

Roll. No. A016	Name: Varun Khadayate
Class B.Tech CSBs	Batch: 1
Date of Experiment: 29-07-2021	Subject: IT/WS

1. Create a variable myage and store your age in it. Subtract 2 from the value of the variable. Add 1 to the value of the variable. Observe the Workspace Window and Command History Window as you do this.

### Command Window

```
>> myage = 21;
```

### Workspace



### Command Window

```
>> myage = myage - 2;
```

### Workspace



### Command Window

```
>> myage = myage + 1;
```

### Workspace



2. Explain the difference between these two statements:

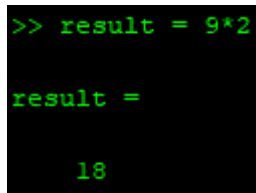
```
result = 9*2
```

```
result = 9*2;
```

Both will store 18 in the variable result. In the first, MATLAB will display this in the Command Window; in the second, it will not.

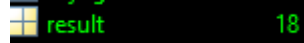
```
result = 9*2
```

Result shown in Command Window and in Worspace



```
result = 9*2;
```

Result shown in Workspace and not in Command Window

A screenshot of the MATLAB Workspace window. It shows a single variable named 'result' with a value of 18. The variable is represented by a small icon with a plus sign and the text 'result' and '18'.

3. Use the built-in function `namelengthmax` to find out the maximum number of characters that you can have in an identifier name under your version of MATLAB.

```
>> namelengthmax
```

```
ans =
```

```
63
```

4. Create two variables to store a weight in pounds and ounces. Use `who` and `whos` to see the variables. Use `class` to see the types of the variables. Clear one of them and then use `who` and `whos` again.

```
>> pounds = 4;  
>> ounces = 3.3;  
>> who
```

Your variables are:

```
ans  myage  ounces  pounds  result
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
ans	1x1	8	double	
myage	1x1	8	double	
ounces	1x1	8	double	
pounds	1x1	8	double	
result	1x1	8	double	

```
>> clear pounds  
>> who
```

Your variables are:

```
ans  myage  ounces  result
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
ans	1x1	8	double	
myage	1x1	8	double	
ounces	1x1	8	double	
result	1x1	8	double	

5. Explore the format command in more detail. Use help format to find options. Experiment with format bank to display dollar values.

```
>> format +
>> 1234

ans =

+

>> -1234

ans =

-

>> format bank
>> 33.4

ans =

    33.40

>> 69.435

ans =

    69.44
```

6. Find a format option that would result in the following output format:

```
>> 5/16 + 2/7
ans =
    67/112
```

```
>> format rat
>> 5/16 + 2/7

ans =

    67/112
```

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

```
25 / 5 * 5
4 + 3 ^ 2
(4 + 3) ^ 2
3 \ 12 + 5
4 - 2 * 3
```

```
>> 25/5*5
```

```
ans =
```

```
25
```

```
>> 4+3^2
```

```
ans =
```

```
13
```

```
>> (4+3)^2
```

```
ans =
```

```
49
```

```
>> 3/12+5
```

```
ans =
```

```
21/4
```

```
>> 4-2*3
```

```
ans =
```

```
-2
```

8. Create a variable pounds to store a weight in pounds. Convert this to kilograms and assign the result to a variable kilos. The conversion factor is

1 kilogram = 2.2 lb.

```
>> pounds = 69
```

```
pounds =
```

```
69
```

```
>> kilos = pounds / 2.2
```

```
kilos =
```

```
31.3636
```

9. Create a variable ftemp to store a temperature in degrees Fahrenheit (F). Convert this to degrees Celsius (C) and store the result in a variable ctemp. The conversion factor is  $C = (F - 32) * 5/9$ .

```
>> f = 75

f =

    75

>> c = (f - 32) * 5/9

c =

    23.8889
```

10. The following assignment statements either contain at least one error, or could be improved in some way. Assume that radius is a variable that has been initialized. First, identify the problem, and then fix and/or improve them:
- ```
33 = number
my variable = 11.11;
area = 3.14 * radius^2;
x = 2 * 3.14 * radius;
```

`33 = number`

The variable is always on the left ***number = 33***

`my variable = 11.11;`

Spaces are not allowed in variable names ***my\_variable = 11.11;***

`area = 3.14 * radius^2;`

Using pi is more accurate than 3.14 ***area = pi \* radius^2;***

`x = 2 * 3.14 * radius;`

x is not a descriptive variable name and Using pi is more accurate than 3.14

***circumference = 2 \* pi \* radius;***

11. Experiment with the functional form of some operators such as plus, minus, and times.

```
>> plus(6,9)

ans =

    15

>> plus(6,-9)

ans =
```

```
-3
>> minus(1,7)
ans =
-6
>> minus(4,-6)
ans =
10
>> times(4,6)
ans =
24
>> times(6,-7)
ans =
-42
```

12. Generate a random

a. real number in the range (0, 20)

```
>> rand * 20
ans =
16.2945
```

b. real number in the range (20, 50)

```
>> rand*(50-20)+20
ans =
47.1738
```

c. integer in the inclusive range from 1 to 10

```
>> randi(10)

ans =

     2
```

d. integer in the inclusive range from 0 to 10

```
>> randi([0, 10])

ans =

    10
```

e. integer in the inclusive range from 50 to 100

```
>> randi([50, 100])

ans =

    82
```

13. Get into a new Command Window, and type `rand` to get a random real number. Make a note of the number. Then, exit MATLAB and repeat this, again making a note of the random number; it should be the same as before. Finally, exit MATLAB and again get into a new Command Window. This time, change the seed before generating a random number; it should be different.

```
>> rand

ans =

    0.0975

>> rng('shuffle')
>> rand

ans =

    0.3191
```

14. What is the difference between `x` and `'x'`?

In an expression, the first would be interpreted as the name of a variable, whereas `'x'` is the character `x`.

15. Explain the difference between constants and variables.

Constants store values that are known and do not change. Variables are used when the value will change, or when the value is not known to begin with (e.g., the user will provide the value).

16. What would be the result of the following expressions?

'b' >= 'c' - 1  
3 == 2 + 1  
(3 == 2) + 1  
xor(5 < 6, 8 > 4)  
10 > 5 > 2  
0 <= result <= 10

```
>> 'b' >= 'c' - 1
'b' >= 'c' - 1
    ↑
Error: Invalid text character. Check for unsupported symbol, invisible character, or pasting of non-ASCII characters.

>> 3 == 2 + 1

ans =

logical

1

>> xor(5 < 6, 8 > 4)

ans =

logical

0

>> 10 > 5 > 2

ans =

logical

0

>> result = 3^2 - 20;
>> result = 3^2 - 20

result =
```



```
-11
```

```
>> 0 <= result <= 10
```

```
ans =
```

```
logical
```

```
1
```

17. Create two variables x and y and store numbers in them. Write an expression that would be true if the value of x is greater than five or if the value of y is less than ten, but not if both of those are true.

```
>> x = 6
```

```
x =
```

```
6
```

```
>> y = 9
```

```
y =
```

```
9
```

```
>> xor(x > 5, y < 10)
```

```
ans =
```

```
logical
```

```
0
```

18. Use the equality operator to verify that  $3 \cdot 10^5$  is equal to  $3e5$ .

```
>> 3*10^5 == 3e5
```

```
ans =
```

```
logical
```

```
1
```

19. In the ASCII character encoding, the letters of the alphabet are in order: 'a' comes before 'b' and also 'A' comes before 'B'. However, which comes first - lower or uppercase letters?

```
>> int32('a')
```

```
ans =
```

```
int32
```

```
97
```

```
>> int32('A')
```

```
ans =
```

```
int32
```

```
65
```

20. Are there equivalents to `intmin` and `intmax` for real number types? Use help to find out.

```
>> help intmax
```

```
intmax Largest positive integer value.
```

```
X = intmax is the largest positive value representable in an int32.
```

```
Any value that is larger than intmax will saturate to intmax when  
cast to int32.
```

```
intmax('int32') is the same as intmax with no arguments.
```

```
intmax(CLASSNAME) is the largest positive value in the integer class  
CLASSNAME. Valid values of CLASSNAME are 'int8', 'uint8', 'int16',  
'uint16', 'int32', 'uint32', 'int64' and 'uint64'.
```

```
intmax('like', Y) returns the largest positive value in the integer  
class with the same data type and complexity (real or complex) as the  
numeric variable Y.
```

```
See also intmin, realmax.
```

```
Documentation for intmax
```

```
>> help intmin
```

```
intmin Smallest integer value.
```

```
X = intmin is the smallest value representable in an int32.
```

```
Any value that is smaller than intmin will saturate to intmin when  
cast to int32.
```

```
intmin('int32') is the same as intmin with no arguments.
```

```
intmin(CLASSNAME) is the smallest value in the integer class CLASSNAME.  
Valid values of CLASSNAME are 'int8', 'uint8', 'int16', 'uint16',
```

'int32', 'uint32', 'int64' and 'uint64'.

intmin('like', Y) returns the smallest value in the integer class with the same data type and complexity (real or complex) as the numeric variable Y.

See also intmax, realmin.

Documentation for intmin

```
>> realmin
```

```
ans =
```

```
2.2251e-308
```

```
>> realmin('double')
```

```
ans =
```

```
2.2251e-308
```

```
>> realmin('single')
```

```
ans =
```

```
single
```

```
1.1755e-38
```

```
>> realmax
```

```
ans =
```

```
1.7977e+308
```

21. Use intmin and intmax to determine the range of values that can be stored in the types uint32 and uint64.

```
>> intmin('uint32')
```

```
ans =
```

```
uint32
```

```
0
```

```
>> intmax('uint32')
```

```
ans =
```

```
uint32
4294967295
>> intmin('uint64')
ans =
uint64
0
>> intmax('uint64')
ans =
uint64
18446744073709551615
```

22. Use the cast function to cast a variable to be the same type as another variable.

```
>> vara = uint16(6 + 9)
vara =
uint16
15
>> varb = 4*7
varb =
28
>> class(varb)
ans =
'double'
>> varb = cast(varb, 'like', vara)
varb =
uint16
28
>> class(varb)
```

```
ans =
```

```
'uint16'
```

23. Use help elfun or experiment to answer the following questions:

a. Is `fix(3.5)` the same as `floor(3.5)`?

```
>> fix(3.5)
```

```
ans =
```

```
3
```

```
>> floor(3.5)
```

```
ans =
```

```
3
```

b. Is `fix(3.4)` the same as `fix(-3.4)`?

```
>> fix(3.4)
```

```
ans =
```

```
3
```

```
>> fix(-3.4)
```

```
ans =
```

```
-3
```

c. Is `fix(3.2)` the same as `floor(3.2)`?

```
>> fix(3.2)
```

```
ans =
```

```
3
```

```
>> floor(3.2)
```

```
ans =
```

```
3
```

d. Is `fix(-3.2)` the same as `floor(-3.2)`?

```
>> fix(-3.2)

ans =

    -3

>> floor(-3.2)

ans =

    -4
```

e. Is `fix(-3.2)` the same as `ceil(-3.2)`?

```
>> fix(-3.2)

ans =

    -3

>> ceil(-3.2)

ans =

    -3
```

24.

a. For what range of values is the function `round` equivalent to the function `floor`?

**For positive numbers:** when the decimal part is less than .5  
**For negative numbers:** when the decimal part is greater than or equal to .5

b. For what range of values is the function `round` equivalent to the function `ceil`?

**For positive numbers:** when the decimal part is greater than or equal to .5  
**For negative numbers:** when the decimal part is less than .5

25. Use `help` to determine the difference between the `rem` and `mod` functions.

```
>> help rem
rem    Remainder after division.
      rem(x,y) returns x - fix(x./y).*y if y ~= 0, carefully computed to
      avoid rounding error. If y is not an integer and the quotient x./y is
      within roundoff error of an integer, then n is that integer. The inputs
      x and y must be real and have compatible sizes. In the simplest cases,
      they can be the same size or one can be a scalar. Two inputs have
      compatible sizes if, for every dimension, the dimension sizes of the
```

inputs are either the same or one of them is 1.

By convention:

`rem(x,0)` is NaN.

`rem(x,x)`, for  $x \neq 0$ , is 0.

`rem(x,y)`, for  $x \neq y$  and  $y \neq 0$ , has the same sign as  $x$ .

Note: `MOD(x,y)`, for  $x \neq y$  and  $y \neq 0$ , has the same sign as  $y$ .

`rem(x,y)` and `MOD(x,y)` are equal if  $x$  and  $y$  have the same sign, but differ by  $y$  if  $x$  and  $y$  have different signs.

See also `mod`.

Documentation for `rem`

Other functions named `rem`

`>> help mod`

`mod` Modulus after division.

`mod(x,y)` returns  $x - \text{floor}(x./y) \cdot y$  if  $y \neq 0$ , carefully computed to avoid rounding error. If  $y$  is not an integer and the quotient  $x./y$  is within roundoff error of an integer, then  $n$  is that integer. The inputs  $x$  and  $y$  must be real and have compatible sizes. In the simplest cases, they can be the same size or one can be a scalar. Two inputs have compatible sizes if, for every dimension, the dimension sizes of the inputs are either the same or one of them is 1.

The statement " $x$  and  $y$  are congruent mod  $m$ " means `mod(x,m) == mod(y,m)`.

By convention:

`mod(x,0)` is  $x$ .

`mod(x,x)` is 0.

`mod(x,y)`, for  $x \neq y$  and  $y \neq 0$ , has the same sign as  $y$ .

Note: `REM(x,y)`, for  $x \neq y$  and  $y \neq 0$ , has the same sign as  $x$ .

`mod(x,y)` and `REM(x,y)` are equal if  $x$  and  $y$  have the same sign, but differ by  $y$  if  $x$  and  $y$  have different signs.

See also `rem`.

Documentation for `mod`

Other functions named `mod`

26. Find MATLAB expressions for the following

$$\sqrt{19}$$

`>> sqrt(19)`

`ans =`

```
4.3589
```

$3^{1.2}$

```
>> 3^1.2
```

```
ans =
```

```
3.7372
```

$\tan(\pi)$

```
>> tan(pi)
```

```
ans =
```

```
-1.2246e-16
```

27. Using only the integers 2 and 3, write as many expressions as you can that result in 9. Try to come up with at least 10 different expressions (Note: don't just change the order). Be creative! Make sure that you write them as MATLAB expressions. Use operators and/or built-in functions.

```
>> 3^2
```

```
ans =
```

```
9
```

```
>> 2^3+(3-2)
```

```
ans =
```

```
9
```

```
>> 3*3
```

```
ans =
```

```
9
```

```
>> 3^3-3*3*2
```

```
ans =
```

```
9
```

```
>> 2^3+abs(2-3)
```

```
ans =
```



```

9
>> 2^3+sign(3)

ans =

9
>> 3/2*2*3

ans =

9
>> 2\3*2*3

ans =

9
>> sqrt(3^(2+2))

ans =

9

```

28. A vector can be represented by its rectangular coordinates  $x$  and  $y$  or by its polar coordinates  $r$  and  $\theta$ .  $\theta$  is measured in radians. The relationship between them is given by the equations:

$$x = r * \cos(\theta)$$

$$y = r * \sin(\theta)$$

```

>> r = 46

r =

46
>> theta = 0.7

theta =

0.7000
>> x = r*cos(theta)

x =

```

```
35.1827
```

```
>> y = r*sin(theta)
```

```
y =
```

```
29.6340
```

29. In special relativity, the Lorentz factor is a number that describes the effect of speed on various physical properties when the speed is significant relative to the speed of light. Mathematically, the Lorentz factor is given as:

$$\gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}}$$

Use  $3 \times 10^8$  m/s for the speed of light,  $c$ . Create variables for  $c$  and the speed  $v$  and from them a variable *lorentz* for the Lorentz factor.

```
>> c = 3e8
```

```
c =
```

```
300000000
```

```
>> v = 2.9e8
```

```
v =
```

```
290000000
```

```
>> lore = 1 / sqrt(1 - v^2/c^2)
```

```
lore =
```

```
3.9057
```

30. A company manufactures a part for which there is a desired weight. There is a tolerance of  $N$  percent, meaning that the range between minus and plus  $N\%$  of the desired weight is acceptable. Create a variable that stores a weight, and another variable for  $N$  (for example, set it to two). Create variables that store the minimum and maximum values in the acceptable range of weights for this part.

```
>> weight = 46.6917
```

```
weight =
```

```
46.6917
```

```
>> N = 7

N =

    7

>> min = weight - weight*0.01*N

min =

    43.4233

>> max = weight + weight*0.01*N

max =

    49.9601
```

31. An environmental engineer has determined that the cost  $C$  of a containment tank will be based on the radius  $r$  of the tank:

$$C = \frac{32430 + 428\pi}{r}$$

Create a variable for the radius, and then for the cost

```
>> radius = 46;
>> c = 32430/radius + 428*pi*radius

c =

    62556.68
```

32. A chemical plant releases an amount  $A$  of pollutant into a stream. The maximum concentration  $C$  of the pollutant at a point which is a distance  $x$  from the plant is:

$$C = \frac{A}{x} \sqrt{\frac{2}{\pi e}}$$

Create variables for the values of  $A$  and  $x$ , and then for  $C$ . Assume that the distance  $x$  is in meters. Experiment with different values for  $x$ .

```
>> A = 80000

A =

    80000.00
```

```

>> x = 100

x =

    100.00

>> C = A/x * sqrt(2/(pi*exp(1)))

C =

    387.15

>> x = 1000;

>> C = A/x * sqrt(2/(pi*exp(1)))

C =

    38.72

>> x = 20000;
>> C = A/x * sqrt(2/(pi*exp(1)))

C =

    1.94

```

33. The geometric mean  $g$  of  $n$  numbers  $x_i$  is defined as the  $n^{\text{th}}$  root of the product of  $x_i$ :

$$g = \sqrt[n]{x_1 x_2 x_3 \dots x_n}$$

(This is useful, for example, in finding the average rate of return for an investment which is something you'd do in engineering economics). If an investment returns 15% the first year, 50% the second, and 30% the third year, the average rate of return would be  $(1.15 * 1.50 * 1.30)^{1/3}$ . ) Compute this.

```

>> x1 = 1.15

x1 =

    1.15

>> x2 = 1.5

x2 =

    1.50

```

```
>> x3 = 1.3
```

```
x3 =
```

```
1.30
```

```
>> g = nthroot(x1*x2*x3, 3)
```

```
g =
```

```
1.31
```

34. Use the **deg2rad** function to convert 180 degrees to radians.

```
>> deg2rad(180)
```

```
ans =
```

```
3.14
```