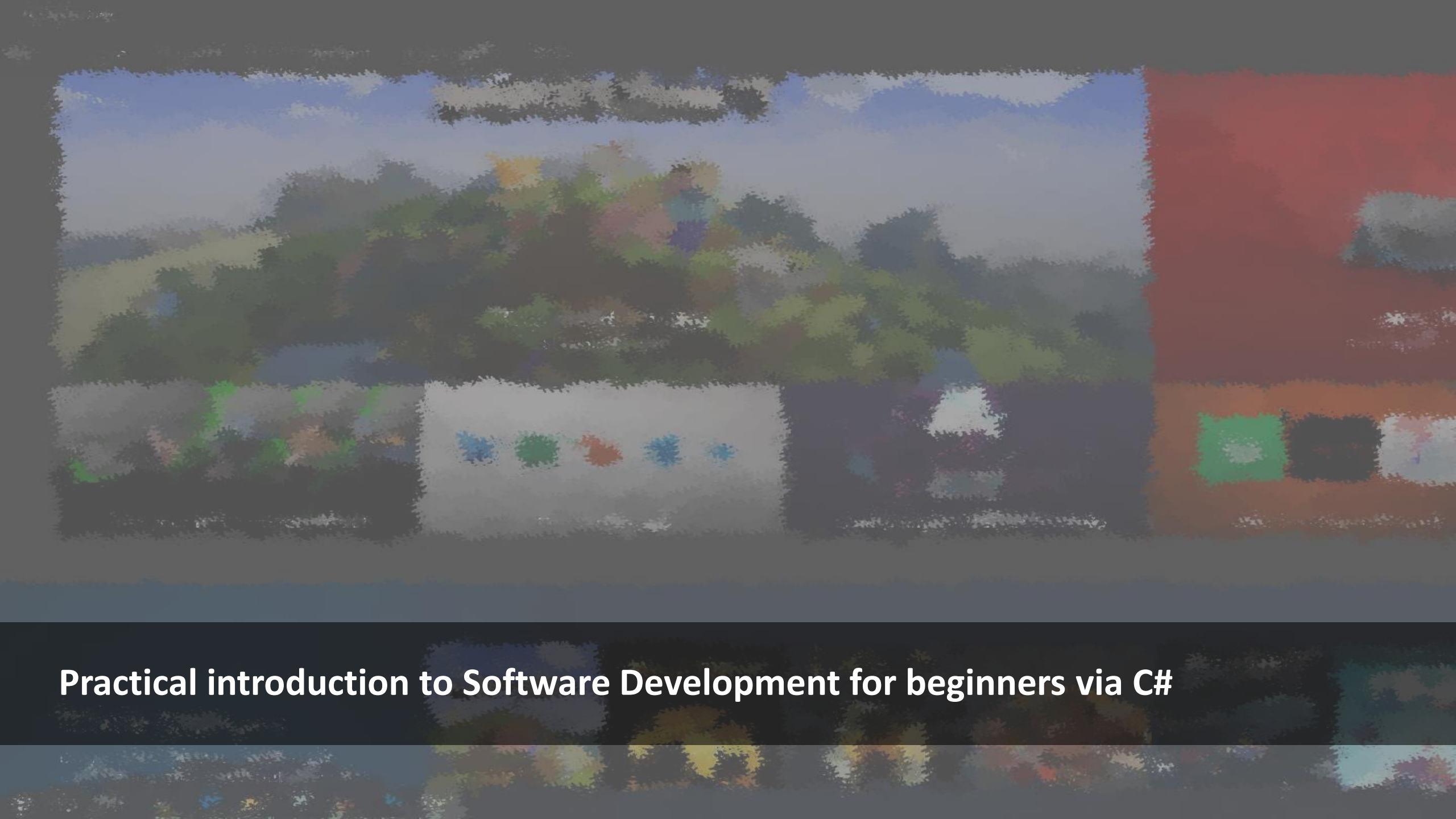
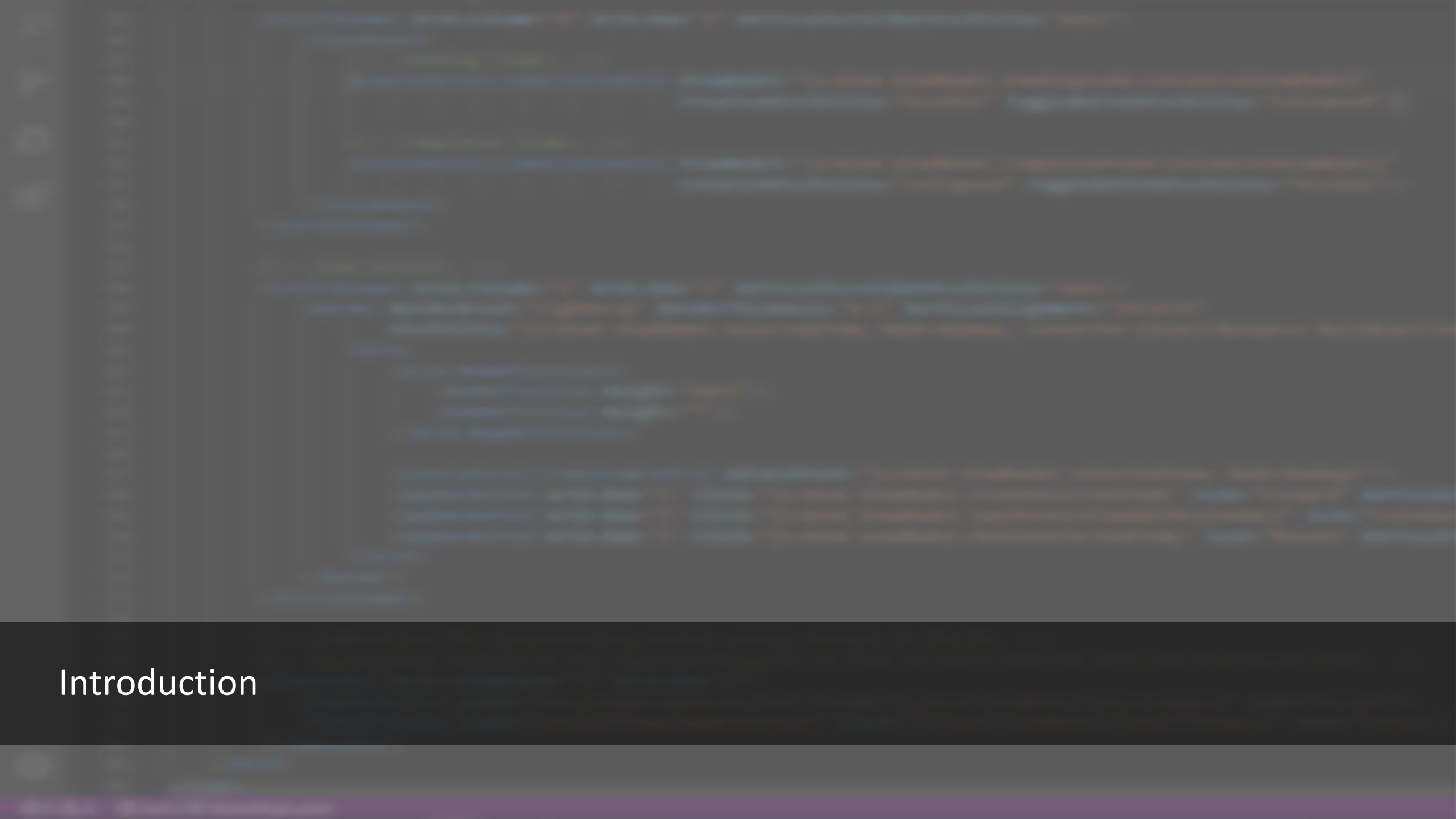


Practical introduction to Software Development for beginners via C#



Introduction

Practical introduction to Software Development for beginners via C#

Learn by doing

```
class BankAccount:
    def __init__(self):
        self._reset()

        self.lock = threading.Lock()


    def get_balance(self):
        self._guard_closed_account()

        return self.balance


    def open(self):
        if self.active:
            raise ValueError("Unsupported operation: Opened account")

        self.active = True


    def deposit(self, amount):
        with self.lock:
            self._guard_closed_account()
            self._guard_negative_amount(amount)

            self.balance += amount


    def withdraw(self, amount):
        with self.lock:
            self._guard_closed_account()
            self._guard_negative_amount(amount)

            if self.balance < amount:
                raise ValueError("Unsupported operation: Insufficient funds")

            self.balance -= amount


    def close(self):
        self._guard_closed_account()

        self._reset()
```

Focus of this course

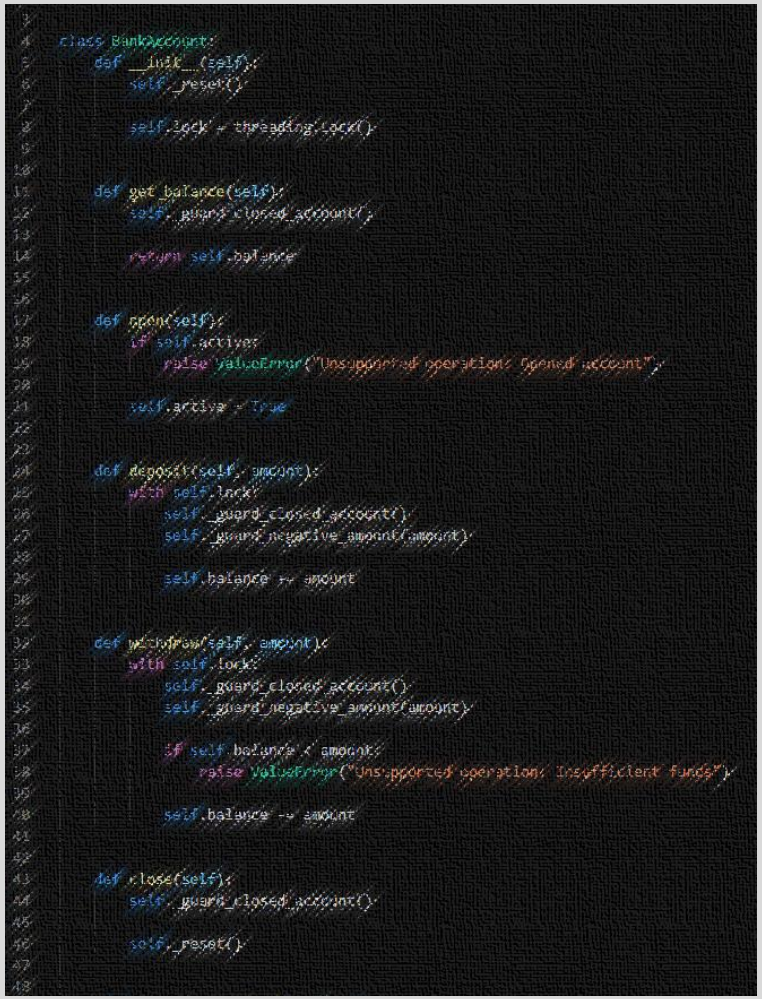
- Teach software development via C# by example
- This is **not** a software *engineering* course or a deep dive into the framework or language
- User research is advised on the subjects of the lesson

Nahuel Ianni

Software Engineer, Agile Coach, Project Manager

Practical introduction to Software Development for beginners via C#

Agenda



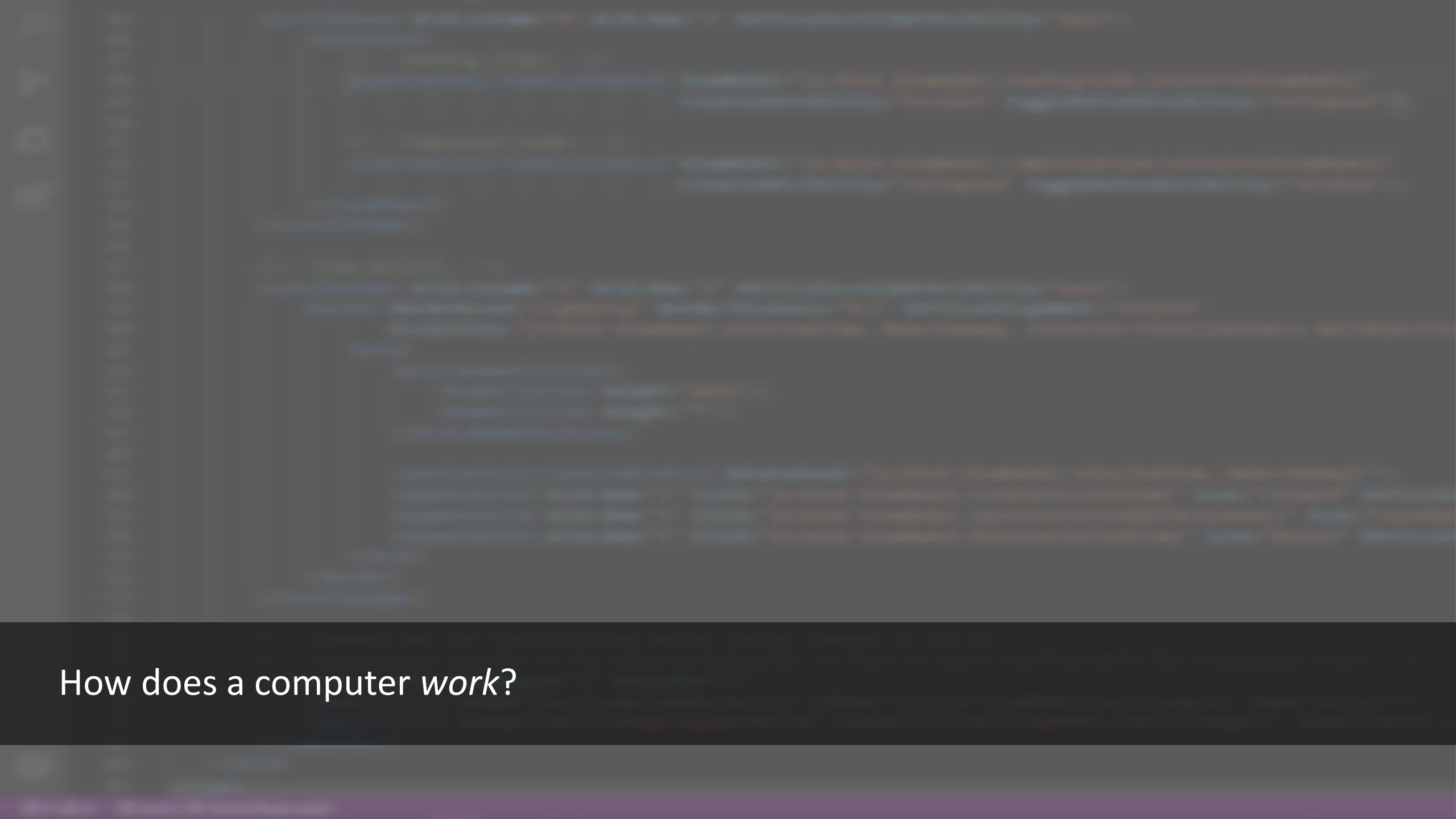
```
class BankAccount:
    def __init__(self):
        self._reset()

        self.lock = threading.Lock()


    def get_balance(self):
        self._guard_closed_account()

        return self.balance


    def open(self):
        if self.active:
            raise ValueError("Unsupported operation: Opened account")

        self.active = True


    def deposit(self, amount):
        with self.lock:
            self._guard_closed_account()
            self._guard_negative_amount(amount)

            self.balance += amount


    def withdraw(self, amount):
        with self.lock:
            self._guard_closed_account()
            self._guard_negative_amount(amount)

            if self.balance < amount:
                raise ValueError("Unsupported operation: Insufficient funds")

            self.balance -= amount


    def close(self):
        self._guard_closed_account()

        self._reset()
```

How does a computer work?

- Fundamental operations
- Example of how a computer works with a tasty analogy
- Code execution remarks

Introduction to C#

- Basics of the language structure and data
- Converting the analogy into code
- Exercise proposal


How does a computer *work*?

How does a computer *work*?

Fundamentals operations

At a fundamental level, a computer understands only of

- Data
- Commands

The basic functions a computer system can do is to

- **Input**

  Transfer data **into** the system, from a variety of sources (e.g. keyboard, speech dictation, LAN connection, etc.)

  The data is temporarily held in the RAM so that it can be processed

- **Processing**

  Process the data stored in the RAM into useful information for the user (e.g. during gaming, project the game onto the monitor while interpreting the user input via the controller)

- **Output**

  Present information to the user, in a variety of forms (e.g. by printing a document, showing a video on screen, playing music over speakers, etc.)

- **Storage**

  Save the information stored in memory in a proper, convenient way (e.g. print a document on paper, save a file on the cloud, etc.)

How does a computer *work*?

Analogy – Let's make a pizza

Ingredients

- List of ingredients needed

Cooking method

- Set of sequential steps detailing how to prepare the dish

How does a computer *work*?

Analogy – Let's make a pizza: Ingredients

Ingredients

- 1 package (1/4 ounce) active dry yeast
- 1 teaspoon sugar
- 1-1/4 cups warm water (110° to 115°)
- 1/4 cup canola oil
- 1 teaspoon salt
- 3-1/2 to 4 cups all-purpose flour
- 1/2 pound ground beef
- 1 small onion, chopped
- 1 can (15 ounces) tomato sauce
- 3 teaspoons dried oregano
- 1 teaspoon dried basil
- 1 medium green pepper, diced
- 2 cups shredded part-skim mozzarella cheese

How does a computer *work*?

Analogy – Let's make a pizza: Cooking method

Cooking method

- In a large bowl, dissolve yeast and sugar in water; let stand for 5 minutes. Add oil and salt. Stir in flour, a cup at a time, until a soft dough forms.
- Turn onto floured surface; knead until smooth and elastic, about 2-3 minutes. Place in a greased bowl, turning once to grease the top. Cover and let rise in a warm place until doubled, about 45 minutes. Meanwhile, cook beef and onion over medium heat until no longer pink; drain.
- Punch down dough; divide in half. Press each into a greased 12-in. pizza pan. Combine the tomato sauce, oregano and basil; spread over each crust. Top with beef mixture, green pepper and cheese.
- Bake at 400° for 25-30 minutes or until crust is lightly browned.

Pizza recipe link

How does a computer *work*?

Analogy – How does a person see the recipe?

Ingredients

1-1/4 cups warm water (110° to 115°)

- Name : Cups of warm water
- Quantity : 1-1/4
- Condition : 110° to 115° degrees Celsius

Cooking method

In a large bowl, dissolve yeast and sugar in water; let stand for 5 minutes. Add oil and salt.

- Precondition : Have a large bowl ready to use
- Action : Dissolve yeast and sugar in water
- Recurrence : Let it stand for 5 minutes
- Action : Add oil and salt

How does a computer *work*?

Analogy – How does the computer see the recipe?

Ingredients - List of data

1-1/4 cups warm water (110° to 115°)

- Object : **Water**
- Quantity : 1-1/4 cups
- Conditional temperature : 110° to 115° degrees Celsius

Cooking method - Sequence of commands

In a large bowl, dissolve yeast and sugar in water; let stand for 5 minutes.
Add oil and salt.

- Method : Prepare dough
- Command : Dissolve yeast and sugar in water
- Loop : For the next 5 minutes
- Command : Let it stand
- Command : Add oil and salt

How does a computer *work*?

Code execution remarks

A computer can't make deductions or guess the meaning a developer had when writing code

- All instructions and parameters need to be clearly defined (e.g. ingredients and cooking steps)
- Any value or step missing or wrongly defined can produce an error in the execution of the software

A computer executes code with perfect precision

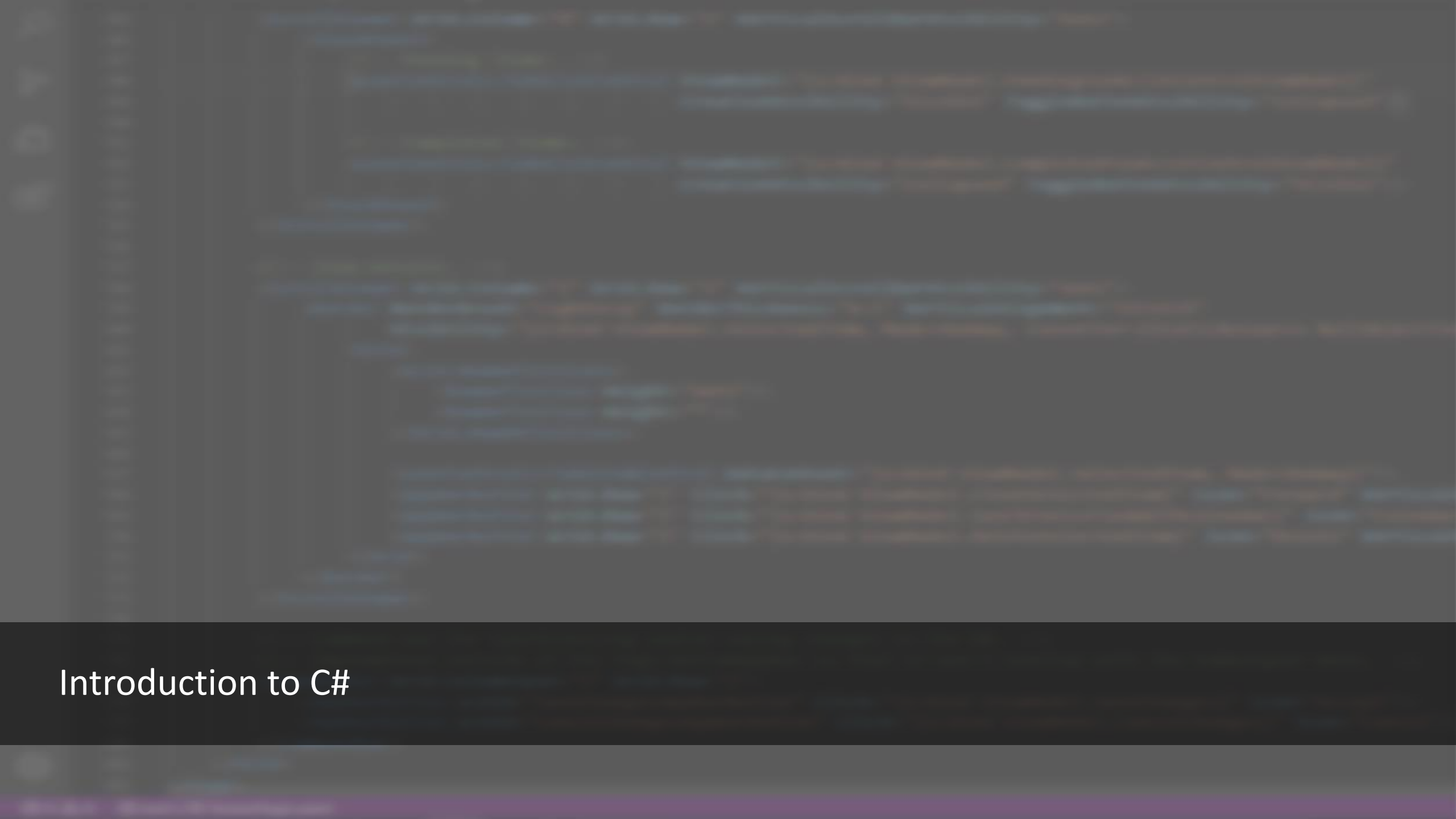
- Every command will get executed as stated
- The computer never fails at executing the specified action or command

Errors in software are the result of human mistakes

- Missing specifications, wrong calculations, hidden bugs; can all *crash the application*

No program can be free of bugs, other than small, trivial applications

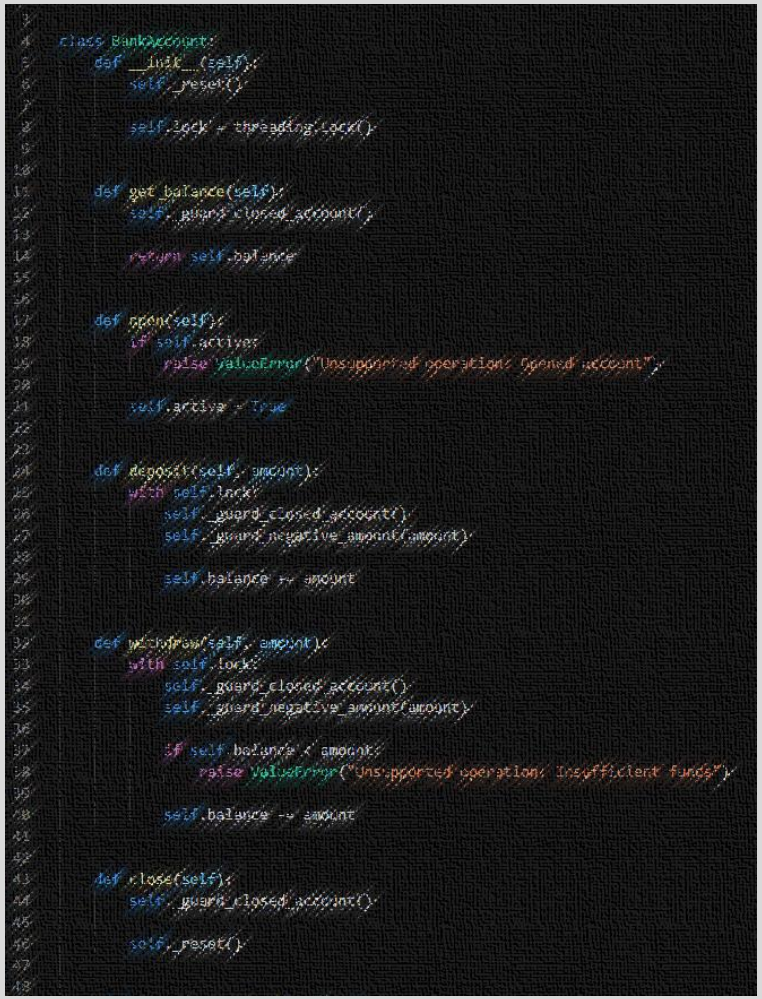
- In order to fully test an application, all relevant use cases must be tested, using real values in devices that emulate that of an end user
- This entails an amount of possible scenarios that would take an extremely long time to cover, and could potentially end up being useful for only a small subset of users in the end
- Advise: Testing is fundamental, do so considering the value of the scenario you are testing in regards to the impact an error might have in it



Introduction to C#

Introduction to C#

A few basics concepts



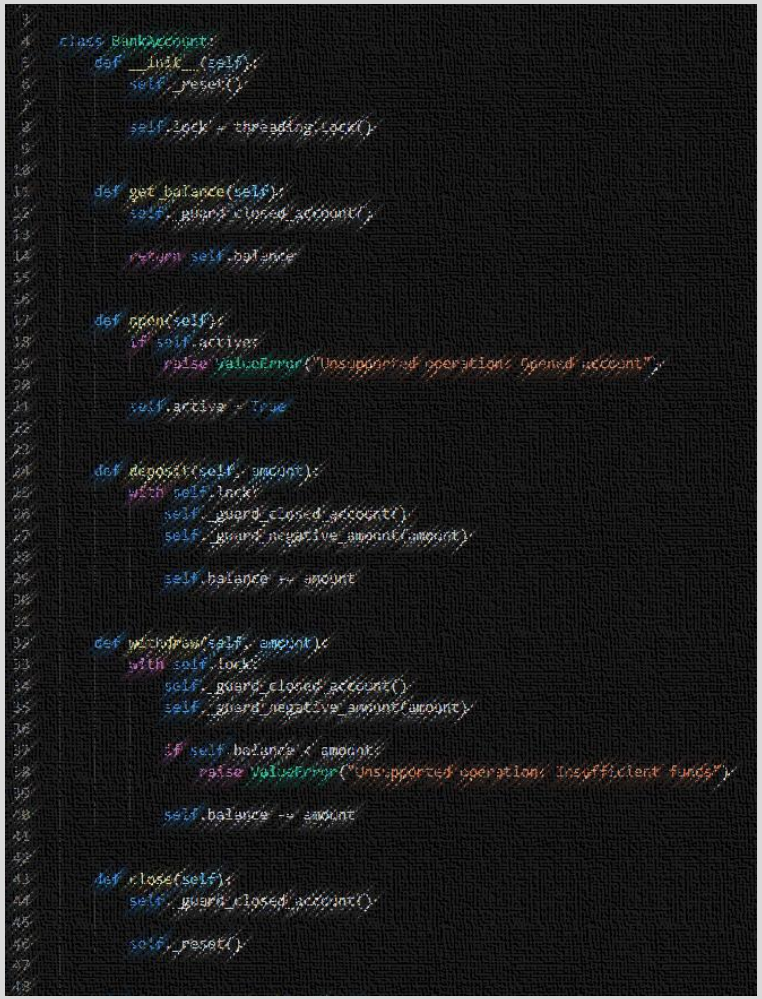
C# is a programming language, part of the .NET Framework family; in which everything is an object with a specific ***type***, ***value***, ***properties*** and ***methods***

Some of the built-in types:

Real world value	Built int type	Example
Numeric	int	int n = 10;
Floating-point numeric	float	float f = 10.5f;
Text - sequence of characters	string	string s1 = “Nahuel”; string s2 = “All 32”;
Boolean	bool	bool b1 = true; bool b2 = false;
Misc.	object	object s1 = “Nahuel”; object b1 = true;

Introduction to C#

A few basics concepts - Syntax



Declaring a type and value

<type> <name> ;

<type> <name> = <value> ;

<type> <name> = new <value> ( ) ;

<type> <name> = new <value> (*parameters*) ;

A method is an action that an object can perform

<object_name> . <method_name> ( ) ;

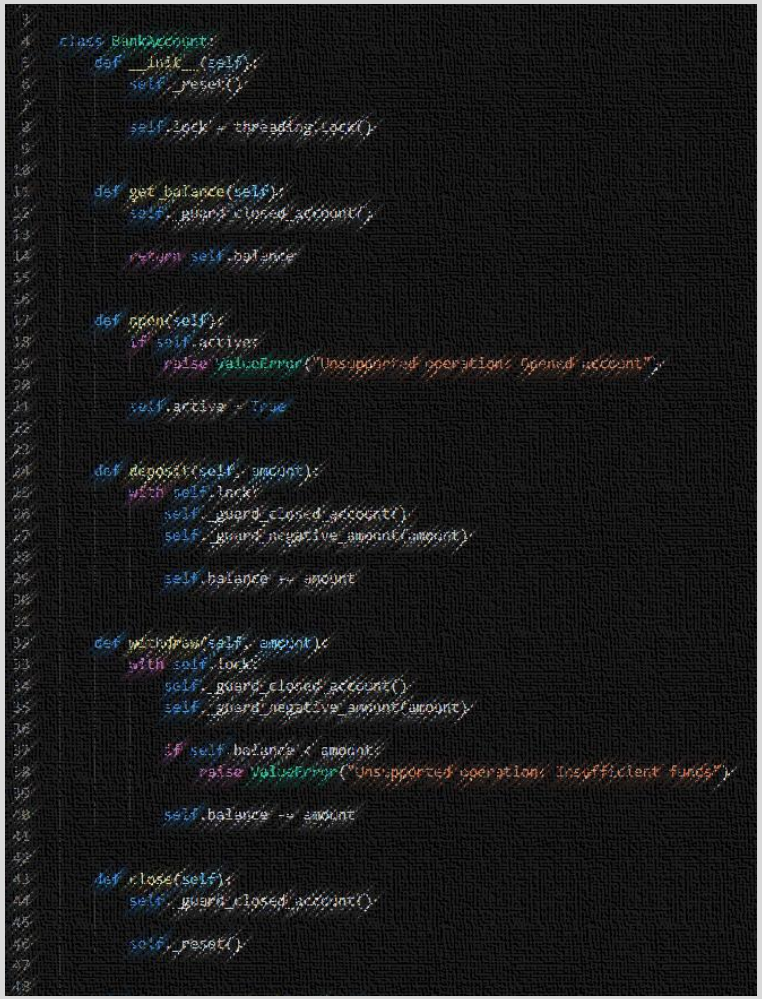
<object_name> . <method_name> (*parameters*) ;

A method can return a value to be used by the application somewhere else

<type> <name> = <object_name> . <method_name> ( ) ;

Introduction to C#

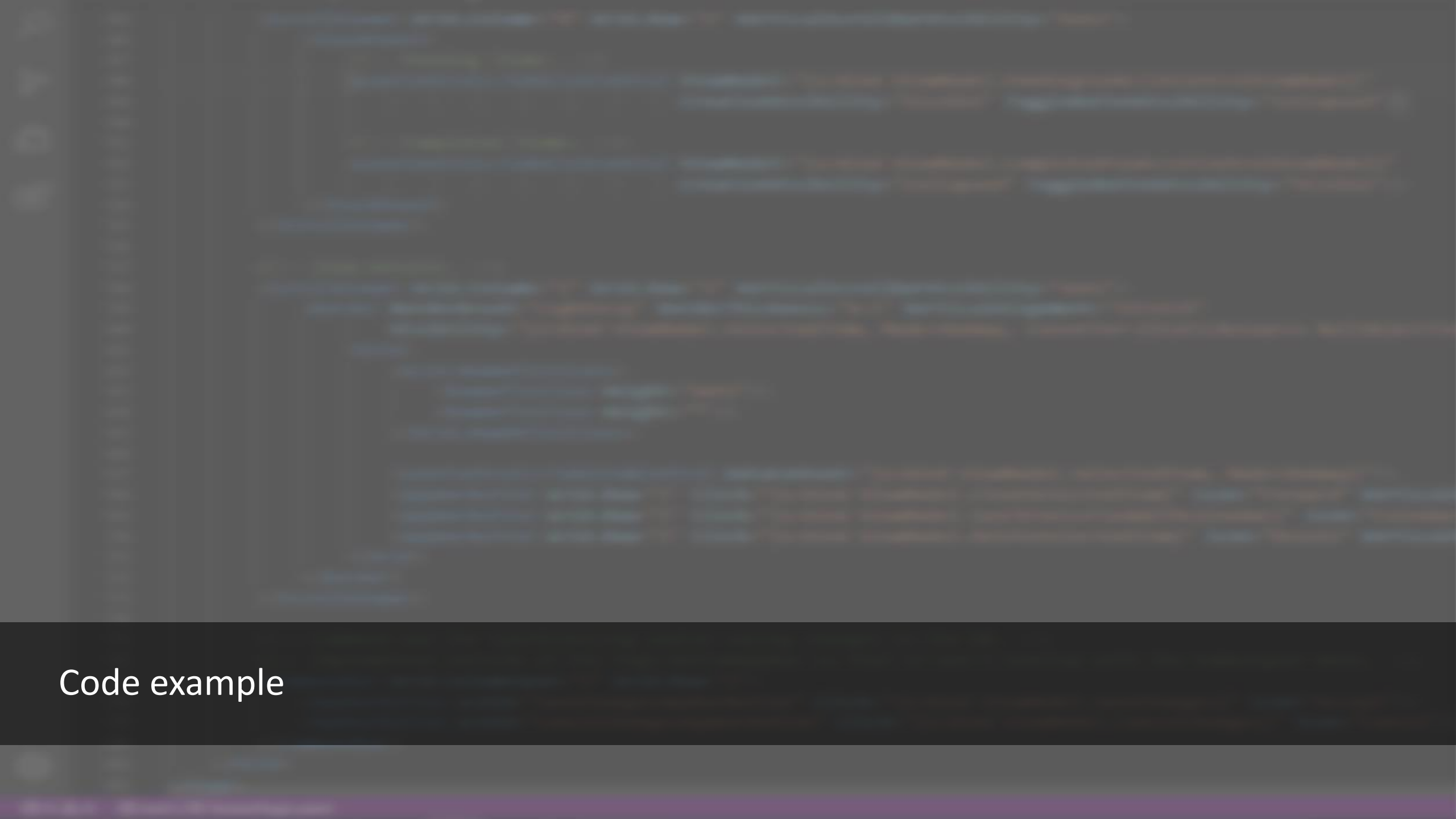
A few basics concepts - Scope



Code in C# must be placed within a scope

- A scope encapsulates contents and limits the visibility of the content to the members it contains
- Different scopes can be nested, creating a declaration space where unique names are required

```
scope_name
{
    <type> <name> = <value> ;

    sub_scope_name
    {
    }
}
```



Code example

Code example

Code example

```
namespace Recipes
{
    class PizzaDough
    {
        string ingredient = "water";
        float  quantity   = 1.25f;
        int    minTemp    = 110;
        int    maxTemp    = 115;

        string yeast      = "yeast";
        string sugar      = "sugar";

        void Prepare(int standTime)
        {
            // Step 1 - Warm up the water
            AddHeat(ingredient, quantity, minTemp, maxTemp);

            // Step 2 - Dissolve yeast and sugar in water
            Dissolve(yeast, sugar, water);

            // Step 3 - Let it stand for a specific amount of time
            Stand(standTime);

            // Step 4 - Add oil and salt
            Finish("oil", "salt");
        }

        void AddHeat(string ingredient, float quantity, int minTemp, int ma
        { /* Function code goes here */ }

        void Dissolve(string i1, string i2, string dissolver)
        { /* Function code goes here */ }

        void Stand(int standTime)
        { /* Function code goes here */ }

        void Finish(string i1, string i2)
        { /* Function code goes here */ }
    }
}
```

How could the instructions look in code?

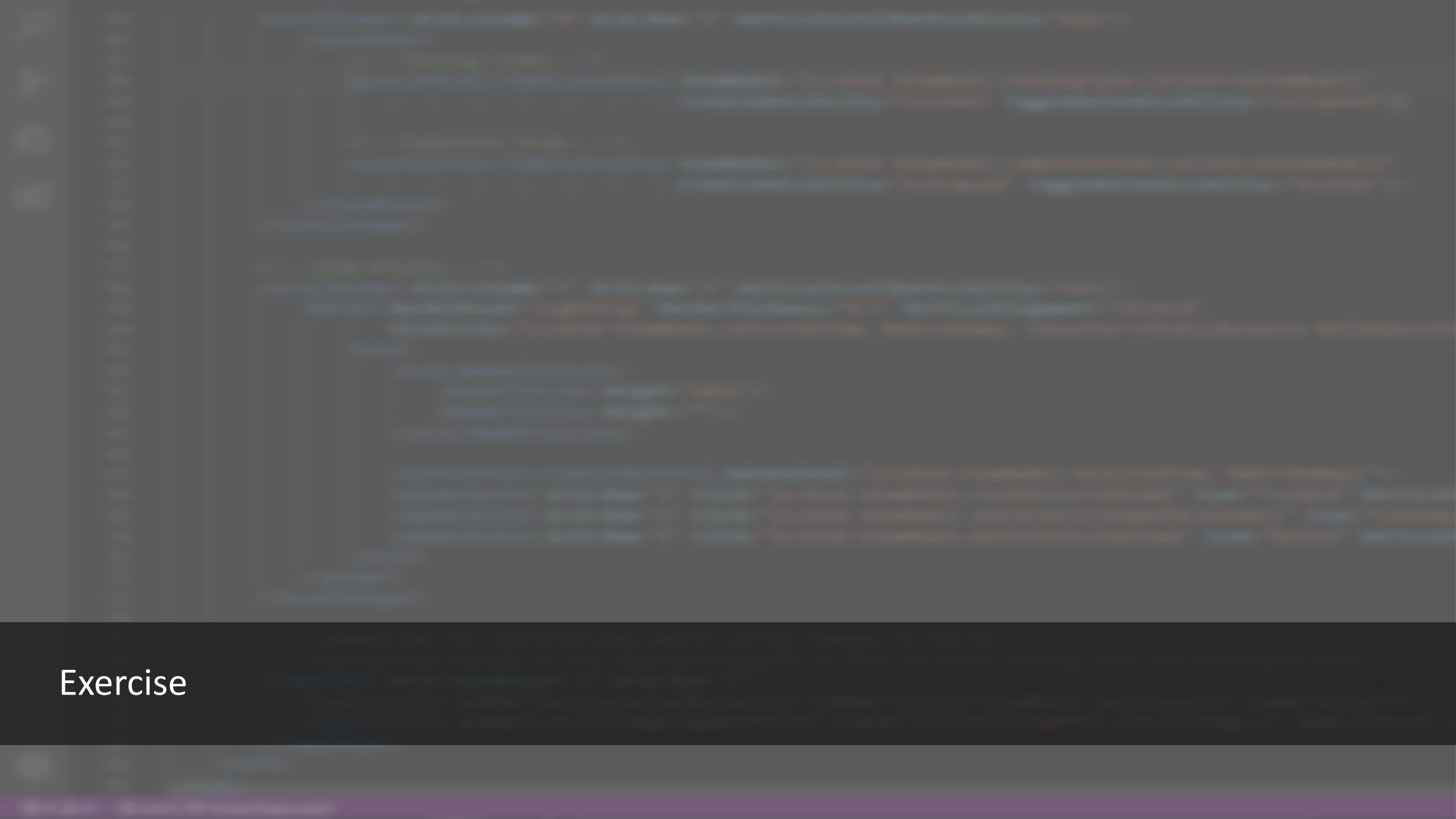
1-1/4 cups warm water (110° to 115°)

In a large bowl:

Dissolve yeast and sugar in water

Let stand for 5 minutes

Add oil and salt.



Exercise

Exercise

Console application

Using Visual Studio, we will create a new project called **Console App (.NET Core)**, with default options

This will create a *Hello World* application, which is traditionally used to introduce beginners to a new programming language
This program simply displays the phrase "Hello World!" on the screen

You can read more on Console Applications, how to execute and debug them on the official .NET Core Console App tutorial from Microsoft:
https://docs.microsoft.com/en-us/dotnet/core/tutorials/with-visual-studio?tabs=csharp

Exercise

Objectives

```
PIZZA DOUGH recipe, by Nahuel
=====================================

Nahuel, let's go through the ingredients needed to prepare your pizza dough.
Don't forget to specify if the ingredient has any special condition!

How much yeast do you need? 1 package, dry

How much sugar do you need? 1 teaspoon

How much water do you need? 1-1/4 cups, warm

How much oil do you need? 1/4 cup, canola oil

How much salt do you need? 1 teaspoon

How much flour do you need? 4 cups, all purpose
```

- Asks the user for the quantity and conditions needed to prepare a pizza dough

```
PIZZA DOUGH recipe, by Nahuel

Ingredients table
|     Ingredient     |          Comments          |
| ================== | ========================== |
|      Yeast         |        1 package, dry      |
|      Sugar         |        1 teaspoon          |
|      Water         |        1-1/4 cups, warm    |
|      Oil           |        1/4 cup, canola oil |
|      Salt          |        1 teaspoon          |
|      Flour         |        4 cups, all purpose |
  ================== | ==========================
```

- Shows the ingredient table on screen

Closing comments

Closing comments

Further research

You can expand the topics covered in this lesson by reading upon the following topics

- Built-in types
  https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/built-in-types

  Pay special attention to *default values* and *casting operators*

- String concatenation and formatting
  https://docs.microsoft.com/en-us/dotnet/csharp/how-to/concatenate-multiple-strings

- .NET Core console app
  https://docs.microsoft.com/en-us/dotnet/core/tutorials/with-visual-studio?tabs=csharp

Closing comments

Summary

In this lesson, you learned about

- The basics on how a machine *works* with code
- What is the C# language
- Some of the built in types in C# and how they correlate to real world values
- How scoping and declarations work in C#
- We touched the surface on what a method is
- We used a Console Application template