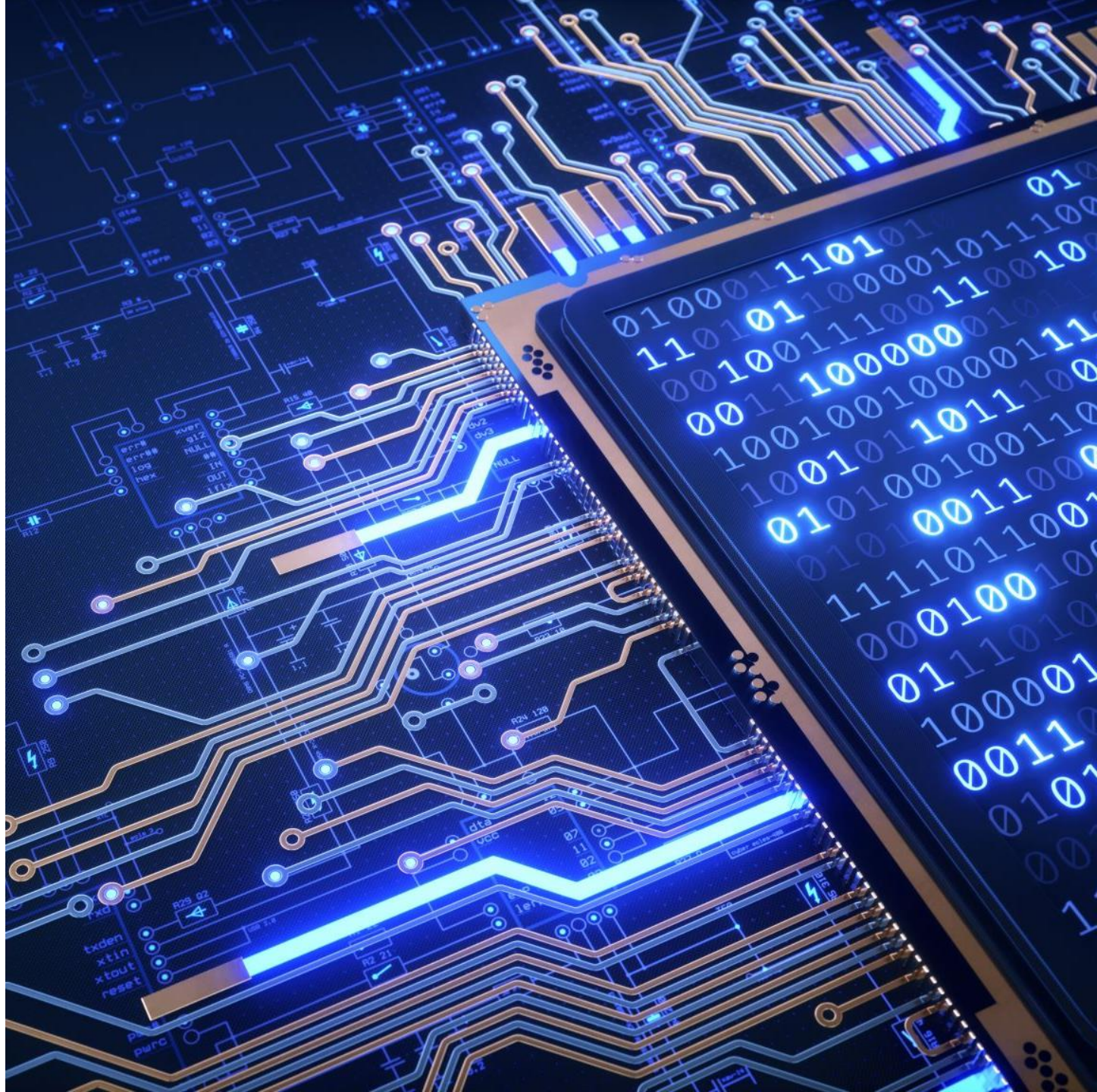
C#

Lambdas and LINQ

Rasmus Lystrøm
Associate Professor
ITU

Agenda

Delegates
Anonymous methods
Lambda expressions
Local functions
Anonymous types
Tuples
Extension methods
LINQ



Delegates

Delegates – Building block for Higher-order functions

```
public delegate int BinaryOperation(int x, int y);

static void Main(string[] args)
{
    var add = new BinaryOperation(
        delegate (int x, int y)
        {
            return x + y;
        }
    );
}
```

Delegates

Demo

Lambda Expressions

Lambda Expressions

```
Action<string> write = s => Console.WriteLine(s);

Func<int, int> square = a => a * a;

Predicate<City> b = c => c.Name.StartsWith("B");

Converter<double, double> ftoC = c => c * 9.0 / 5.0 + 32.0;
```

Local functions I

```
static void Main(string[] args)
{
    int square(int a) { return a * a; };

    Console.WriteLine(square(16));
}
```

Local functions II

```
public static int LocalFunctionFactorial(int n)
{
    return nthFactorial(n);

    int nthFactorial(int number) => number < 2
        ? 1
        : number * nthFactorial(number - 1);
}
```

Local functions III

```
public static int LambdaFactorial(int n)
{
    Func<int, int> nthFactorial = default(Func<int, int>);

    nthFactorial = number => number < 2
        ? 1
        : number * nthFactorial(number - 1);

    return nthFactorial(n);
}
```

Prerequisites

Anonymous types

```
var question = new
{
    Title = "The answer...?",
    Answer = 42
};
```

(Tuples)

```
var s = Tuple.Create("Clark Kent", "Superman");

var b = ("Bruce Wayne", "Batman");

var f = (name: "Barry Allen", alterEgo: "The Flash");

IEnumerable<(float x, float y)> GenerateCoordinates()
{
    yield return (1.3f, 23.45f);
}
```

Data: Collection Initializer

```
IEnumerable<City> cities = new[]
{
    new City(1, "Berlin"),
    new City(2, "Hamburg"),
    new City(3, "Frankfurt")
};
```

Data: Collection + Object Initializer

```
IEnumerable<City> cities = new[]
{
    new City { Id = 1, Name = "Berlin" },
    new City { Id = 2, Name = "Hamburg" },
    new City { Id = 3, Name = "Frankfurt" }
};
```

Extension Methods

Extension methods 1/2

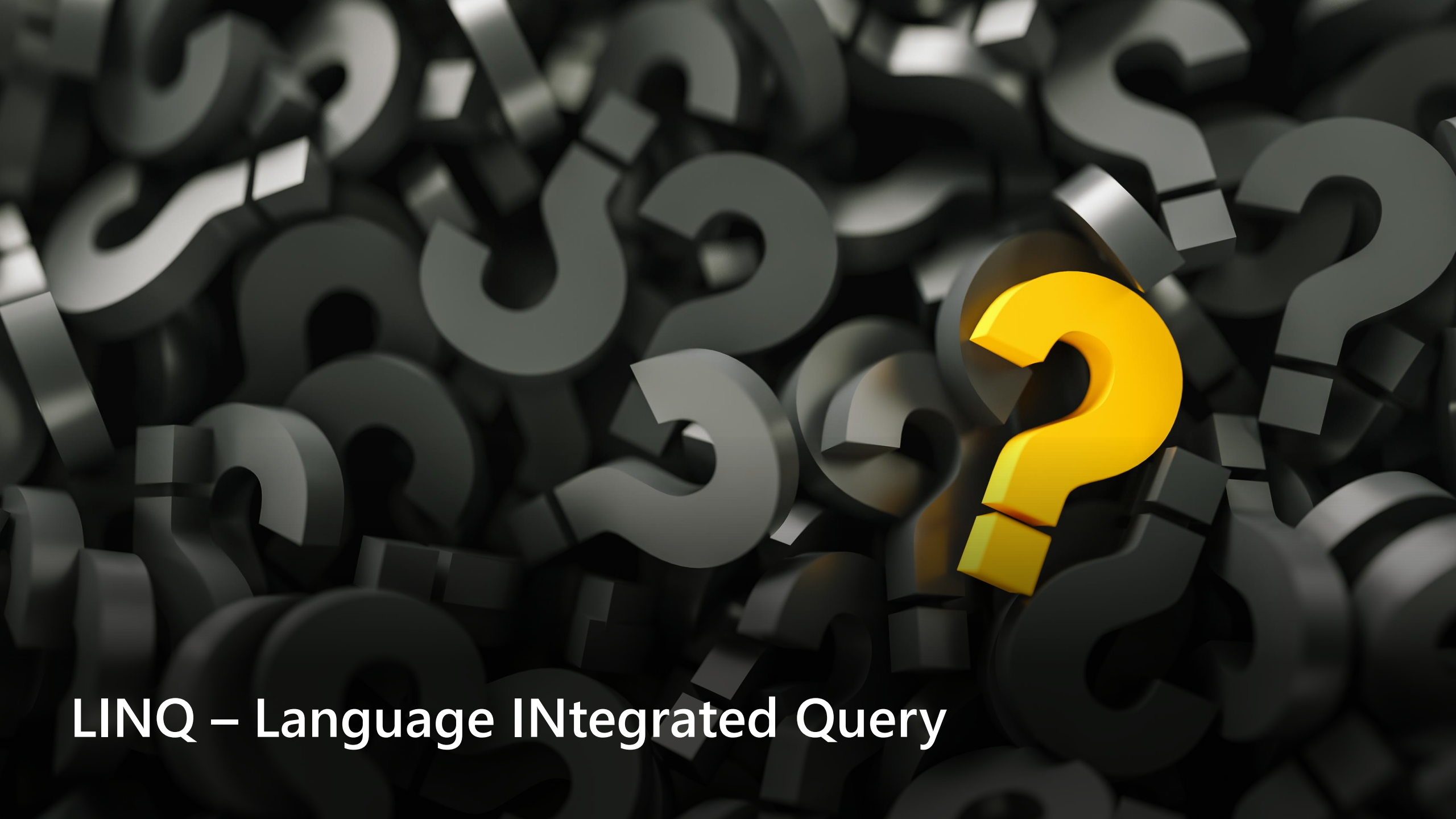
```
var count = cities.Count();

var sorted = cities.OrderBy(c => c.Name);

var filtered = cities.Where(c => c.Name.Contains("i"));

var pick = cities.FirstOrDefault(c => c.Id == 2);

var all = cities.All(c => c.Name.Length < 10);

var any = cities.Any(c => c.Name.StartsWith("B"));

var select = cities.Select(c => c.Name);
```

Extension methods 2/2

```
public static class Extensions
{
    public static int WordCount(this string str)
    {
        return str.Split(
            new char[] { ' ', '.', '?' },
            StringSplitOptions.RemoveEmptyEntries).Length;
    }
}
```

Extension Methods

Demo



LINQ – Language INtegrated Query

LINQ

```
var sorted = from c in cities
             where c.Name.Contains("i")
             orderby c.Name descending
             select new { Name = c.Name };
```

Extension Methods Version

```
var sorted = cities.Where(c => c.Name.Contains("i"))
                   .OrderByDescending(c => c.Name)
                   .Select(c => new { Name = c.Name });
```

LINQ

Demo


Thank you