LINQ

LINQ stand for:

Language Integrated Query

LINQ

- **Query Language**

Query Definition

que·ry (kwîr′ē)

n. pl. **que·ries**

1. A question; an inquiry.

2. A doubt in the mind; a mental reservation.

3. A notation, usually a question mark, calling attention to an item in order to question its validity or accuracy.

tr.v. **que·ried**, **que·ry·ing**, **que·ries**

1. To express doubt or uncertainty about; question: *query someone's motives*.

2. To put a question to (a person). See Synonyms at **ask**.

3. To mark (an item) with a notation in order to question its validity or accuracy.

For Data

Linq is a feature in C# that provides a set of query operators and syntax to **query, manipulate, and transform data**.

What Linq can do

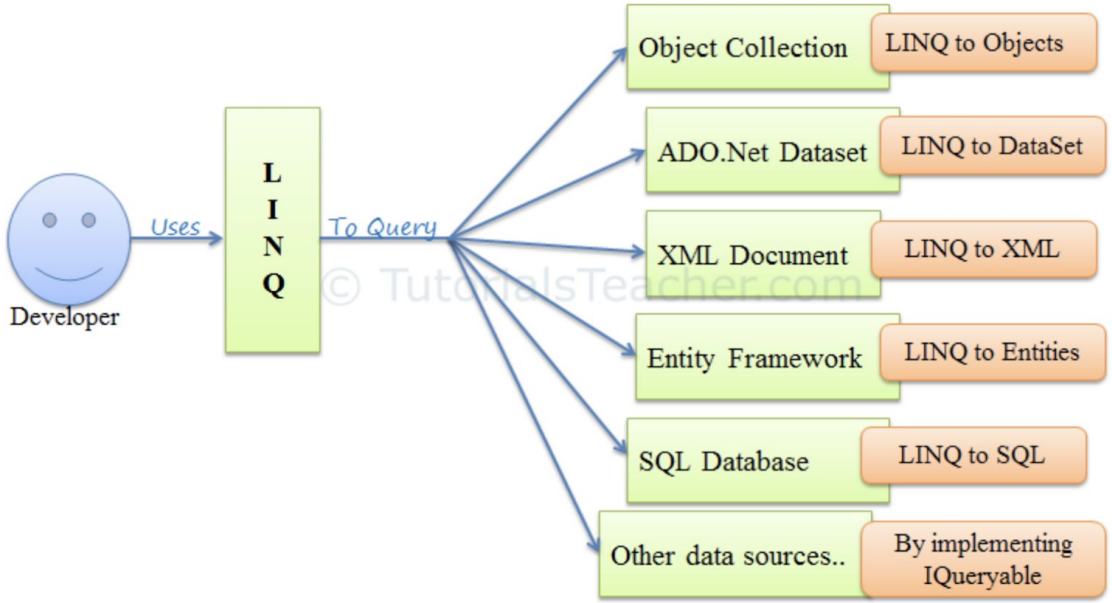
- Selecting data (projection)
- Filtering data
- Sorting data
- Joining data from multiple sources
- Grouping data
- Performing mathematical operations on data

Bottom Line

- **Used to Query Data**

LINQ for Data

- Notice that we use the generic term “data” and didn’t indicate what type of data.
- That’s because LINQ can be used to query many different types of data, including:
  - LINQ to Objects - what we will use!
  - LINQ to SQL - querying relational databases
  - LINQ to Entities (Entity Framework) - supports other database providers, not just SQL
  - LINQ to XML - allows querying XML documents
  - etc.



LINQ Usage

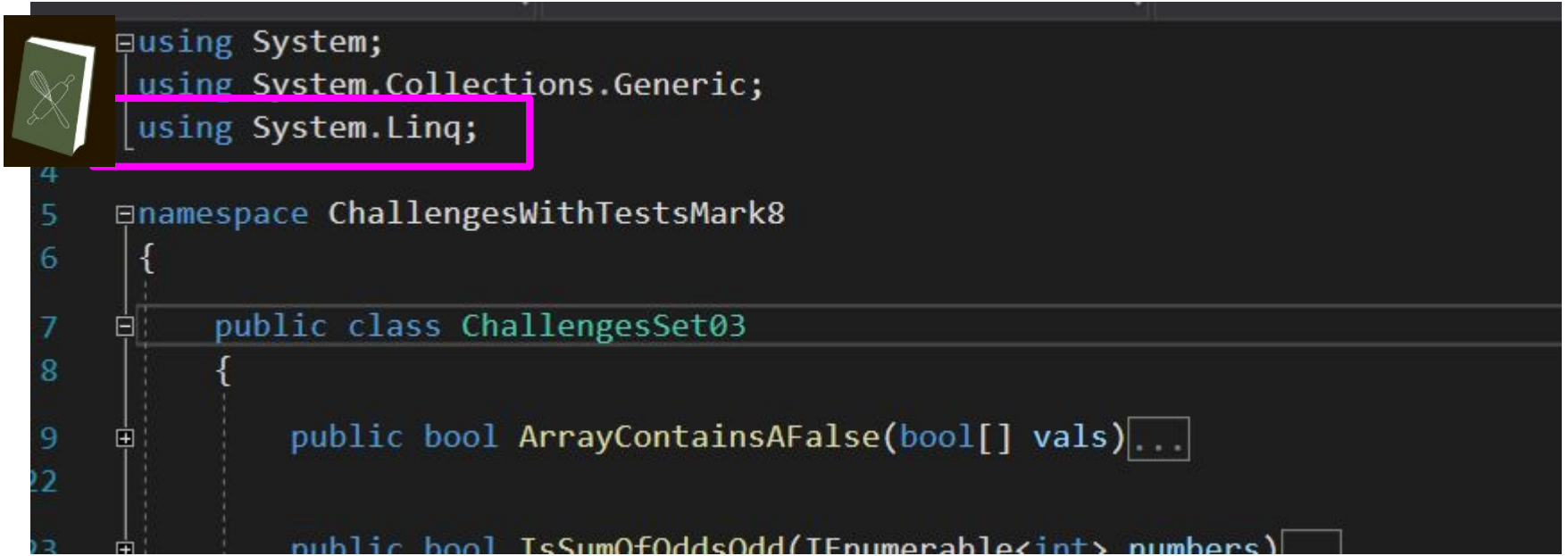
LINQ queries return results as objects. It enables you to uses object-oriented approach on the result set and not to worry about transforming diffent formats of results into objects.

Syntax

- Another way to describe LINQ: it is **programming language syntax that is used to query** data.

System.Linq

- Will need the using directive System.Linq at the top of the file
- You can then use built-in methods from that namespace


using System;
using System.Collections.Generic;
using System.Linq;
namespace ChallengesWithTestsMark8
{
public class ChallengesSet03
{
public bool ArrayContainsAFalse(bool[] vals)...

Reminder:

- Arrays and Lists conform to the IEnumerable interface.

```
int[] array1 = new int[5];
```

```
var names = new List<string> { "<name>", "Ana", "Felipe" };
```

Work with List\<T> - Introduction to C# tutorial | Microsoft Learn

System.Linq

- System.Linq extends the IEnumerable interface.
- We can utilize Linq to query anything that conforms to that interface in C#, such as **arrays and lists.**

```
int[] array1 = new int[5];
```

```
var names = new List<string> { "<name>", "Ana", "Felipe" };
```

What do we typically use with arrays and lists?

Loops!

LINQ

- Linq eliminates the need for loops

Before:

1 reference | 6/6 passing

```
public int Sum(int[] numbers)
{
    while (numbers != null)
    {
        int sum = 0;
        for (int i = 0; i < numbers.Length; i++)
        {
            sum += numbers[i];
        }
        return sum;
    }
    return 0;
}
```

After:

```
1 reference | 6/6 passing
public int Sum(int[] numbers)
{
    if (numbers == null)
    {
        return 0;
    }
    return numbers.Sum();
}
```

LINQ - Syntax

1. Query Syntax
2. Method Syntax

Query Syntax

```
using System.Linq;

namespace SpringClean
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] dogs = {"K 9", "Brian Griffin",
            "Scooby Doo", "Old Yeller", "Rin Tin Tin",
            "Rambo", "Lassie",
            "Snoopy"};

            // Get strings with spaces and put in
            // alphabetical order

            // from states where data comes from and
            // where to store each piece as you cycle

            // where defines conditions that must be met

            // orderby defines what data is used for
            // ordering results (ascending / descending)

            // select defines the variable that is
            // checked against the condition
            var dogSpaces = from dog in dogs
                            where dog.Contains(" ")
                            orderby dog ascending
                            select dog;

            foreach (var i in dogSpaces)
            {
                Console.WriteLine(i);
            }

            Console.WriteLine();

        }

    }
}
```

You might find similarities with this and SQL.

```
using System.Linq;

namespace SpringClean
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] dogs = {"K 9", "Brian Griffin",
            "Scooby Doo", "Old Yeller", "Rin Tin Tin",
            "Rambo", "Lassie",
            "Snoopy"};

            // Get strings with spaces and put in
            // alphabetical order

            // from states where data comes from and
            // where to store each piece as you cycle

            // where defines conditions that must be met

            // orderby defines what data is used for
            // ordering results (ascending / descending)

            // select defines the variable that is
            // checked against the condition
            var dogSpaces = from dog in dogs
                            where dog.Contains(" ")
                            orderby dog ascending
                            select dog;

            foreach (var i in dogSpaces)
            {
                Console.WriteLine(i);
            }

            Console.WriteLine();

        }

    }
}
```

Query Syntax Advantages:

Most people comfortable with SQL will prefer Query Syntax...

- Easy to read!
- Query syntax is automatically converted to method syntax at compile-time.

However: there are some methods that do not have a direct query syntax equivalent. For some operations, you'll need to switch to method syntax or combine both.

```
using System;
using System.Linq;

namespace SpringClean
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] dogs = {"K 9", "Brian Griffin",
            "Scooby Doo", "Old Yeller", "Rin Tin Tin",
            "Rambo", "Lassie",
            "Snoopy"};

            // Get strings with spaces and put in
            // alphabetical order

            // from states where data comes from and
            // where to store each piece as you cycle

            // where defines conditions that must be met

            // orderby defines what data is used for
            // ordering results (ascending / descending)

            // select defines the variable that is
            // checked against the condition
            var dogSpaces = from dog in dogs
                            where dog.Contains(" ")
                            orderby dog ascending
                            select dog;

            foreach (var i in dogSpaces)
            {
                Console.WriteLine(i);
            }

            Console.WriteLine();

        }
    }
}
```

Method Syntax

```
List<string> animalNames = new List<string>
{"fawn", "gibbon", "heron", "ibex", "jackalope"};

IEnumerable<string> longAnimalNames =
    animalNames.Where(name => name.Contains("o"));

foreach (var i in animalNames)
{
    Console.WriteLine(i);
}
```

Note: LINQ method syntax can do everything that query syntax can do. It's just a different way to format the instructions.

```
List<string> animalNames = new List<string>
{"fawn", "gibbon", "heron", "ibex", "jackalope"};

IEnumerable<string> longAnimalNames =
    animalNames.Where(name => name.Contains("o"));

foreach (var i in animalNames)
{
    Console.WriteLine(i);
}
```

Most people comfortable with C# prefer method syntax over query syntax...

- Method syntax is stylistically more similar to other C# code.

```
List<string> animalNames = new List<string>
{"fawn", "gibbon", "heron", "ibex", "jackalope"};

IEnumerable<string> longAnimalNames =
    animalNames.Where(name => name.Contains("o"));

foreach (var i in animalNames)
{
    Console.WriteLine(i);
}
```

Which is better?

- Method or Query Syntax?

Subjective!

- In practice, the choice between method and query syntax often comes down to personal preference, readability, and the specific operations you're performing.

Methods

Here is a brief list of the most common methods that are used in Linq:

- Where() - Filters a sequence of values based on a predicate.
- Select() - Projects each element of a sequence into a new form.
- OrderBy() - Sorts the elements of a sequence in ascending order.
- OrderByDescending() - Sorts the elements of a sequence in descending order.
- Sum() - Sums all elements
- Average() - Calculates average of all elements
- Count() - Returns the number of elements in a sequence.
- Min() - Finds min value in list/sequence
- Max() - Finds max value in list/sequence
- Take() - Returns a specified number of contiguous elements from the start of a sequence.
- Append() - Appends a value to the end of the sequence.
- ThenBy() - Performs a subsequent ordering of the elements in a sequence in ascending order.

Methods

Here is a brief list of the most common methods that are used in Linq:

- Where() - Filters a sequence of values based on a predicate.
- Select() - Projects each element of a sequence into a new form.
- OrderBy() - Sorts the elements of a sequence in ascending order.
- OrderByDescending() - Sorts the elements of a sequence in descending order.
- Sum() - Sums all elements
- Average() - Calculates average of all elements
- Count() - Returns the number of elements in a sequence.
- Min() - Finds min value in list/sequence
- Max() - Finds max value in list/sequence
- Take() - Returns a specified number of contiguous elements from the start of a sequence.
- Append() - Appends a value to the end of the sequence.
- ThenBy() - Performs a subsequent ordering of the elements in a sequence in ascending order.

More Methods to use in Exercise

- **ThenBy()** - used for subsequent orderings. Ex. order by first name, and then by last name (what if they have the same first name?)
- **SetValue(value, index)** - used to set a value at a specific position in an array
- **StartsWith()** - used to check whether the beginning of a string instance matches a specified string.
- **Append()** - used to add an element to the end of a sequence. It creates a new sequence that includes the original elements plus the new element.

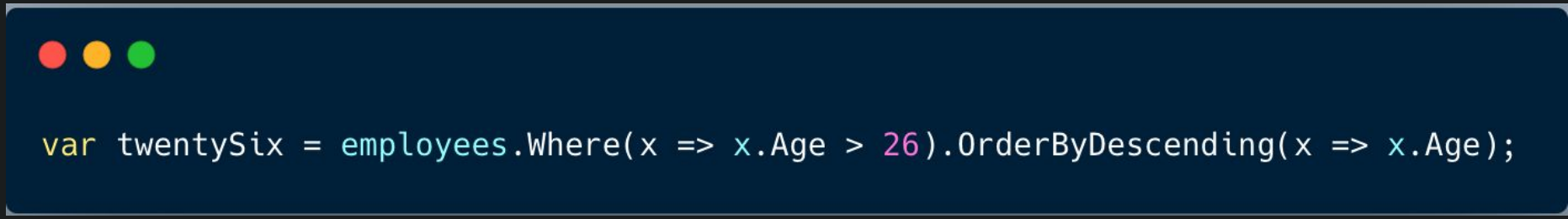
(That’s not the full list)

Microsoft Documentation for full list

Enumerable Class (System.Linq) | Microsoft Learn

Note: You can chain methods together!

For Example:



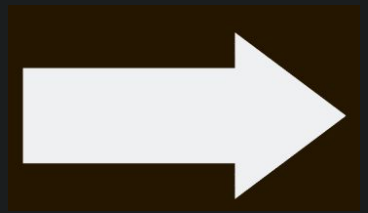
Lambda expression:

Lambda expressions in C# are used like anonymous functions... a method without a name.

The difference being that, in Lambda expressions, you don't need to specify the type of the value that you input, thus making it more flexible to use.

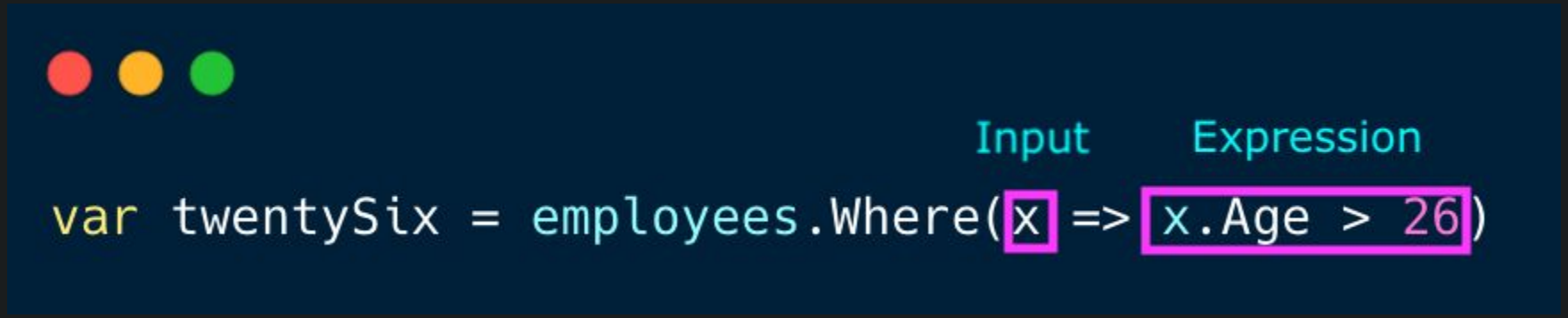
Note: The => is the lambda operator which is used in all lambda expressions.

```
var twentySix = employees.Where(x => x.Age > 26).OrderByDescending(x => x.Age);
```



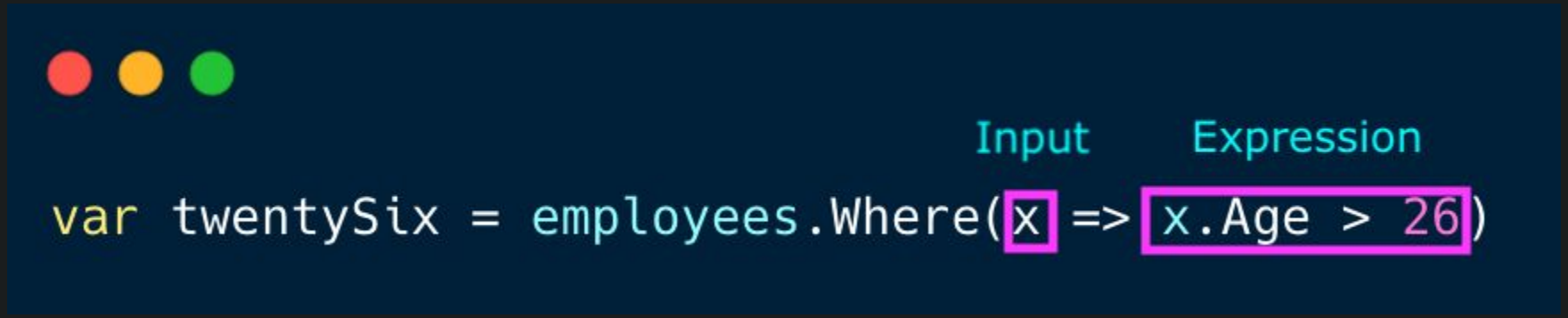
the lambda expression is used as an argument for the method call.

```
var twentySix = employees.Where(x => x.Age > 26).OrderByDescending(x => x.Age);
```



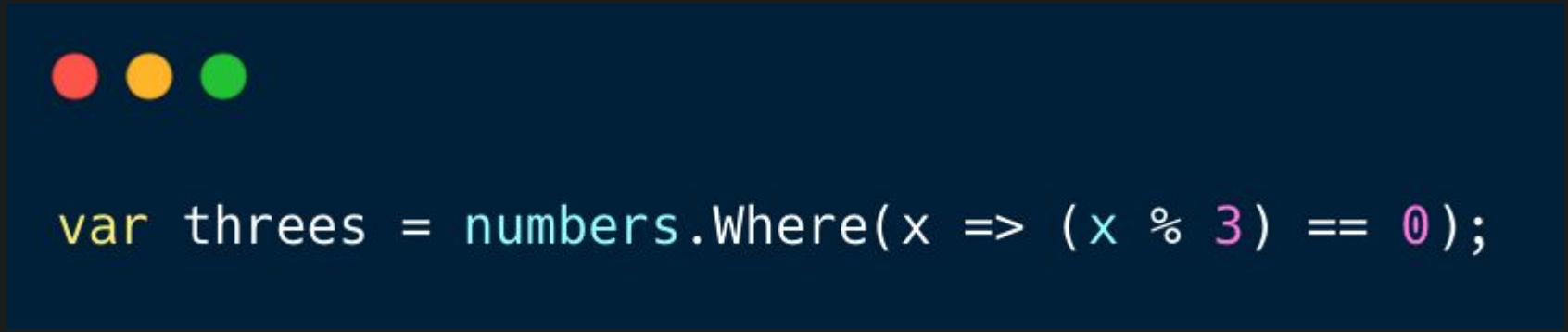
The Lambda expression is divided into two parts:

- the left side is the input
- the right is the expression



x is a temporary variable. You can name it anything you want (remember foreach statement)

```
foreach (var item in array)
{
    if (item % 2 == 0)
    {
        Console.WriteLine(item);
    }
}
```



- Filters! to find numbers divisible by 3

```
var threes = numbers.Where(x => (x % 3) == 0);
```

- Filters! to find numbers divisible by 3

Task:

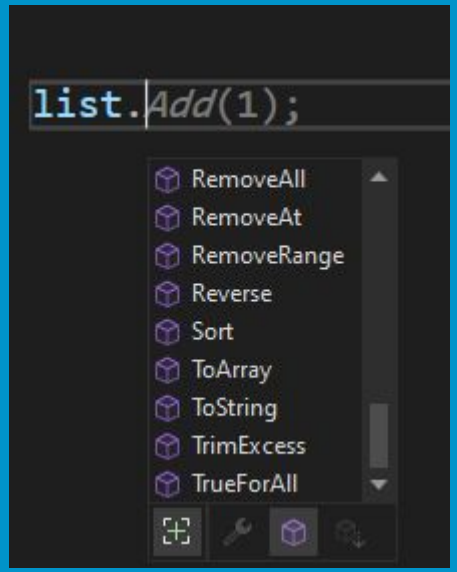
- Convert this to use LINQ (Method Syntax)

```
var array = new int[] {1, 24, 29, 30, 42, 302};
```

```
foreach (var item in array)
{
    if (item % 2 == 0)
    {
        Console.WriteLine(item);
    }
}
```

Not sure where to start?

- Name of the collection
- .



Name of the collection - dot

UTILIZE INTELLISENSE!

Using Linq

- (Method Syntax)

```
var array = new int[] {1, 24, 29, 30, 42, 302};
```

```
IEnumerable<int> myNewList = array.Where(x => x % 2 == 0);
```

Using Linq

- (Method Syntax)

```
var array = new int[] {1, 24, 29, 30, 42, 302};
```

```
IEnumerable<int> myNewList = array.Where(x => x % 2 == 0);
```

The returned value is a collection – so you will still need to iterate through the collection to see the items.

Where to start

Name of collection dot