MATLAB: A Practical Introduction to Programming and Problem Solving

Fifth Edition

PRACTICE PROBLEM SOLUTIONS

Stormy Attaway

College of Engineering

Boston University

**Chapter 1**

*Practice 1.1*

Think about what the results would be for the following expressions, and then type them in to verify your answers:

>> 1\2

ans =

2

>> - 5 ^ 2

ans =

-25

>> (-5) ^ 2

ans =

25

>> 10-6/2

ans =

7

>> 5\*4/2\*3

ans =

30

*Practice 1.2*

Generate a random

* real number in the range (0,1)

rand

* real number in the range (0, 100)

rand\*100

* real number in the range (20, 35)

rand\*(35-20)+20

* integer in the inclusive range from 1 to 100

randi(100)

* integer in the inclusive range from 20 to 35

randi([20, 35])

*Practice 1.3*

Think about what would be produced by the following expressions, and then type them in to verify your answers.

>> 3 == 5 + 2

ans =

0

>> 'b' < 'a' + 1

ans =

0

>> 10 > 5 + 2

ans =

1

>> (10 > 5) + 2

ans =

3

>> 'c' == 'd' - 1 && 2 < 4

ans =

1

>> 'c' == 'd' - 1 || 2 > 4

ans =

1

>> xor('c' == 'd' - 1, 2 > 4)

ans =

1

>> xor('c' == 'd' - 1, 2 < 4)

ans =

0

>> 10 > 5 > 2

ans =

0

*Practice 1.4*

* Calculate the range of integers that can be stored in the types **int16** and **uint16**. Use **intmin** and **intmax** to verify your results.

>> 2^16

ans =

65536

>> 2^15

ans =

32768

>> intmin('int16')

ans =

-32768

>> intmax('int16')

ans =

32767

>> intmin('uint16')

ans =

0

>> intmax('uint16')

ans =

65535

* Enter an assignment statement and view the type of the variable in the Workspace Window. Then, change its type and view it again. View it also using **whos**.

>> clear

>> mynumber = 3\*11;

>> whos

Name Size Bytes Class Attributes

mynumber 1x1 8 double

>> mynumber = int32(mynumber)

mynumber =

int32

33

>> whos

Name Size Bytes Class Attributes

mynumber 1x1 4 int32

*Practice 1.5*

* Find the numerical equivalent of the character 'x'.
* Find the character equivalent of 107.

>> double('x')

ans =

120

>> char(107)

ans =

'k'

*Practice 1.6*

Use the **help** function to find out what the rounding functions **fix**, **floor**, **ceil**, and **round** do. Experiment with them by passing different values to the functions, including some negative, some positive, some with fractions less than 0.5 and some greater. *It is very important when testing functions that you thoroughly test by trying different kinds of arguments!*

>> help fix

fix Round towards zero.

fix(X) rounds the elements of X to the nearest integers

towards zero.

See also floor, round, ceil.

Reference page for fix

Other functions named fix

>> help floor

floor Round towards minus infinity.

floor(X) rounds the elements of X to the nearest integers

towards minus infinity.

See also round, ceil, fix.

Reference page for floor

Other functions named floor

>> help ceil

ceil Round towards plus infinity.

ceil(X) rounds the elements of X to the nearest integers

towards infinity.

See also floor, round, fix.

Reference page for ceil

Other functions named ceil

>> help round

round rounds towards nearest decimal or integer

round(X) rounds each element of X to the nearest integer.

round(X, N), for positive integers N, rounds to N digits to the right

of the decimal point. If N is zero, X is rounded to the nearest integer.

If N is less than zero, X is rounded to the left of the decimal point.

N must be a scalar integer.

round(X, N, 'significant') rounds each element to its N most significant

digits, counting from the most-significant or left side of the number.

N must be a positive integer scalar.

round(X, N, 'decimals') is equivalent to round(X, N).

For complex X, the imaginary and real parts are rounded independently.

Examples

--------

% Round pi to the nearest hundredth

>> round(pi, 2)

3.14

% Round the equatorial radius of the Earth, 6378137 meters,

% to the nearest kilometer.

round(6378137, -3)

6378000

% Round to 3 significant digits

format shortg;

round([pi, 6378137], 3, 'significant')

3.14 6.38e+06

If you only need to display a rounded version of X,

consider using fprintf or num2str:

fprintf('%.3f\n', 12.3456)

12.346

fprintf('%.3e\n', 12.3456)

1.235e+01

See also floor, ceil, fprintf.

Reference page for round

Other functions named round

>>

>> fix(5.2)

ans =

5

>> fix(5.8)

ans =

5

>> floor(5.2)

ans =

5

>> floor(5.8)

ans =

5

>> ceil(5.2)

ans =

6

>> ceil(5.8)

ans =

6

>> round(5.2)

ans =

5

>> round(5.8)

ans =

6

>> fix(-5.2)

ans =

-5

>> fix(-5.8)

ans =

-5

>> floor(-5.2)

ans =

-6

>> floor(-5.8)

ans =

-6

>> ceil(-5.2)

ans =

-5

>> ceil(-5.8)

ans =

-5

>> round(-5.2)

ans =

-5

>> round(-5.8)

ans =

-6

>> fix(4.5)

ans =

4

>> floor(4.5)

ans =

4

>> ceil(4.5)

ans =

5

>> round(4.5)

ans =

5