# ~~Playing~~ Working with Strings

**Program input/output**

```
month = input('Enter a month:  ');
disp(['There are  '  num2str(numDays(month))  '  days that month']);
```

**Text labels and graph properties**

```
plot(xcoords, ycoords, 'g:*',  'Linewidth',  2);
title('golden ratio data');
app.pictureLabel.Text = 'my cool picture';
```


**Cell arrays of strings**

```
names = {'Betts'  'Martinez'  'Benintendi'  ...  'Bradley'  'Leon'};
```

**Reading/writing image files**

```
im = imread('albright.jpg');
imwrite(im,  'myImage.jpg');
```

**Processing strings**

```
ubbified = ubbi('Stella')
Stubelluba
```

# What Lies Beneath – the ASCII Code

**ASCII code**, from the *American Society for Communication and Information Interchange*, is a numerical code to represent letters, digits, punctuation, and control signals

Original ASCII code represents characters using a 7-bit code (numbers 0 to 127)

new international standard – **Unicode** – represents thousands of characters for multiple languages


| ASCII | Char | ASCII | Char | ASCII | Char | ASCII | Char |
|---|---|---|---|---|---|---|---|
| 0 | ^@ | 32 | space | 64 | @ | 96 | ` |
| 1 | ^A | 33 | ! | 65 | A | 97 | a |
| 2 | ^B | 34 | " | 66 | B | 98 | b |
| 3 | ^C | 35 | # | 67 | C | 99 | c |
| 4 | ^D | 36 | $ | 68 | D | 100 | d |
| 5 | ^E | 37 | % | 69 | E | 101 | e |
| 6 | ^F | 38 | & | 70 | F | 102 | f |
| 7 | ^G | 39 | ' | 71 | G | 103 | g |
| 8 | ^H | 40 | ( | 72 | H | 104 | h |
| 9 | ^I | 41 | ) | 73 | I | 105 | i |
| 10 | ^J | 42 | * | 74 | J | 106 | j |
| 11 | ^K | 43 | + | 75 | K | 107 | k |
| 12 | ^L | 44 | , | 76 | L | 108 | l |
| 13 | ^M | 45 | - | 77 | M | 109 | m |
| 14 | ^N | 46 | . | 78 | N | 110 | n |
| 15 | ^O | 47 | / | 79 | O | 111 | o |
| 16 | ^P | 48 | 0 | 80 | P | 112 | p |
| 17 | ^Q | 49 | 1 | 81 | Q | 113 | q |
| 18 | ^R | 50 | 2 | 82 | R | 114 | r |
| 19 | ^S | 51 | 3 | 83 | S | 115 | s |
| 20 | ^T | 52 | 4 | 84 | T | 116 | t |
| 21 | ^U | 53 | 5 | 85 | U | 117 | u |
| 22 | ^V | 54 | 6 | 86 | V | 118 | v |
| 23 | ^W | 55 | 7 | 87 | W | 119 | w |
| 24 | ^X | 56 | 8 | 88 | X | 120 | x |
| 25 | ^Y | 57 | 9 | 89 | Y | 121 | y |
| 26 | ^Z | 58 | : | 90 | Z | 122 | z |
| 27 | ^[ | 59 | ; | 91 | [ | 123 | { |
| 28 | ^\ | 60 | < | 92 | \ | 124 | \| |
| 29 | ^] | 61 | = | 93 | ] | 125 | } |
| 30 | ^^ | 62 | > | 94 | ^ | 126 | ~ |
| 31 | ^_ | 63 | ? | 95 | _ | 127 | del |

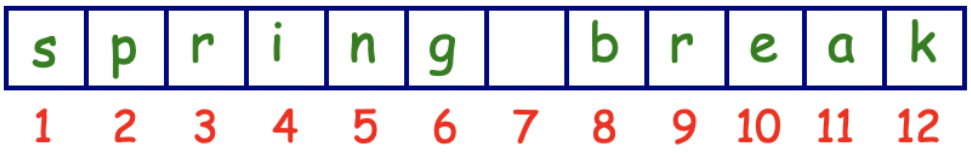
Character strings in MATLAB are stored in special numerical vectors of ASCII values

The **double** and **char** functions translate between a character string and its ASCII values

```
% create a sample string

myString = 'violet';

% convert to the numerical ASCII codes

numCode = double(myString)

% convert back to characters

myString = char(numCode)
```

# String Processing Freebies

**myString**




```
% create a sample string

myString = 'spring break';

length(myString)

% indexing strings

myString(6)

myString(3:6)

myString([6 3 11 8])

myString(8:12) = 'fling'

% MATLAB autoexpands a string when you add a character
% at a location that's beyond the current length

myString(20) = '*'
```

# Conditional Expressions & Strings

```
% create a sample string

myString = 'spring fling';

% compare a string to a single character with ==

myString == 'g'

% get the indices of locations where a character occurs

find(myString == 'g')

% use a logical vector as an index

myString(myString == 'f') = 'b'
```

```
% comparing strings with ==

'ellen' == 'silly'

% what if the strings have different length?

'ellen' == 'serious'

% what function do we need to user in this case?
```

```
% create a sample string

hamlet = 'to be or not to be';

% create a new string from the non-space characters

letters = hamlet(hamlet ~= ' ')

hamlet

% alter the string by removing the spaces

hamlet(hamlet == ' ') = [];

hamlet
```

# String Processing Newbies

```
% convert a string to all lower or all upper case letters
% with the lower and upper functions

shakespeare = 'To Be Or Not To Be'

lows = lower(shakespeare)

ups = upper(shakespeare)
```

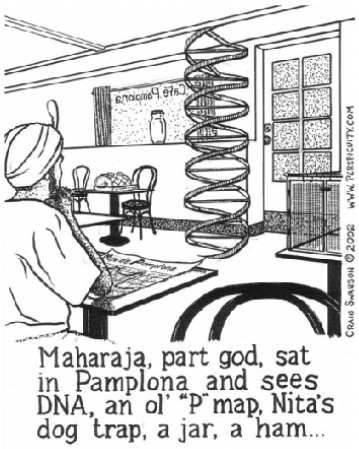
```
% strrep replace all occurrences of one substring with another

newString = strrep(shakespeare, 'Be', 'Play')

shakespeare

% find all the starting indices of a substring

strfind(newString, 'ay')
```

# Putting It All Together

Let's first write a function that tests whether an input string is a ***palindrome***. Here are some examples of palindromes:

Murder for a jar of red rum
Oozy rat in a sanitary zoo
Able was I ere I saw Elba*

***Apparently Napoleon didn't really say this**




We can do this in three steps:
(1) remove the spaces from the string
(2) convert all letters to lower case
(3) check if the string is the same, forwards and backwards

**isPalindrome** incorporates these three steps:

```
function answer = isPalindrome (str)
str = str(str ~= ' ');        % remove spaces from input string
str = lower(str);             % convert to lower case letters
% is the string the same, forwards and backwards?
answer = all(str == str(end:-1:1));
```

```
% let's try some examples

pal1 = isPalindrome('Murder for a jar of red rum')

pal2 = isPalindrome('Oozy rat in a sanitary zoo')

pal3 = isPalindrome('This is not a palindrome')

% note that the last statement in the isPalindrome function
% hints at a way to reverse a string

str = 'once upon a time'

reverseStr = str(end:-1:1)

% in fact, MATLAB provides a built-in function for this

reverseStr = reverse(str)
```

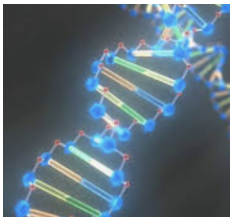
Let's create our own function, **reverseStr**, that returns the reverse of an input string. In the example below, the for loop assigns the variable letter to each character of the input string, progressing one-by-one from left to right, and places each letter to the left of the new string that is being created

```
function newString = reverseStr(str)
% returns a new string with the characters of the
% input string shown in reverse order

newString = '';
for letter = str
   newString = [letter newString];
end
```

Consider the execution of the for loop, given the input string 'april':

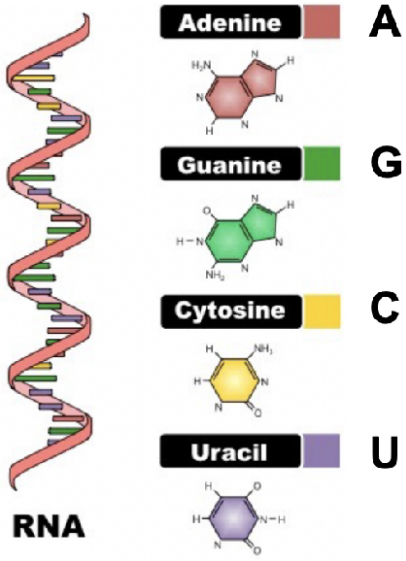
```
>> revStr = reverseStr('april')
```

| letter | newString |
|---|---|
| 'a' | 'a' |
| 'p' | 'pa' |
| 'r' | 'rpa' |
| 'i' | 'irpa' |
| 'l' | 'lirpa' |



## The Building Blocks of Life

A DNA molecule is a sequence of nucleotides. DNA is transcribed to RNA (a sequence of nucleotides), and RNA is then translated into a protein. RNA sequences are comprised of four nucleotides, abbreviated A, G, C, and U. Each set of three contiguous RNA nucleotides (called a *codon*) codes for a single amino acid. A protein is made of a chain of amino acids hooked together.

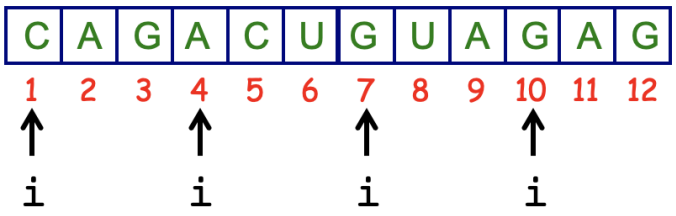



We'll write a function to translate an RNA sequence into amino acids

**RNA sequence:**
CAGACUGUAGAGAGAGCUCUA

**amino acids:**
Gln Thr Val Glu Arg Ala Leu

In the **rna2amino** function, we hide the details of converting a codon to an amino acid in the **translateCodon** function




```
function aminoAcids = rna2amino (nucleotides)
% translates a sequence of nucleotides into amino acids

% step through codons, translate each one to an amino
% acid, and add the amino acid to the aminoAcids string
aminoAcids = '';
for i = 1:3:length(nucleotides)-2
    aminoAcids = [aminoAcids ' ' ...
                  translateCodon(nucleotides(i:i+2))];
end
```

**codon:** AGU ⟶ **amino acid:** Serine

| first position | second position | | | | third position |
|---|---|---|---|---|---|
| | **U** | **C** | **A** | **G** | |
| **U** | Phe | Ser | Tyr | Cys | **U** |
| **U** | Phe | Ser | Tyr | Cys | **C** |
| **U** | Leu | Ser | Stop | Stop | **A** |
| **U** | Leu | Ser | Stop | Trp | **G** |
| **C** | Leu | Pro | His | Arg | **U** |
| **C** | Leu | Pro | His | Arg | **C** |
| **C** | Leu | Pro | Gln | Arg | **A** |
| **C** | Leu | Pro | Gln | Arg | **G** |
| **A** | Ile | Thr | Asn | Ser | **U** |
| **A** | Ile | Thr | Asn | Ser | **C** |
| **A** | Ile | Thr | Lys | Arg | **A** |
| **A** | Met | Thr | Lys | Arg | **G** |
| **G** | Val | Ala | Asp | Gly | **U** |
| **G** | Val | Ala | Asp | Gly | **C** |
| **G** | Val | Ala | Glu | Gly | **A** |
| **G** | Val | Ala | Glu | Gly | **G** |

U ⟶ 0
C ⟶ 1
A ⟶ 2
G ⟶ 3

```
function aminoAcid = translateCodon (codon)
% translates a single 3-letter codon into an amino acid

% create a cell array of amino acids, named aminoNames
createAminoTable

% convert input codon string to a numerical representation
% that allows the use of an index into the translation table
% to find the right amino acid
n1 = find(codon(1)=='UCAG')-1;
n2 = find(codon(2)=='UCAG')-1;
n3 = find(codon(3)=='UCAG')-1;

% calculate index and determine corresponding amino acid
index = 1 + n3 + 4*n1 + 16*n2;
aminoAcid = aminoNames{index};
```

```
% translate some RNA sequences to amino acids

amino1 = rna2amino('GCUCUCUGC')

amino2 = rna2amino('UAUCUAUCUAUCUAUCUAUCUAUC')

amino3 = rna2amino('AGAUGUAGAGCUACACUGAGAGUGAGU')
```