- Previous Lecture:
  - Structure
  - Structure arrays
- Today's Lecture:
  - Working with large data files
  - Built-in **sort** function
- Announcement:
  - P5 Part A posted, due 4/10 at 6pm
  - Part B TBA

```
function cellArray2file(CA, fname)
% CA is a cell array of strings.
% Create a .txt file with the name
% specified by the string fname.
% The i-th line in the file is CA{i}

fid= fopen([fname '.txt'], 'w');
for i= 1:length(CA)
    fprintf(fid, '%s\n', CA{i});
end
fclose(fid);
```



```
function CA = file2cellArray(fname)
% fname is a string that names a .txt file
%   in the current directory.
% CA is a cell array with CA{k} being the
%   k-th line in the file.

fid= fopen([fname '.txt'], 'r');
k= 0;
while ~feof(fid)
   k= k+1;
   CA{k}= fgetl(fid);
end
fclose(fid);
```


Where exactly are the xyz data?

```
1-4   14-15 18-20    33-38 41-46 49-54
```

Column nos. of interest

```
ATOM    14  N   HIS A  25      68.656  24.973  44.142  1.00128.26          N
ATOM    15  CA  HIS A  25      69.416  24.678  42.939  1.00128.26          C
ATOM    16  C   HIS A  25      68.843  23.458  42.227  1.00128.26          C
ATOM    17  O   HIS A  25      68.911  23.354  41.007  1.00128.26          O
ATOM    18  CB  HIS A  25      70.881  24.416  43.300  1.00154.92          C
ATOM    19  CG  HIS A  25      71.188  22.977  43.573  1.00154.92          C
ATOM    20  ND1 HIS A  25      71.886  22.184  42.689  1.00154.92          N
ATOM    21  CD2 HIS A  25      70.877  22.182  44.625  1.00154.92          C
ATOM    22  CE1 HIS A  25      71.993  20.963  43.183  1.00154.92          C
ATOM    23  NE2 HIS A  25      71.388  20.935  44.356  1.00154.92          N
ATOM    24  N   TRP A  26      68.271  22.546  43.005  1.00 87.09          N
ATOM    25  CA  TRP A  26      67.702  21.311  42.475  1.00 87.09          C
ATOM    26  C   TRP A  26      66.187  21.378  42.339  1.00 87.09          C
ATOM    27  O   TRP A  26      65.577  20.508  41.718  1.00 87.09          O
```

x y z

Just getting what you need from a data file

- Read past all the header information
- When you come to the lines of interest, collect the xyz data
  - Line starts with '**ATOM**'
  - Cols 18-19 is '**CA**'


```
fid = fopen('1bl8.dat']);
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(18:19),'CA')
         x = [x; str2double(s(33:38))];
         y = [y; str2double(s(41:46))];
         z = [z; str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Update the x, y, z arrays

`sprintf` returns a string

Suppose `x` is a length-n array. Then

```
s = sprint('%10.2f',x);
```

is eqiuvalent to

```
s = [];
for i=1:length(x)
   s = [s sprintf('%10.2f',x(i))];
end
```



```
function matrix2file(M,nbrFormat,fname)
% M is a 2D array of numbers
% Creates .dat file with name specified by the
%   string fname.
% The ith line in the file is M(i,:) displayed with
%   the format specified by the string nbrFormat

[nr,nc] = size(M);
fid = fopen([fname '.dat'],'w');
for i=1:nr
     str = sprintf(nbrFormat,M(i,:));
     fprintf(fid,'%s\n',str);
end
fclose(fid);
```


A detailed sort-a-file example

Create a new file

`statePopSm2Lg.txt`

that is structured the same as **`statePop.txt`** except that *the states are ordered from smallest to largest according to population.*

First, get the populations into an array

```
C = file2cellArray('StatePop');
n = length(C);
pop = zeros(n,1);
for i=1:n
  S = C{i};
  pop(i) = str2double(S(16:24));
end
```


Built-In function `sort`

Syntax: `[y,idx] = sort(x)`

x:	10	20	5	90	15
y:	5	10	15	20	90
idx:	3	1	5	2	4

`y(k) = x(idx(k))`

Sort from little to big

```
% C is cell array read from statePop.txt
% pop is vector of state pop (numbers)
[s,rank] = sort(pop);
Cnew = cell(n,1);
for i=1:length(C);
    ithSmallest = rank(i);
    Cnew{i} = C{ithSmallest};
end

cellArray2file(Cnew,'statePopSm2Lg')
```