String processing, File I/O

ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Matlab Strings: one character ~ one number

```
>> double('hello')
ans =
   104   101   108   108   111

>> %newline is not a new row in the matrix
>> char([104   101   108  10  108   111 ])
ans =
hel
lo

>> %two '' encodes a single ' in a string
>> msg='ahmet''s presentation'
msg = ahmet's presentation
```

Exercise

- Write a Matlab function str_rand(n) that returns a random alphanumeric (consisting of lower/upper case letters and digits) string of length n.

Concatenating Strings

```
>> %% Using vector notation:
>> name=['ahmet', ' ', 'sacan']
name =
ahmet sacan

>> %% Using strcat()
>> name=strcat('ahmet', ' ', 'sacan')
name =
ahmet sacan

>> %%Using sprintf()
>> name = sprintf( '%s%s%s', 'ahmet', ' ', 'sacan')
name =
ahmet sacan
```

Multiple strings

- As multiple rows of char arrays of the same length:

```
>> msgs=char('hello','yes','no')
msgs =
hello
yes
no
>> ['#' msgs(2, :) '#']
ans =
#yes  #
```

Multiple strings

- As cell arrays:

```
>> msgs={'hello','yes','no'}
msgs =
    'hello'    'yes'    'no'
>> ['#' msgs(2) '#']
ans =
    '#'    'yes'    '#'
>> ['#' msgs{2} '#']
ans =
#yes#
>> msgs={'hello','world';'yes','no'}
msgs =
    'hello'    'world'
    'yes'      'no'
```

String functions

```
>> lower('Hello 123')
hello 123

>> upper('Hello 123')
HELLO 123

>> ischar('Hello 123')
     1

>> isletter('Hello 123')
     1     1     1     1     1     0     0     0     0

>> isspace('Hello 123')
0     0     0     0     0     1     0     0     0
```

String functions: dealing with blanks

```
>> blanks(5)
ans =

>> [ '#' blanks(5) '#' ]
ans =
#     #

>> [ '#' deblank('   ahmet   sacan  ') '#' ]
ans =
#   ahmet sacan#

>> [ '#' strtrim('   ahmet   sacan  ') '#' ]
ans =
#ahmet sacan#
```

Comparing Strings

- strcmp('str1','str2')
- strcmpi('str1','Str2')
- strncmp('str1', 'str2', n)
- strncmpi('str1', 'Str2', n)

```
>> [strcmp('yes','yes'), strcmp('yes','no')]

>> strcmp('yes','YES')

>> strcmpi('yes','YES')

>> strcmp(lower('yes'),lower('YES'))

>> strcmp('yes','yes ')

>> strcmp('yes',deblank('yes '))
```

Exercise

- Write a function isprefix(s,pre) that returns true if pre is a prefix of s.

Comparing Strings

- strcmp({'str1','str2'}, 'str1')

- find(strcmp({'str1','str2'}, 'str2'))

Finding Strings

- k=strfind(haystack, needle)
- C=strfind(haystackcell, needle)

```
>> strfind('Find the starting indices of the pattern string', 'in')
2 15 19 45

>> cstr = {'How much wood would a woodchuck chuck', 'if a woodchuck could chuck wood?'};
>> idx = strfind(cstr, 'wood');
idx{1}=[10 23]
idx{2}=[6 28]
```

Exercise

- s='hello world';
- Replace all letters 'o' in s with 'x'. Use only two sets of parentheses and no auxiliary variables.
  - a. Using vector indexing.
  - b. Using strfind() function.

Replacing strings

- strrep(origstr, oldstr, newstr)

```
>> strrep('flow crow','ow','y')
```

Splitting strings

- strsplit(s,delimiter)
- strsplit(s,{delimiter1, delimiter2})
- strsplit(... ,'CollapseDelimiters',true)

Evaluating strings

- eval(str)

```
>> eval('2+2')

>> x=[2 6 8 3]
>> eval('plot(x)')

>> plotwhat='sin'
>> eval( ['plot(0:.1:10, ' plotwhat '(0:.1:10))'] )
```

Conversion from numbers

```
>> ['#' num2str(35.6) '#']
ans =
#35.6#
>> ['#' int2str(35.6) '#']
ans =
#36#
>> ['#' int2str(35) '#']
ans =
#35#
>> ['#' mat2str([1 2; 3 4.5]) '#']
ans =
#[1 2;3 4.5]#
```

%% sprintf()

Conversion to numbers

- [x, status] = str2num(str)
  - str2num is not supported by Progbank! Use str2double instead.

String Input	Numeric Output	Output Class
'500'	500	1-by-1 scalar double
'500 250 125 67'	500, 250, 125, 67	1-by-4 row vector of double
'500; 250; 125; 62.5'	500.0000 250.0000 125.0000 62.5000	4-by-1 column vector of double
'1 23 6 21; 53:56'	1 23 6 21 53 54 55 56	2-by-5 matrix of double
'12e-3 5.9e-3'	0.0120 0.0059	vector of double
'uint16(500)'	500	16–bit unsigned integer

Conversion to numbers (parsing)

- x = str2double(str)
- x = str2double(cellarray)

```
>> str2double( '123.45e7' )
>> str2double( '123 + 45i' )
>> str2double( '3.14159' )
>> str2double( '2.7i - 3.14' )
>> str2double( {'2.71' '3.1415'} )
>> str2double( '1,200.34' )

>> str2double('[3 4 5]')
>> str2num('[3 4 5]')
>> str2num('3 4 5')
```

Parsing strings: sscanf

```
>> s=sprintf('it is %f degrees outside.',63.5)
s =
it is 63.500000 degrees outside.

>> str2double( s(7:15) )
ans =
   63.5000

>> l=numel('it is '); str2double( s(l:l+8) )
ans =
   63.5000

>> sscanf(s, 'it is %f degrees outside.')
ans =
   63.5000
```

sscanf

- [A, count] = sscanf(str, format)

```
>> A=sscanf('grade=2.7 grade=3.1', 'grade=%f')
A =
   2.7
   3.1
>> % spaces are ignored, unless %c is used.
>> [A count]=sscanf('hello yes no','%s')
A = helloyesno
count = 3
>> [A count]=sscanf('hello yes no','%c')
A = hello yes no
count = 12
```

Parsing strings: textscan

- C = textscan(str|fid, 'format', N, 'param',value)

```
>> str = '0.41 8.24 3.57 6.24 9.27';
>> C = textscan(str, '%3.1f %*1d')
C = [5x1 double]
C{1} = [0.4; 8.2; 3.5; 6.2; 9.2]
```

textscan

```
>> s=['09/12/2005 Level1 12.34 45 1.23e10 inf Nan Yes 5.1+3i', sprintf('\n'), '10/12/2005 Level2 23.54 60 9e19 -inf 0.001 No 2.2-.5i', sprintf('\n'), '11/12/2005 Level3 34.90 12 2e5   10  100   No 3.1+.1i']
s =
09/12/2005 Level1 12.34 45 1.23e10 inf Nan Yes 5.1+3i
10/12/2005 Level2 23.54 60 9e19 -inf  0.001 No 2.2-.5i
11/12/2005 Level3 34.90 12 2e5   10  100   No 3.1+.1i

>> C = textscan(s, '%s %s %f %d %u %f %f %s %f')
C = {3x1 cell}  {3x1 cell}  [3x1 single]  [3x1 int8]  [3x1 uint32]  [3x1 double]  [3x1 double]  {3x1 cell}  [3x1 double]
```

>> C{1}	>> C{2}	>> C{3}	>> C{4}
'09/12/2005'	'Level1'	12.3400	45
'10/12/2005'	'Level2'	23.5400	60
'11/12/2005'	'Level3'	34.9000	12

textscan

```
09/12/2005 Level1 12.34 45 1.23e10 inf Nan Yes 5.1+3i
10/12/2005 Level2 23.54 60 9e19 -inf  0.001 No 2.2-.5i
11/12/2005 Level3 34.90 12 2e5   10  100   No 3.1+.1i
```

```
>> C = textscan(s, '%s %s %f %d %u %f %f %s %f');
C{2} = ['Level1'; 'Level2'; 'Level3']

>> C = textscan(s, '%s Level%u %f %d8 %u %f %f %s %f');
C{2} = [1; 2; 3]

>> C = textscan(s, '%s %*[^\n]')
C= {3x1 cell}
C{1} = ['09/12/2005'; '10/12/2005'; '11/12/2005']
```

Exercise

```
>> s=[
'ATOM      1  N   THR A   1      17.047  14.099   3.625  1.00 13.79           N  ' sprintf('\n') ...
'ATOM      2  CA  THR A   1      16.967  12.784   4.338  1.00 10.80           C  ' sprintf('\n') ...
'ATOM      3  C   THR A   1      15.685  12.755   5.133  1.00  9.19           C  ' sprintf('\n') ...
'ATOM      4  O   THR A   1      15.268  13.825   5.594  1.00  9.85           O  '];
```

```
>> % Parse atom coordinates
>> C=textscan(s,'ATOM %d %s %s %s %d %f %f %f %*[^\n]')
>> C{6}
   17.0470
   16.9670
   15.6850
   15.2680
>> C=textscan(s,'%*32c %f %f %f %*[^\n]')
C =
    [4x1 double]    [4x1 double]    [4x1 double]
>> cell2mat(C)
   17.0470   14.0990    3.6250
   16.9670   12.7840    4.3380
   15.6850   12.7550    5.1330
   15.2680   13.8250    5.5940
```