TEXT MANIPULATION FUNCTIONS + NOTES

strtok()

ex. strtok(strvar, delimiter)

note that default delimiter is always a space

```
ex. example = 'Well hello'
   [f l] = strtok(example)
      f =
         'Well'
      l =
         ' hello'
```

note that l includes delimiter (space)

strcat()

ex. strcat(var1, var2)

ex. strcat('Well', 'hello')

```
ans =
    'Wellhello'
```

ex. strcat('Well', ' hello')

```
ans =
    'Well hello'
```

strrep()

ex. strrep(string, oldstring, newstring)

ex. strrep('hello', 'lo', 'p!')

```
ans =
    'help!'
```

→ strrep('hello', 'lo', 'lp me!')

```
ans =
    'help me!'
```

sprintf()

```
ex. var = sprintf( )
    input(var)
```

Functions need to know that work on strings but <u>not</u> char. vectors

+ strings() strjoin() strsplit() join()

Advantage of cell array over struct?
→ you can index into a cell array, good for looping or vectorizing code

Advantage of struct over cell array?
→ fieldnames are mnemonic!

IMPORTANT TEXT MANIPULATION EXAMPLE!

sports = ["baseball" "football" "track and field"]

```
>> sports(1:2)
   1 x 2 string array
   "baseball" "football"
```

*not sequential commands

```
>> sports{1:2}
   ans =
         "baseball"
   ans =
         "football"
```

```
>> sports(2) = [ ]
   1 x 2 string array
   "baseball" "track and field"
```

```
>> sports{2} = [ ]
   1 x 3 string array
   "baseball" <missing> "track and field"
```

```
>> "I love " + sports
   ["I love baseball" "I love football" etc...]
```

```
>> sports{2}(1:3)
   'foo'
```

(sports(2){1}(1:3) also works)

```
>> sports(3)
   "track and field"
```

↑ 1x1 string scalar