- Similar to arrays in that they are a sequential collection of values that can hold references to any type  
- Need to add the following header to the top of our code to use lists:



**Key Features:**  
- Have unlimited length  
- Automatically track the number of elements in the list  
- Have methods to work with multiple elements at a time  
- Arrays are faster to use when we have a pre-determined number of elements.  
- Lists are better when the number is unknown and they have extra features.

- When constructing a list, the C# compiler constructs an array and stores the elements there. If the list gets longer than the array’s length, the compiler copies the list’s elements to a new, longer array. To the developer, it looks like the list is infinitely long.

**Creating and Adding:**

- Created similarly to arrays in that the type has to be specified ahead of time and the new keyword is used to instantiate the list (basically you have to type it twice)

A computer code with white text

Description automatically generated with medium confidence

- More elements can be added to the list using the Add() method



-Elements can be accessed by using indices in square brackets



- Can also use the same notation to replace existing elements within a list



**Object Initialization:**

- Used to add new elements to a list all in one line  
- Used when creating a new list with values already known, additional values can still be added using .Add()

A black background with white text

Description automatically generated

**Add Range:**

- Used to add multiple elements at a time after list has already been instantiated   
A close up of a logo

Description automatically generated

**Count and Contains:**

**Count:**

- Used to find the number of elements in a list

A screen shot of a computer code

Description automatically generated

**Contains:**

- Used to check if an element exists within a list

A screen shot of a computer code

Description automatically generated

**Removing:**

- Used to remove a specific element from a list  
- Store return value in bool variable - “success is true” if successful and “success is false” if unsuccessful  
- Shifts all elements down one position after removal

A computer code with text

Description automatically generated

**Clear:**

- Used to remove all elements from a list at once

A computer screen shot of a program code

Description automatically generated

**Looping Through the List:**

- When using for loops, use .Count to stay within the bounds of the list  
- In foreach loops, counting is handled for us (preferred function)

A computer screen with numbers and symbols

Description automatically generated A screen shot of a computer

Description automatically generated

**Working with Ranges:**

- Used when wanting to add, access, or remove multiple elements at one time  
- A range is a subsequence of a list  


* AddRange() — takes an array or list as an argument. Adds the values to the end of the list. Returns nothing.

A computer code with text

Description automatically generated with medium confidence

* InsertRange() — takes an int and array or list as an argument. Adds the values at the int index. Returns nothing.

A screenshot of a computer code

Description automatically generated

* RemoveRange() — takes two int values. The first int is the index at which to begin removing, and the second int is the number of elements to remove. Returns nothing.

A screen shot of a computer

Description automatically generated

* GetRange() — takes two int values. The first int is the index of the first desired element, and the second int is the number of elements in the desired range. Returns a list of the same type.

A screen shot of a computer code

Description automatically generated A computer screen with text

Description automatically generated

**Generic Collections:**

- Generic collections are data structures that are defined with a generic type. Each class is defined generally without a specific type in mind. When we make an actual instance, we define the specific type.  
- The list class is in a group of **generic collections**. They don’t exist in the default set of System classes, so we need to reference them with this line - using System.Collections.Generic;

A computer code with white text

Description automatically generated with medium confidence A computer screen shot of a program

Description automatically generated