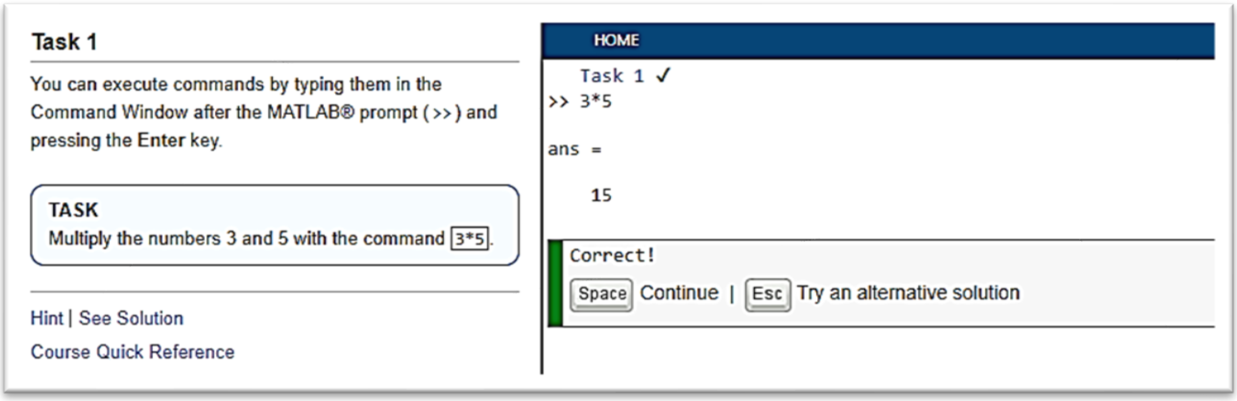


Task 1

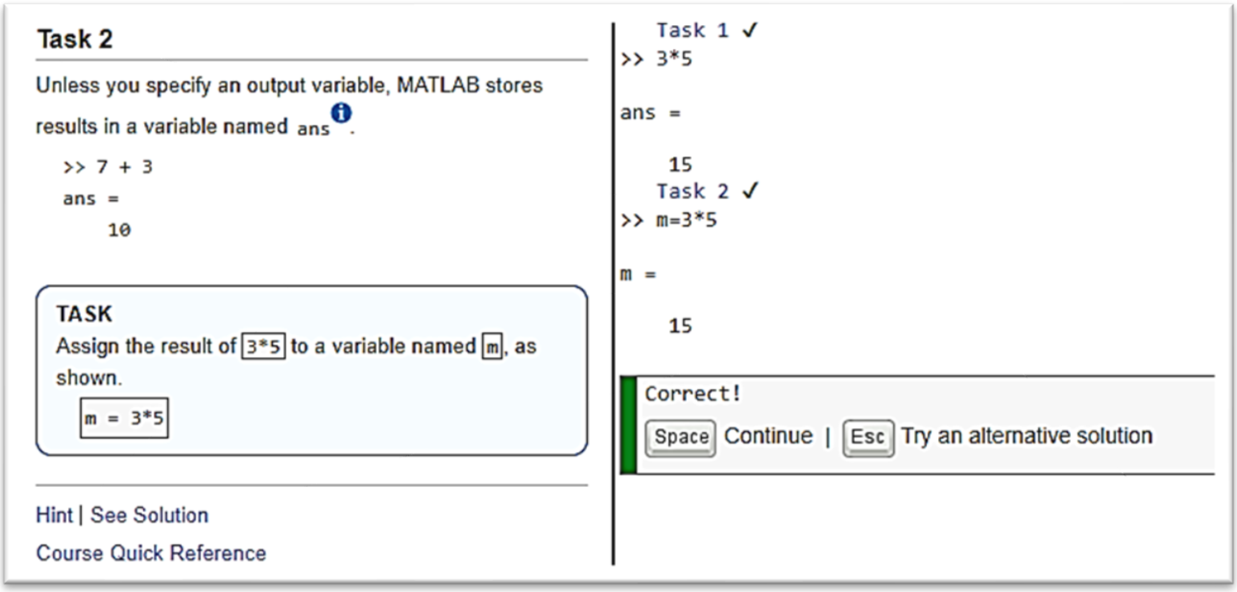
You can execute commands by typing them in the Command Window after the MATLAB® prompt (>>) and pressing the **Enter** key.

TASK
Multiply the numbers 3 and 5 with the command `3*5`.

Hint | See Solution
Course Quick Reference

```
  Task 1 ✓
>> 3*5

ans =

    15
```

Correct!
Space Continue | Esc Try an alternative solution



Task 2

Unless you specify an output variable, MATLAB stores results in a variable named `ans`.

```
>> 7 + 3
ans =
    10
```

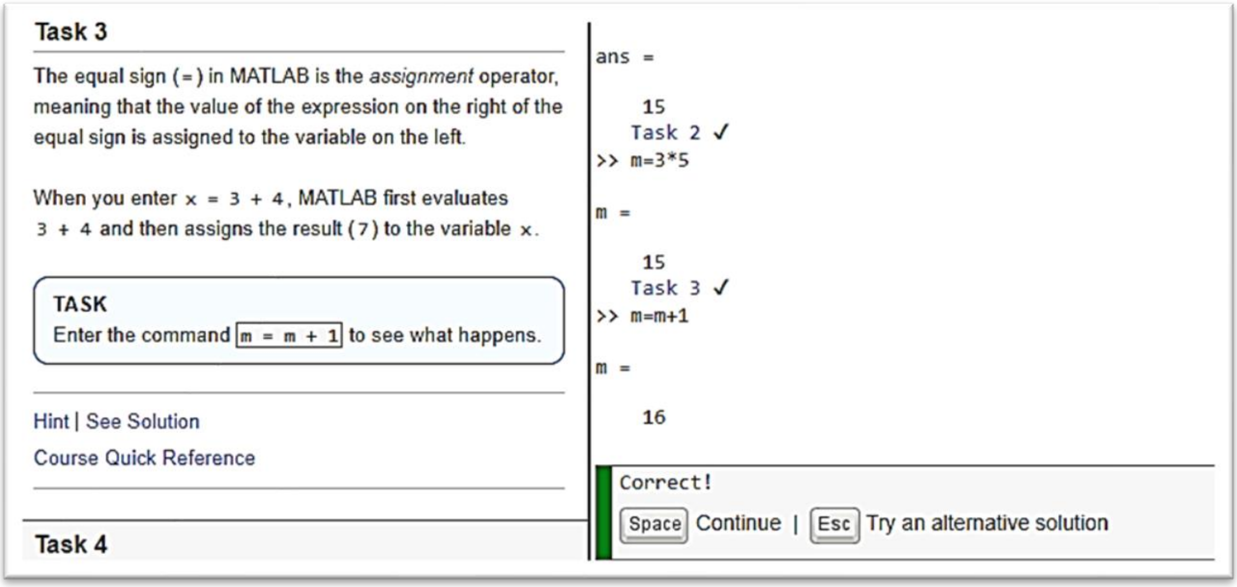
TASK
Assign the result of `3*5` to a variable named `m`, as shown.

```
m = 3*5
```

Hint | See Solution
Course Quick Reference

```
  Task 1 ✓
>> 3*5

ans =

    15
  Task 2 ✓
>> m=3*5

m =

    15
```

Correct!
Space Continue | Esc Try an alternative solution



Task 3

The equal sign (`=`) in MATLAB is the *assignment* operator, meaning that the value of the expression on the right of the equal sign is assigned to the variable on the left.

When you enter `x = 3 + 4`, MATLAB first evaluates `3 + 4` and then assigns the result (`7`) to the variable `x`.

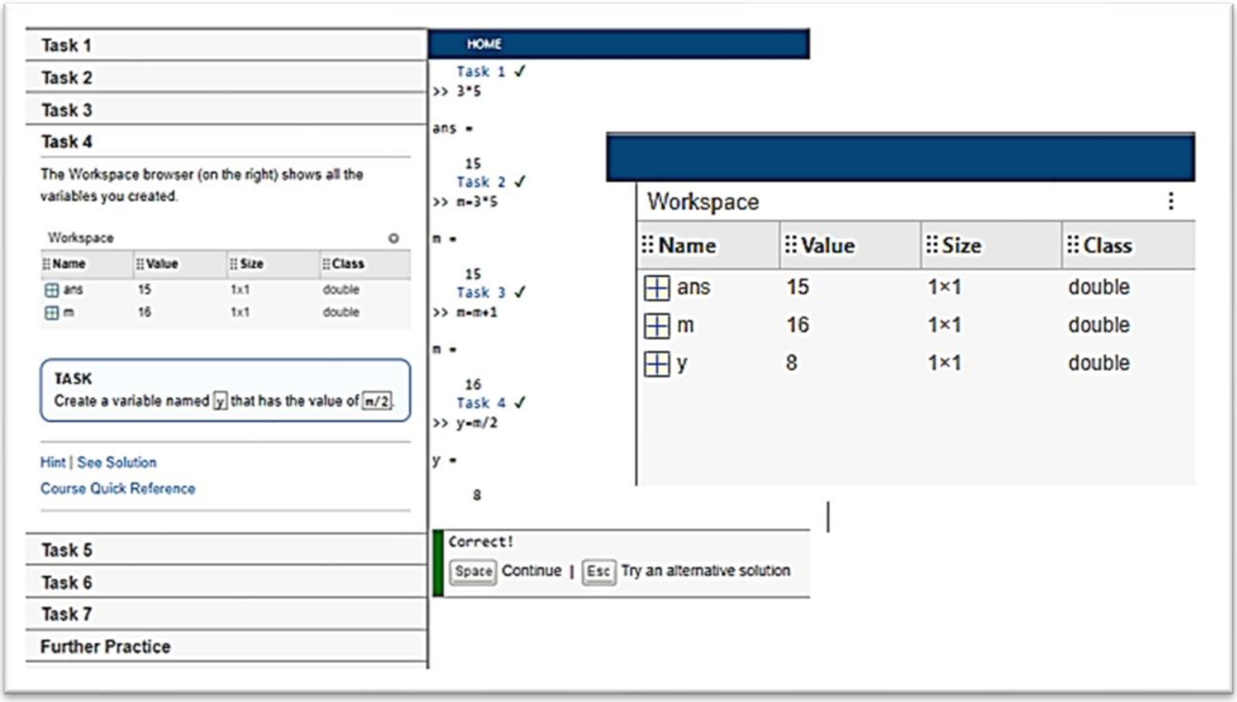
TASK
Enter the command `m = m + 1` to see what happens.

Hint | See Solution
Course Quick Reference

Task 4

```
ans =

    15
  Task 2 ✓
>> m=3*5

m =

    15
  Task 3 ✓
>> m=m+1

m =

    16
```

Correct!
Space Continue | Esc Try an alternative solution



The Workspace browser (on the right) shows all the variables you created.

Workspace

Name	Value	Size	Class
ans	15	1x1	double
m	16	1x1	double

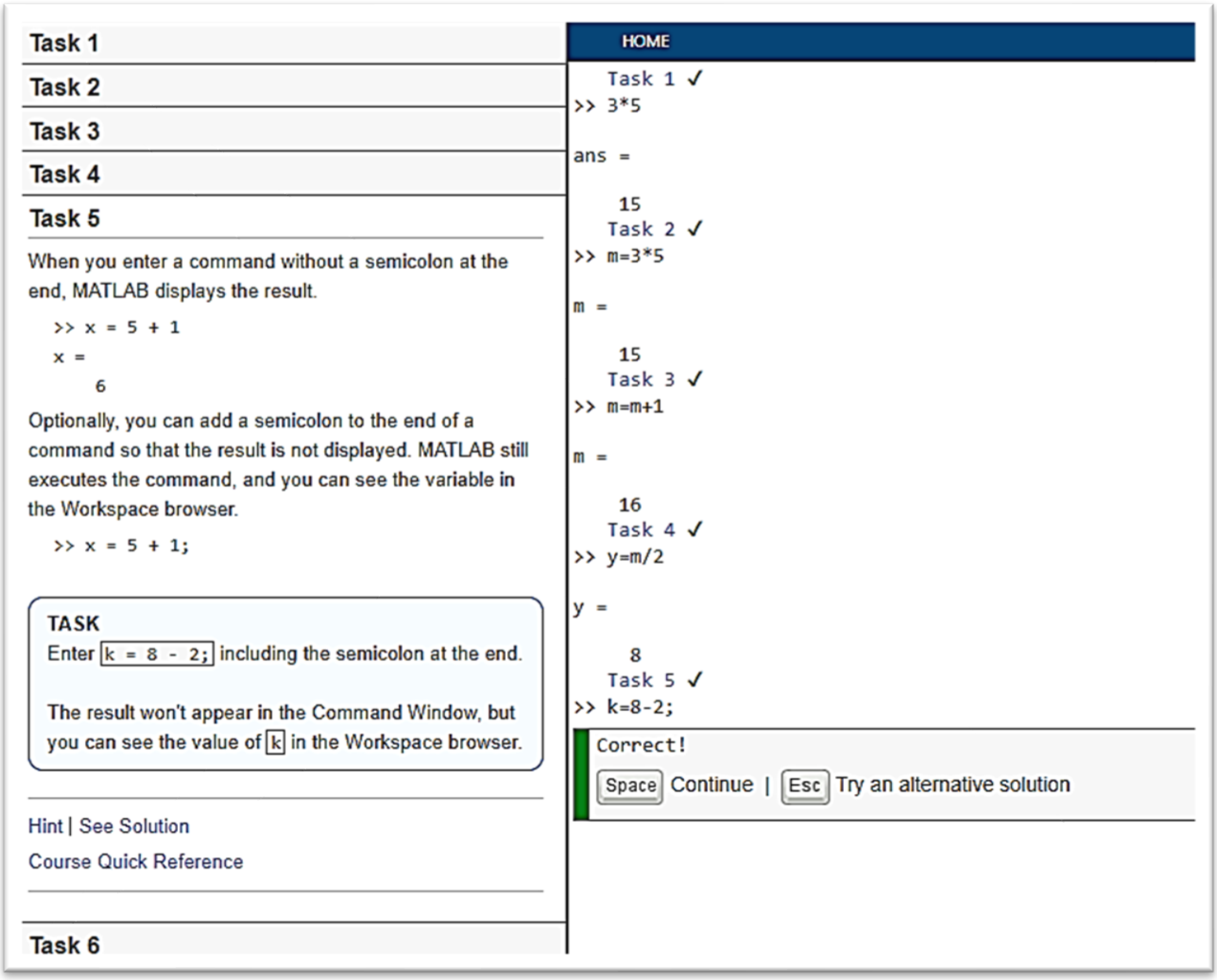
TASK
Create a variable named y that has the value of m/2.

```
  Task 1 ✓
>> 3*5

ans =

   15
  Task 2 ✓
>> m=3*5

m =

   15
  Task 3 ✓
>> m=m+1

m =

   16
  Task 4 ✓
>> y=m/2

y =

    8
```

Correct!

Workspace

Name	Value	Size	Class
ans	15	1×1	double
m	16	1×1	double
y	8	1×1	double



When you enter a command without a semicolon at the end, MATLAB displays the result.

```
>> x = 5 + 1
x =
     6
```

Optionally, you can add a semicolon to the end of a command so that the result is not displayed. MATLAB still executes the command, and you can see the variable in the Workspace browser.

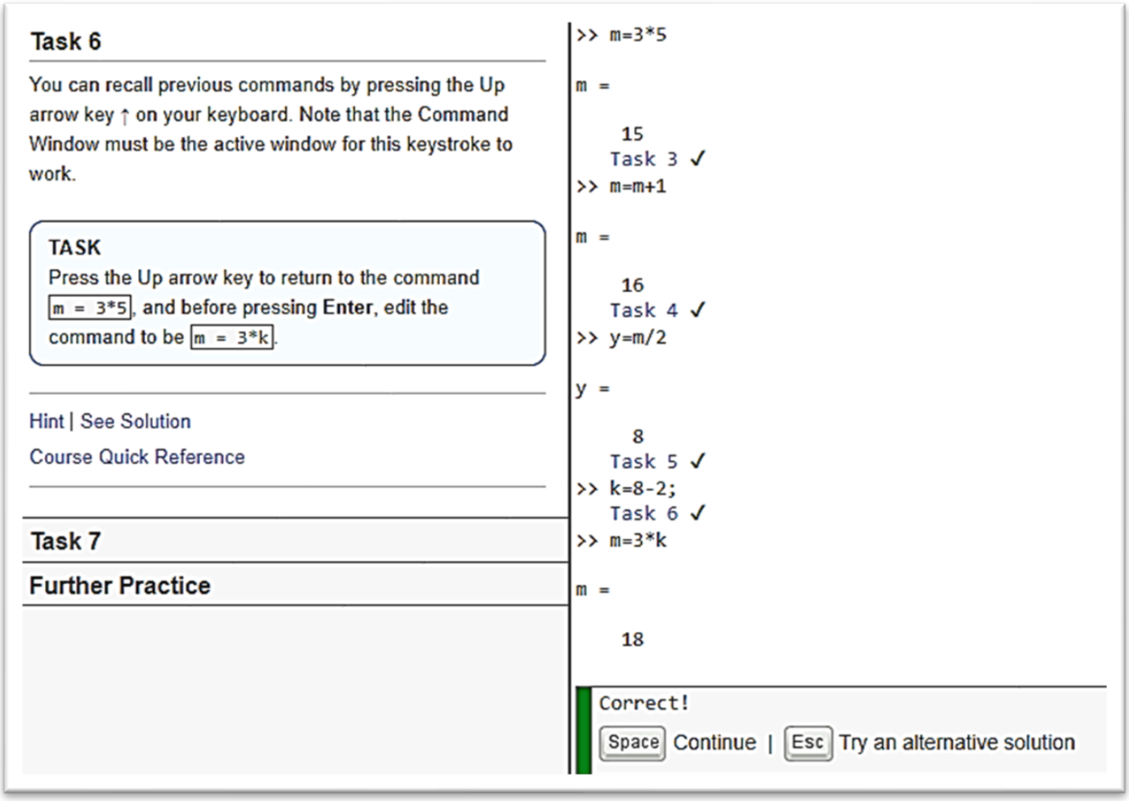
```
>> x = 5 + 1;
```

TASK
Enter `k = 8 - 2;` including the semicolon at the end.

The result won't appear in the Command Window, but you can see the value of k in the Workspace browser.

```
  Task 1 ✓
>> 3*5

ans =

   15
  Task 2 ✓
>> m=3*5

m =

   15
  Task 3 ✓
>> m=m+1

m =

   16
  Task 4 ✓
>> y=m/2

y =

    8
  Task 5 ✓
>> k=8-2;
```

Correct!



Task 6

You can recall previous commands by pressing the Up arrow key ↑ on your keyboard. Note that the Command Window must be the active window for this keystroke to work.

TASK
Press the Up arrow key to return to the command `m = 3*5`, and before pressing **Enter**, edit the command to be `m = 3*k`.

Hint | See Solution

Course Quick Reference

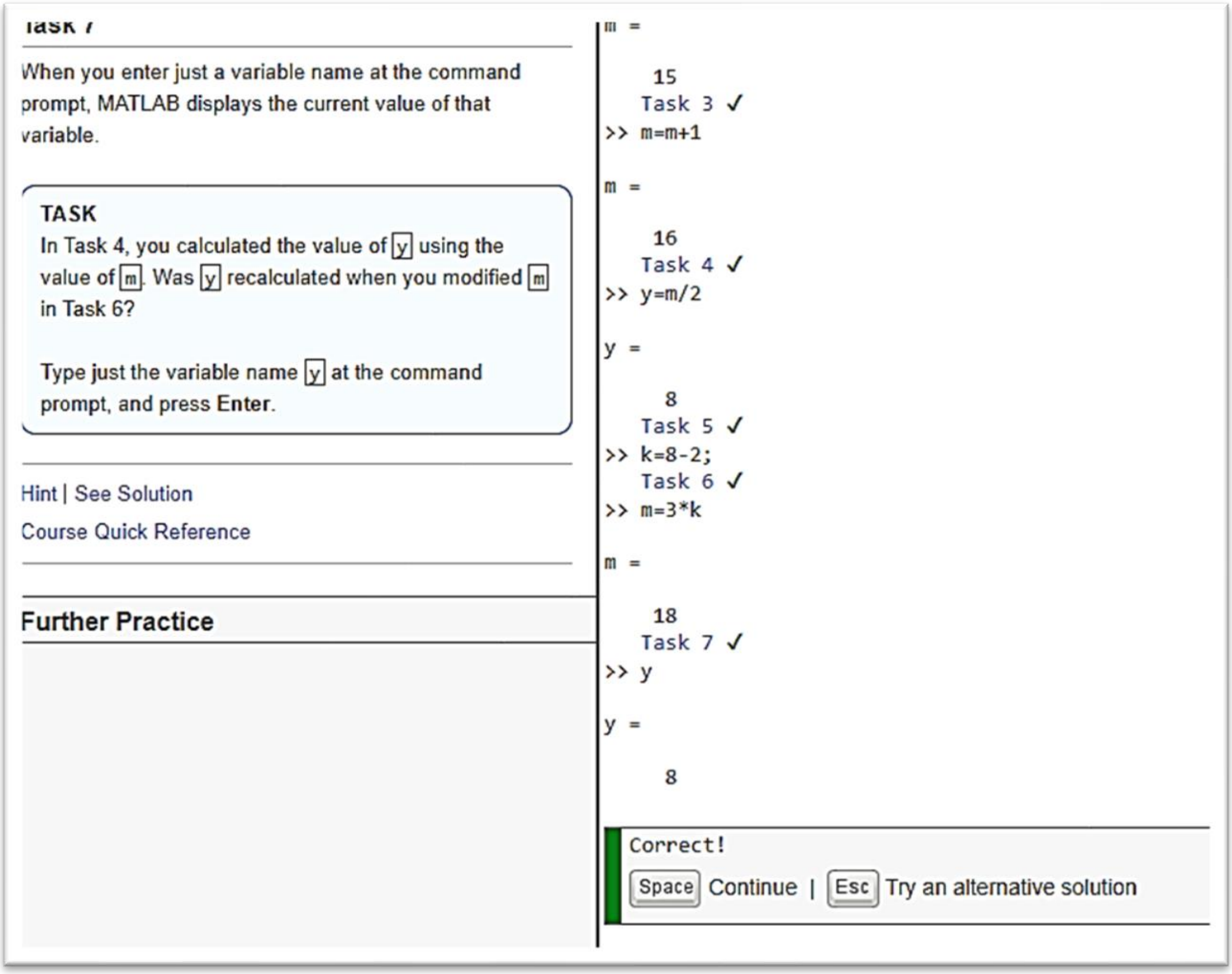
Task 7

Further Practice

```
>> m=3*5

m =

    15
   Task 3 ✓
>> m=m+1

m =

    16
   Task 4 ✓
>> y=m/2

y =

     8
   Task 5 ✓
>> k=8-2;
   Task 6 ✓
>> m=3*k

m =

    18
```

Correct!

Space Continue | Esc Try an alternative solution



Task 7

When you enter just a variable name at the command prompt, MATLAB displays the current value of that variable.

TASK
In Task 4, you calculated the value of `y` using the value of `m`. Was `y` recalculated when you modified `m` in Task 6?

Type just the variable name `y` at the command prompt, and press **Enter**.

Hint | See Solution

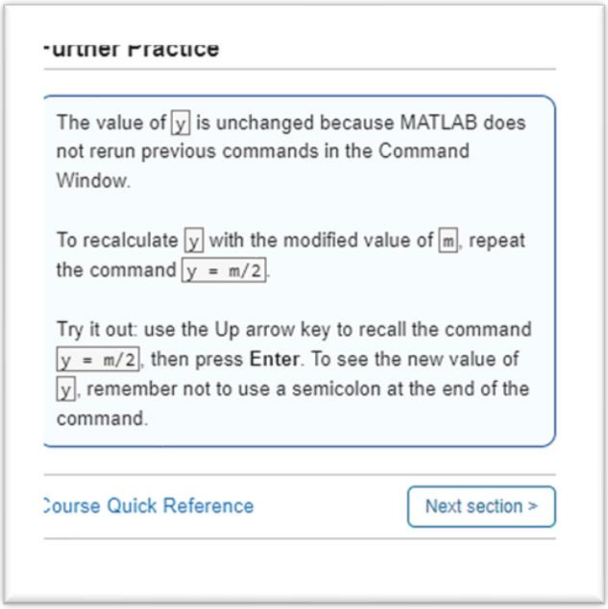
Course Quick Reference

Further Practice

```
m =

    15
   Task 3 ✓
>> m=m+1

m =

    16
   Task 4 ✓
>> y=m/2

y =

     8
   Task 5 ✓
>> k=8-2;
   Task 6 ✓
>> m=3*k

m =

    18
   Task 7 ✓
>> y

y =

     8
```

Correct!

Space Continue | Esc Try an alternative solution

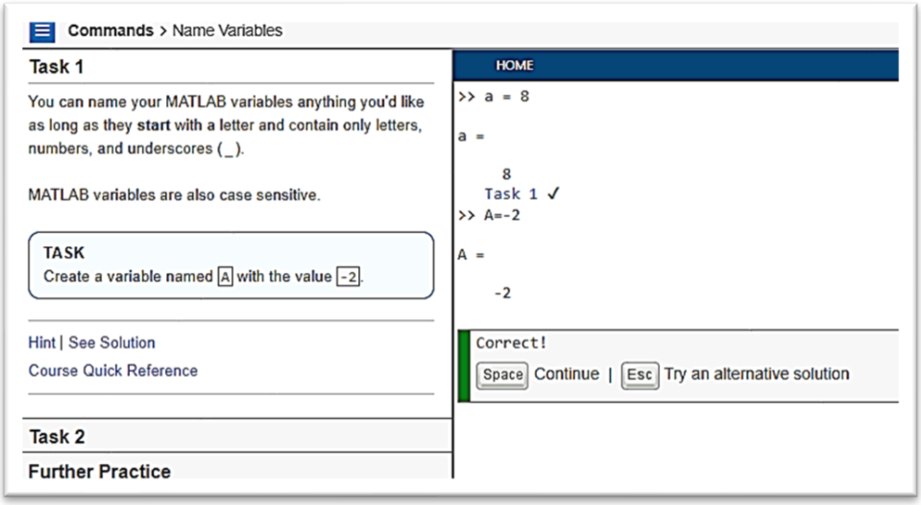


urther Practice

The value of y is unchanged because MATLAB does not rerun previous commands in the Command Window.

To recalculate y with the modified value of m, repeat the command `y = m/2`.

Try it out: use the Up arrow key to recall the command `y = m/2`, then press **Enter**. To see the new value of y, remember not to use a semicolon at the end of the command.



Commands > Name Variables
HOME

Task 1

You can name your MATLAB variables anything you'd like as long as they **start** with a letter and contain only letters, numbers, and underscores (_).

MATLAB variables are also case sensitive.

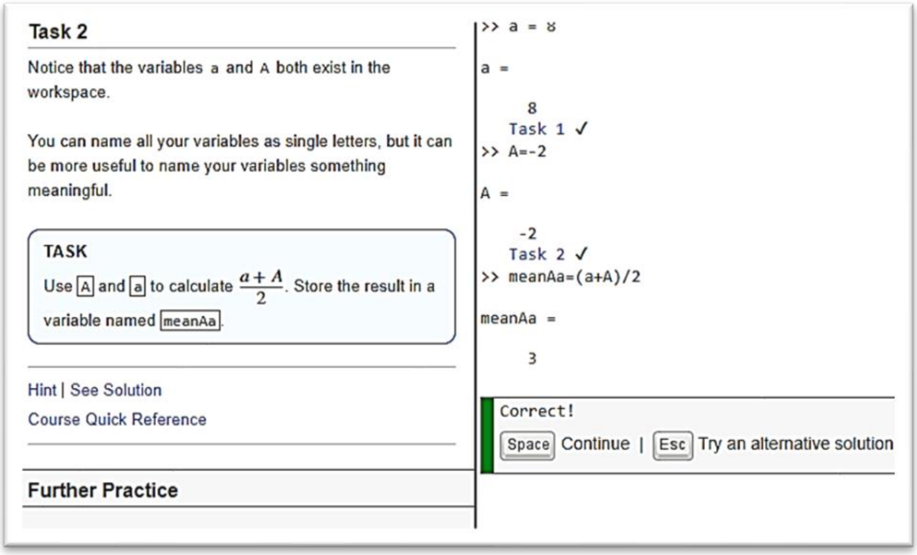
TASK
Create a variable named A with the value -2.

Hint | See Solution
Course Quick Reference
Task 2
Further Practice

```
>> a = 8

a =

     8
  Task 1 ✓
>> A=-2

A =

    -2
```

Correct!

Space Continue | Esc Try an alternative solution



Task 2

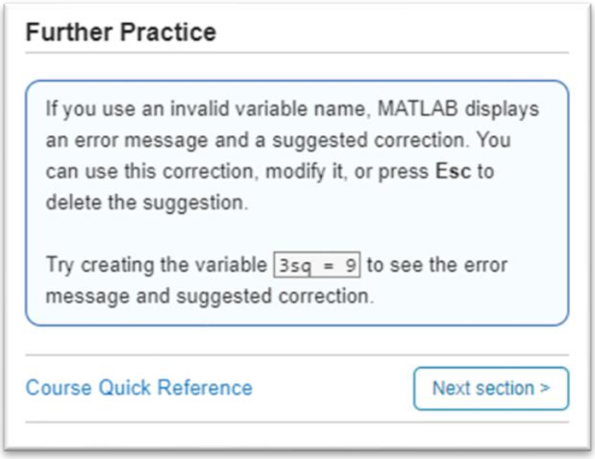
Notice that the variables a and A both exist in the workspace.

You can name all your variables as single letters, but it can be more useful to name your variables something meaningful.

TASK
Use A and a to calculate $\frac{a+A}{2}$. Store the result in a variable named meanAa.

```
>> a = 8

a =

     8
  Task 1 ✓
>> A=-2

A =

    -2
  Task 2 ✓
>> meanAa=(a+A)/2

meanAa =

     3
```

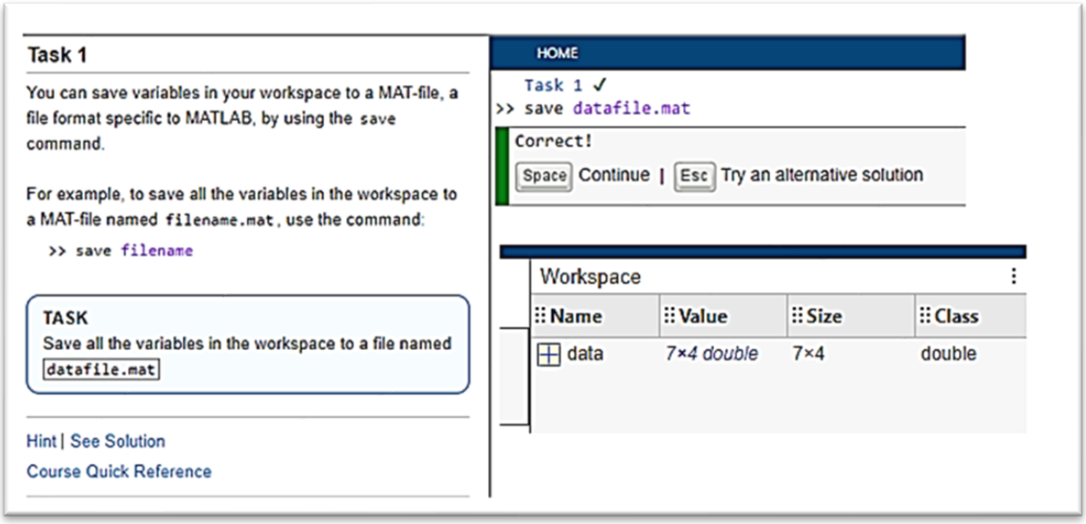
Correct!



Further Practice

If you use an invalid variable name, MATLAB displays an error message and a suggested correction. You can use this correction, modify it, or press **Esc** to delete the suggestion.

Try creating the variable `3sq = 9` to see the error message and suggested correction.



Task 1

You can save variables in your workspace to a MAT-file, a file format specific to MATLAB, by using the `save` command.

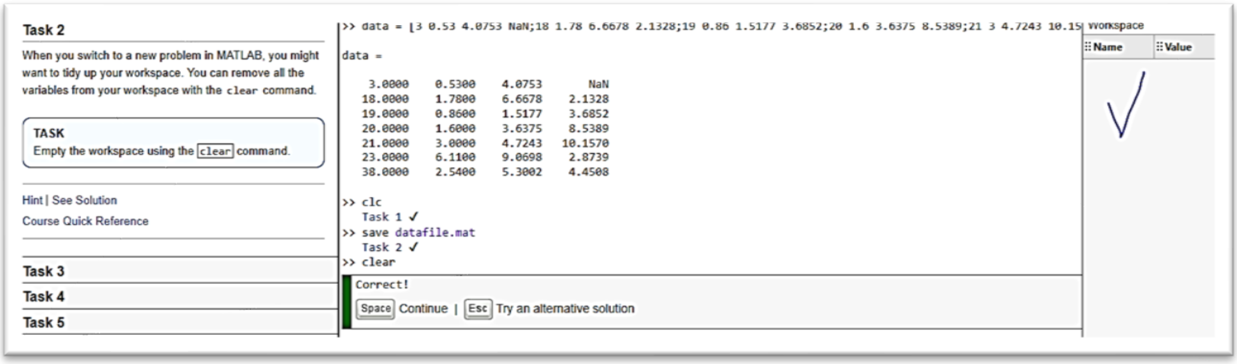
For example, to save all the variables in the workspace to a MAT-file named `filename.mat`, use the command:

```
>> save filename
```

TASK
Save all the variables in the workspace to a file named `datafile.mat`

```
   Task 1 ✓
>> save datafile.mat
Correct!
```

Name	Value	Size	Class
data	*7×4 double*	7×4	double

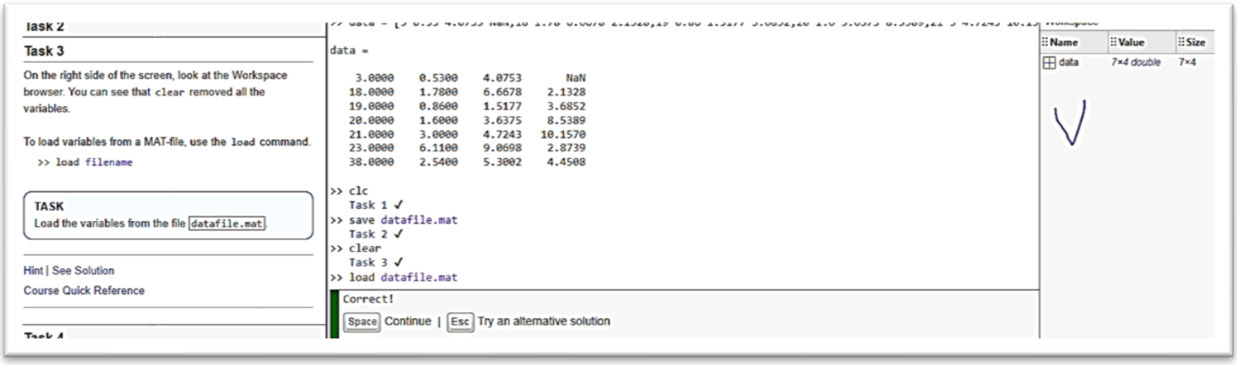


Task 2

When you switch to a new problem in MATLAB, you might want to tidy up your workspace. You can remove all the variables from your workspace with the `clear` command.

TASK
Empty the workspace using the `clear` command.

```
>> data = [3 0.53 4.0753 NaN;18 1.78 6.6678 2.1328;19 0.86 1.5177 3.6852;20 1.6 3.6375 8.5389;21 3 4.7243 10.15

data =

    3.0000    0.5300    4.0753       NaN
   18.0000    1.7800    6.6678    2.1328
   19.0000    0.8600    1.5177    3.6852
   20.0000    1.6000    3.6375    8.5389
   21.0000    3.0000    4.7243   10.1570
   23.0000    6.1100    9.0698    2.8739
   38.0000    2.5400    5.3002    4.4508

>> clc
   Task 1 ✓
>> save datafile.mat
   Task 2 ✓
>> clear
Correct!
```



Task 2

Task 3

On the right side of the screen, look at the Workspace browser. You can see that `clear` removed all the variables.

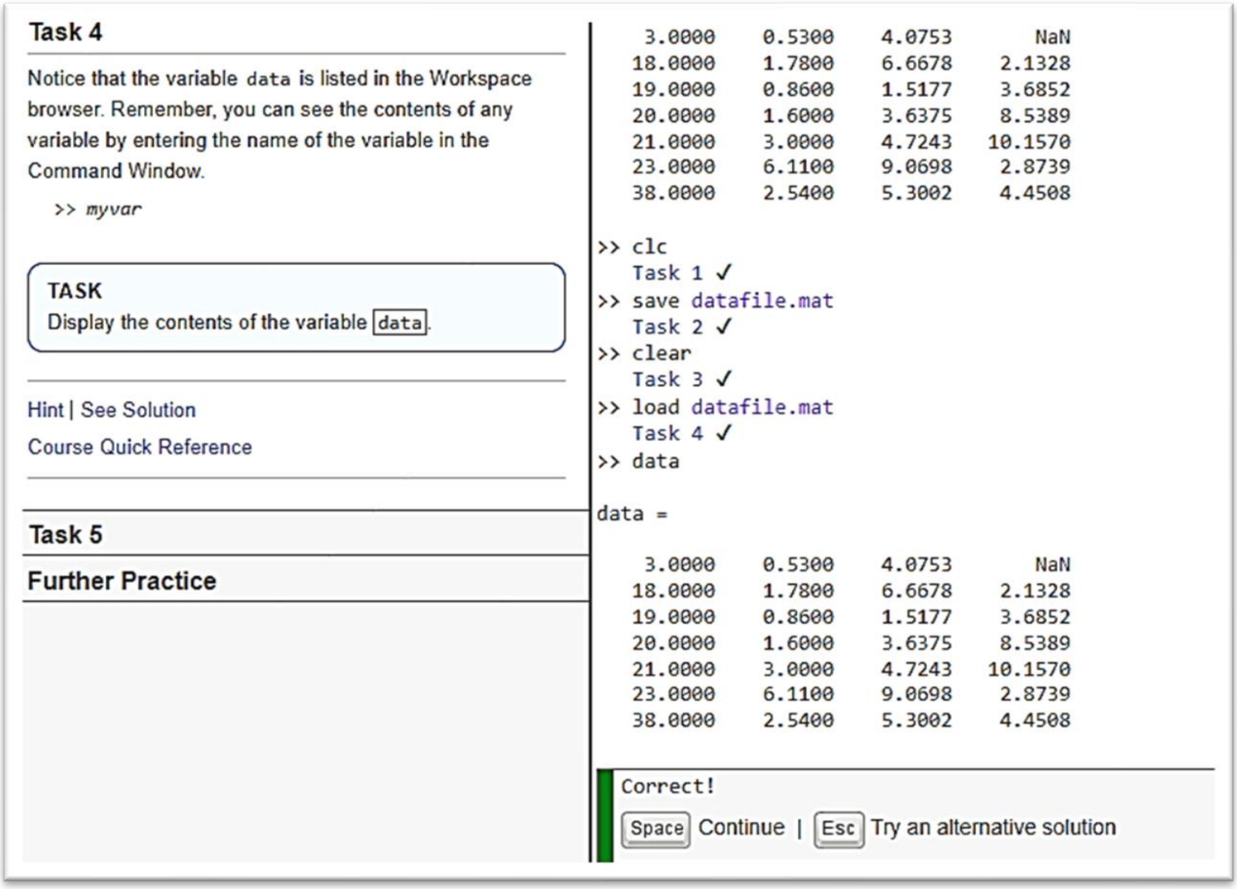
To load variables from a MAT-file, use the `load` command.

```
>> load filename
```

TASK
Load the variables from the file `datafile.mat`

```
>> data = [3 0.53 4.0753 NaN;18 1.78 6.6678 2.1328;19 0.86 1.5177 3.6852;20 1.6 3.6375 8.5389;21 3 4.7243 10.15

data =

    3.0000    0.5300    4.0753       NaN
   18.0000    1.7800    6.6678    2.1328
   19.0000    0.8600    1.5177    3.6852
   20.0000    1.6000    3.6375    8.5389
   21.0000    3.0000    4.7243   10.1570
   23.0000    6.1100    9.0698    2.8739
   38.0000    2.5400    5.3002    4.4508

>> clc
   Task 1 ✓
>> save datafile.mat
   Task 2 ✓
>> clear
   Task 3 ✓
>> load datafile.mat
Correct!
```

Name	Value	Size
data	*7×4 double*	7×4



Task 4

Notice that the variable `data` is listed in the Workspace browser. Remember, you can see the contents of any variable by entering the name of the variable in the Command Window.

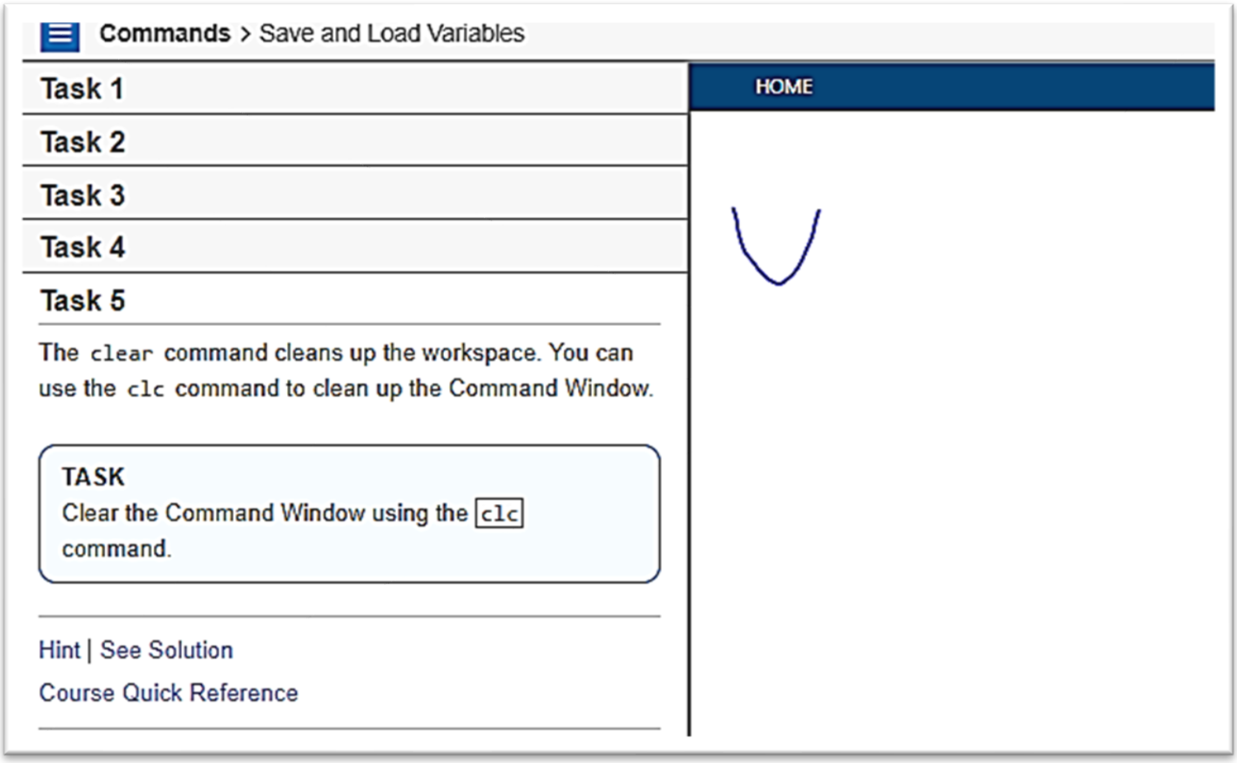
```
>> myvar
```

TASK
Display the contents of the variable `data`.

```
    3.0000    0.5300    4.0753       NaN
   18.0000    1.7800    6.6678    2.1328
   19.0000    0.8600    1.5177    3.6852
   20.0000    1.6000    3.6375    8.5389
   21.0000    3.0000    4.7243   10.1570
   23.0000    6.1100    9.0698    2.8739
   38.0000    2.5400    5.3002    4.4508

>> clc
   Task 1 ✓
>> save datafile.mat
   Task 2 ✓
>> clear
   Task 3 ✓
>> load datafile.mat
   Task 4 ✓
>> data

data =

    3.0000    0.5300    4.0753       NaN
   18.0000    1.7800    6.6678    2.1328
   19.0000    0.8600    1.5177    3.6852
   20.0000    1.6000    3.6375    8.5389
   21.0000    3.0000    4.7243   10.1570
   23.0000    6.1100    9.0698    2.8739
   38.0000    2.5400    5.3002    4.4508
```

Correct!

Space Continue | Esc Try an alternative solution



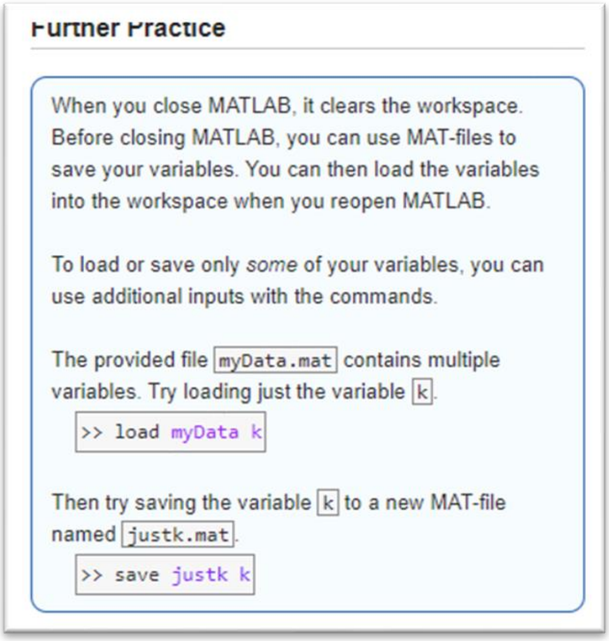
Commands > Save and Load Variables
Task 1
Task 2
Task 3
Task 4
HOME

Task 5

The `clear` command cleans up the workspace. You can use the `clc` command to clean up the Command Window.

TASK
Clear the Command Window using the `clc` command.

Hint | See Solution
Course Quick Reference



Further Practice

When you close MATLAB, it clears the workspace. Before closing MATLAB, you can use MAT-files to save your variables. You can then load the variables into the workspace when you reopen MATLAB.

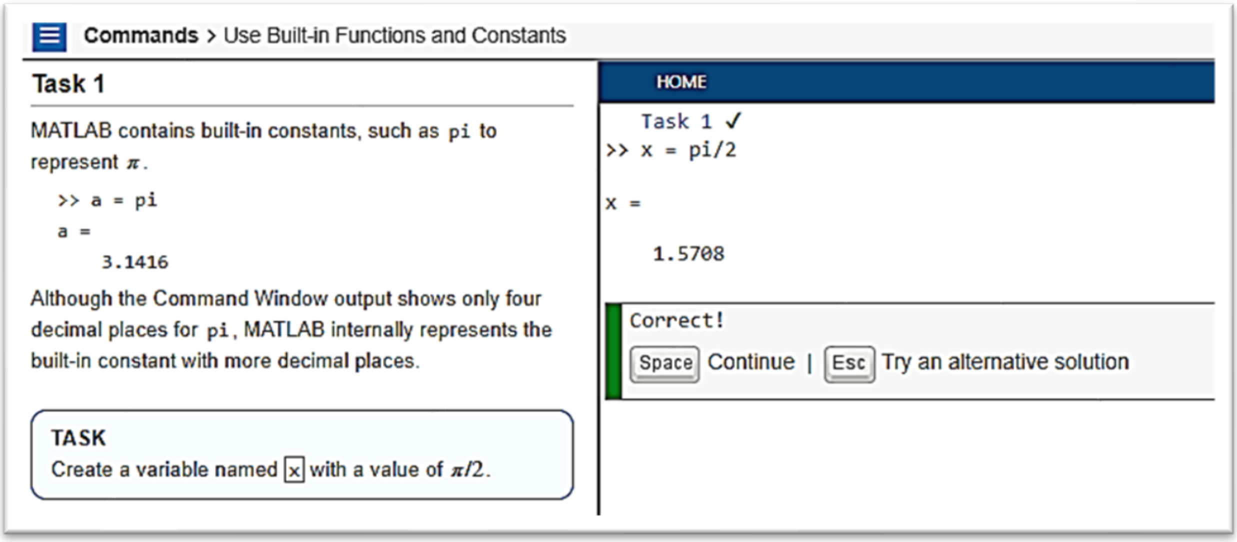
To load or save only *some* of your variables, you can use additional inputs with the commands.

The provided file `myData.mat` contains multiple variables. Try loading just the variable `k`.

```
>> load myData k
```

Then try saving the variable `k` to a new MAT-file named `justk.mat`.

```
>> save justk k
```



Commands > Use Built-in Functions and Constants

Task 1

MATLAB contains built-in constants, such as `pi` to represent π.

```
>> a = pi
a =
      3.1416
```

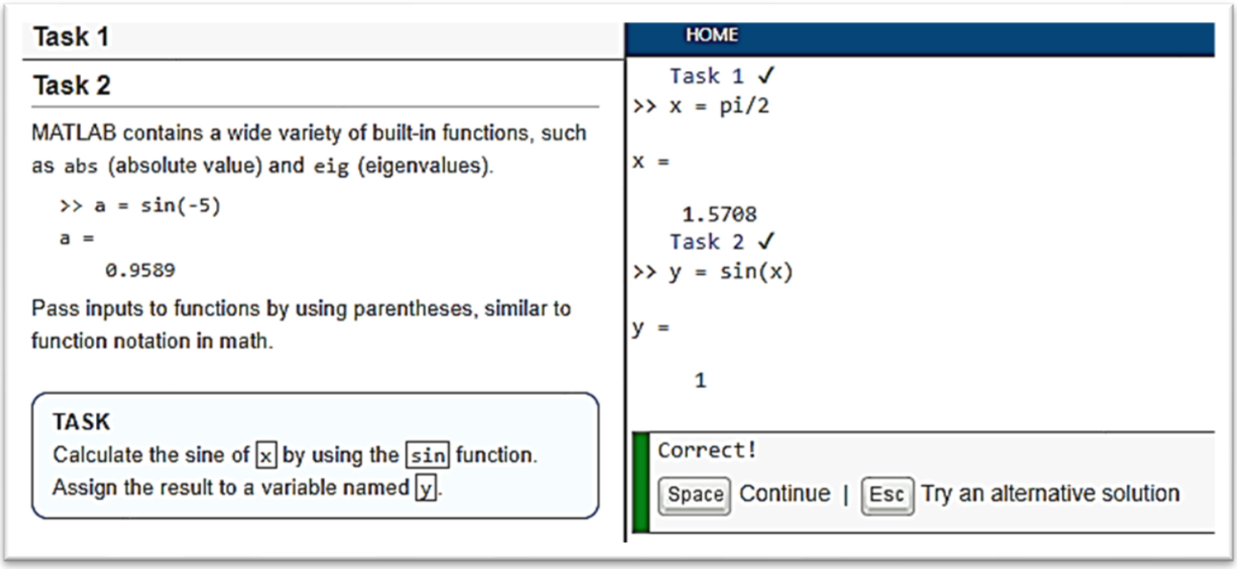
Although the Command Window output shows only four decimal places for `pi`, MATLAB internally represents the built-in constant with more decimal places.

TASK
Create a variable named `x` with a value of $\pi/2$.

```
   Task 1 ✓
>> x = pi/2

x =

    1.5708
```

```
Correct!
```

Space Continue | Esc Try an alternative solution



Task 1

Task 2

MATLAB contains a wide variety of built-in functions, such as `abs` (absolute value) and `eig` (eigenvalues).

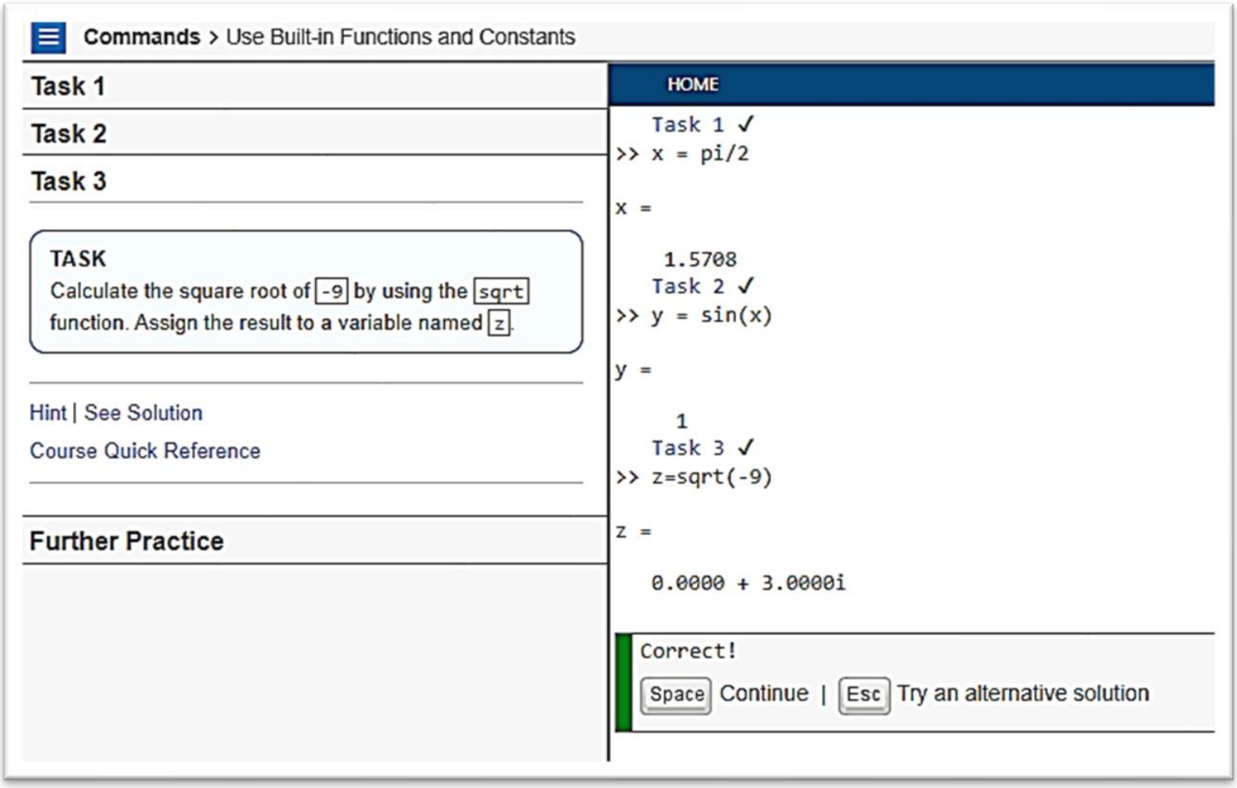
```
>> a = sin(-5)
a =
      0.9589
```

Pass inputs to functions by using parentheses, similar to function notation in math.

TASK
Calculate the sine of `x` by using the `sin` function. Assign the result to a variable named `y`.

```
   Task 1 ✓
>> x = pi/2

x =

    1.5708
   Task 2 ✓
>> y = sin(x)

y =

     1
```

```
Correct!
```



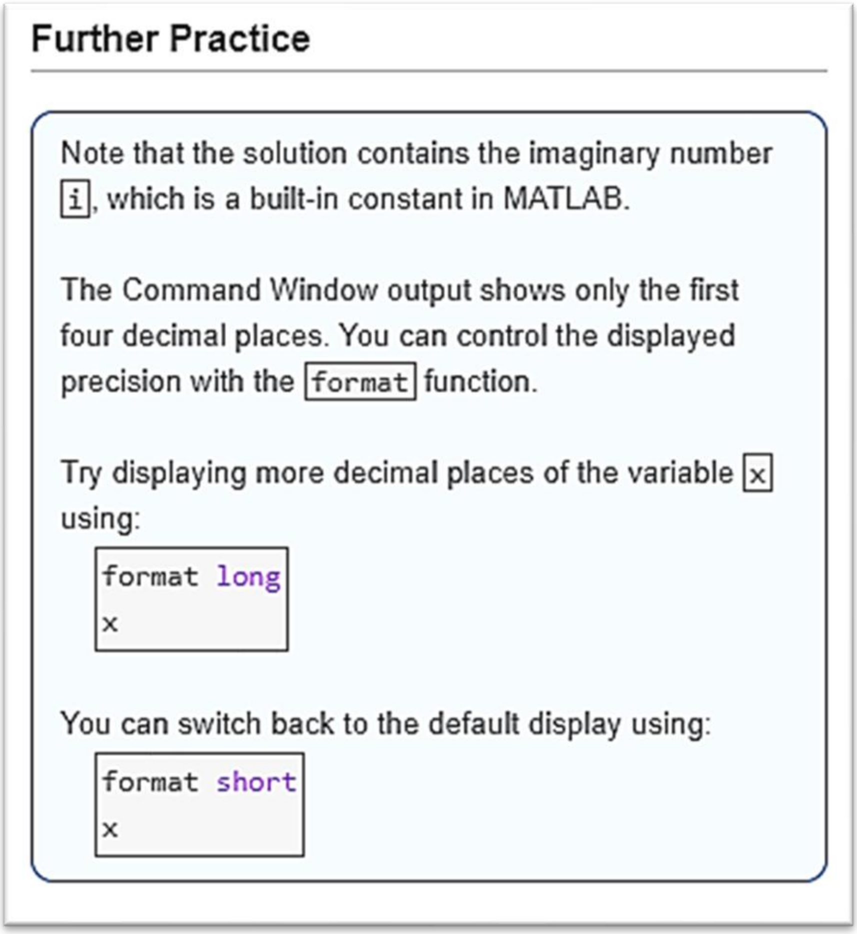
Commands > Use Built-in Functions and Constants
Task 1
Task 2
Task 3
HOME
Hint | See Solution
Course Quick Reference
Further Practice
Space Continue | Esc Try an alternative solution

TASK

Calculate the square root of `-9` by using the `sqrt` function. Assign the result to a variable named `z`.

```
   Task 1 ✓
>> x = pi/2

x =

    1.5708
   Task 2 ✓
>> y = sin(x)

y =

     1
   Task 3 ✓
>> z=sqrt(-9)

z =

   0.0000 + 3.0000i
```

Correct!



Further Practice

Note that the solution contains the imaginary number `i`, which is a built-in constant in MATLAB.

The Command Window output shows only the first four decimal places. You can control the displayed precision with the `format` function.

Try displaying more decimal places of the variable `x` using:

```
format long
x
```

You can switch back to the default display using:

```
format short
x
```