- A *string* is a group of characters surrounded by quotation marks, like "https://codecademy.com" or "To be or not to be."   
- A string is just a collection of a smaller data type, *char*, which is a single character like “a” or “?”.



**Escape Characters:**

- Used when quotes are needed within the text of a string, prevents the program from reading the quotation marks as the end of the string input  
A computer screen shot of text

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**New Line:**

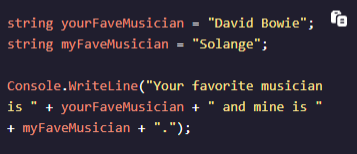
- Used to move text down to a new line using an escape character “\n”

A blue background with white text

Description automatically generated

**String Concatenation:**

- Combining strings using the “+” symbol  
- If concatenating a value other than a string, C# will automatically convert it to a string



**String Interpolation:**

- Allows us to insert variable values and expressions in the middle of string without needing to consider spaces and punctuation separately  
- Prepend $ before first quotation mark (no space inbetween) and variable names are encapsulated in {}

A computer screen shot of a program

Description automatically generated

**More Info About Strings:**

- Using built-in methods you can get more information about strings  
- A function inside a class is called a method. As C# is an [object-oriented](https://www.codecademy.com/resources/docs/general/programming-paradigms/object-oriented-programming) programming language, all functions are declared inside classes, making them methods. Thus, methods and functions are synonymous.

**Length:**

- Used to find how many characters a string is composed of by appending the *.Length* method to a variable name

A screen shot of a computer

Description automatically generated

**IndexOf:**

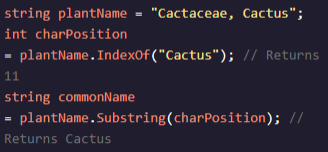
- Used to find the specific position of a search term in a larger string  
- Positions start at index 0  
- If a search term does not exist it will return -1, if we pass in an empty string it will return 0

A black background with white text

Description automatically generated

**Substring:**

- Grabs part of a string using the specified character position, continues until the end of the string, and then returns a new string.

  
- .Substring() is useful if we only want to use part of a string but keep the original data intact. So in this instance, we want to keep the string plantName, but just grab the "Cactus" portion of the string.  
- We use .IndexOf() to find where "Cactus" starts, then use .Substring() with the position information to save "Cactus" to the new variable commonName.

A computer screen shot of a code

Description automatically generated

- We can also pass .Substring() a second argument, which will determine the number of characters in the resulting substring. For example, the following code shows how we can use .Substring() with two arguments to specify the length of our substring:

**Bracket Notation:**

- Uses [ ] and an integer value to identify a particular value in a collection   
- Answers question later on “what the hell is at this position?”

A computer screen shot of a computer code

Description automatically generated  
- Similar to the example above, we first use .IndexOf() to grab the character position, which in this case is 15. We then take the string value and append it with a set of brackets [] and place the charPosition value inside the brackets.

A screenshot of a computer program

Description automatically generated

**ToUpper, ToLower:**

- Returns a new string containing the original strings in new case, either all uppercase or all lowercase

A computer screen shot of text

Description automatically generated