Data Structure

Chul Min Yeum

Assistant Professor

Civil and Environmental Engineering

University of Waterloo, Canada

AE121: Computational Method

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

Last updated: 2019-06-19

Cell Array

- A **cell array** is a type of data structure that can **store different types of values** in its elements
- A cell array could be a vector (row or column) or a matrix
- It is an array, so indices are used to refer to the elements

For example, in “Pressure Calculation” assignment, what if # of measurements or # of days are different at each stations?

Creating Cell Arrays

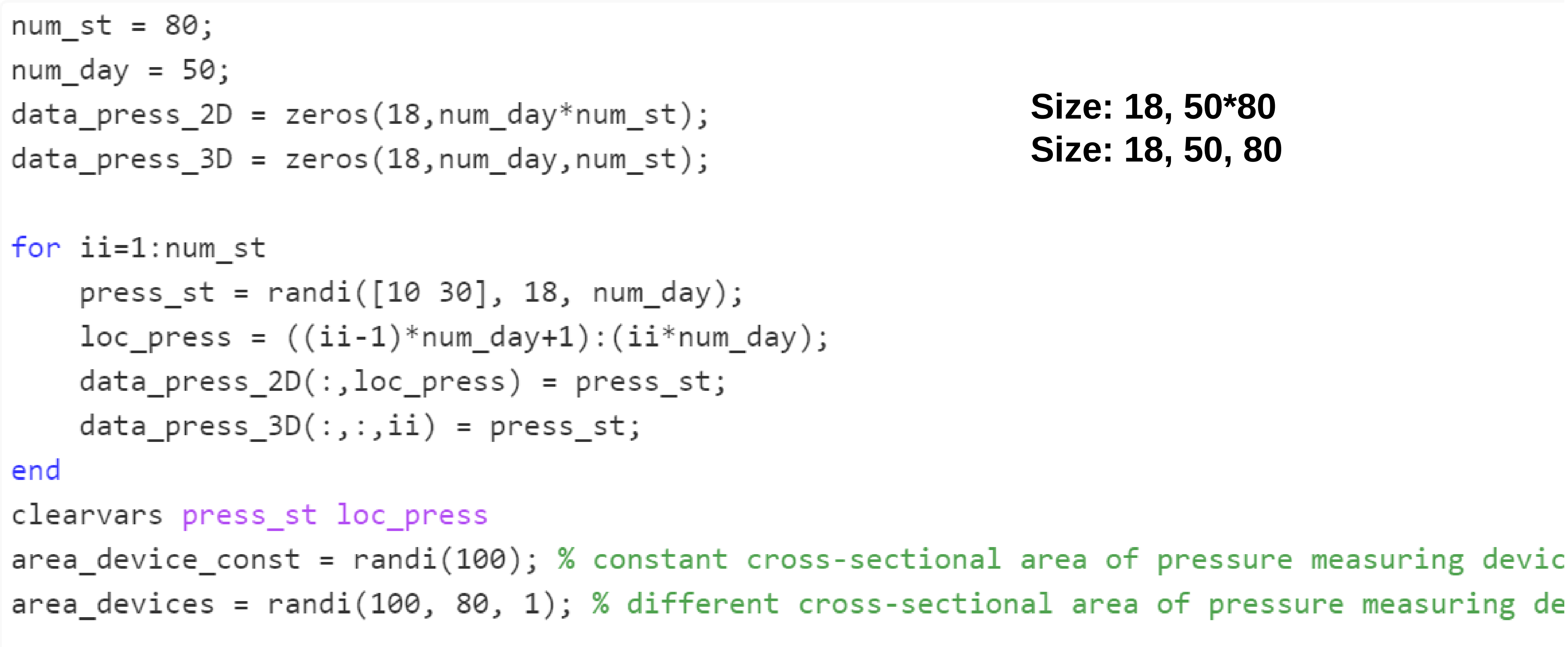
- The syntax used to create a cell array is curly braces { } instead of [ ]
- The direct method is to put values in the row(s) separated by commas or spaces, and to separate the rows with semicolons (so, same as other arrays) – the difference is using { } instead of [ ]
- The cell function can also be used to preallocate by passing the dimensions of the cell array, e.g.

```
cell(4,2)
```

Revisit: Pressure Calculation

Pressure data have been measurement from 80 stations (Station 1, 2, ... 80), at 1hr intervals, starting at 6:00AM and ending at 11:00PM (all 18 measuremets in each location) during fifty (50) days.

Here is the script to generate syntatic values for generating data.

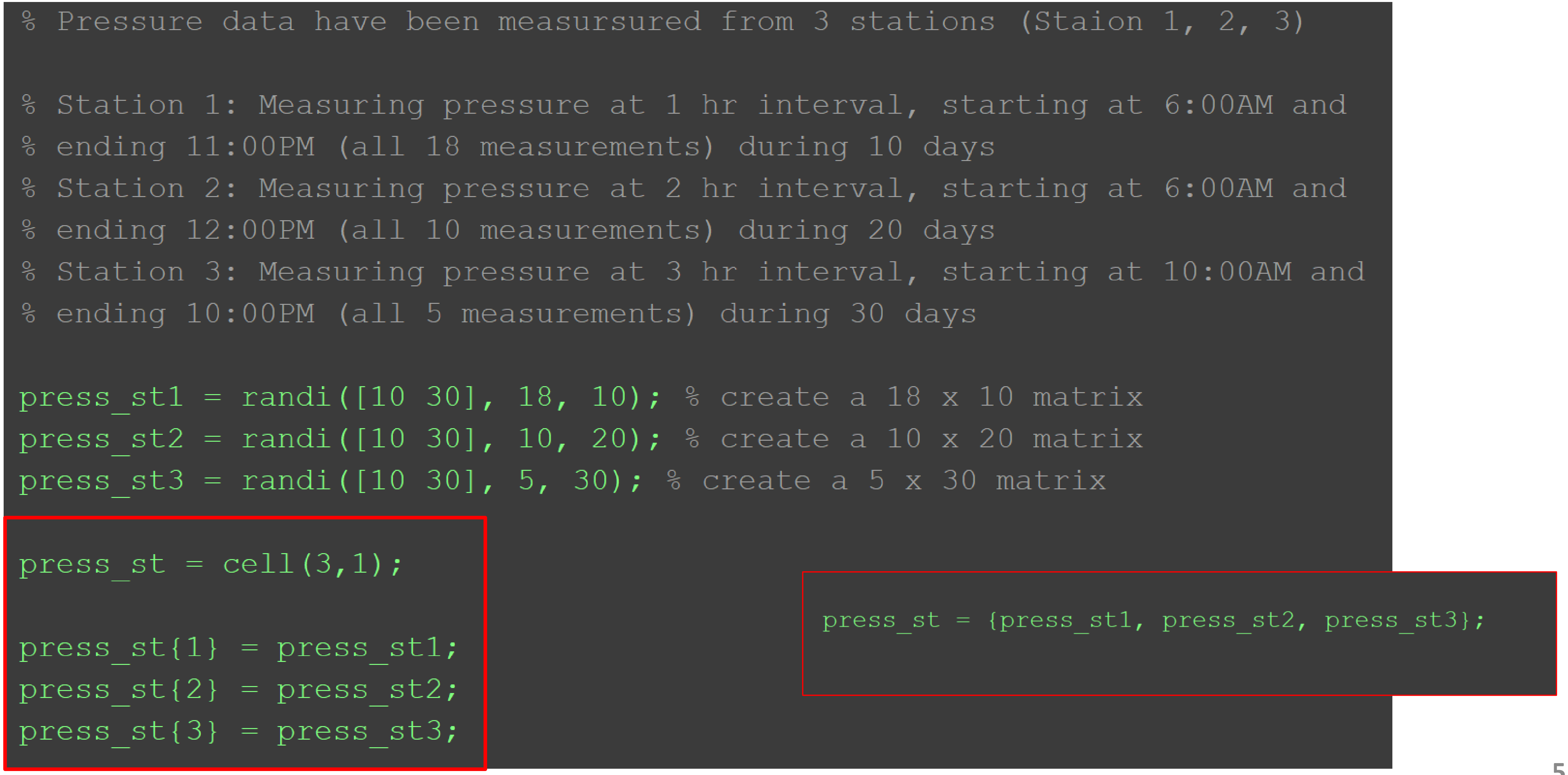


```
num_st = 80;
num_day = 50;
data_press_2D = zeros(18,num_day*num_st);
data_press_3D = zeros(18,num_day,num_st);

for ii=1:num_st
    press_st = randi([10 30], 18, num_day);
    loc_press = ((ii-1)*num_day+1):(ii*num_day);
    data_press_2D(:,loc_press) = press_st;
    data_press_3D(:,:,ii) = press_st;
end
clearvars press_st loc_press
area_device_const = randi(100); % constant cross-sectional area of pressure measuring devic
area_devices = randi(100, 80, 1); % different cross-sectional area of pressure measuring de
```

Size: 18, 50*80

Size: 18, 50, 80

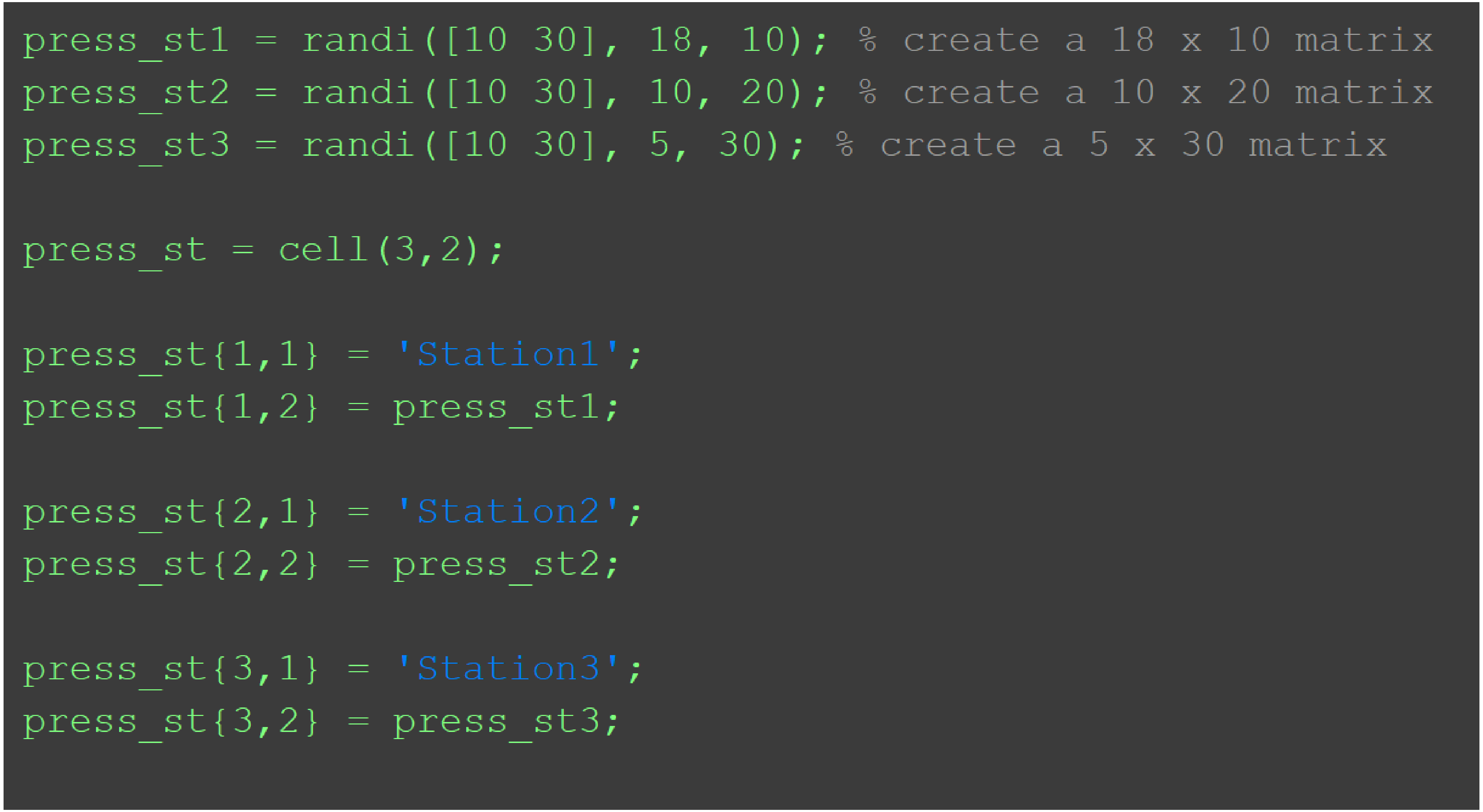
Example: Use of a Cell Array



```
% Pressure data have been measursured from 3 stations (Staion 1, 2, 3)

% Station 1: Measuring pressure at 1 hr interval, starting at 6:00AM and
% ending 11:00PM (all 18 measurements) during 10 days
% Station 2: Measuring pressure at 2 hr interval, starting at 6:00AM and
% ending 12:00PM (all 10 measurements) during 20 days
% Station 3: Measuring pressure at 3 hr interval, starting at 10:00AM and
% ending 10:00PM (all 5 measurements) during 30 days

press_st1 = randi([10 30], 18, 10); % create a 18 x 10 matrix
press_st2 = randi([10 30], 10, 20); % create a 10 x 20 matrix
press_st3 = randi([10 30], 5, 30); % create a 5 x 30 matrix

press_st = cell(3,1);

press_st{1} = press_st1;
press_st{2} = press_st2;
press_st{3} = press_st3;
```

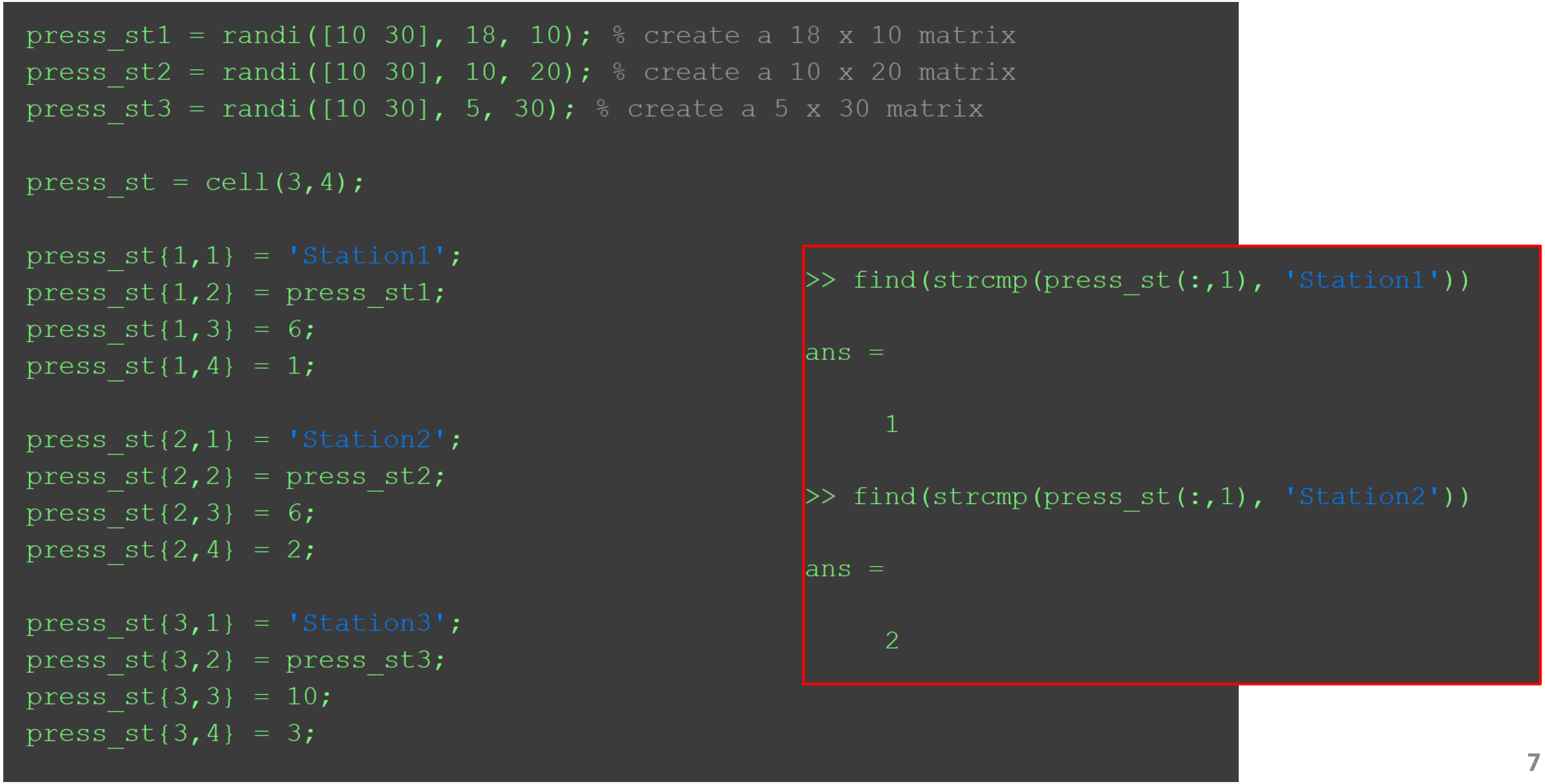
```
press_st = {press_st1, press_st2, press_st3};
```

Example: Containing Multiple Data Types



```
press_st1 = randi([10 30], 18, 10); % create a 18 x 10 matrix
press_st2 = randi([10 30], 10, 20); % create a 10 x 20 matrix
press_st3 = randi([10 30], 5, 30); % create a 5 x 30 matrix

press_st = cell(3,2);

press_st{1,1} = 'Station1';
press_st{1,2} = press_st1;

press_st{2,1} = 'Station2';
press_st{2,2} = press_st2;

press_st{3,1} = 'Station3';
press_st{3,2} = press_st3;
```

Example: Containing Multiple Data Types (Continue)



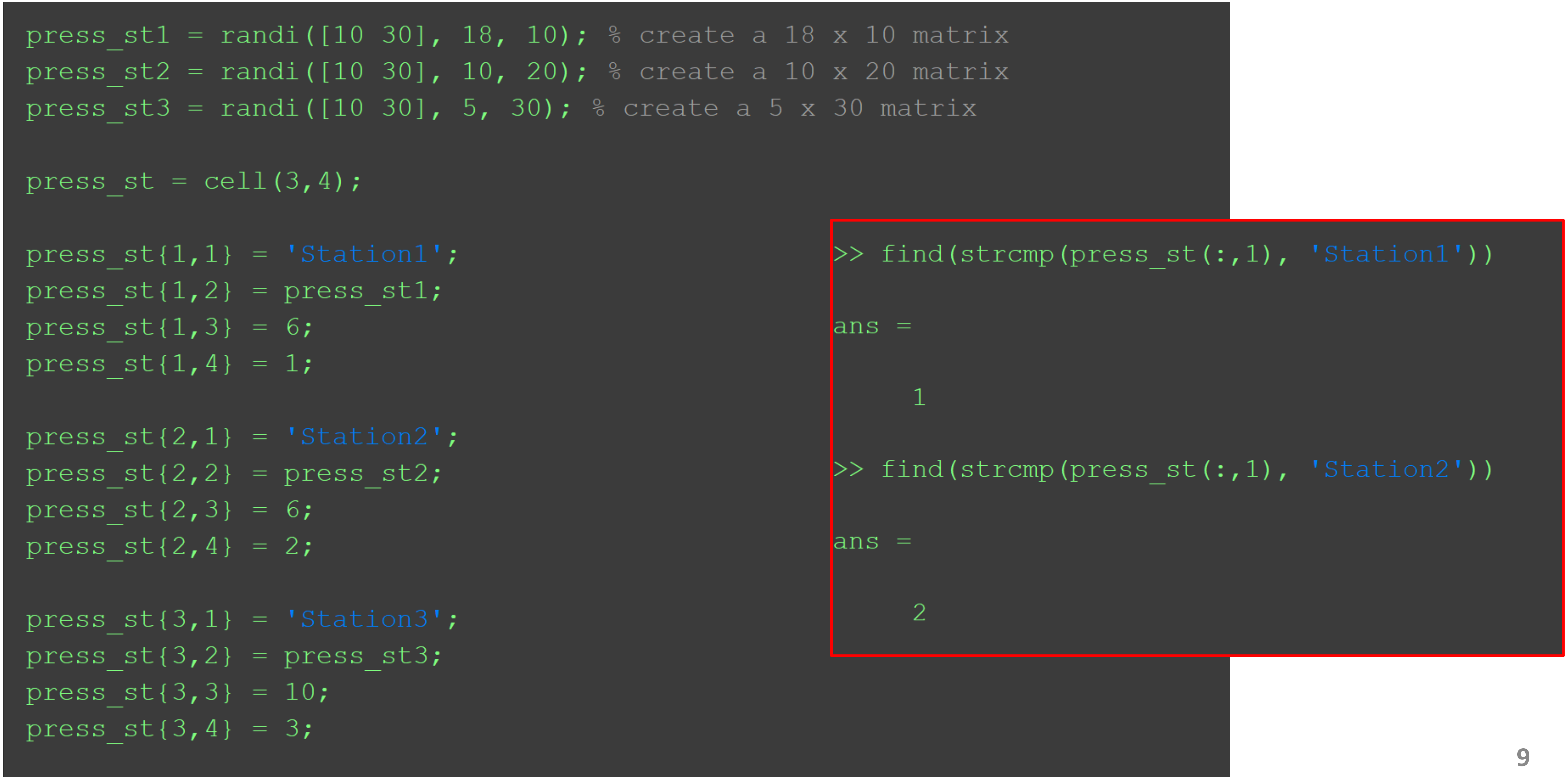
```
press_st1 = randi([10 30], 18, 10); % create a 18 x 10 matrix
press_st2 = randi([10 30], 10, 20); % create a 10 x 20 matrix
press_st3 = randi([10 30], 5, 30); % create a 5 x 30 matrix

press_st = cell(3,4);

press_st{1,1} = 'Station1';
press_st{1,2} = press_st1;
press_st{1,3} = 6;
press_st{1,4} = 1;

press_st{2,1} = 'Station2';
press_st{2,2} = press_st2;
press_st{2,3} = 6;
press_st{2,4} = 2;

press_st{3,1} = 'Station3';
press_st{3,2} = press_st3;
press_st{3,3} = 10;
press_st{3,4} = 3;
```

```
>> find(strcmp(press_st(:,1), 'Station1'))

ans =

     1

>> find(strcmp(press_st(:,1), 'Station2'))

ans =

     2
```

Referring to Cell Array Elements

- **The elements in cell arrays are cells**
- There are two methods of referring to parts of cell arrays:
  - you can refer to the cells; **this is called cell indexing and parentheses are used**
  - you can refer to the contents of the cells; this is called content indexing and curly braces are used
- For example:
- >> ca = {2:4, 'hello'};
- >> ca(1)
- ans =
-    [1x3 double]
- >> ca{1}
- ans =
-     2   3   4

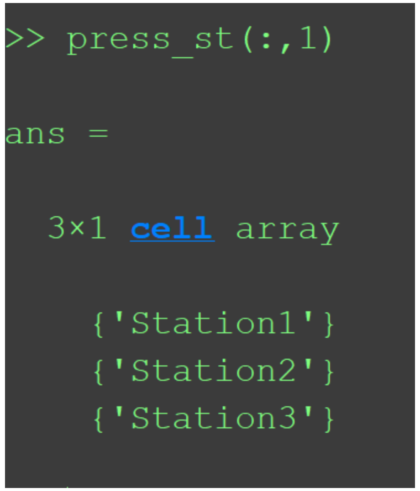
Example: Two Methods to Refer the Cell Array



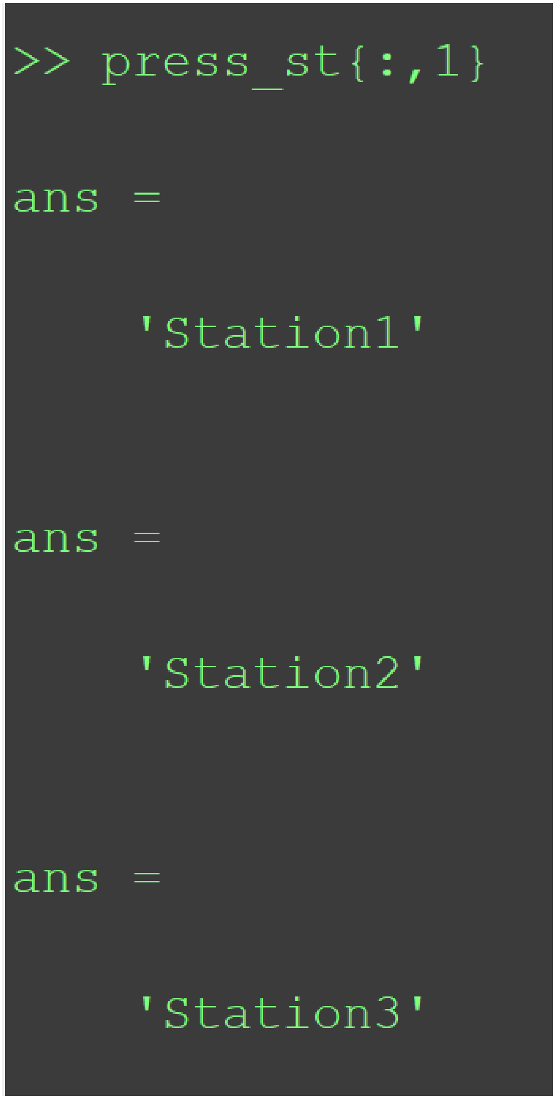
```
press_st1 = randi([10 30], 18, 10); % create a 18 x 10 matrix
press_st2 = randi([10 30], 10, 20); % create a 10 x 20 matrix
press_st3 = randi([10 30], 5, 30); % create a 5 x 30 matrix

press_st = cell(3,4);

press_st{1,1} = 'Station1';
press_st{1,2} = press_st1;
press_st{1,3} = 6;
press_st{1,4} = 1;

press_st{2,1} = 'Station2';
press_st{2,2} = press_st2;
press_st{2,3} = 6;
press_st{2,4} = 2;

press_st{3,1} = 'Station3';
press_st{3,2} = press_st3;
press_st{3,3} = 10;
press_st{3,4} = 3;
```

```
>> find(strcmp(press_st(:,1), 'Station1'))

ans =

     1

>> find(strcmp(press_st(:,1), 'Station2'))

ans =

     2
```

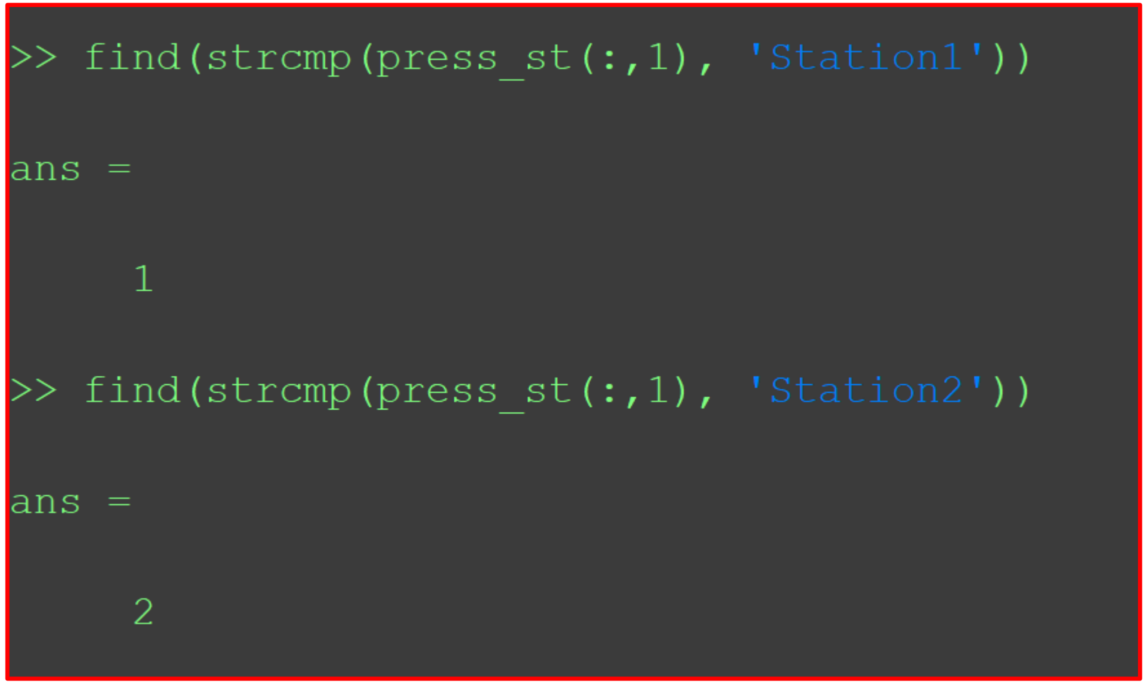
Example: Two Methods to Refer the Cell Array (Continue)



```
>> press_st(:,1)

ans =

  3×1 cell array

    {'Station1'}
    {'Station2'}
    {'Station3'}
```

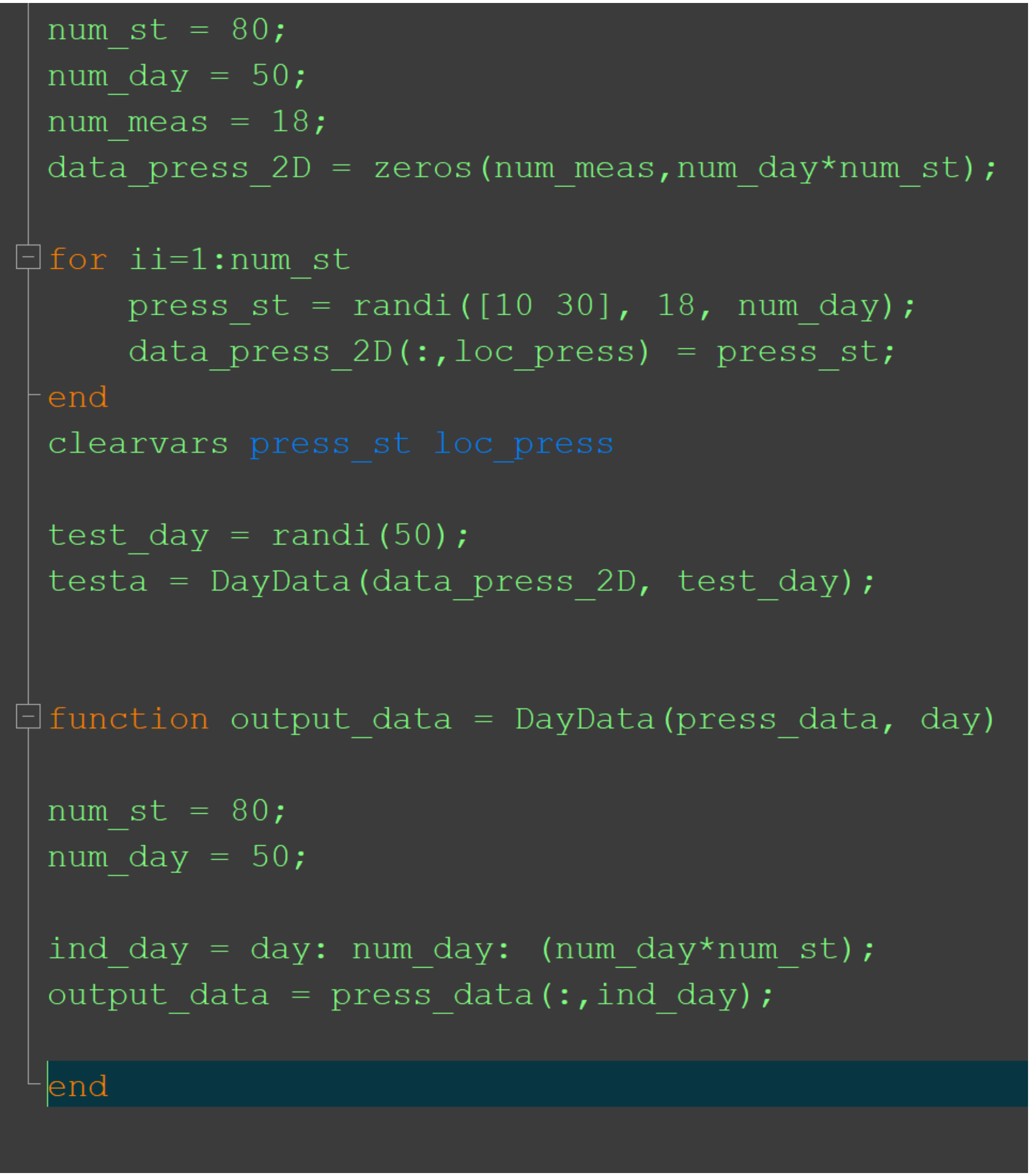


```
>> press_st{:,1}

ans =

    'Station1'


ans =

    'Station2'


ans =

    'Station3'
```

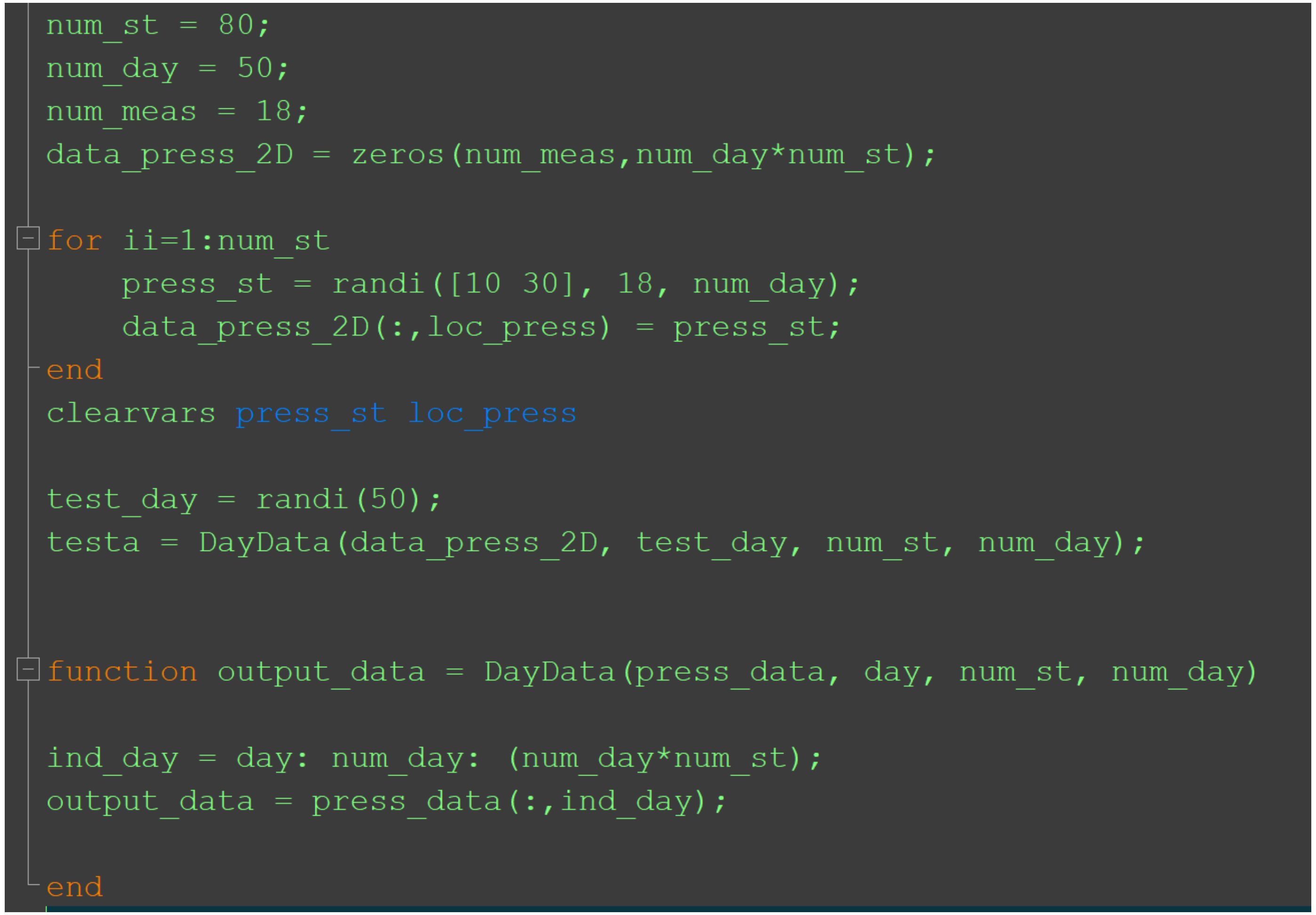


```
>> find(strcmp(press_st(:,1), 'Station1'))

ans =

     1

>> find(strcmp(press_st(:,1), 'Station2'))

ans =

     2
```

Example: Multiple Input Data in Functions

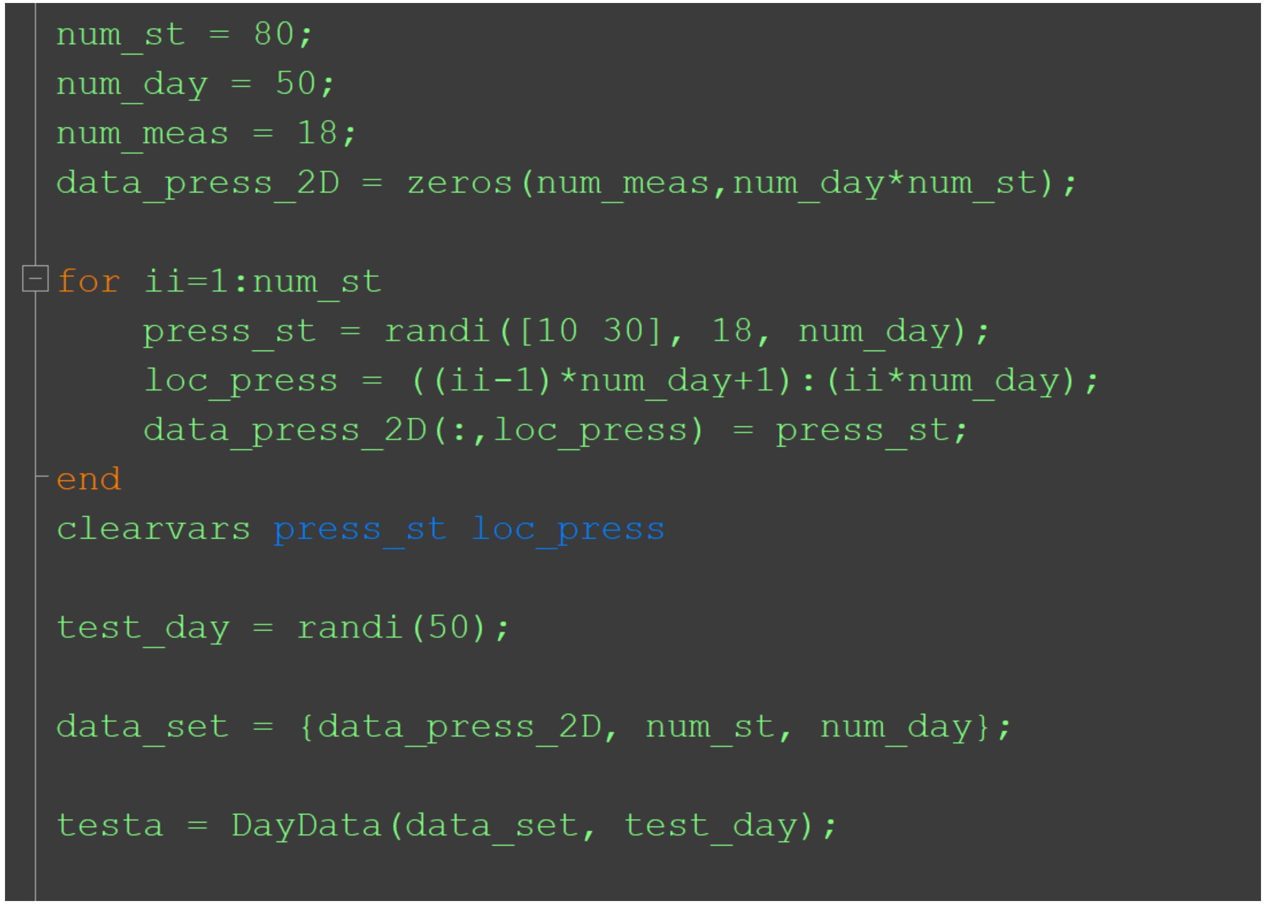


```
num_st = 80;
num_day = 50;
num_meas = 18;
data_press_2D = zeros(num_meas,num_day*num_st);

for ii=1:num_st
    press_st = randi([10 30], 18, num_day);
    data_press_2D(:,loc_press) = press_st;
end
clearvars press_st loc_press

test_day = randi(50);
testa = DayData(data_press_2D, test_day);


function output_data = DayData(press_data, day)

num_st = 80;
num_day = 50;

ind_day = day: num_day: (num_day*num_st);
output_data = press_data(:,ind_day);

end
```

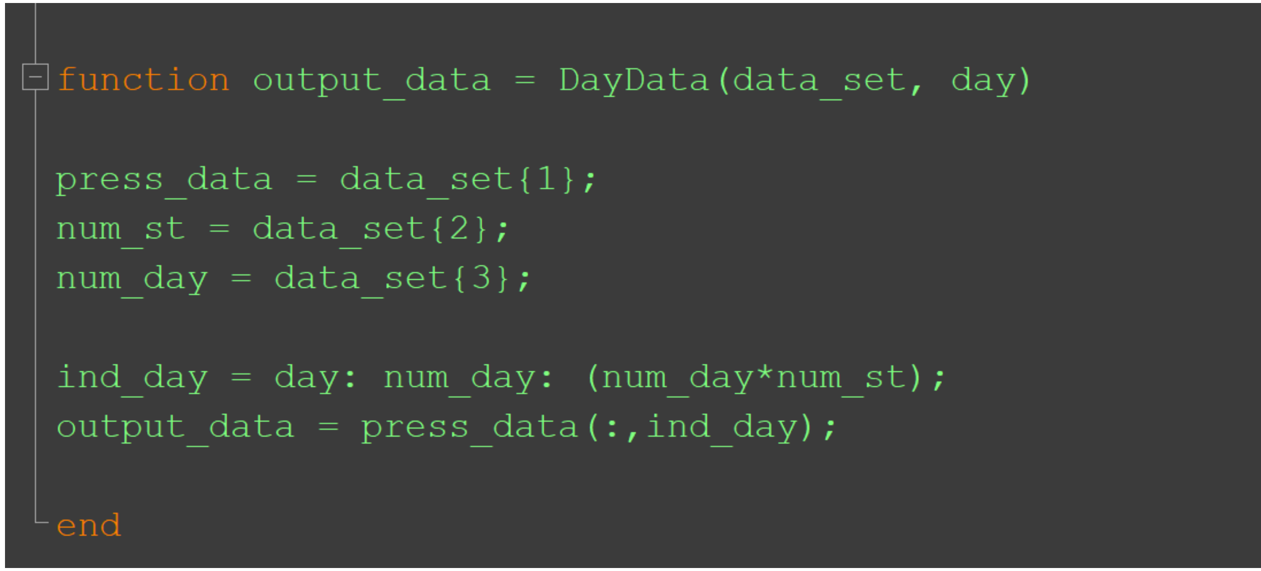


```
num_st = 80;
num_day = 50;
num_meas = 18;
data_press_2D = zeros(num_meas,num_day*num_st);

for ii=1:num_st
    press_st = randi([10 30], 18, num_day);
    data_press_2D(:,loc_press) = press_st;
end
clearvars press_st loc_press

test_day = randi(50);
testa = DayData(data_press_2D, test_day, num_st, num_day);


function output_data = DayData(press_data, day, num_st, num_day)

ind_day = day: num_day: (num_day*num_st);
output_data = press_data(:,ind_day);

end
```

Example: Multiple Input Data in Functions (Continue)



```
num_st = 80;
num_day = 50;
num_meas = 18;
data_press_2D = zeros(num_meas,num_day*num_st);

for ii=1:num_st
    press_st = randi([10 30], 18, num_day);
    loc_press = ((ii-1)*num_day+1):(ii*num_day);
    data_press_2D(:,loc_press) = press_st;
end
clearvars press_st loc_press

test_day = randi(50);

data_set = {data_press_2D, num_st, num_day};

testa = DayData(data_set, test_day);
```

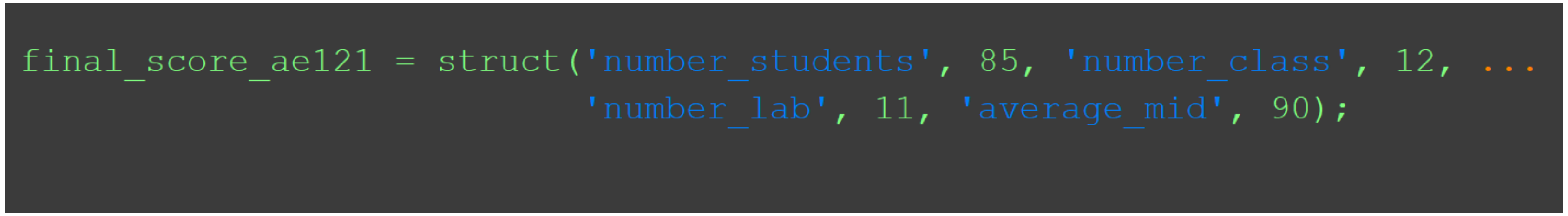


```
function output_data = DayData(data_set, day)

press_data = data_set{1};
num_st = data_set{2};
num_day = data_set{3};

ind_day = day: num_day: (num_day*num_st);
output_data = press_data(:,ind_day);

end
```

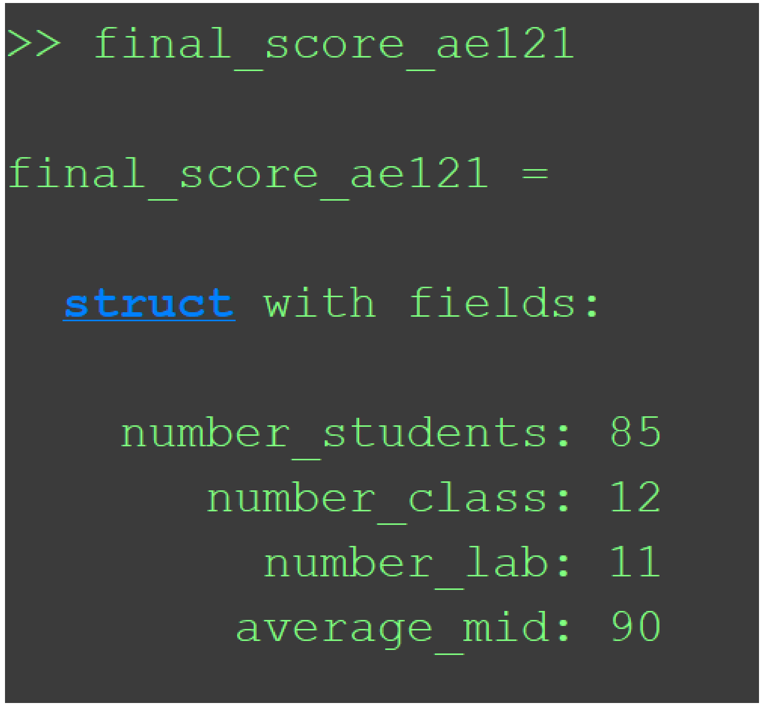
Structure Variables

- Structures store values of different types, in fields
- Fields are given names; they are referred to as
- **structurename.fieldname** using the dot operator
- Structure variables can be initialized using the struct function, which takes pairs of arguments (field name as a string followed by the value for that field)
- To print, disp will display all fields; fprintf can only print individual fields

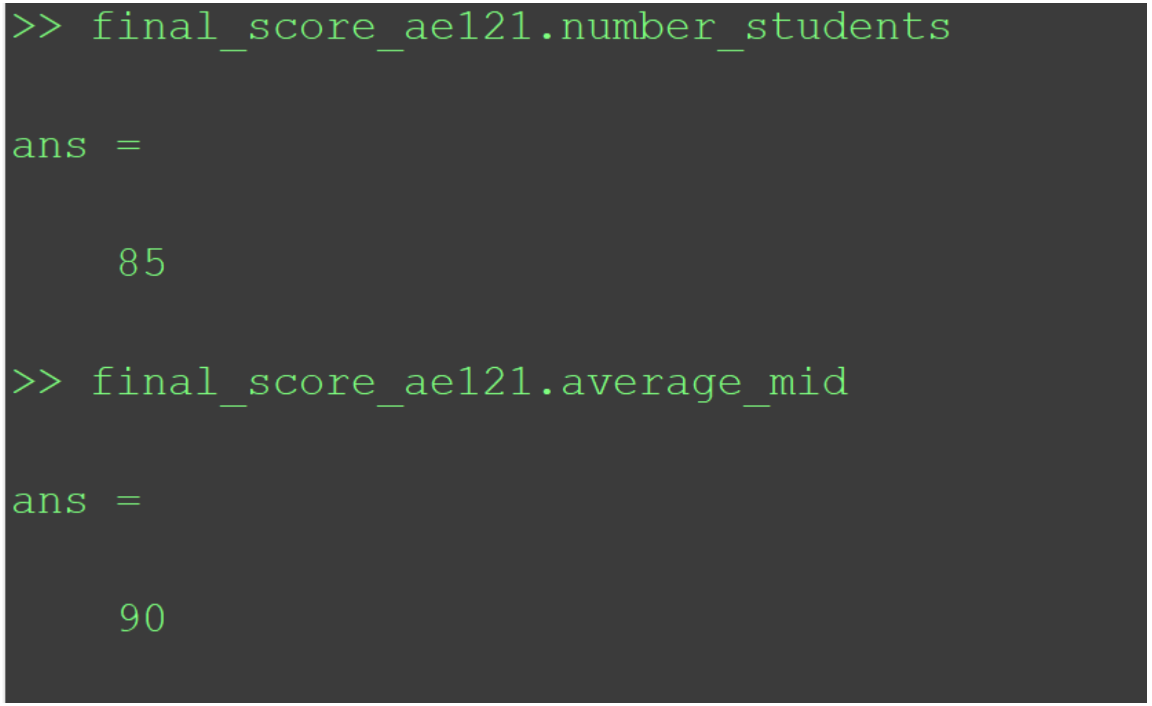
Example: Simple Example



```
final_score_ae121 = struct('number_students', 85, 'number_class', 12, ...
                           'number_lab', 11, 'average_mid', 90);
```

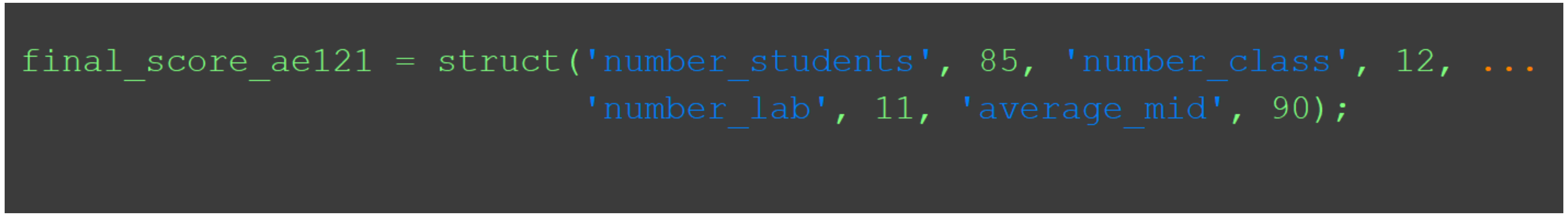


```
>> final_score_ae121

final_score_ae121 =

  struct with fields:

    number_students: 85
       number_class: 12
         number_lab: 11
        average_mid: 90
```

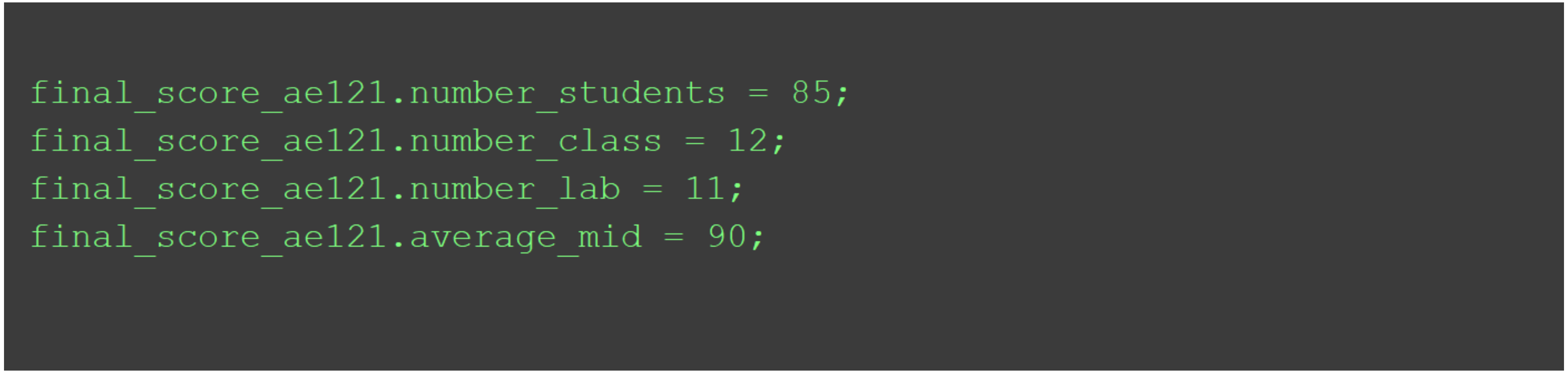


```
>> final_score_ae121.number_students

ans =

    85

>> final_score_ae121.average_mid

ans =

    90
```

Example: Simple Example (Continue)



```
final_score_ae121 = struct('number_students', 85, 'number_class', 12, ...
                           'number_lab', 11, 'average_mid', 90);
```



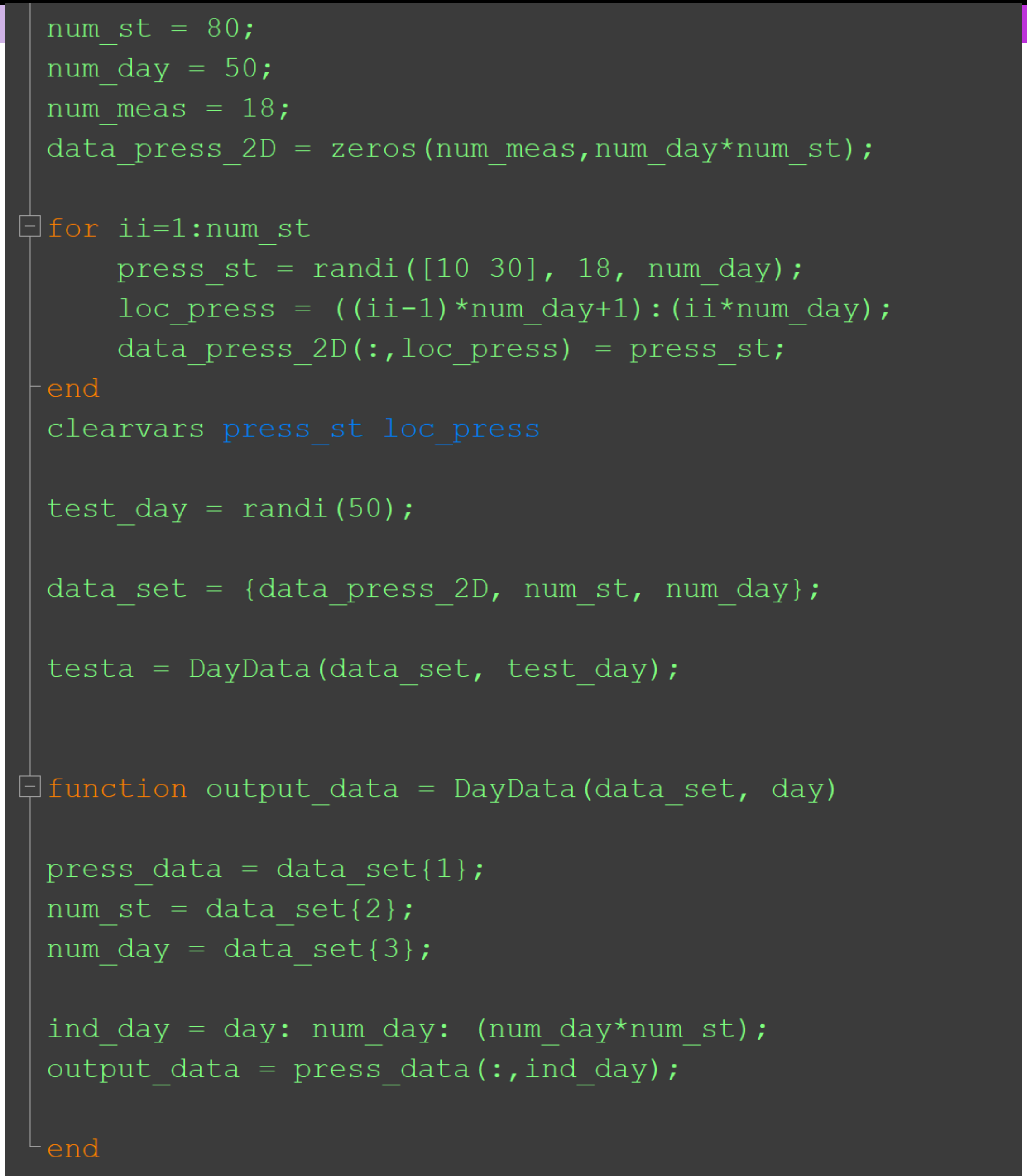
```
final_score_ae121.number_students = 85;
final_score_ae121.number_class = 12;
final_score_ae121.number_lab = 11;
final_score_ae121.average_mid = 90;
```

Cell Arrays vs. Structs

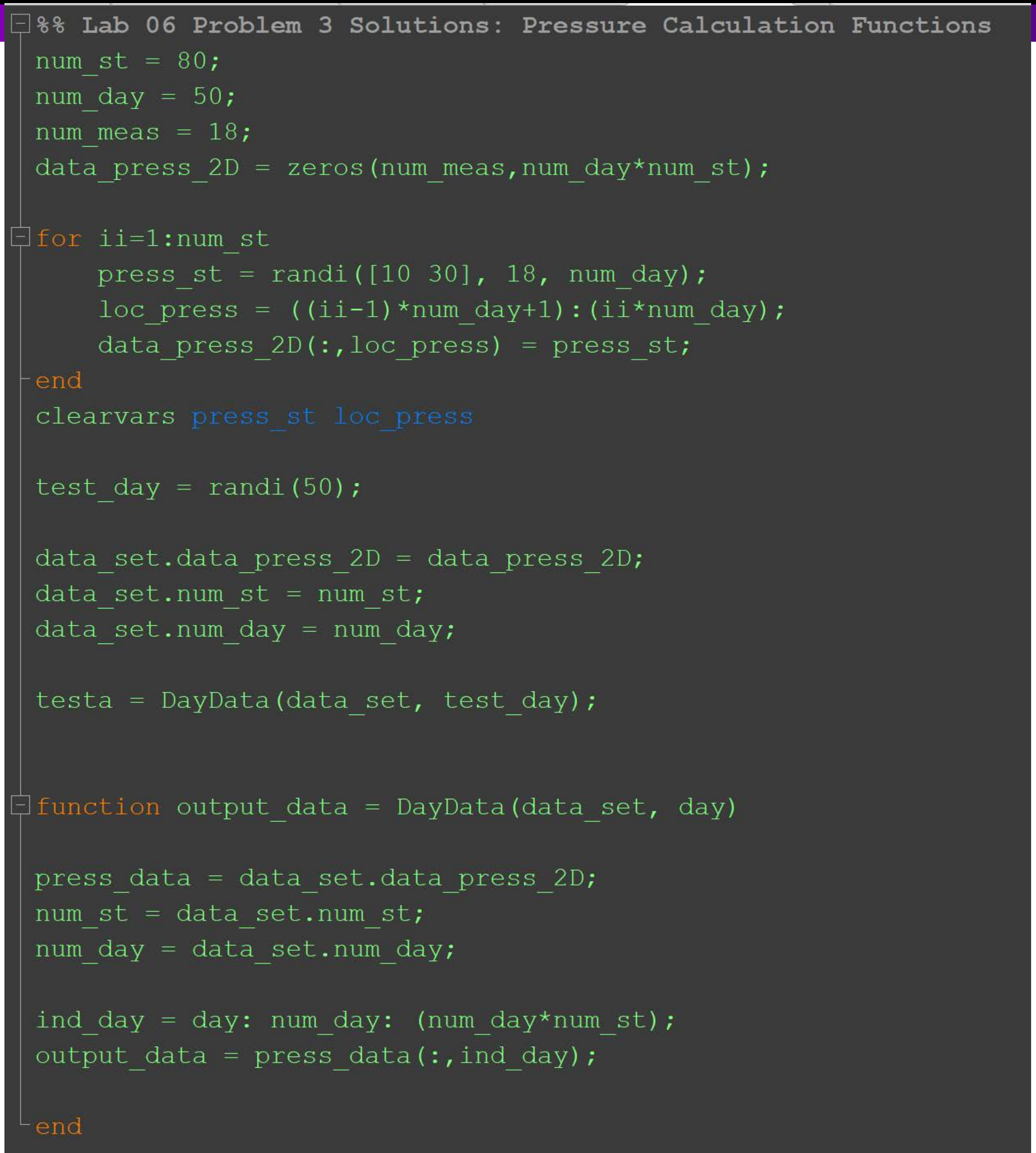
- Cell arrays are arrays, so they are indexed
  - That means that you can loop though the elements in a cell array – or have MATLAB do that for you by using vectorized code
- Structs are not indexed, so you cannot loop
  - However, the field names are mnemonic so it is more clear what is being stored in a struct
- For example:

  variable{1} vs. variable.weight: which is more mnemonic?

Example: Multiple Input Data in Functions



```
num_st = 80;
num_day = 50;
num_meas = 18;
data_press_2D = zeros(num_meas,num_day*num_st);

for ii=1:num_st
    press_st = randi([10 30], 18, num_day);
    loc_press = ((ii-1)*num_day+1):(ii*num_day);
    data_press_2D(:,loc_press) = press_st;
end
clearvars press_st loc_press

test_day = randi(50);

data_set = {data_press_2D, num_st, num_day};

testa = DayData(data_set, test_day);


function output_data = DayData(data_set, day)

press_data = data_set{1};
num_st = data_set{2};
num_day = data_set{3};

ind_day = day: num_day: (num_day*num_st);
output_data = press_data(:,ind_day);

end
```



```
%% Lab 06 Problem 3 Solutions: Pressure Calculation Functions
num_st = 80;
num_day = 50;
num_meas = 18;
data_press_2D = zeros(num_meas,num_day*num_st);

for ii=1:num_st
    press_st = randi([10 30], 18, num_day);
    loc_press = ((ii-1)*num_day+1):(ii*num_day);
    data_press_2D(:,loc_press) = press_st;
end
clearvars press_st loc_press

test_day = randi(50);

data_set.data_press_2D = data_press_2D;
data_set.num_st = num_st;
data_set.num_day = num_day;

testa = DayData(data_set, test_day);


function output_data = DayData(data_set, day)

press_data = data_set.data_press_2D;
num_st = data_set.num_st;
num_day = data_set.num_day;

ind_day = day: num_day: (num_day*num_st);
output_data = press_data(:,ind_day);

end
```

Slide Credits and References

- Stormy Attaway, 2018, Matlab: A Practical Introduction to Programming and Problem Solving, 5th edition
- Lecture slides for “Matlab: A Practical Introduction to Programming and Problem Solving”
- Holly Moore, 2018, MATLAB for Engineers, 5th edition