operatori

**Hanno sintassi speciale anche per gli operatori logici**

```
A & B    % "and": vero se A è vero _e_ B è vero
A | B    % "or": vero se A è vero _o_ B è vero
~A        % "not": vero se A è falso
```

- `A` e `B` sono due espressioni che denotano un valore logico
- Il risultato è un valore logico

Qualche esempio:

```
(1 < 2) & (-1 ~= 1)   % Risp.: ans = logical 1
(1 < 2) & (-1 == 1)   % Risp.: ans = logical 0
(1 < 2) | (-1 == 1)   % Risp.: ans = logical 1
(1 > 2) | (-1 == 1)   % Risp.: ans = logical 0
```

# Valori Logici e Numerici

Come accennato, i valori logici sono interpretabili come valori numerici:

$$\texttt{true} \longrightarrow 1$$

$$\texttt{false} \longrightarrow 0$$

Vale anche l'inverso: valori numerici sono interpretabili come logici:

$$0 \longrightarrow \texttt{false}$$

$$\neq 0 \longrightarrow \texttt{true}$$

Attenzione: ogni numero $\neq 0$ viene interpretato come "vero":

```
2 & 0    % Risp.: ans = logical 0
1 & 2    % Risp.: ans = logical 1
-1 | 0   % Risp.: ans = logical 1
```

# Associatività e Priorità

Consideriamo l'espressione:

```
2 * a + b + 5
```

Sappiamo (per regole di matematica) che va interpreta come:

```
((2 * a) + b) + 5
```

Questa interpretazione si basa su due proprietà degli operatori:

- Priorità, che determina quali operatori debbano essere risolti prima
- Associatività, per risolvere applicazioni multiple di un operatore

Matlab utilizza le stesse proprietà per interpretare gli operatori

# Associatività e Priorità

| Operatori, per priorità decrescente | Associatività |
|---|---|
| chiamata a funzione | sx |
| elevamento a potenza | sx |
| operatori + e – unari, operatore ~ | sx |
| moltiplicazione e divisione | sx |
| somma e sottrazione | sx |
| operatori di confronto | sx |
| operatore and logico | sx |
| operatore or logico | sx |

# Associatività e Priorità

Qualche esempio:

```
2 * 3 + 4           % --> (2 * 3) + 4
2 * plus(3, 4)      % --> (plus(2,3)) * 2
2 + 1 == 1 + 2      % --> (2 + 1) == (1 + 2)
1 == 1 | 2 < 3    % --> (1 == 1) | (2 < 3)
```

Per forzare un ordine diverso, si possono usare le parentesi

```
2 * (3 + 4)
```

Se c'è un assegnamento, esso viene eseguito dopo la valutazione:

```
a = 2^3 + 1  % in a viene inserito 7
```

# Ottenere Aiuto

## Matlab mette a disposizione un'enormità di funzioni

Ognuna ha la sua definizione!

- Il suo nome
- I suoi parametri
- La sua specifica (sotto-programma)

## Come fare ad orientarsi?

- Prima soluzione: imparare a memoria
- Seconda soluzione: Google o il manuale di Matlab
- Terza soluzione: usare una funzione!

# Ottenere Aiuto

**Per conoscere la specifica di una funzione con nome noto**

Potere usare i comandi **`help`** o **`doc`**

```
help <nome funzione>
doc <nome funzione>
```

- **`help`** visualizza un messaggio sulla finestra dei comandi
- **`doc`** apre una finestra esterna nella GUI

Provate con:

```
help plus
doc plus
```

# Ottenere Aiuto

## Se conoscete solo :parte del nome di una funzione

Potete iniziare a scrivere a poi premere [TAB]:

- Matlab mostrerà i possibili completamenti
- I completamenti sono nomi di variabili (come già visto)...
- ...E nomi di funzioni (quelli che ci interessano per **`help`**)

Provate con:

```
pl [TAB]
```

È una funzionalità molto utile

# Qualche Semplice Esercizio

Considerate le seguenti espressioni (ed assegnamenti):

```
a = 10 * 2 + 3
b = 2^3 - 1
log(exp(a))
(a + 2) - (b - 2)
c = a + b == 30
a + b * c
c & (a < 2^4)
a + b + c
abs(-2^3) == b + 1
```

- Cercate di capire cosa dovrebbero restituire
- Verificate cosa restituiscono effettivamente in Matlab
- Ricordate che i valori logici sono assimilabili a valori numerici