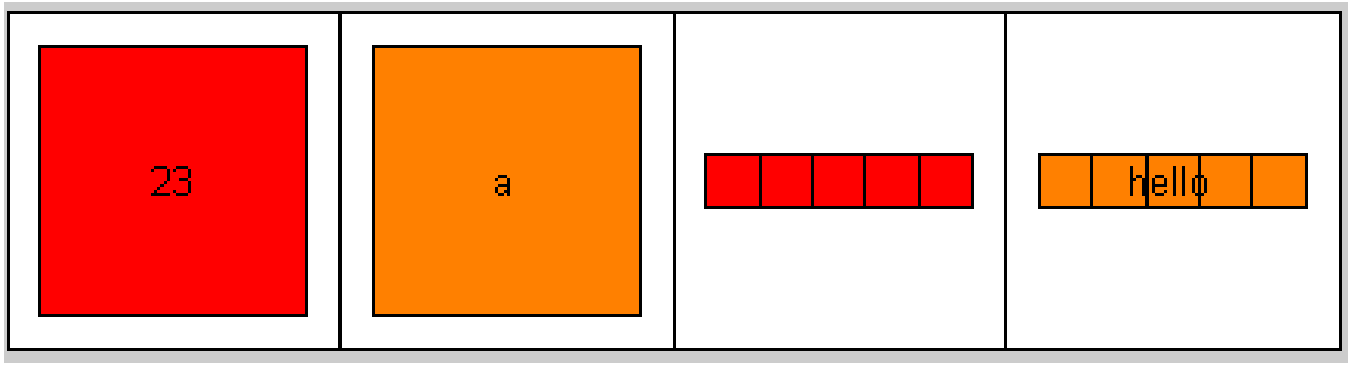
Cell Arrays

Data Structures

- Data structures: store more than one value, organize data
- Vector/Matrix: all values are of the same type.
- Cell array: values can be of different types.
- Structures: organize data into a group, with each data entry having a field name.

Creating cell arrays

```
>> cellrowvec = { 23, 'a', 1:2:9, 'hello' }
cellrowvec =
    [23]    'a'    [1x5 double]    'hello'
>> cellplot ( cellrowvec )
```



Creating cell arrays

```
>> cellrowvec = { 23, 'a', 1:2:9, 'hello' }
```

```
cellrowvec =
    [23]    'a'    [1x5 double]    'hello'
```

```
>> cellcolvec = { 23; 'a'; 1:2:9; 'hello' }
```

```
cellcolvec =
    [       23]
    'a'
    [1x5 double]
    'hello'
```

```
>> cellmat = { 23, 'a'; 1:2:9, 'hello' }
```

```
cellmat =
    [       23]    'a'
    [1x5 double]    'hello'
```

```
>> emptycellmat = cell(2,2)
```

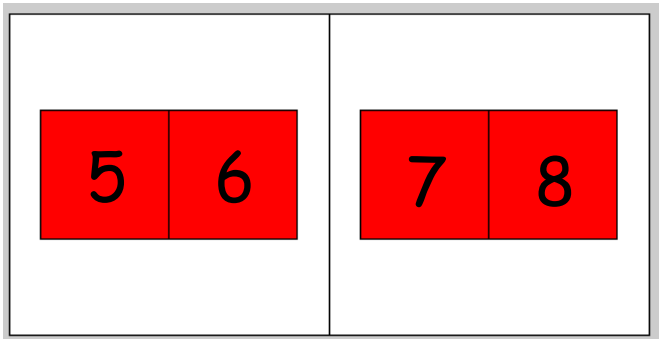
```
emptycellmat =
    []    []
    []    []
```

Indexing cell arrays

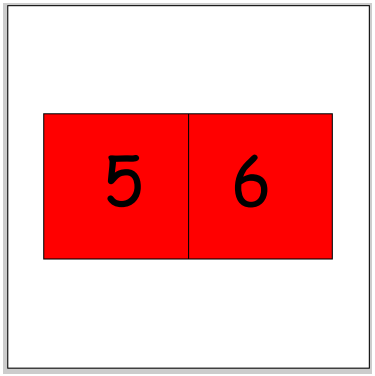
- Matrix indexing: ( )
  - Creates a new cell vector/matrix from the indexed entries.
- Content indexing: { }
  - Extracts the contents of the indexed entries.

Indexing cell arrays

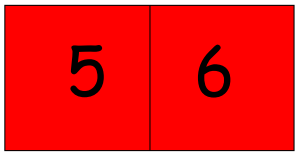
```
>> a = { 5:6, 7:8 }
a =
    [1x2 double]    [1x2 double]
```




```
>> a(1)
ans =
    [1x2 double]
```




```
>> a{1}
ans =
     5   6
```




```
>> iscell ( a(1) ),  iscell( a{1} )
```

Indexing cell arrays

```
>> m = { 'a', 5:6; 7:8, 'hello' }
>> m{1,2}
ans =
    5     6

>> m{1,1:2}
ans =
a
ans =
   5  6

>> [ x y ] = m{1,1:2}
x =
a
y =
   5  6
```

```
>> m{3}(2)


>> m(1:2)
ans =
    'a'    [1x2 double]

>> celldisp ( m(1:2) )
ans{1} =
a
ans{2} =
    7     8
```

Indexing cell arrays

```
>> m = { 'a', 5:6; 7:8, 'hello' }
>> m(end)
>> m(1,end)
>> m{end}
>> m{end}(2)
```

- To delete elements, must use matrix indexing.

```
>> m(1,:) = [ ]
```

Size of Cell arrays

- length/numel/size do not care about the "content" in cell elements.

```
>> m = { 'a', 1:100; 7:8, 'hello'; 9, [ ] }
>> size(m)
>> size( m(2) )
>> size( m{2} )
>> numel(m)
```

Cell of strings

- A set of strings can be stored in a char matrix:

```
>> s = 'mike'
>> s(2,1:5) = 'frank'
>> s(3,1:8) = 'mary ann'
```

- Easier to use cell of strings:

```
>> t = {'mike', 'frank'}
>> t{3} = 'mary ann'
```

Cell of strings

```
>> s = char('mike', 'frank', 'mary ann')
mike
frank
mary ann

>> t = cellstr ( s )
  'mike'
  'frank'
  'mary ann'

>> char ( t )
mike
frank
mary ann
```

```
>> iscellstr ( s )

>> iscellstr( t )
```

varargin, nargin

- varargin: a cell array of input arguments
- nargin: number of input arguments the function is called by

```
function [y]=smallest( x, varargin )
y =x
for i = 1:numel(varargin)
  if varargin{i} < y
    y=varargin{i};
  end
end
fprintf ( 'nargin=%d\n', nargin )
```

```
>> smallest( 5, 3 )
nargin=2
ans =
  3

>> smallest( 5, 3, 2, 10 )
nargin=4
ans =
  2
```

Optional input arguments: exist,varargin,nargin

```
function [s,m,d]=foo1(a,b)
s=a+b; m=a*b; d=a/b;
```

```
function [s,m,d]=foo2(a,b)
if ~exist('b','var'); b=3; end
if ~exist('a','var'); a=2; end
s=a+b; m=a*b; d=a/b;
```

```
function [s,m,d]=foo3(varargin)
if numel(varargin) < 1
        varargin{1}=2; end
if nargin < 2
        varargin{2}=3; end
a=varargin{1}; b=varargin{2};
s=a+b; m=a*b; d=a/b;
```

```
>> foo2(2)
ans = 5
>> foo2(2,4)
ans = 6
>> foo2
ans = 5
```

```
>> foo3(2)
ans = 5
>> foo3(2,4)
ans = 6
>> foo3
ans = 5
```

Optional output arguments: varargout, nargout

```
function [s,m,d]=foo1(a,b)
s=a+b; m=a*b; d=a/b;
```

```
function [s,varargout]=foo2(a,b)
s=a+b; m=a*b; d=a/b;
varargout{1}=m;
varargout{2}=d;
```

```
function [s,varargout]=foo3(a,b)
s=a+b;
if nargout>1;
        varargout{1}=a*b; end
if nargout>2;
        varargout{2}=a/b;
```

```
>> foo2(2,3)
ans = 5
>> [a,b,c]=foo2(2,3)
a = 5
b = 6
c = 0.6667
```

```
% foo3 outputs are the
same as foo2.
```