

Mixed bags

Working with strings and cell arrays



CS112 Scientific Computation

Department of Computer Science
Wellesley College

Strings

Program input/output

```
month = input('Enter a month: ');  
disp(['There are ' num2str(numDays(month)) ' days in that month']);
```

Text labels and graph properties

```
plot(xcoords, ycoords, 'g*', 'Linewidth', 2);  
title('golden ratio data');
```

Cell arrays of strings

```
names = {'Varitek' 'Ortiz' 'Ramirez' 'Lowell' 'Lugo' 'Youkilis' 'Crisp' 'Pedroia'};
```

String processing

```
>> test = ubbi('I am flying to America!')  
test =  
ubI ubam flubyubing tubo ubAmuberubicuba!
```



Cell arrays 2

What lies beneath - the ASCII code

The [ASCII code](#), established by the [American Society for Communication and Information Interchange](#), is a numerical code for representing letters, digits, punctuation, and control signals

The [original ASCII code](#) represents characters using a 7-bit code (numbers from 0 to 127)

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	NUL	32	20	Space	64	40	@	96	60	
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	Endtrans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	

Cell arrays 3

MATLAB character strings

Character strings in MATLAB are stored in special numerical vectors of ASCII values

The `double` and `char` functions translate between a character string and its ASCII values:

```
>> string1 = 'violet';
>> numcode = double(string1)
numcode =
    118    105    111    108    101    116
>> char(numcode)
ans =
violet
```

Cell arrays 4

String processing freebies

```
>> string1 = 'spring break';
```

```
>> length(string1)
```

```
ans =
```

```
>> string1(6)
```

```
ans =
```

```
>> string1(3:6)
```

```
ans =
```

```
>> string1([6 3 11 8])
```

```
ans =
```

```
>> string1(8:12) = 'fling'
```

```
ans =
```

```
>> string1(20) = '*';
```

string1

s	p	r	i	n	g		b	r	e	a	k
1	2	3	4	5	6	7	8	9	10	11	12



Cell arrays 5

Conditional expressions & strings

```
>> string1 == 'g'
```

```
ans =
```

```
000001000001
```

string1

s	p	r	i	n	g		f	l	i	n	g
1	2	3	4	5	6	7	8	9	10	11	12

ans

0	0	0	0	0	1	0	0	0	0	0	1
1	2	3	4	5	6	7	8	9	10	11	12

```
>> string1(string1 == 'f') = 'b'
```

```
string1 =
```

```
>> string1 == 'CS112'
```

```
* remember strcmp
```

Cell arrays 6

More fun with strings

```
>> string2 = 'to be or not to be';  
>> find(string2 == ' ')  
ans =
```

```
>> string2Letters = string2(string2 ~= ' ')  
string2Letters =
```

```
>> string2(string2 == ' ') = []  
string2 =
```



Cell arrays 7

String processing newbies

```
>> lower('To Be Or Not To Be')    converts letters to lower case  
ans =
```

```
>> upper(ans)                      converts letters to upper case  
ans =
```

```
>> strrep('to be or not to be', 'be', 'play')  
ans =                               replaces occurrences of 2nd  
                                     input string with 3rd input string,  
                                     in the 1st input string...
```

```
>> strfind(ans, 'ay')              finds all occurrences of second  
ans =                               input string in first input string...  
                                     returns indices of first character
```

Cell arrays 8

Time out exercises

What actions are performed by the following statements?

```
newString = '';  
for letter = string1  
    newString = [letter newString];  
end
```

string1

s	p	r	i	n	g	t	i	m	e
1	2	3	4	5	6	7	8	9	10

What test is performed by the following function?

```
function answer = test(str)  
    str = str(str ~= ' ');  
    str = lower(str);  
    answer = all(str == str(end:-1:1));  
  
    >> answer = test('Murder for a jar of red rum')
```



Maharaja, part god, sat in Pamplona and sees DNA, an ol' 'P' map, Nita's dog trap, a jar, a ham...

Cell arrays 9

Collecting strings with cell arrays

We have used a **cell array** to store a collection of strings

```
>> myPets = {'mona' 'cleo'};
```

We can access the contents of individual locations of a cell array using an index placed *inside curly braces*:

```
>> myPets{1}  
ans =  
mona  
  
for index = 1:length(myPets)  
    disp(myPets{index});  
end
```



Exercise: Write a function with a single input that is a cell array, which prompts the user for a string and determines whether the user's string is contained in the input cell array

Cell arrays 10

Collecting multiple types of data

The *real power* of cell arrays is that they allow us to store multiple types of data in one structure:

```
>> myCell = {'Ellen' 3.14159 [2 5 1 7] [1 2; 3 4]}
myCell =
    'Ellen' [3.14159] [1x4 double] [2x2 double]
```

```
>> celldisp(myCell)
```

```
myCell{1} =
```

```
Ellen
```

```
myCell{2} =
```

```
3.1416
```

```
myCell{3} =
```

```
2 5 1 7
```

```
myCell{4} =
```

```
1 2
```

```
3 4
```

Create a cell array from scratch with the `cell` function:

```
>> newCell = cell(1,3);
```

```
>> newCell{1} = 'Sohie';
```

```
>> newCell{2} = 'SCI E127';
```

```
>> newCell{3} = sohieImage;
```

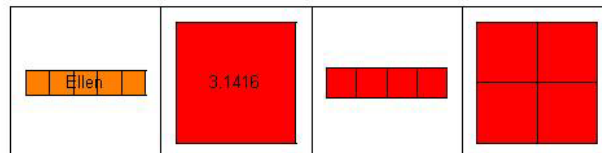
Cell arrays 11

A thousand words...

The `cellplot` function provides a high-level picture of a cell array:

```
>> myCell = {'Ellen' 3.14159 [2 5 1 7] [1 2; 3 4]}
myCell =
    'Ellen' [3.14159] [1x4 double] [2x2 double]
```

```
>> cellplot(myCell)
```



Cell arrays 12

Accessing the contents of cell arrays

Contents of individual locations of a cell array can be accessed with an index surrounded by curly braces:

```
>> myCell = {'Ellen' 3.14159 [2 5 1 7] [1 2; 3 4]};
>> disp([myCell{1} 's favorite number is ' num2str(myCell{2})])
ans =
Ellen's favorite number is 3.14159

>> myCell{3}(2)
ans =
5

>> sum(sum(myCell{4}))
ans =
10
```



Cell arrays 13

Into thin air...

```
mountains = {'Everest' 'K2' 'Kanchenjunga' 'Lhotse I' 'Makalu I' ...
'Lhotse II' 'Dhaulagiri' 'Manaslu I' 'Cho Oyu' ...
'Nanga Parbat' 'Annapurna'} ...
{'Himalayas' 'Karakoram' 'Himalayas' 'Himalayas'
'Himalayas' 'Himalayas' 'Himalayas' 'Himalayas' ...
'Himalayas' 'Himalayas' 'Himalayas'} ...
{'Nepal-China' 'Kashmir' 'Nepal-India' 'Nepal-China' ...
'Nepal-China' 'Nepal-China' 'Nepal' 'Nepal' 'Nepal-China' ...
'Kashmir' 'Nepal'} ...
[29028 28250 28208 27923 27824 27560 26810 ...
26760 26750 26660 26504];
mount = input('Enter the name of a mountain: ', 's');
```

Exercise:

Write a loop that prints all of the information about the user's input mountain



Cell arrays 14