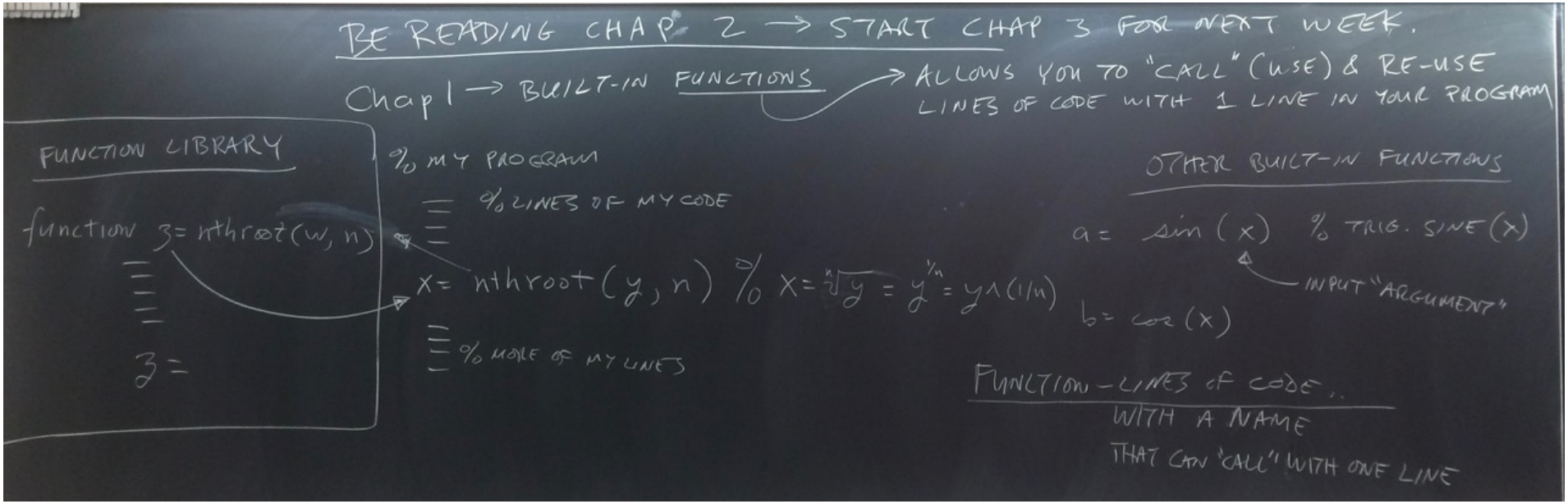
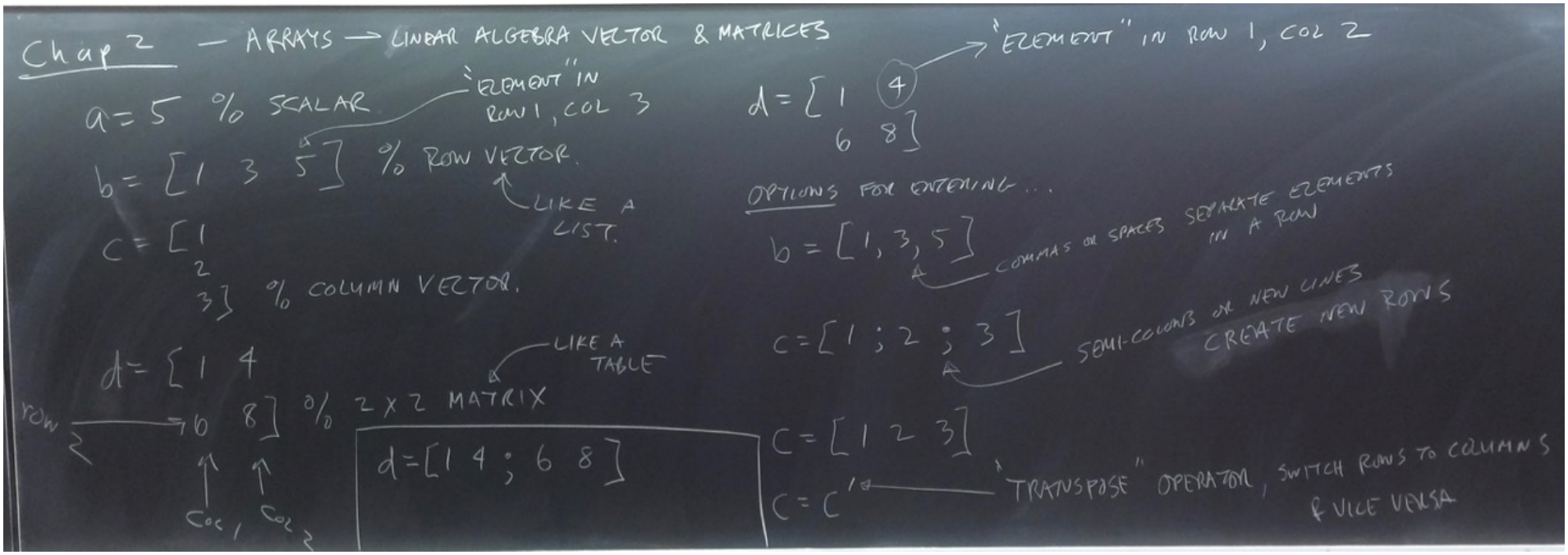
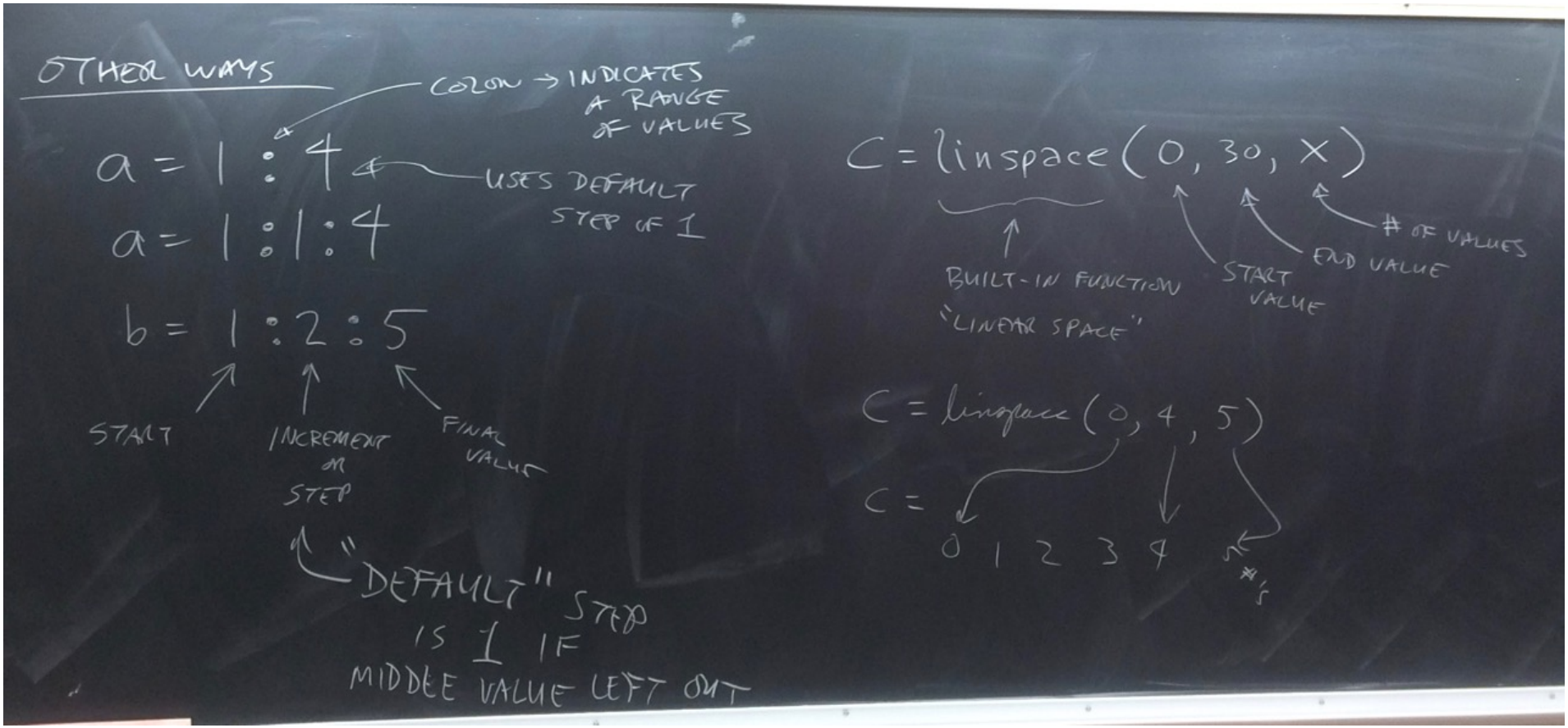
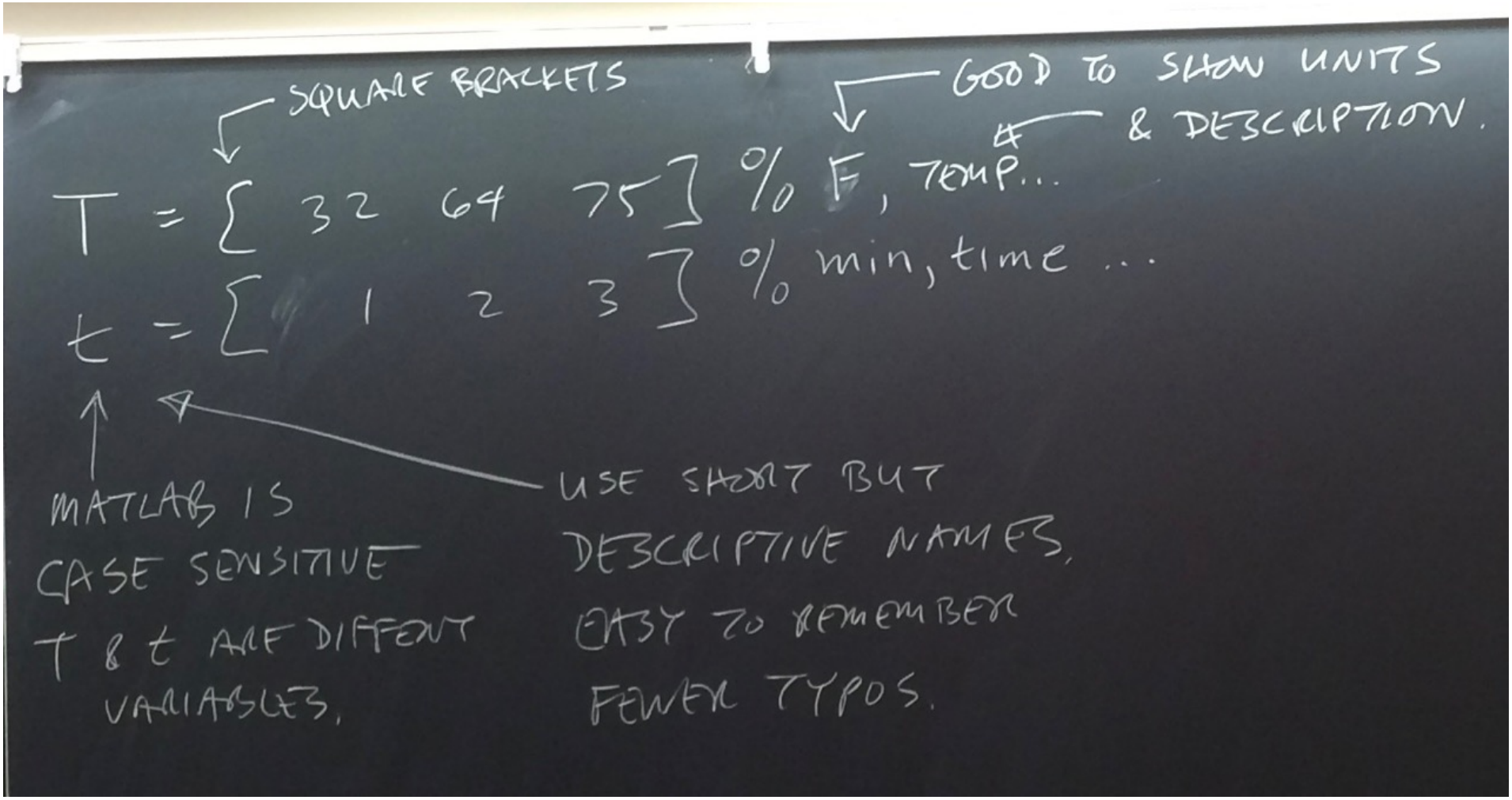
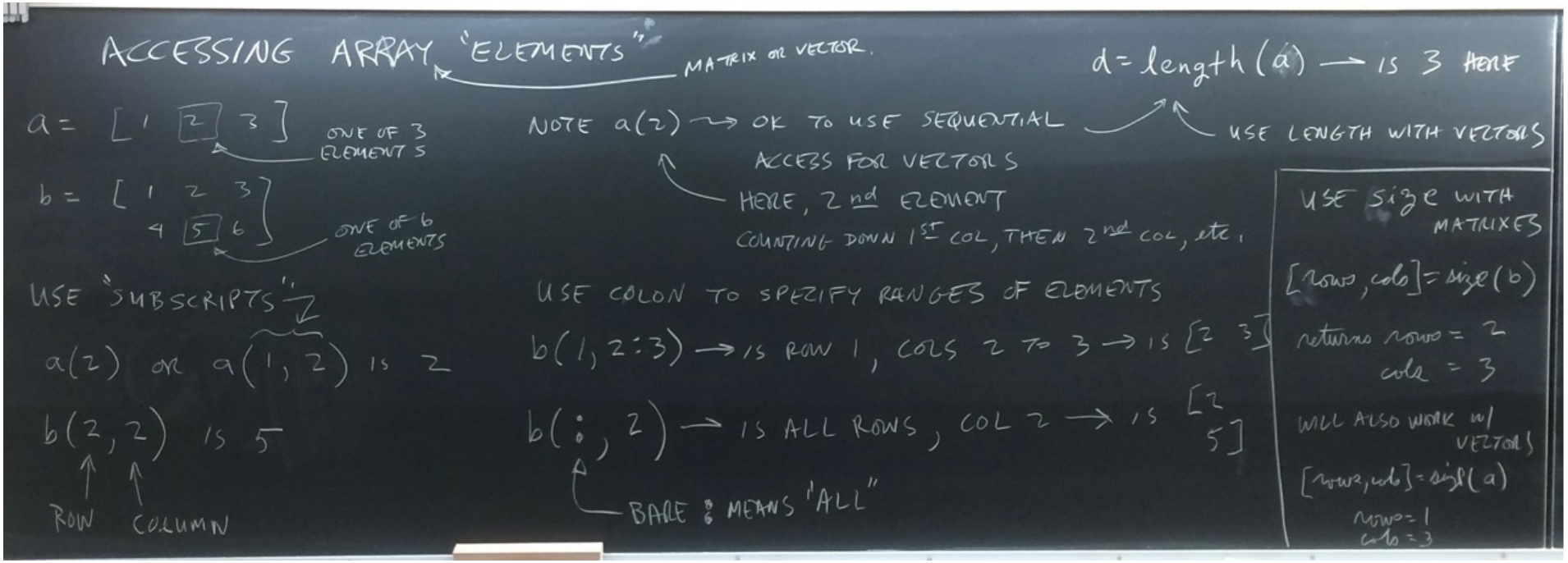
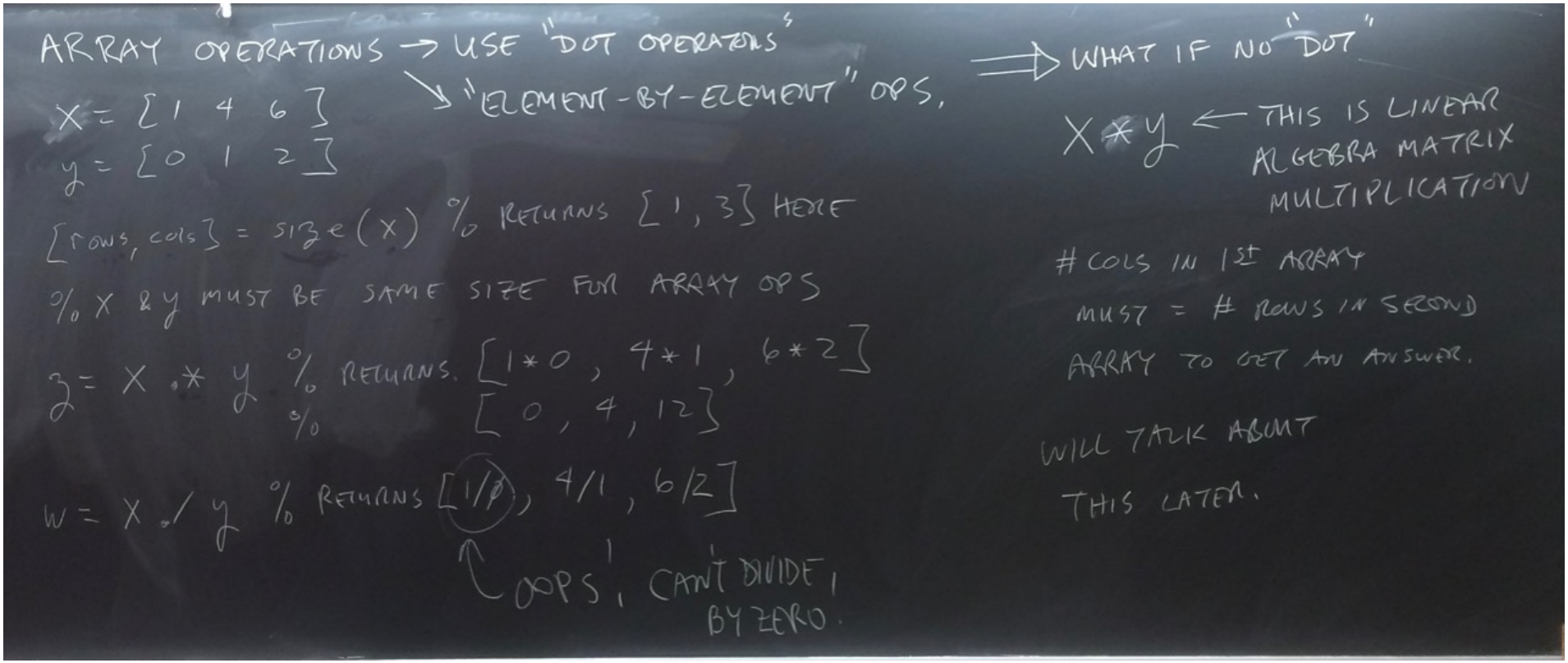

BE READING CHAP 2 → START CHAP 3 FOR NEXT WEEK.
Chap 1 → BUILT-IN FUNCTIONS → ALLOWS YOU TO "CALL" (USE) & RE-USE LINES OF CODE WITH 1 LINE IN YOUR PROGRAM
FUNCTION LIBRARY
function z = nthroot(w, n)
z =
% MY PROGRAM
% LINES OF MY CODE
x = nthroot(y, n) % x = ∛y = y^(1/n) = y^(1/3)
% MORE OF MY LINES
OTHER BUILT-IN FUNCTIONS
a = sin(x) % TRIG. SINE (x)
INPUT "ARGUMENT"
b = cos(x)
FUNCTION - LINES of CODE.. WITH A NAME THAT CAN 'CALL' WITH ONE LINE


Chap 2 — ARRAYS → LINEAR ALGEBRA VECTOR & MATRICES
a = 5 % SCALAR
"ELEMENT" IN ROW 1, COL 3
b = [1 3 5] % ROW VECTOR.
LIKE A LIST.
c = [1 2 3] % COLUMN VECTOR.
d = [1 4 6 8] % 2 X 2 MATRIX
LIKE A TABLE
ROW 2
COL 1
COL 2
d = [1 4 ; 6 8]
d = [1 4 6 8]
"ELEMENT" IN ROW 1, COL 2
OPTIONS FOR ENTERING...
b = [1, 3, 5]
COMMAS OR SPACES SEPARATE ELEMENTS IN A ROW
c = [1 ; 2 ; 3]
SEMI-COLONS OR NEW LINES CREATE NEW ROWS
C = [1 2 3]
C = C'
"TRANSPOSE" OPERATOR, SWITCH ROWS TO COLUMNS & VICE VERSA


OTHER WAYS
COLON → INDICATES A RANGE OF VALUES
a = 1 : 4
USES DEFAULT STEP OF 1
a = 1 : 1 : 4
b = 1 : 2 : 5
START
INCREMENT OR STEP
FINAL VALUE
"DEFAULT" STEP IS 1 IF MIDDLE VALUE LEFT OUT
C = linspace(0, 30, X)
BUILT-IN FUNCTION "LINEAR SPACE"
START VALUE
END VALUE
# OF VALUES
C = linspace(0, 4, 5)
C =
0 1 2 3 4
5 #'s


SQUARE BRACKETS
GOOD TO SHOW UNITS & DESCRIPTION.
T = [ 32 64 75 ] % F, TEMP...
t = [ 1 2 3 ] % min, time ...
MATLAB IS CASE SENSITIVE T & t ARE DIFFENT VARIABLES.
USE SHORT BUT DESCRIPTIVE NAMES. EASY TO REMEMBER FEWER TYPOS.


ACCESSING ARRAY "ELEMENTS"
MATRIX OR VECTOR.
d = length(a) → IS 3 HERE
a = [1 2 3]
ONE OF 3 ELEMENTS
NOTE a(2) → OK TO USE SEQUENTIAL ACCESS FOR VECTORS
HERE, 2nd ELEMENT
COUNTING DOWN 1ST COL, THEN 2ND COL, etc.
USE LENGTH WITH VECTORS
b = [1 2 3
4 5 6]
ONE OF 6 ELEMENTS
USE "SUBSCRIPTS"
a(2) OR a(1, 2) IS 2
b(2, 2) IS 5
ROW
COLUMN
USE COLON TO SPECIFY RANGES OF ELEMENTS
b(1, 2:3) → IS ROW 1, COLS 2 TO 3 → IS [2 3]
b(:, 2) → IS ALL ROWS, COL 2 → IS [2 5]
BARE : MEANS "ALL"
USE size WITH MATRIXES
[rows, cols] = size(b)
returns rows = 2
cols = 3
WILL ALSO WORK W/ VECTORS
[rows, cols] = size(a)
rows = 1
cols = 3


ARRAY OPERATIONS → USE "DOT OPERATORS"
↘ "ELEMENT-BY-ELEMENT" OPS.
x = [1 4 6]
y = [0 1 2]
[rows, cols] = size(x) % RETURNS [1, 3] HERE
% x & y MUST BE SAME SIZE FOR ARRAY OPS
z = x .* y % RETURNS [1*0, 4*1, 6*2]
% [0, 4, 12]
w = x ./ y % RETURNS [1/0, 4/1, 6/2]
↑ OOPS! CAN'T DIVIDE BY ZERO.
⟹ WHAT IF NO "DOT"
x*y ← THIS IS LINEAR ALGEBRA MATRIX MULTIPLICATION
# COLS IN 1st ARRAY MUST = # ROWS IN SECOND ARRAY TO GET AN ANSWER.
WILL TALK ABOUT THIS LATER.