```
% Keeps the output compact

format compact

% To stop execution in command window Ctrl + c


% ----- GETTING USER INPUT -----

% Receive string input by using 's'

% The ; suppresses showing the variable assignment

% name = input("What's your name : ", 's');

% Checks to make sure the user entered something

% if ~isempty(vInput)

% %s puts the string value in the output

% fprintf("Hello %s\n", name)

% end


% You can receive vectors

% vInput = input("Enter a vector : ");

% disp(vInput)


% ----- VARIABLES -----

%{

Variables start with a letter and then numbers and _
Data types are classes which also have associated
methods for working with that data
Types include int8, int16, int32, int64, char, logical,
double, single(generic int) and unsigned uint8, ...
%}


% chars hold single characters

c1 = 'A'

% class displays the data type

class(c1)

% You can store strings in single quotes

s1 = 'A string'
class(s1)

% Booleans map true to 1 and false to 0

5 > 2
```

```
b1 = true

% Show max & min values

intmin('int8')
intmax('int8')
% Largest double

realmax
% Largest int

realmax('single')

% You can continue expressions using ...

% Suppress output with a ;

v1 = 1 + 2 + 3 ...
+ 4;

% ----- CASTING -----

% Everything defaults to double

v2 = 8
class(v2)

% Cast to int8

v3 = int8(v2)
class(v3)

% Convert char to double

v4 = double('A')

% Convert to char

v5 = char(64)

% ----- MATH OPERATORS & SPRINTF -----

% sprintf formats a string

% %d : Integers, %f : Floats, %e : exponential notation

% %c : Characters, %s : Strings


fprintf('5 + 4 = %d\n', 5 + 4)
fprintf('5 - 4 = %d\n', 5 - 4)
fprintf('5 * 4 = %d\n', 5 * 4)

% Define you want only 2 decimals

fprintf('5 / 4 = %0.2f\n', 5 / 4)

% Exponentiation

fprintf('5^4 = %d\n', 5^4)

```

```
% Modulus (Escape % by doubling)

fprintf('5 %% 4 = %d\n', mod(5,4))

% Generate a random value between 10 & 20

randi([10,20])

% Precision is accurate to 15 digits by default

bF = 1.1111111111111111
bF2 = bF + 0.1111111111111111
fprintf("bF2 = %0.16f\n", bF2)

% ----- MATH FUNCTIONS -----

% help elfun shows a list


fprintf('abs(-1) = %d\n', abs(-1))
fprintf('floor(2.45) = %d\n', floor(2.45))
fprintf('ceil(2.45) = %d\n', ceil(2.45))
fprintf('round(2.45) = %d\n', round(2.45))
fprintf('exp(1) = %f\n', exp(1)) % e^x
fprintf('log(100) = %f\n', log(100))
fprintf('log10(100) = %f\n', log10(100))
fprintf('log2(100) = %f\n', log2(100))
fprintf('sqrt(100) = %f\n', sqrt(100))
fprintf('90 Deg to Radians = %f\n', deg2rad(90))

% ----- CONDITIONALS -----

% Relational Operators : >, <, >=, <=, == and ~=

% Logical Operators : ||, &&, ~ (Not)


age = 12
if age >= 5 && age <= 6
    disp("You're in Kindergarten")
elseif age >= 7 && age <= 13
    disp("You're in Middle School")
elseif age >= 14 && age <= 18
    disp("You're in High School")
else
    disp("Stay Home")
end

true || false
~true

% Switch is used when you have a limited number of

% options

switch age
    case 5
        disp("Go to Kindergarten")
    case num2cell(6:13)
        disp("Go to Middle School")
    case {14,15,16,17,18}
```

```
        disp("Go to High School")
    otherwise
        disp("Stay Home")
end

% ----- VECTORS -----

% Vectors are either row or column vectors or

% 1 dimensional arrays

vt1 = [5 3 2 1]

% Elements in vector

vL = length(vt1)

% Sort in ascending order or (, 'descend')

vt1 = sort(vt1)

% Create a range

vt2 = 5:10

% Create a range with a step

vt3 = 2:2:10

% Concatenate vectors

vt4 = [vt1 vt2]

% Get a value with an index starting at 1

vt4(1)

% Get the last value

vtEnd = vt4(end)

% Change a value

vt4(1) = 12

% Add to the end (0 is added to indexes between)

vt4(11) = 33

% Get 1st 3 values

vt4(1:3)

% Get 2, 4th and 6th

vt4([2 4 6])

% Create a column vector

vt5 = [2;3;4]
% And another row
```

```
245
246 vt6 = [1 2 3]
247
248 % Vector multiplication
249
250 % We need a column and row vector
251
252 % Multiply vt6(1,1) by each row in vt5(1,:)
253
254 vtMult = vt5 * vt6
255
256 % Dot Product
257
258 % (1*4) + (5*2) + (3*6) = 32
259
260 vt7 = [4 5 6]
261 % ' Tranposes the vector
262
263 vtDotP = vt6 * vt7'
264 % The dot function does the same
265 vtDotP2 = dot(vt6, vt7)
266
267 % Cross Product
268 % [a1, a2, a3] [b1, b2, b3]
269 % (1,1) = (a2*b3) - (a3*b2)
270 % (1,1) = (2*6) - (3*5) = -3
271 % (2,1) = (a3*b1) - (a1*b3)
272 % (3,1) = (a1*b2) - (a2*b1)
273 vtCross = cross(vt6, vt7)
274
275 % Create linearly spaced vector with for elements
276 % between 1 and 20
277 vt8 = linspace(1,20,4)
278
279 % Logarithmically spaced vector 10^1, 10^2, 10^3
280 vt9 = logspace(1,3,3)
281
282 % ----- MATRICES -----
283 % Matrices have rows and columns
284 m1 = [2 3 4; 4 6 8]
285
286 % Number of values in a row
287 mNRV = length(m1)
288
289 % Total number of values
290 mNV = numel(m1)
291
292 % Get row and column size
293 mS = size(m1)
294
295 % Store rows and columns in different variables
296 [nRows, nCols] = size(m1)
297
298 % Generate random matrix with 2 rows between 10 & 20
299 m2 = randi([10,20], 2)
300
301 % Use row then column to get a value
302 m2(1,2)
303
304 % Change a value
305 m2(1,2) = 22
```

```
306
307 % Change all row values
308 m2(1,:) = 25
309
310 % Change all column values
311 m2(:,1) = 36
312
313 % Get 1st value in the last row
314 mR1Last = m2(end,1)
315
316 % Get the 2nd value in last column
317 MR2Last = m2(2,end)
318
319 % Delete the 2nd column
320 m2(:,2)
321
322 % ----- LOOPING -----
323 % For loop
324 for i = 1:10
325     disp(i)
326 end
327
328 % Decrement and stepping
329 for i = 10:-1:0
330     disp(i)
331 end
332
333 % Specify values
334 for i = [2 3 4]
335     disp(i)
336 end
337
338 % Cycle through a matrix
339 m4 = [2 3 4; 4 6 8]
340 for i = 1:2
341     for j = 1:3
342         disp(m4(i,j))
343     end
344 end
345
346 % Cycle through a vector
347 lVect = [6 7 8]
348 for i = 1:length(lVect)
349     disp(lVect(i))
350 end
351
352 % While loop
353 i = 1
354
355 % Print out only evens
356 while i < 20
357     if (mod(i,2)) == 0
358         disp(i)
359
360         % MatLab doesn't have ++, +=
361         i = i + 1;
362
363         % Skip back to the top of the loop
364         continue
365     end
366
```

```
    i = i + 1;

    if i >= 10

        % Leave the loop
        break
    end
end

% ----- MATRIX FUNCTIONS -----

m3 = [2 3 4; 4 6 8; 8 12 16; 16 24 32]

% You can added, subtract, etc. matrice

m4 = [1:3; 4:6]
m5 = [2:4; 5:7]

m4 + m5
m4 .* m5

% Functions perform operations on each value

sqrt(m3)
m3 = m3 * 2

% Sum adds all the columns

sum(m3)

% Matrix Multiplication

% Columns of m6 must equal rows in m7

m6 = [1 2 3;
    4 5 6]

m7 = [1 1 1 1;
    2 2 2 2;
    3 3 3 3]

% m8(1,1) = (1*1) + (2*2) + (3*3) = 14

m8 = m6 * m7

% Check how many values are greater then 3

gT3M = m4 > 3
sum(gT3M, 'all')

% Check for equality

isequal(m4, m5)

% Find matching value indexes

find(m3 > 24)

% Multiply column values

prod(m3)
```

```
428
429 % 1st row stays the same and each after is a sum of
430
431 % the proceeding and current row
432
433 % cumsum(m3, 'reverse') starts in opposite order
434
435 cumsum(m3)
436
437 % Like cumsum, but with multiplication
438
439 cumprod(m3)
440
441 % Flip 1st column to last
442
443 fliplr(m3)
444
445 % Flip rows
446
447 flipud(m3)
448
449 % Rotate 90 degrees
450
451 rot90(m3)
452
453 % Rotate 180 degrees
454
455 rot90(m3,2)
456
457 % Convert into a 2 x 6 matrix
458
459 reshape(m3, 2, 6)
460
461 % Duplicate matrix into new matrix 2 rows by 1 column
462
463 repmat(m3, 2, 1)
464
465 % Duplicates elements into new matrix 2 rows for each value
466
467 repelem(m3, 2, 1)
468
469 % ----- CELL ARRAYS -----
470
471 % Stores values of different types
472
473 cA1 = {'Doug Smith', 34, [25 8 19]}
474
475 % Define the number of spaces to set aside
476
477 cA2 = cell(5)
478
479 % Get by index
480
481 cA1{1}
482 cA1{3}(2)
483
484 % Add more data
485
486 cA1{4} = 'Patty Smith'
487
488 % Get size
```

```
length(cA1)

% Delete a value

cA1(4) = []

% Output

for i = 1:length(cA1)
    disp(cA1{i})
end

% Cell Array to char matrix

cA3 = {'Doug' 'Patty'}
nameMat = char(cA3)

% Char array to cell array

cA4 = cellstr(nameMat)

% ----- STRINGS -----

% Strings are vectors of characters

str1 = 'I am a string'

% Get length

length(str1)

% Character at index

str1(1)

% Get substring

str1(3:4)

% Join strings (Escape ' with '')

str2 = strcat(str1, ' that''s longer')

% Find all index matches for a string

strfind(str2, 'a')

% Replace any matches

strrep(str2, 'longer', 'bigger')

% Split string using delimiter ' '

strArray = strsplit(str1, ' ')
class(strArray)
strArray(1)

% Convert cell array into a string

cA5 = {'I' 'like' 'chickens'}
```

```
str3 = strjoin(cA5)

% Integer to string

nStr = int2str(99)

% Float to str

fStr = num2str(3.14)

% Compare for equality

strcmp(str1, str2)

% Check if is a character

isletter('num 2')

% Check if all are letters

% ('alphanum' letters or numbers)

isstrprop('word2', 'alpha')

% Is it a string

ischar('Some words 2')

% Sort ascending (, 'descend')

sort(str3)

% Delete whitespace

strtrim(str1)

% Uppercase & lowercase

lower(str1)
upper(str1)

% ----- STRUCTURES -----

% Custom data type that stores related data in fields

dougSmith = struct('name', 'Doug Smith', ...
    'age', 34, 'purchases', [12 23])

% Access data

disp(dougSmith.age)

% Add a field

dougSmith.wife = 'Patty Smith'

% Remove a field

dougSmith = rmfield(dougSmith, 'wife')

% Check for a field
```

```
611
612 isfield(dougSmith, 'wife')
613
614 % Get fields
615
616 fieldnames(dougSmith)
617
618 % Store structs in a vector
619
620 customers(1) = dougSmith
621 sallySmith = struct('name', 'Sally Smith', ...
622     'age', 34, 'purchases', [18])
623 customers(2) = sallySmith
624
625 % Get data
626
627 disp(customers(2).name)
628
629
630 % ----- TABLES -----
631
632 % Tables are labeled rows of data in a table format
633
634 name = {'Jim'; 'Pam'; 'Dwight'};
635 age = [28; 27; 31];
636 salary = [35000; 26000; 75000];
637 id = {'1', '2', '3'};
638
639 % RowNames defines the name used for each row
640
641 employees = table(name, age, salary, 'RowName', id)
642
643 % Get avg salary
644
645 meanSalary = mean(employees.salary)
646
647 % Add column of data
648
649 employees.vDays = [10; 14; 16]
650
651 % Show just 2 by id
652
653 employees({'1', '2'},:)
654
655 % Get by name
656
657 employees(ismember(employees.name,{'Jim' 'Dwight'}), :)
658
659
660 % ----- FILE IO -----
661
662 % Generate a random 8x8 matrix between 10 & 50
663
664 randM = randi([10,50], 8)
665
666 % Save the file as a text file and overwrite
667
668 save sampdata.dat randM -ascii
669
670 % Read the data into a matrix with the same name as
671
```

```
% the data file

load sampdata.dat
disp(sampdata)

% Display file data

type sampdata.dat

% Saving variables to a file

% save yourfile (saves every variable)

save myData

% Read file

load myData
who

% Append data

v4 = 123
save -append myData v4

% ----- OBJECT ORIENTED PROGRAMMING -----

% Real world objects have attributes (height, weight)

% & capabilities (run, eat). Classes model real

% world objects by storing attributes as properties

% and model capabilities using methods

% A class as a blueprint then creates objects

% Classes must be created in their own file with

% class name == file name


a1 = Shape(10, 20);
disp(a1)
Shape.setGetNumShapes
a1.getArea

a2 = Shape(5, 10)
disp(a2)
Shape.setGetNumShapes

a1 > a2

a3 = Trapezoid(10, 4, 6);
disp(a3)
a3.getArea

%{

% ----- ANONYMOUS FUNCTIONS -----

```

```
% Anonymous functions are one line functions

% nameOfFunc = @ (attr) functionBody;

cubeVol = @ (l, w, h) l * w * h;
cV = cubeVol(2,2,2)

% Pass function to function

mult3 = @ (x) x * 3;
sol = doMath(mult3, 4)

% Return a function

mult4 = doMath2(4);
sol2 = mult4(5)

% ----- RECURSIVE FUNCTIONS -----

% Recursive functions call themselves

% Calculate factorial

% 1st : result = 4 * factorial(3) = 4 * 6 = 24

% 2nd : result = 3 * factorial(2) = 3 * 2 = 6

% 3rd : result = 2 * factorial(1) = 2 * 1 = 2


fact4 = factorial(4)


% ----- FUNCTIONS -----

% Get cylinder volume

% tic toc is used to calculate how long it took

% for code to execute

tic
cylinderVol(20, 30)
toc

% Try to change a variable value in a function

changeMe = 5
changeVal()
disp(changeMe)

% Function with no arguments

getRandomNum

% Return more then 1 value

[coneV, cylVol] = getVols(10,20)

% eval will execute code saved as a string

```

```
toExecute = sprintf("total = %d + %d", 5, 4)
eval(toExecute)

% Except a variable number of arguments

theSum = getSum(1, 2, 3, 4)

% Return a variable number of values

listOfNums = getNumbers(10)

% Create a function that finds the volume of a cylinder

% function returnVar = funcName(arguments)

function vol = cylinderVol(radius, height)
vol = pi * radius^2 * height
end

% Try to change a variable value in a function

% If you return nothing leave it out

function changeVal()
% This is a local or variable specific to the function

changeMe = 10
class(changeMe)
end

% Return a random value

function randNum = getRandomNum
randNum = randi([1,100])
end

% Return multiple volumes

function [coneVol, cylinVol] = getVols(radius, height)
cylinVol = pi * radius^2 * height
coneVol = 1/3 * cylinVol
end

% Variable arguments stored in varargin

function sum = getSum(varargin)
sum = 0;
% Cycle through contents while adding

for k = 1:length(varargin)
    sum = sum + varargin{k}(1);
end
end

% Returns a variable number of values

function [varargout] = getNumbers(howMany)
for k = 1:howMany
    varargout{1}(k) = k;
end
end
```

```
855
856 % Receives a function
857
858 function sol = doMath(func, num)
859 sol = func(num);
860 end
861
862 % Return a function
863
864 function func = doMath2(num)
865 func = @(x) x * num;
866 end
867
868 function val = factorial(num)
869 % Every recursive function must reach a point
870
871 % where it no longer makes a function call
872
873 if num == 1
874     val = 1;
875 else
876     val = num * factorial(num - 1);
877 end
878 end
879
880 %}
881
882
883 %{
884
885 % ----- PLOTTING -----
886
887 % Define x & y values
888
889 xVals = 1:5
890 yVals = [2 4 8 5 9]
891 yVals2 = [1 5 7 6 8]
892
893 % Creates window labeled figure 1 with plot
894
895 figure(1)
896
897 % Colors : blue(b), black(k), cyan(c), green(g),
898
899 % magenta(m), red(r), yellow(y), white(w)
900
901 % Plot Symbols : . o x + * s d v ^ < > p h
902
903 % Line Types : -, :, -., - -
904
905
906 % Green dotted line with + at points 2 line width
907
908 plot(xVals, yVals, 'g+:','LineWidth',2)
909
910 % Draw over previous plot
911
912 hold on
913
914 % Draw black stars on points
915
```

```
plot(xVals, yVals2, 'k*')

% Defines look of each plot

legend('yVals', 'yVals2')

% Show grid

grid on

% Define x & y lables & Title

xlabel('Days')
ylabel('Money')
title('Money made Today')

% Creates window labeled figure 2 with bar chart

figure(2)
bar(xVals, yVals, 'r')
%}
```