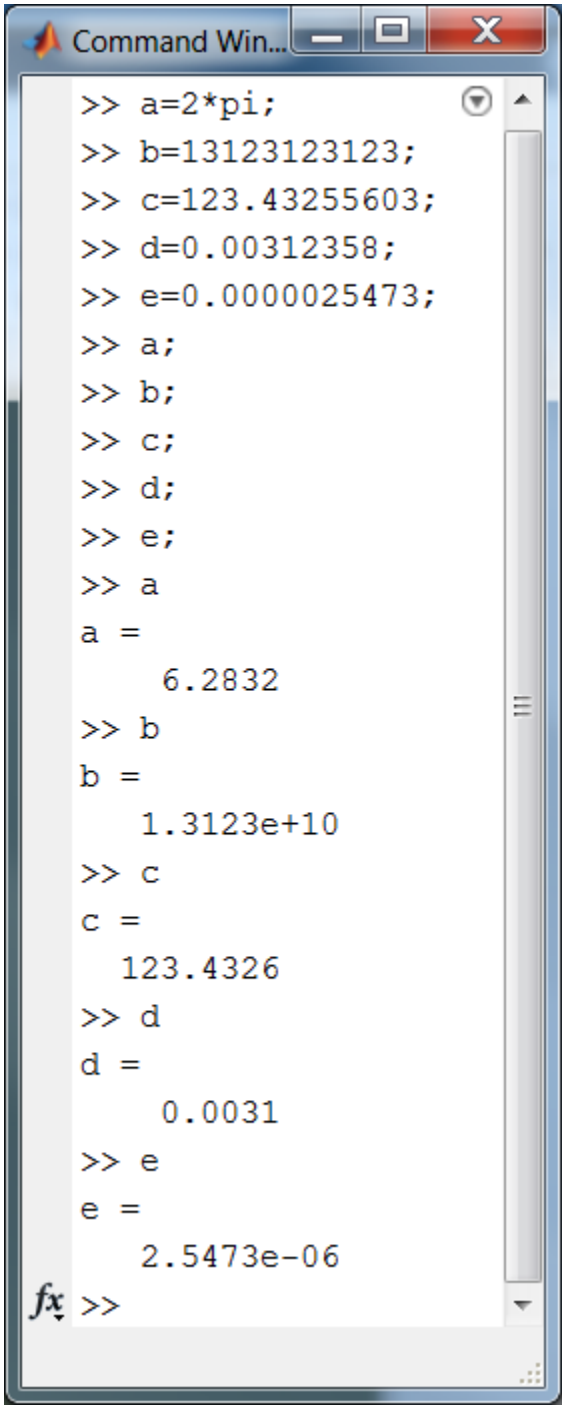
Module 2

Results Display

Semicolons

- A semicolon at end of a statement will suppress the result display.
- With no semicolon at the end of a statement, MATLAB displays result in the Command Window
- No control over appearance of result, e.g., how many lines, what precision in numbers.



Displaying Output

- MATLAB commands for displaying outputs:
  - **`disp()`** for some control of appearance
  - **`fprintf()`** for full control

disp function

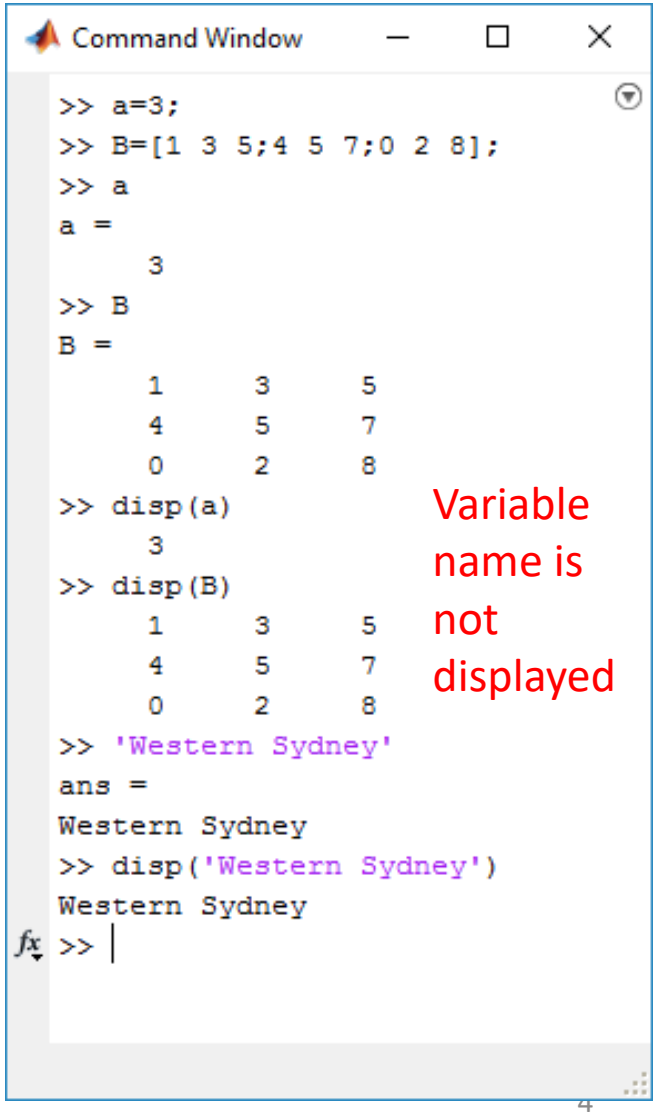
`disp()`

displays values and/or text

- Displays on a new line
- Doesn't display name of the variable

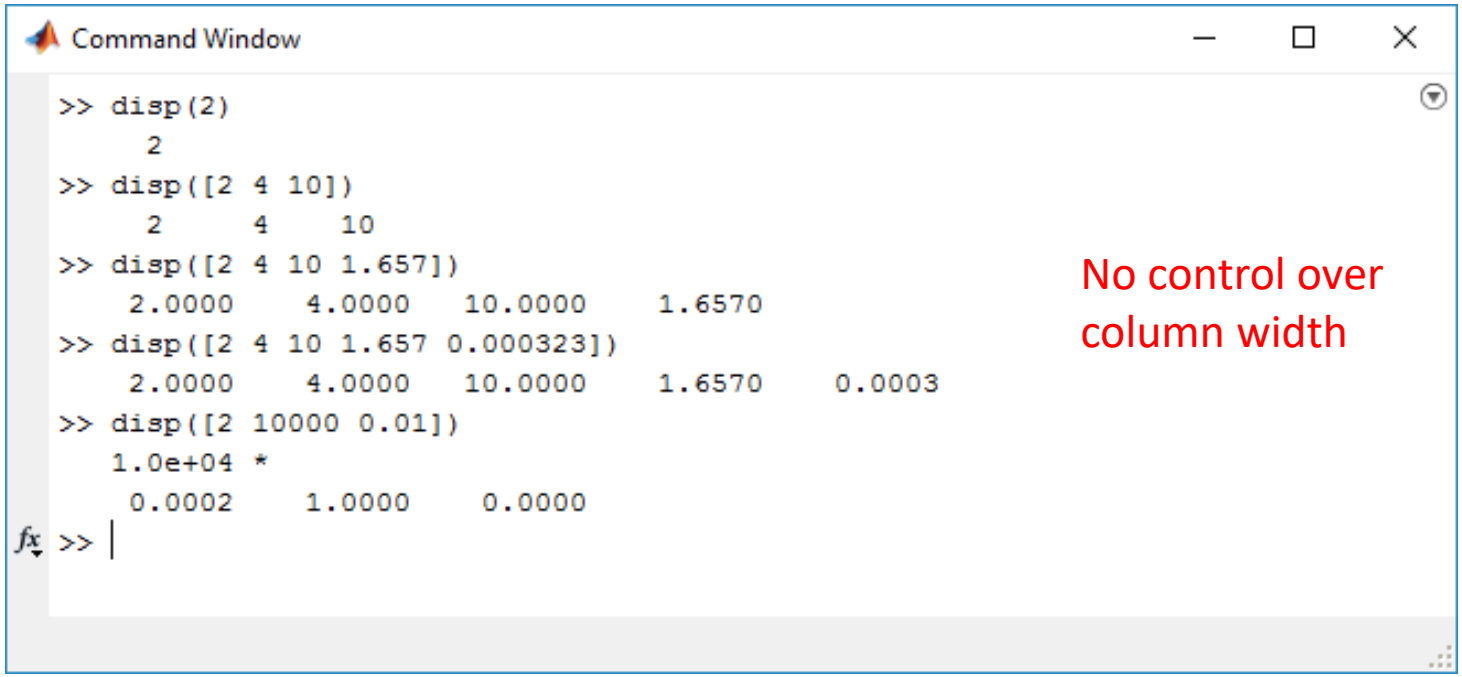
`disp(variable)` or

`disp('string')`



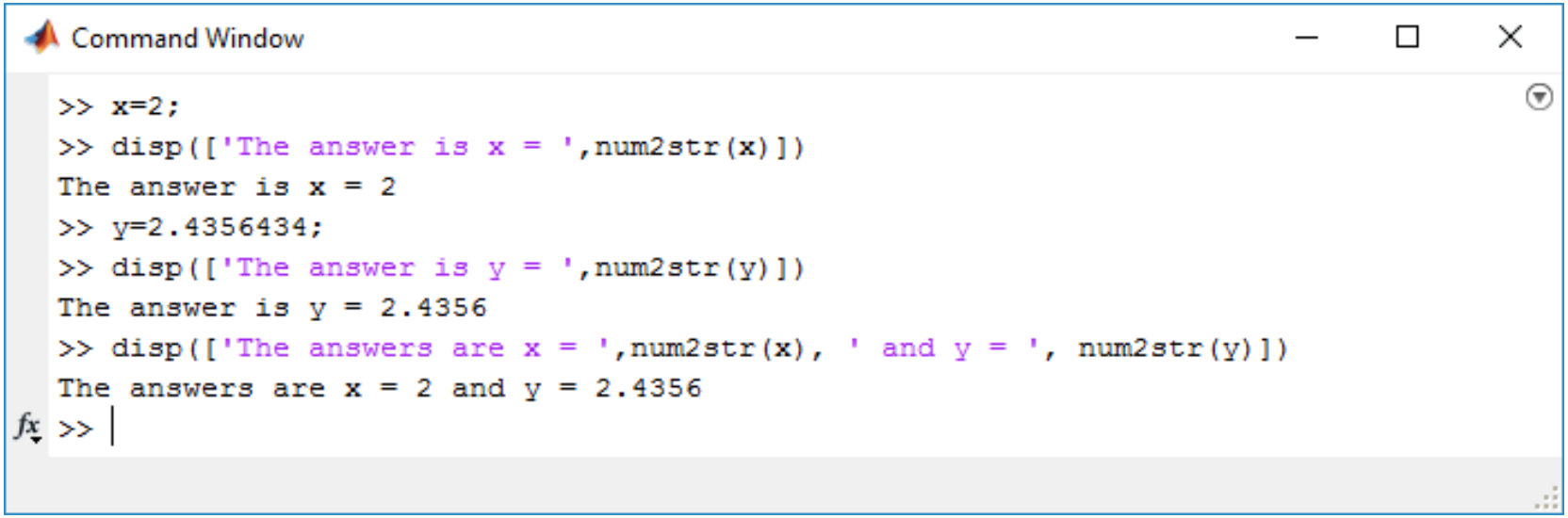
disp function

- Displaying numbers



disp function

- Displaying numbers and strings – convert the number to strings using **`num2str()`** and combine with strings to form a string array



fprintf function

`fprintf()`

- Means file print formatted
  - *formatted text* is text that can be read by people
  - *unformatted text* looks random to people but computers can read it
- Can write to screen or to a file
- Can mix numbers and text in output
- Have full control of output display
- **Not straight forward**

fprintf – display text

- Display text with

  `fprintf('Text to display')`

example:

```
>> fprintf( 'Western Sydney' )
Western Sydney>>
```

Prompt at the end of the line

There's a problem! The Command Window displays prompt (>>) at end of text, not at start of next line!

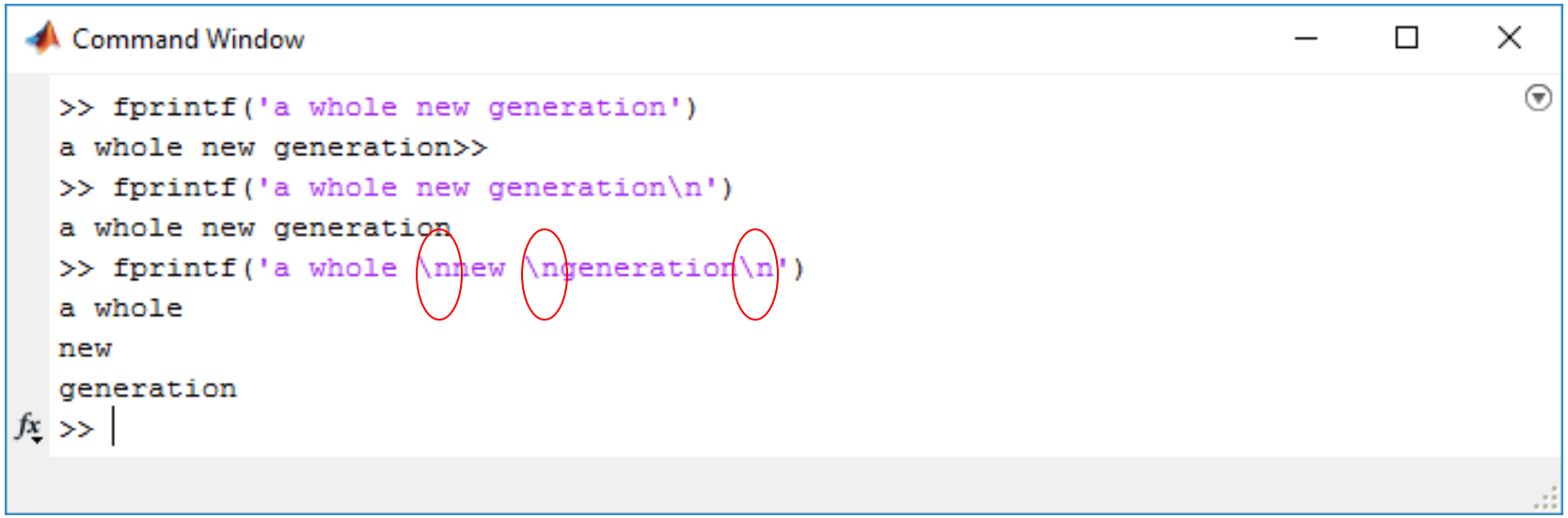
New Line Character \n

When using `fprintf()`, to make the next thing that MATLAB writes appear on the start of a new line, add the two

characters \n at the end of the fprintf text

```
>> fprintf('Western Sydney\n' )
Western Sydney
>>
```

- Can use multiple \n characters as necessary



fprintf - general use

fprintf(format spec, n1, n2, n3)

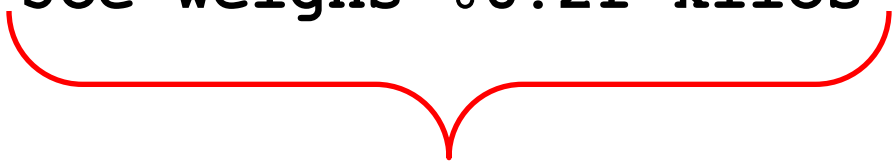
Variable to be displayed

Formatting operator

```
>> fprintf( 'Joe weighs %6.2f kilos', n1 )
```

Format specification

```
>> fprintf( 'Joe weighs %6.2f kilos', n1 )
```



Format specification

Format specification

- Entire format specification is a string
- May contain text and/or formatting operators
- Must be enclosed in SINGLE quotes, not double quotes, aka quotation marks (" ")

```
>> fprintf( 'Joe is %d weighs %f kilos', age, weight )
```

Arguments

- Number of arguments and formatting operators must be the same
- Leftmost formatting operator formats leftmost argument, 2nd to left formatting operator formats 2nd to left argument, etc.

Formatting operator

```
>> fprintf( 'Joe weighs %6.2f kilos', n1 )
```

- Formatting operator contains a number of parts, as shown below

- Formatting operator starts with a % symbol
- We will mostly be focusing on field width, precision and conversion character
- Formatting operator ends with the 'conversion character' (conversion character is f in this example, to ensure n1 as a floating-point number)

Conversion Characters

Value Type	Conversion	Details
Integer, signed	%d or %i	Base 10
Integer, unsigned	%u	Base 10
	%o	Base 8 (octal)
	%x	Base 16 (hexadecimal), lowercase letters a–f
	%X	Same as %x, uppercase letters A–F
Floating-point number	%f	Fixed-point notation (Use a precision operator to specify the number of digits after the decimal point.)
	%e	Exponential notation, such as 3.141593e+00 (Use a precision operator to specify the number of digits after the decimal point.)
	%E	Same as %e, but uppercase, such as 3.141593E+00 (Use a precision operator to specify the number of digits after the decimal point.)
	%g	The more compact of %e or %f, with no trailing zeros (Use a precision operator to specify the number of significant digits.)
	%G	The more compact of %E or %f, with no trailing zeros (Use a precision operator to specify the number of significant digits.)
Characters or strings	%c	Single character
	%s	Character vector or string array. The type of the output text is the same as the type of formatSpec.

Formatting operator

```
>> fprintf( 'Joe weighs %f kilos', n1 )
```

Common conversion characters

- %f fixed point (decimal always between 1's and 0.1's place,
  e.g., 3.14, 56.8
- %e scientific notation, e.g., 2.99e+008
- %d integers
- %g whichever is shorter, %f or %e
- %s string of characters

```
>> e = exp( 1 );
>> fprintf( 'e is about %4.1f\n', e )
e is about  2.7
>> fprintf( 'e is about %10.8f\n', e )
e is about 2.71828183
>> fprintf( 'e is about %10.8e', e )
e is about 2.71828183e+000
>> fprintf( 'e is about %10.2e', e )
e is about  2.72e+000
>> fprintf( 'e is about %f\n', e )
e is about 2.718282
```

Noteworthy ideas

Use escape characters to display characters used in formatting specification

- To display a percent sign, use %% in the text
- To display a single quote, use '' in the text (two sequential single quotes)
- To display a backslash, use \\ in the text (two sequential backslashes)

Make the following strings

- Mom's apple 3.14
- Mom's apple 3.1415926
- Mom's apple 3.1e+000

```
>> fprintf( 'Mom''s apple %.2f\n', pi )
Mom's apple 3.14
>> fprintf( 'Mom''s apple %.7f\n', pi )
Mom's apple 3.1415927
>> fprintf( 'Mom''s apple %.1e\n', pi )
Mom's apple 3.1e+000
```

Format specifications are often long. Can break up the formatting specification string by

1. Put an open square bracket ([ ) in front of first single quote
2. Put a second single quote where you want to stop the line
3. Follow that quote with an ellipsis ("…" three periods)
4. Press ENTER, which moves cursor to next line
5. Type in remaining text in single quotes
6. Put a close square bracket ( ])
7. Put in the rest of the fprintf command

Example

```
>> weight = 57.3;
>> age = 17;
>> fprintf( ['Tim weighs %.1f kgs'...
' and is %d years old'], weight, age )

 Tim weighs 57.3 kgs and is 17 years old
```

Exercise

- **%8.2f** specifies that there can be no less than 8 characters in the displayed output, and that two of them are to follow the decimal point.

```
weight= 57638.75;
fprintf('The weight is %8.2f pounds \n', weight)
fprintf('The weight is %50.2f pounds \n', weight)
```

```
The weight is 57638.75 pounds
The weight is                                        57638.75 pounds
```

  - Notice the large gap in the second output between 'is' and '57638.75 pounds'.
  - Use %% to have a % sign show up in output.

`fprintf()` is *vectorized*, i.e., when vector or matrix in arguments, command repeats until all elements displayed

- Uses matrix data column by column

```
x=1:5;
y=sqrt(x);
T=[x; y]
fprintf('If the number is: %i, its square root is: %f\n',T)
```

Create a vector x.

Create a vector y.

Create 2 × 5 matrix T, first row is x, second row is y.

The `fprintf` command displays two numbers from T in every line.

When this script file is executed the display in the Command Window is:

```
T =
    1.0000    2.0000    3.0000    4.0000    5.0000
    1.0000    1.4142    1.7321    2.0000    2.2361
If the number is: 1, its square root is: 1.000000
If the number is: 2, its square root is: 1.414214
If the number is: 3, its square root is: 1.732051
If the number is: 4, its square root is: 2.000000
If the number is: 5, its square root is: 2.236068
```

The 2×5 matrix `T`.

The `fprintf` command repeats 5 times, using the numbers from the matrix `T` column after column.