MATLAB

Strings

Strings

MATLAB has two different types of text strings – *character arrays* and *cell arrays*

•Main internal difference is how stored in memory

•User manipulates two types slightly differently

Character arrays - best when considering individual letters of text

Cell arrays - best when considering words

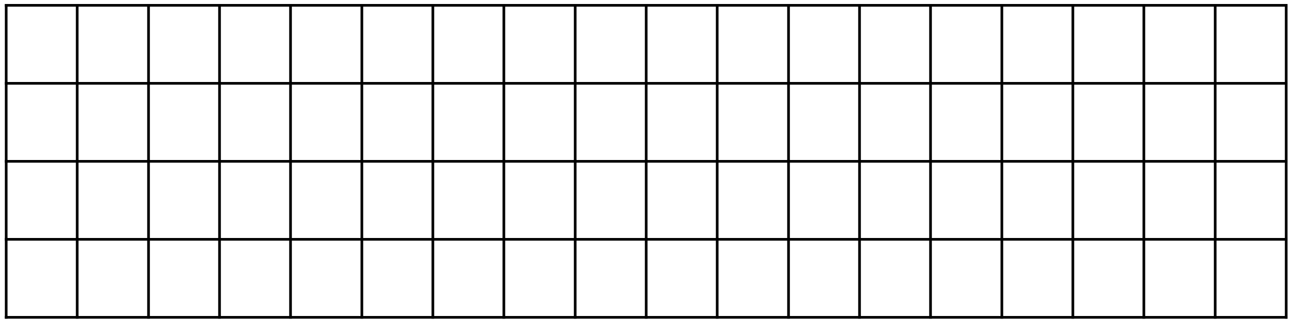
Character Arrays

Text stored in two-dimensional array

Key point – All rows must have same number of columns

- If not enough text in a row, row is *padded* on right with blanks, i.e., MATLAB adds enough space characters to end of text to make row correct length

Character Arrays



Four lines of text stored in a 4x18 array

MATLAB makes all rows as long as longest row

•First three rows above have enough space characters added on ends to make each row 18 characters long

Character Arrays

ASIDE

Each character actually occupies two bytes of memory because MATLAB accepts Unicode

- Unicode is common standard for working with non-English languages
- For more information, search for "Unicode" in MATLAB help system

Character Arrays

Pros

- Internally simple
- Can easily use with MATLAB functions that operate on arrays, e.g.,

```
>> num_a = sum(seuss(:) == 'a')
ans = 6
```

Character Arrays

To make a character-array variable with text in it, set variable equal to text in single quote marks:

```
>> s = 'Hello world'
s = Hello World
```

Warning to C/C++ programmers:

Use a single quote mark ('), not a double quote mark (")

Character Arrays

Multiple lines

```
>> subjects = [ 'math'; 'physics' ]
```

Gives error. Reason is two rows don't have same number of columns (letters)

There are two ways to fix problem

Character Arrays

Fix One – pad by hand

```
>> subjects=['math   ';...
             'physics' ]
               7 characters

subjects =
math
physics
```

Character Arrays

Fix Two – use `char()`

```
>> subjects=char('math','physics')

subjects = math
           physics


>> whos subjects

Name       Size Bytes Class Attributes

subjects   2x7  28    char
```

↑ "Attributes" column always empty for these slides so will omit from now on

Character Arrays

Often want to stick two text strings together

Example (pseudocode)

```
name = "edges"
if user wants JPEG output
    file = name + .JPG
else
    file = name + .TIF
```

Character Arrays

Sticking one text string to the end of another is called *concatenation* or *appending*

To concatenate character array constants and/or variables, put all between square brackets [], separating each by a space or comma

Character Arrays

Example

```
name = 'edges';
if userEntered == 1
    file = [ name '.jpg' ];
else
    file = [ name '.tif' ];
end
```

Character Arrays

Try It

Make variables with the names "Harold" and "Maude", then use concatenation to store "Harold and Maude" in the variable "film"

```
>> young = 'Harold';
>> old = 'Maude';
>> film = [ young ' and ' old ]
          film = 'Harold and Maude'
```

Comparing Character Arrays

`strcmp( s1, s2 )` returns 1 if the two strings (character arrays) are identical, returns 0 otherwise

- Strings may be different lengths
- Function is *case-sensitive*, i.e., letters must be in same case to be equal
  - For case insensitive comparison, use `strcmpi( s1, s2 )`

i = insensitive

Comparing Character Arrays

Try It

```
>> s1 = 'Matlab';
>> s2 = 'matlab'
>> strcmp(s1,s2)
ans = 0
>> strcmpi(s1,s2)
ans = 1
>> strcmp( s1(2:end), s2(2:end) )
ans = 1
```

Comparing Character Arrays

To get a character-by-character comparison use `==`

- Strings must be same length
- Comparison is *case-sensitive*
  - For case-insensitive comparison, use `upper()` or `lower()` (to be discussed soon) on both strings first
- Can use logical and relational operators to analyze text

Comparing Character Arrays

```
>> s1 = 'Matlab';
>> s2 = 'Maltab';
>> s1 == s2
ans = 1 1 0 0 1 1
% number of matching letters
>> sum( s1==s2 )
ans = 4
% index of first mismatch
>> find( s1~=s2, 1 )
ans = 3
```

Categorizing Characters

`isletter()` determines which characters in an array are letters. `isspace()` determines which are whitespace (blank, tab, newline)

```
>> bond = 'Agent 007';
>> isletter( bond )
ans = 1 1 1 1 1 0 0 0 0
>> isspace( bond )
ans = 0 0 0 0 0 1 0 0 0
```

Categorizing Characters

Often use `isletter`() or `isspace()` in conjunction with `any()` or `all()`

Example – get file name from user, but no spaces allowed (use MATLAB function `input()`)

```
>> name = input( 'File name: ', 's' );
>> if any( isspace( name ) )
disp( 'Illegal name - no spaces allowed' );
end
```

Categorizing Characters

Can check for lots of other types of characters by using `isstrprop( s, 'property' )`, e.g.,

- `'alpha'` – letter
- `'alphanum'` – letter or number
- `'punct'` – punctuation

```
>> isstrprop('www.muohio.edu', 'punct')
ans = 0 0 0 1 0 0 0 0 0 0 1 0 0 0
```

Type `help isstrprop` for all properties

Finding Characters

`findstr( s1, s2 )` returns vector of indexes where shorter string is in longer

Example

```
>> s1 = 'am';
>> s2 = 'Sam I am';
>> findstr( s1, s2 )
ans = 2 7
>> findstr( s2, s1 )
ans = 2 7
```

Modifying Characters

whitespace is any character for which `isspace()` returns true, i.e.,

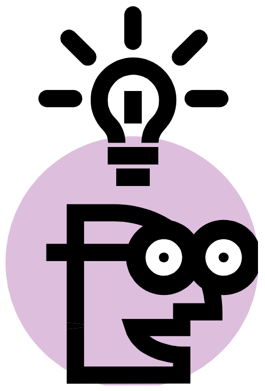
- spaces
- newlines
- carriage returns
- tabs
- vertical tabs
- form feeds

Modifying Characters

Common functions

- `s2 = deblank( s1 )` – returns string with trailing whitespace removed
- `s2 = strtrim( s1 )` – returns string with leading and trailing whitespace removed
- `s2 = lower( s1 )` – returns string with all letters in lower case
- `s2 = upper( s1 )` – returns string with all letters in upper case
- `s2 = strjust( s1 )` – returns string left, right, or center justified

Modifying Characters



Tip

When comparing strings make sure

- There is no leading or trailing space
- Both strings are all in the same case

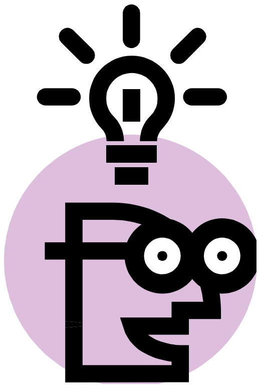
This is especially useful if text is entered by user or comes from a file

Modifying Characters

Tip Example

```
>> g1 =  'Green Eggs and Ham ';
>> g2  =  'Green eggs and Ham';
>> length(g1) == length(g2)
   ans = 0
>> g1=strtrim( g1 ), g2=strtrim( g2 );
>> length(g1) == length(g2)
   ans = 1
>> g1 == g2
    ans = 0
>> lower(g1)==lower(g2)
   ans = 1
```

Modifying Characters



Tip

Can use text in switch statements but make sure to trim and convert case first

Try It

Write image_type.m

Modifying Characters

image_type.m

```
function [] = image_type( extension )
switch extension
    case 'JPG'
        disp( 'JPEG file' );
    case 'TIF'
        disp( 'TIFF file' );
    otherwise
        disp( 'Unknown file' );
end
end
```

Modifying Characters

Try It – image_type.m

```
>> image_type( 'jpg' )
>> image_type( 'TIF ' )
>> image_type( 'TIFF' )
```

```
>> image_type( 'jpg' )
'Unknown file'
>> image_type( 'TIF ' )
'Unknown file'
>> image_type( 'TIFF' )
'Unknown file'
```

Modifying Characters

```
function[]= image_type( extension )
extension = upper(strtrim(extension));
switch extension
    case 'JPG'
        disp( 'JPEG file' );
    case 'TIF'
        disp( 'TIFF file' );
    otherwise
        disp( 'Unknown file' );
end
```

Modifying Characters

Try It

```
>> image_type( 'jpg' )
>> image_type( 'TIF ' )
>> image_type( 'TIFF' )
```

```
>> image_type( 'jpg' )
'JPEG file'
>> image_type( 'TIF ' )
'TIFF file'
>> image_type( 'TIFF' )
'Unknown file'
```

Replacing Characters

Use `strrep()` to find and replace characters in a string with other characters

`str = strrep( str1, str2, str3 )` finds and replaces all occurrences of the string `str2` in `str1` with the string `str3`

- `str2` and `str3` can be different lengths

```
>> s = 'Brown is excellent; Brown is expensive';
>> s2 = strrep( s, 'Brown', 'Dartmouth' )
s2 = Dartmouth is excellent; Dartmouth is expensive
```

Modifying Characters

Try It

In "Native of miami Valley" use string replacement to:

1 – Make first letter of last word lower case

```
>> s = 'Native of miami Valley';
>> s = strrep( s, 'V', 'v' )
s = Native of miami valley
```

2 – Capitalize the third word

```
>> s = strrep( s, 'mia', 'Mia' )
>> s = Native of Miami valley
```

Replacing Characters

For more on replacing characters, see

- `strtok()`
- `strmatch()`
- `textscan()`
- Regular expressions

Character Arrays

Questions?