

Introduction to Matlab programming

Class 7: *Advance data types*

1. Define a variable *mycell* with a cell array that contains the following information

(1,1) 'My vector'	(1,2) [1, 2, 3, 4]
(2,1) 'I am a logical person'	(2,2) <i>false</i>
(3,1) 'Best days'	(3,2) { 'Sat' , 'Sun' }
(4,1) [1, 2, 3; 5, 6, 7]	(4,2) <i>pi</i>

- How is the data displayed when you call *mycell*?
 - Check the size of the cell array using the function *size*
 - Extract the cells (1,1), (2,1) and (3,1) into a new cell array called *mysubcell*. **Don't type the content again!** Do you do this operation using () or { }?
 - Transform the vertical cell array *mysubcell* into a horizontal cell array.
2. Create a function that accept a string as input argument and checks and displays a message depending on whether it is an additive primary color, a subtractive primary color or none of them. Inside the function define two cell arrays (vector) as

```
PrimaryAdd = {'red','green','blue'}
PrimarySub = {'cyan','magenta','yellow'}
```

- Develop the function using *if-else* statements
 - Develop the function using *switch-case* statements
 - How do you improve the previous functions in such a way that they accept any combination of lower or upper case letters in the input string.
3. Use the function *textscan* to read the file "Table2.txt".

Notes:

- First open the file externally (out of Matlab) to check how is it organized
- Remember that you have to use the functions *fopen* and *fclose* to open and close the file respectively. Never forget to close the file after using it! If this happens, then write *fclose('all')* in the command window.
- After reading the file, the current file position is moved to the next unread data or the end of the file. If for some reason you need to move the current file position to the beginning of the file, then use the function *frewind*.
- The file "Table2.txt" is a tabulated file where columns are separated by tabs.

Calculate the mean of each numeric column in the file

Bio-challenge 1

- a) Create a structure called *Cells* (don't call it *cell* because it is an internal Matlab instruction) with the following fields
- *birthtime* Time of the birth in minutes (from the beginning of the exp.)
 - *deathtime* Time of death, also in minutes
 - *alive* Logical variable that indicates if the variable is alive
 - *size* Size of cell in μm^2 .
 - *rep_rate* After how many minutes it divided itself (mitosis)
- b) Create a function that returns a structure array of *Cells*. The function must accept the following input arguments:
- *cellsize* Size of the cells in μm^2
 - *rep_rate* Reproduction rate
 - *chambersize* Size of the chamber, in μm^2 . You could specify this value also in mm^2 , but remember to do the necessary units conversions later.

For simplicity, we will consider at the moment that every cell has the same size and reproduction time, and that they don't die. Therefore, the returned *Cells* structure array should contain the same information for fields *size*, *rep_rate*, *alive* and *deathtime*. As the cells don't die, set the field *deathtime* to [].

Apply a loop to simulate the reproduction of this cell over time until the whole chamber is covered. The returned *Cells* structure array is a list of all the cells that are in the chamber at the end, whose appearance time is stored in the field *birthtime*.

- c) *To be continued...*