# DATA TYPES

# CONVERTING STRINGS TO NUMBERS

```
>> de = double('!')
>> dq = double('Let''s go!')

de =
    33

dq =
    76   101   116    39   115    32   103   111    33

>> de_lettered = char(de)
>> dq_lettered = char(dq)

de_lettered =
!

dq_lettered =
Let's go!
```

# CONVERTING STRINGS TO NUMBERS

```
for i = 1:8
    for j = (i:11:91)
        thiscode = j+31;
        fprintf('%5.0f %s',thiscode,char(thiscode));
    end
    fprintf('\n');
end
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 32 | 43 + | 54 6 | 65 A | 76 L | 87 W | 98 b | 109 m | 120 x |
| 33 ! | 44 , | 55 7 | 66 B | 77 M | 88 X | 99 c | 110 n | 121 y |
| 34 " | 45 - | 56 8 | 67 C | 78 N | 89 Y | 100 d | 111 o | 122 z |
| 35 # | 46 . | 57 9 | 68 D | 79 O | 90 Z | 101 e | 112 p | |
| 36 $ | 47 / | 58 : | 69 E | 80 P | 91 [ | 102 f | 113 q | |
| 37 % | 48 0 | 59 ; | 70 F | 81 Q | 92 \ | 103 g | 114 r | |
| 38 & | 49 1 | 60 < | 71 G | 82 R | 93 ] | 104 h | 115 s | |
| 39 ' | 50 2 | 61 = | 72 H | 83 S | 94 ^ | 105 i | 116 t | |

```
Num1 = 123.456
Str1 = '567.890'
strOfNum1 = num2str(Num1);
numOfStr1 = str2num(Str1);
```

# CELL ARRAYS

# CELL ARRAYS

```
 MyMatrix1 = [
'oranges'
'bananas']

MyMatrix2 = [
  'apples'
  'oranges']

MyMatrix1 =

oranges
bananas
```

연결(Concatenate)된 행렬의 차원이 일치하지 않습니다. : %Why is there error message for MyMatrix2?

```
MyCells = {
    'apples'
    'oranges'}
for i = 1:2
     thisword = MyCells{i}
end
```

# CELL ARRAYS

```
Whos


MyCells =
    'apples'
    'oranges'

thisword =
apples

thisword =
oranges

  Name          Size                Bytes  Class     Attributes
  MyCells       2x1                   250  cell
  i             1x1                     8  double
  thisword      1x7                    14  char
```

# CELL ARRAYS: MIXING NUMBERS AND CHARACTER STRINGS

```
 c = {[ 1 2 3]
     [4 5 6 7]
     ['rats mice']; [' voles']
     [1 3]}
c_second_row = c{2}
c_second_row_middle_numbers = c{2}(2:3)
c_third_row = c{3}
c_third_row_second_character = c{3}(2)
```

```
c =
    [1x3 double]
    [1x4 double]
    'rats mice'
    ' voles'
    [1x2 double]

c_second_row =
                4     5     6     7

c_second_row_middle_numbers =
                      5     6

c_third_row =
              rats mice

c_third_row_second_character =
                           a
```

# CELL ARRAYS: MIXING NUMBERS AND CHARACTER STRINGS

```
>> Names_and_Numbers = {
'Bob' [90 95]
'Jane' 100
}

Names_and_Numbers =

    'Bob'      [1x2 double]
    'Jane'     [       100]
```

```
>> Name1 = cell2mat(Names_and_Numbers(1,1))

Name1 =

Bob

>> Numbers1 =
cell2mat(Names_and_Numbers(1,2))

Numbers1 =

     90    95
```

# CELL ARRAYS: MIXING NUMBERS AND CHARACTER STRINGS

```
 for produce = {'Apple' 'Artichoke' 'Banana' 'Broccoli'...
                'Cherry' 'Cauliflower'}
    productName = char(produce); % convert cell to char
    switch productName
        case {'Apple' 'Banana' 'Cherry'}
            fprintf('%s is a fruit.\n', productName);
        case {'Artichoke' 'Broccoli' 'Cauliflower'}
            fprintf('%s is a vegetable.\n', productName);
    end
end
```

***Apple is a fruit.***
***Artichoke is a vegetable.***
***Banana is a fruit.***
***Broccoli is a vegetable.***
***Cherry is a fruit.***
***Cauliflower is a vegetable.***

# STRUCTURES

# STRUCTURES

```
>> trials = [
1 2 200
2 2 200
1 1 400
]

trials =

     1     2   200
     2     2   200
     1     1   400
```

Here, let's assume that you want to have a data matrix where you want to specify the side, brightness and the duration.
The first column: side (1 = left, 2 = right)
The second column: brightness (1 = dim, 2 = bright)
The third column: duration ( ms)

# STRUCTURES

Or, instead, you can make a structure.

```
%Initialize struct fields and values
trial(1).side = 'left';
trial(1).brightness = 'bright';
trial(1).duration = 200;

trial(2).side = 'right';
trial(2).brightness = 'bright';
trial(2).duration = 200;

trial(3).side = 'left';
trial(3).brightness = 'dim';
trial(3).duration = 400;
```

# STRUCTURES

```
>> trial

trial =

1x3 struct array with fields:

    side
    brightness
    duration


>> trial(3)

ans =

          side: 'left'
    brightness: 'dim'
      duration: 400
```

```
>> trial(3).side

ans =

left

>> [trial(:).duration]

ans =

   200   200   400
```

# STRUCTURES

```
subject(1).RTs = [
    500 400 350
    450 375 325
    ];
subject(1).errors = [
    10 8 6
    4 3 2
    ];
subject(2).RTs = [
    600 500 450
    550 475 425
    500 425 400
    ];
subject(2).errors = [
    10 8 6
    4 3 2
    3 2 1
    ] ;
subject(2).debrief = true;
subject(2).comment = 'That was a really cool experiment!';
```

# STRUCTURES

```
subject =

1x2 struct array with fields:

    RTs
    errors
    debrief
    comment

>> subject(1)

ans =

        RTs: [2x3 double]
     errors: [2x3 double]
    debrief: []
    comment: []
```

```
>> subject(2)

ans =

        RTs: [3x3 double]
     errors: [3x3 double]
    debrief: 1
    comment: 'That was a really cool experiment!'
```

# STRUCTURES

***Save data into file***

```
outfilename = 'RTdata.txt';
outfile = fopen(outfilename,'wt');
% print header line
fprintf(outfile,'sub\tRT\tErrors\n');
% print data table
for subjectnumber = 1:2

fprintf(outfile,'%3d\t%5.1f\t%3.1f\n',subjectnu
mber,...
     mean(subject(subjectnumber).RTs(:)),...
     mean(subject(subjectnumber).errors(:)));
end
type('RTdata.txt')
```

# VARIOUS FUNCTIONS
## (FOR HANDLING DATA)

# FUNCTION DEAL()

```
[mystruct(1:8).initiallyZeroVariable] = deal(0);
[mystruct.initiallyEmpty] = deal([]);
[mystruct.random] = ...
    deal(randi(10),randi(10),randi(10),randi(10),...
         randi(10),randi(10),randi(10),randi(10));
[mystruct.integers] = deal(8,7,6,5,4,3,2,1);
ms_1 = mystruct(1)
ms_2 = mystruct(2)
ms_8 = mystruct(8)
```

```
ms_1 =
    initiallyZeroVariable: 0
            initiallyEmpty: []
                   random: 9
                 integers: 8

ms_2 =
    initiallyZeroVariable: 0
            initiallyEmpty: []
                   random: 10
                 integers: 7

ms_8 =
    initiallyZeroVariable: 0
            initiallyEmpty: []
                   random: 6
                 integers: 1
```

# FUNCTION DEAL()

```
% Reading field of struct to cell array using deal
[TheIntegersCellArray{1:length(mystruct)}] = ...
     deal(mystruct(:).integers)
% Converting cell array to matrix using cell2mat
TheIntegerMatrix = cell2mat(TheIntegersCellArray)
```

```
TheIntegersCellArray =

    [8]    [7]    [6]    [5]    [4]    [3]    [2]    [1]


TheIntegerMatrix =

     8     7     6     5     4     3     2     1
```

# FUNCTION DEAL()

```
>> [subject(1:8).data] = deal([]);
>> [subject.sex] = deal('female');
>> subject

subject =
1x8 struct array with fields:

    data
    sex

>> subject(1)
ans =

    data: []
     sex: 'female'
```

```
>> [cArray{1:8}] = deal(subject.sex)
cArray =

  열 1 ~ 5

    'female'    'female'    'female'
'female'    'female'

  열 6 ~ 8

    'female'    'female'    'female'

>> [a, b, c, d, e, f, g, h] = deal(cArray{:});
>> a
a =

female
```

# FUNCTION STRFIND()

```
s = ['How much wood could a wood chuck chuck if
a wood chuck could chuck wood?'];
all_wood_in_s = strfind(s,'wood')
all_could_in_s = strfind(s,'could')
all_should_in_s = strfind(s,'should')
any_wood_in_s = any(strfind(s,'wood'))
any_should_in_s = any(strfind(s,'should'))
```

```
all_wood_in_s =
    10    23    45    68

all_could_in_s =
    15    56

all_should_in_s =
     []

any_wood_in_s =
     1

any_should_in_s =
     0
```

# FUNCTION STRREP()

```
s = ['How much wood could a wood chuck chuck if a wood chuck could chuck wood?'];
s1 = strrep(s,'wood','cider');
s2 = strrep(s1,'chuck','press');
```

***>> s1 =***

***How much cider could a cider chuck chuck if a cider chuck could chuck cider?***

***>> s2***

***s2 =***

***How much cider could a cider press press if a cider press could press cider?***

# READING DATA FROM FILE

```
infilename = 'RTdata.txt';
infile = fopen(infilename);
firstline = fgetl(infile); %read the header line
headers = textscan(firstline,'%s');
cell_of_headers = headers{1}(1:3)';
matrix_of_numbers = [];

while ~feof(infile)
    nextline = fgetl(infile);
    nextvalues = textscan(nextline,'%f');
    matrix_of_numbers = [matrix_of_numbers;
nextvalues{1}(1:3)'];
end
```

```
fclose(infile);


cell_of_headers =

    'sub'    'RT'    'Errors'

matrix_of_numbers =

    1.0000  400.0000    5.5000
    2.0000  480.6000    4.3000
```

# READING DATA FROM FILE

```
% DoSimon.m
fin = fopen('Simon.txt');
allRTs = [];
%Skip the header line
headerline = fgetl(fin);
while ~feof(fin)
    aline = fgetl(fin);

    %Read in the variables
    cellvalues =
textscan(aline,'%d %s %s %s %s %s %f');
    Trnum = cell2mat(cellvalues(1));
    side = char(cellvalues{2});
    stim = char(cellvalues{3});
    comp = char(cellvalues{4});
    Key = char(cellvalues{5});
    Resp = char(cellvalues{6});
    RT = cell2mat(cellvalues(7));
```

```
 % Assemble the correct trial RT's
    if strcmp(Resp,'correct')
        allRTs = [allRTs RT];
    end

end
meanRT = mean(allRTs);
fprintf('Mean of correct RTs is %f\n',meanRT);
```

# EXERCISE

Create a 5×3 cell array, G, in which the first row contains the name of one student (Adam, Brad, Charley, David, or Emily) in the first column of the cell array; the student's corresponding numerical average (90, 92, 96, 95, 88) in the second column of the cell array; and the student's letter grade (A-, A-, A, A, B+) in the third column.

Represent the same data as above in a 5×1 struct array, studentStruct(1:5), with two fields, name, and average initialized as above.  Write a program to compute the letter grade based on studentStruct(i).average, and record it in studentStruct(i).letter for each student.

# EXERCISE

Generate a data set using the code below, and verify the accuracy of your program by comparing its checksum output with that in the output below

```
% % Code % %
rng('default')
for n = 1:20
    r1 = randn;
    r2 = mean([r1 r1 randn]) + .4;
    subject(n).score1 = r1;
    subject(n).score2 = r2;
end
subject
checksum1 = sum([subject(:).score1])
checksum2 = sum([subject(:).score2])
```

```
% Output %

 subject =
 1x20 struct array with fields:
     score1
     score2
 checksum1 =
     4.5867
 checksum2 =
    13.5765
```

Now, compute the correlation coefficient between score1 and score2.

# EXERCISE

Write a program to administer a computerized questionnaire on a topic of interest to you. Use a structure data type and allow participants to answer with whole sentences or phrases for at least some items. Save the data in an external file. Record the time to answer each question.