Input and Output Statements

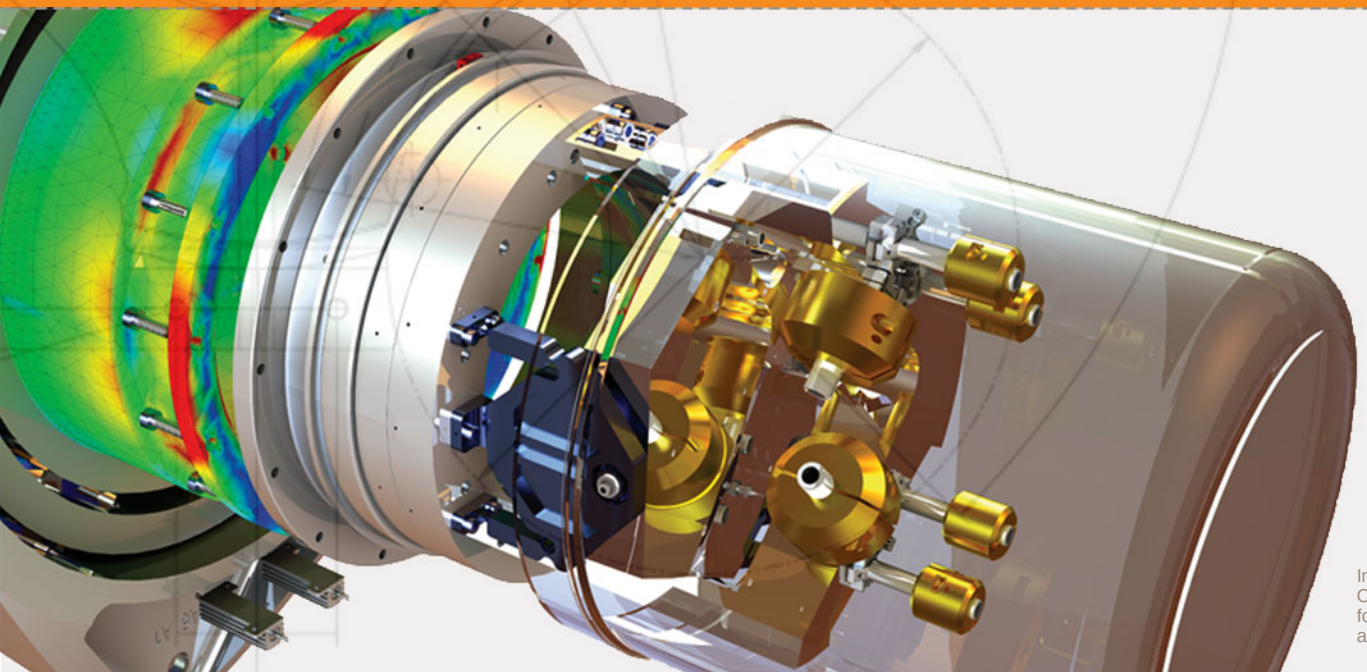


Image courtesy of National Optical Astronomy Observatory, operated by the Association of Universities for Research in Astronomy, under cooperative agreement with the National Science Foundation.

Overview

- Input
  - Input statements
  - Menu statements
- Output
  - Display statements
  - fprintf statements
  - Message boxes

Entering Inputs into a Program

There are a variety of ways to enter input variables into a MATLAB program. Three methods will be introduced initially:

1. Use ***hard-coded variables*** in the MATLAB program.
2. Use ***input statements*** within the program to prompt the user in the MATLAB command window to enter input variables.
3. Use ***menu statements*** within the program to prompt the user to click on a button to select an input.

Using Hard-Coded Variables

- So far, whenever we need a value in our program, we have simply created a variable:

```
x = 5;
pizza = 'pepperoni';
```

- Why not use hard-coded variables all the time?
  - Does not allow the user to choose different values
  - Program is only useful for one specific case
  - You might not know the needed values beforehand

Example

- Assume we write a script (circle_properties.m) with the following commands:

```
radius = 5;
circumference = 2*pi*radius;
area = pi*radius^2;
```

- What if we want to check a new circle with radius 10?

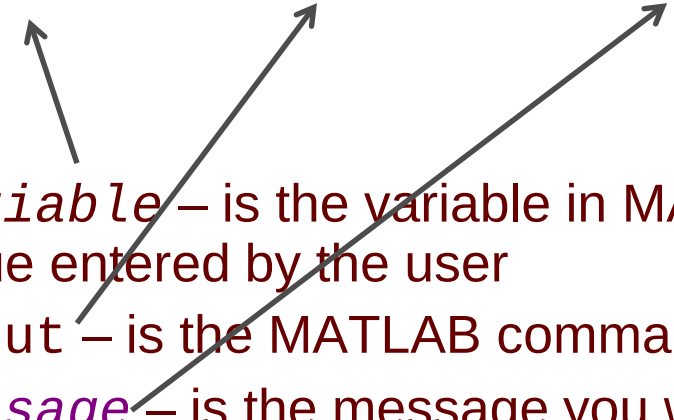
```
radius = 10;
circumference = 2*pi*radius;
area = pi*radius^2;
```

- What if you don't know the value?

Using Input Statements for Numbers

- One way to allow the user to choose a value is to use an input statement:

```
variable = input('message');
```



where

- *variable* – is the variable in MATLAB where you wish to store the value entered by the user
- `input` – is the MATLAB command used
- *message* – is the message you want to display to the user

Example

- Assume we write a script (circle_properties.m) with the following commands:

```
radius = input('Please enter the radius: ');
circumference = 2*pi*radius;
area = pi*radius^2;
```

- When we run this script:

```
>> circle_properties
Please enter the radius: 5
```

  - Value of 5 is stored into variable `radius`
  - `circumference` and `area` are computed as normal

Using Input Statements for Strings

- What happens if you want to allow the user to enter a letter or a word instead of a number?

```
>> letter = input('Please enter a letter: ');
Please enter a letter:
Error using input
Undefined function or variable 'a'.
```

- Add the second argument to the input statement to specify that the value will be a string:

```
variable = input('message', 's');
```

Using Menu Statements

- A menu statement can be used to allow the user to select from a set of predefined choices
- When you use a menu statement, MATLAB will create a new window with a set of buttons
- A menu statement takes the following form:

```
variable = menu('message', 'opt1', 'opt2',…);
```

where:

- *variable* – is the variable in MATLAB where you wish to store the button pressed by the user
- menu – is the MATLAB command used
- *message* – is the message you want to display to the user
- *'opt1','opt2',…* – define the number of buttons and the text displayed on them

Using Menu Statements

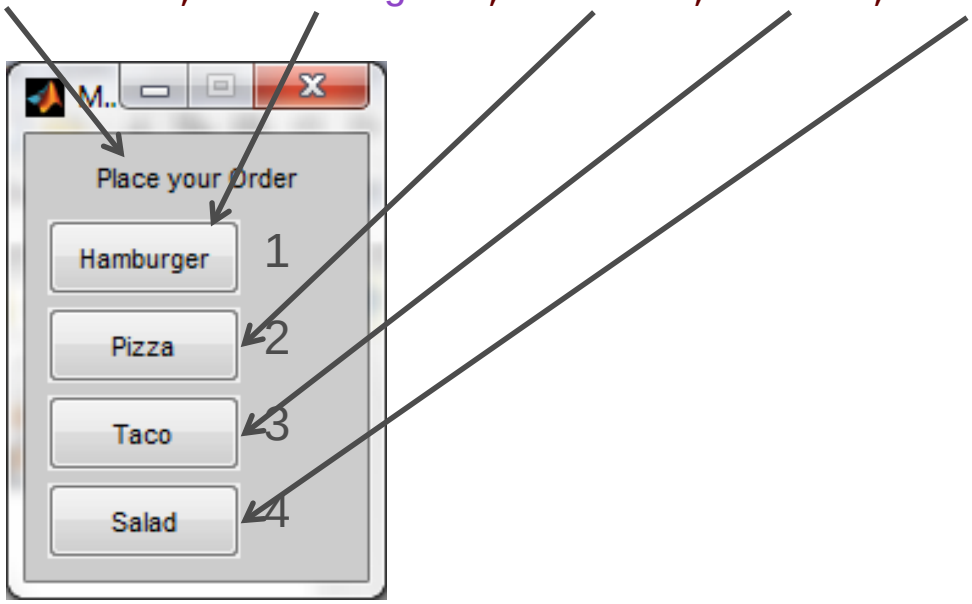
- What value is stored in the variable when the user presses one of the buttons?

```
variable = menu('message', 'opt1', 'opt2',…);
                           1       2
```

  - The order you list the options in your message statement determines the order the buttons are displayed
  - The first button is assigned a value of 1, the second a value of 2,…
  - When the user presses a button the value of the button is stored

Example

```
>> lunch = menu('Place your Order','Hamburger','Pizza','Taco','Salad')
```



When a button is pressed, the value of the button is stored into the variable `lunch`

Button	Value of `lunch`
Hamburger	1
Pizza	2
Taco	3
Salad	4

Displaying the Outputs of a Program

There are a variety of ways to display and store the output variables of a MATLAB program. Four methods will be introduced initially:

1. Do not use ***suppression*** for a given statement in the MATLAB program.
2. Use ***display statements*** (disp) within the program that will display the desired outputs in the MATLAB command window.
3. Use ***fprintf statements*** within the program that will that will display the desired outputs in the MATLAB command window and/or write the resulting outputs to a file.
4. Use ***message boxes*** within the program that will open a new window to display a message to the user.

Unsuppressing Statements

- The easiest way to display a value in MATLAB is to not suppress it
  - When you place a `;` at the end of a line in MATLAB, you are telling MATLAB that you do not want to see the result
  - If you leave the `;` off, MATLAB will show you the result

```
>> x = 10+5;
>>

>> x = 10+5

x =

    15

>>
```

- This works in a script as well
  - If you leave off a `;` in a script, the result will be displayed in the command window

Using Display Statements

- Often, we want to add some additional information to the value we display:
  - Message describing a value
  - Units
- You can display anything you want to the command prompt using the `disp` command:

```
disp(constant or variable);
```

where

- `disp` – is the MATLAB command used
- *`constant or variable`* – is what you want to display

Example

- Going back to our circle example, we can now add a couple of lines to display the results:

```
radius = input('Please enter the radius: ');
circumference = 2*pi*radius;
area = pi*radius^2;
disp('Circumference:'); disp(circumference);
disp('Area:'); disp(area);
```

- Running the script now, we would see:

```
>> circle_properties
Please enter the radius: 5
Circumference:
   31.4159

Area:
   78.5398

>>
```

Using fprintf Statements

- If we want additional formatting, we can use the `fprintf` statement
  - Mix strings with values
  - Specify the number of decimal places used
- `fprintf` statement takes the following form:

```
fprintf('Formatted Message', val1, val2,…);
```

where

- `fprintf` – is the MATLAB command used
- *`Formatted Message`* – is the message you want to display with formatting symbols
- *`val1, val2,…`* – are the values or strings that you want to insert into your formatted message

Using fprintf Statements

Insertion Formats

- `%s` – inserts a string
- `%i` – inserts an integer number
- `%f` – inserts a floating point (decimal) number

Escape Characters

- `\n` – moves to the next line
- `\t` – inserts a tab
- `\\` – displays the `\` character
- `%%` – displays the `%` character
- `''` – displays the `'` character

- When using `%f`, there are some additional settings you can use

  `%W.Pf`

  where W is the number of characters used to display the value and P is the number of decimal places displayed

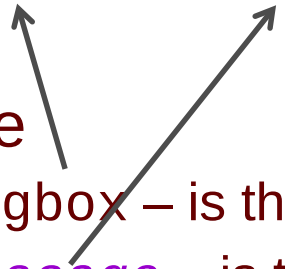
Examples

```
>> day = 'Wednesday'; weather = 'sunny'; temp = 75;

>> fprintf('Today is %s \n',day)
Today is Wednesday

>> fprintf('The weather today is %s and the temperature
is %i \n', weather,temp)
The weather today is sunny and the temperature is 75

>> fprintf('The weather today is %s and the temperature
is %0.3f \n',weather,temp)
The weather today is sunny and the temperature is 75.000

>> fprintf('The weather today is %s and the temperature
is %0.3f \n',temp,weather)
The weather today is K and the temperature is 115.000
```

Using Message Boxes

- If we want to output a message outside of the command window, we can use a message box
  - A message box creates a new window (similar to a menu) which displays a message
- Message boxes take the following form:

```
msgbox('Message');
```



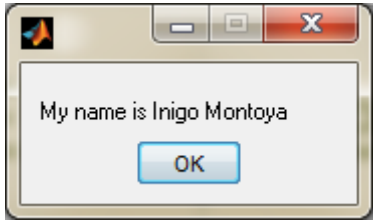
where

- `msgbox` – is the MATLAB command used
- *`Message`* – is the message you want to display to the user
  The *`Message`* has to be a string

Examples

- Simple message box:

```
>> msgbox('My name is Inigo Montoya');
```



- Constructing strings for message boxes

```
number_of_fingers = 6;
hand = 'right';
message = ['You don''t happen to have '
          num2str(number_of_fingers)
              ' fingers on your ' hand ' hand?'];
msgbox(message);
```