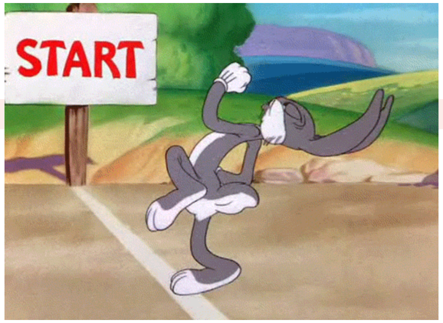
01 Hello World

Programming fundamentals

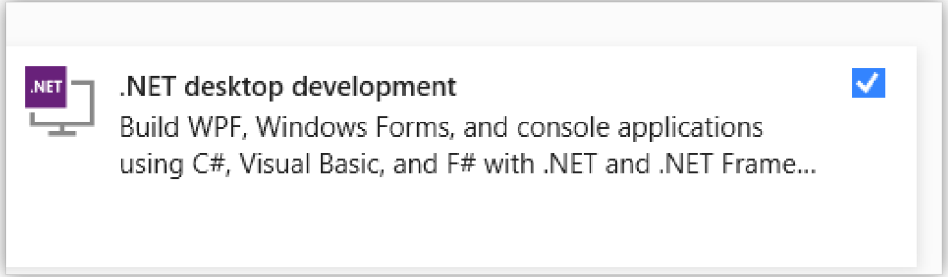
YP0616 - YP0601

Elke Boonen & Tristan Vandevelde

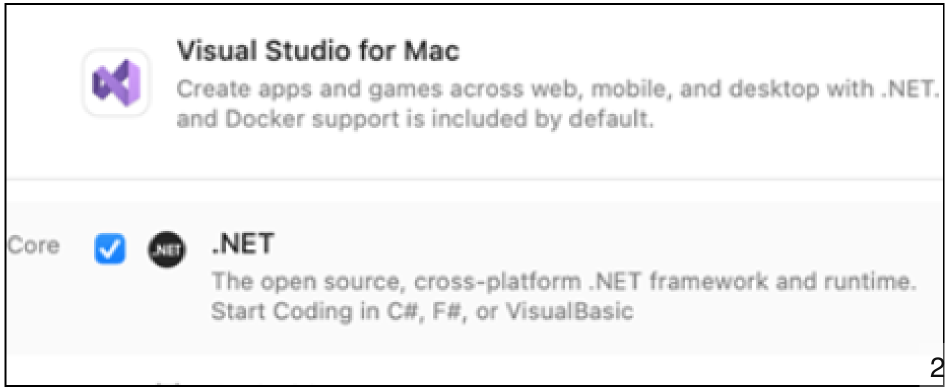
Before we start!



- **Install Visual Studio on Windows** (community or enterprise)
    - https://visualstudio.microsoft.com/
    - https://www.academicsoftware.eu/dashboard
    - **Install *.NET desktop development:* *--> console applications using C#.***


.NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

- **Install Visual Studio on Mac**
    - https://visualstudio.microsoft.com/vs/mac/
    - installation tutorial


Visual Studio for Mac
Create apps and games across web, mobile, and desktop with .NET. and Docker support is included by default.

Core
.NET
The open source, cross-platform .NET framework and runtime. Start Coding in C#, F#, or VisualBasic

Learning objectives (ECTS)

- **Basic principles** (types, operators, expressions) & **structures** (loop & if)
- **Arrays**, **lists**, **dictionaries**
- **Methods** and **functions**
- Basic principles of **OO**
- **Files**, in-and output **IO**
- **Exception** handling

Learning materials

- **Canvas** LMS https://thomasmore.instructure.com/
  - Presentations
  - E-Book: Fundamentals of Computer Programming with C#
  - Cheatsheet C#
  - Assigments (CodeGrade)
- **Online**
  - https://docs.microsoft.com/en-us/dotnet/csharp/
  - https://github.com/ElkeBoonen/ProgrammingFundamentals (code from slides)
  - https://github.com/ElkeBoonen/ProgrammingFundamentals-Students (code from class)
- **Software**
  - Visual Studio (Community) https://visualstudio.Microsoft.com/

Schedule

Before autumn break	After autumn break
01 Hello world	07 Exception handling
02 Variables & expression	08 Recap
03 If-structures	09 Collections
04 Loops	10 Methods
05 Files (IO)	11 OO
06 Arrays	12 OO
	13 Exam prep

Schedule is always subject to unexpected circumstances

Evaluation

- **1st term**
  - Permanent Evaluation (30 %):
    - CodeGrade exercises (each week, from week 02)
  - Computer Exam (70 %) use of cheatsheet only!
- **2nd term**
  - Computer Exam (100 %) use of cheatsheet only!

01 Hello world!

- Why programming?
- Why C#
- Hello World!
- Input/output
- CodeGrade

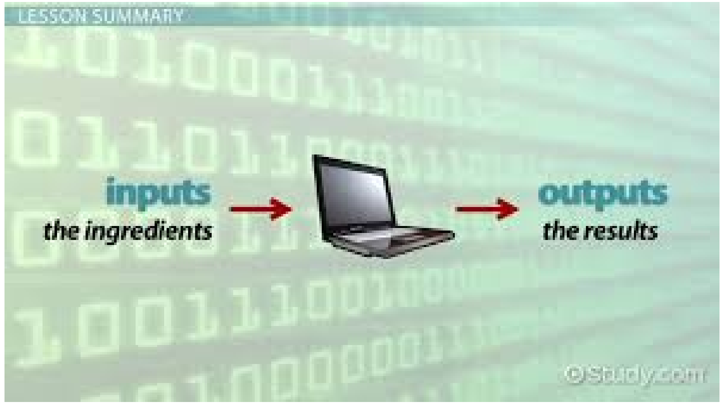
What is an algorithm

- **A sequence** of well-defined **instructions** to perform a special task
  - *Recipe*
- **Solve a problem** or **perform** a **computation**
  - *I am hungry! So we need bread!*
- **Input to output**
  - *From ingredients to bread*

WATER
YEAST
SUGAR
SALT
FLOUR
OIL
KNEAD dough
1h
PUT IN WARM PLACE
MAKING BREAD
at home
DIVIDE DOUGH ...
... INTO PIECES
PRESS EACH PIECE ...
... INTO A FLAT DISK
C°
BAKE ON EACH SIDE
WELL DONE

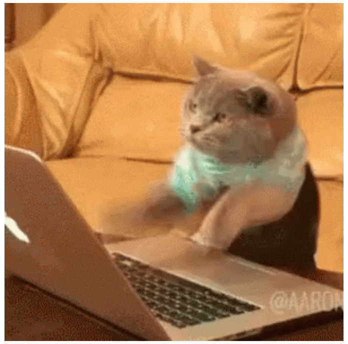
What is a program?

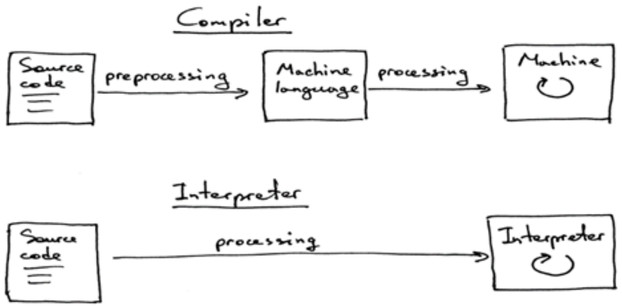
- A **series of instructions performed by a computer**
- A program is **always a combination** of
  - sequences
  - selections
  - loops
  - methods
  - ready-made objects
  - objects you write yourself



What is a program?

- **Coding** = **translating algorithm** to **code**
- **Programming** = writing an algorithm in a programming language
- Programming **is thinking**!
- The written program is **converted to machine language** by
  - **Compiler**: transfer full program (eg: C#, Java...)
  - **Interpreter**: transfer line by line (eg: PHP, Python..)
  - **Assembler**: transfer to zeros and ones





Execution of program

- Computer executes **program step by step**
- A computer is **very strict**
- **Error** in **code** = **error** in **execution**

Why should you learn to program?

“ Everybody in this country should learn to program a computer, because it teaches you how to think

- Steve Jobs -

https://www.youtube.com/embed/uL3047AJRgk?enablejsapi=1

From A# to Z++

- Programming is only 50 years old
- More than 700 programming languages!
- Why is there not **just one universal language?**

01 Hello world!

- ~~Why programming~~
- Why C#
- Hello World!
- Input/output
- CodeGrade

Why C#

- It's **not about the language**,
  it't **about the algorithm!**
- But why C#?
  - Object oriented
  - Needs to be compiled
  - Well documented
  - Easier than C, C++ or Java
  - Just a good starting point for learning to code

Visual Studio (community)

- It's just **the best IDE** (Integrated Development Environment) in town!
- **Intellisense** like **code completion**, **quick info**, **member lists**...
  - always remember CTRL+Space, just saying...
- **Download Visual Studio** (Community) (also Mac-edition) https://visualstudio.microsoft.com/
- **Install** *.NET desktop development: Build WPF, Windows Forms, and console applications using C#.*

01 Hello world!

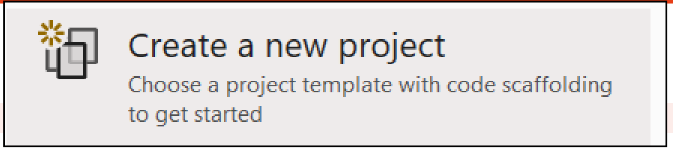
- ~~Why programming~~
- ~~Why C#~~
- Hello World!
- Input/output
- CodeGrade

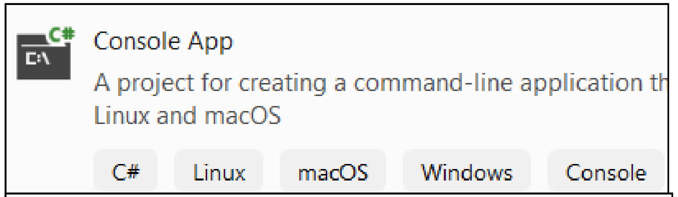
Hello to the world!

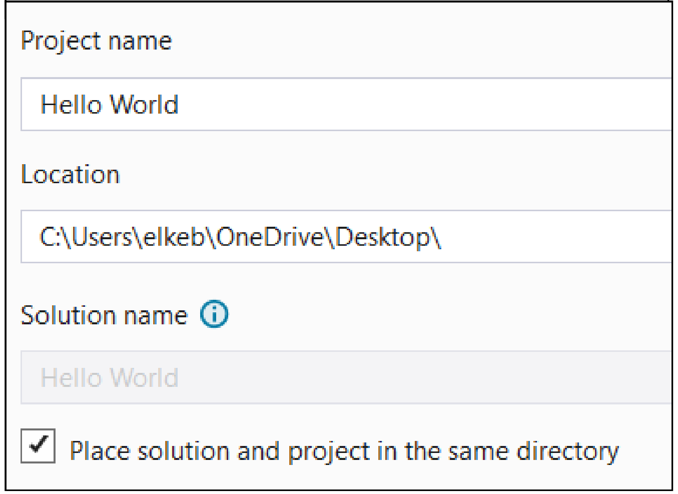
- **First program!**
  - first program in any programming language
  - just saying hi to the world!
- Easy peasy lemon squeezy
- Show text on screen

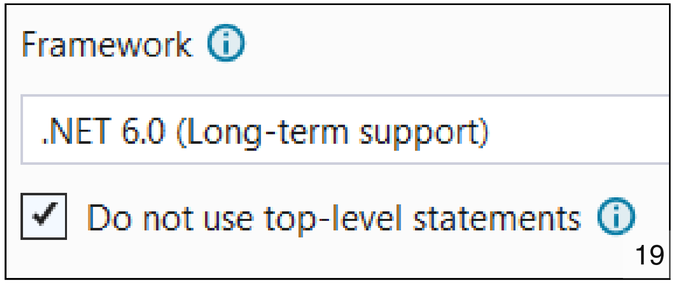
First project

- Open Visual Studio
- Create **a new project**
  - **Choose Console.App (.Net Core) C#**
  - ***Mac-users : Apps - Console app***
- **Project name**: Hello World
  - **Location**: choose a location!
  - **Solution**: Create new solution
  - **✔ Place solution and project in the same directory**
- **Framework** .NET 6.0
  - **✔ Do not use top-level statements**
  - ***Mac-users: choose .NET 3.1***


Create a new project
Choose a project template with code scaffolding to get started


Console App
A project for creating a command-line application th
Linux and macOS
C# Linux macOS Windows Console


Project name
Hello World
Location
C:\Users\elkeb\OneDrive\Desktop\
Solution name
Hello World
✓ Place solution and project in the same directory

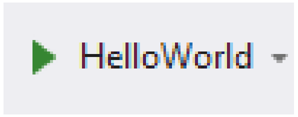

Framework
.NET 6.0 (Long-term support)
✓ Do not use top-level statements

Take a look at the code

```
using System;

namespace HelloWorld
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

```
Hello World!
```

- Hit F5 or 

- Terminal opens

  (press any key to close)

Putting it out there

- Prints **text to output** screen
- **Text** must be **placed between " "**
- **Every line of simple code** must **end** with **semicolon ;**
- **C# is case sensistive!**

```csharp
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Try Console.Write(), what happens?

Everything starts in Main

```
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

- **Main-method**
- **Execution starts here**!
- Main-method = **code block**
- Everything i**nside block code** is **surrounded** by **braces { }**
- **Don't use ; after declaration of method or after { }**
- **Main is mandatory** in every C#-program
- **Everything inside Main will get executed**

Add some extra writelines in Main, what happens?

The class is half full

```
using System;

namespace HelloWorld
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

- **Creates** a **class named Program!**
  - **internat =** limited to the assembly in which it is declared
- **C# is object-oriented** so creating a **class** is **mandatory**!
- **Every class** is a **code block**, so code in class is surrounded by **braces { }**
- **Don't use ;** after declaration of class or after { }

To namespace and beyond

```
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

- **Creates** a namespace **HelloWorld** (container for all associated classes)
- **A namespace** is a **code block**, so content is **surrounded by braces { }**
- **Don't use ; after declaration of namespace or after { }**

To use or not to use

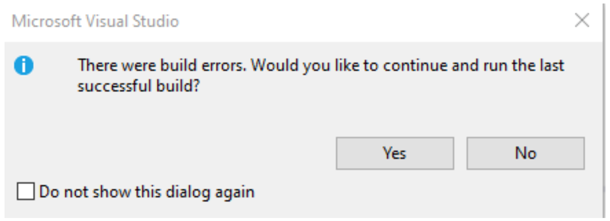
```
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

- **Without using System** we would write **System.Console.WriteLine**
- The **using-statement imports all functionality from the System namespace** which we need to print text to a screen.

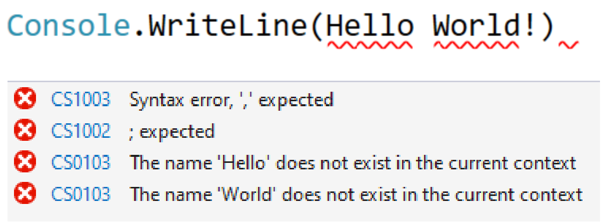
Computer says no

- Your **program won't run!**
  - Message: *there where build errors*
  - *Hit **NO***
- **Red wavy lines indicate** the **error**
- **Start debugging** by **reading** the description of the error
- **Try to fix it! Test** and **re-test**



Microsoft Visual Studio

There were build errors. Would you like to continue and run the last successful build?

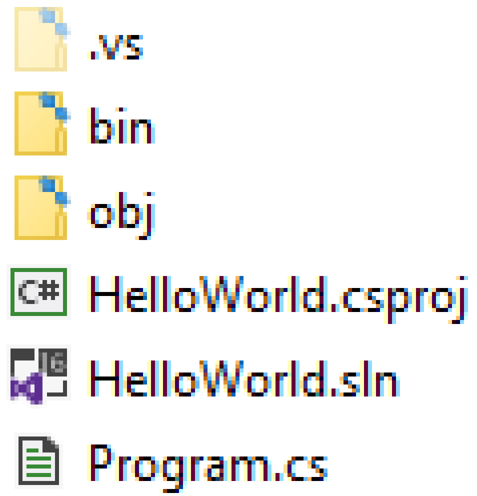


```
Console.WriteLine(Hello World!)
```

```
CS1003  Syntax error, ',' expected
CS1002  ; expected
CS0103  The name 'Hello' does not exist in the current context
CS0103  The name 'World' does not exist in the current context
```

“ Breathe in, breathe out, in any case never hit your computer!

What's in a project



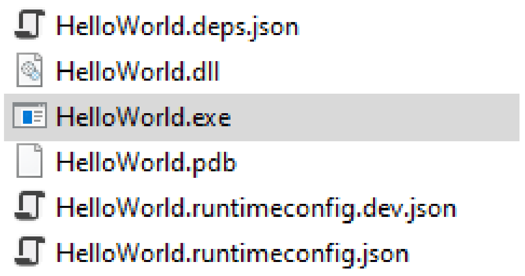
.vs
bin
obj
HelloWorld.csproj
HelloWorld.sln
Program.cs

- Find the project in file explorer
  - RMB project name in solution explorer,
  - select Open folder in File explorer)
- HelloWorld.sln (solution le VS)
- Program.cs (actual code)
- bin (binary files, executable code)
- obj (object files, temporary to build binary)

Execute, make it happen!

Execute HelloWorld outside VS

- Open executable in **file explorer**
  - navigate bin, debug, netcoreapp3.1, double-click HelloWorld.exe
  - Look closely, because it closes quickly.. Why?



HelloWorld.deps.json
HelloWorld.dll
HelloWorld.exe
HelloWorld.pdb
HelloWorld.runtimeconfig.dev.json
HelloWorld.runtimeconfig.json

- Execute in terminal, open **Powershell**

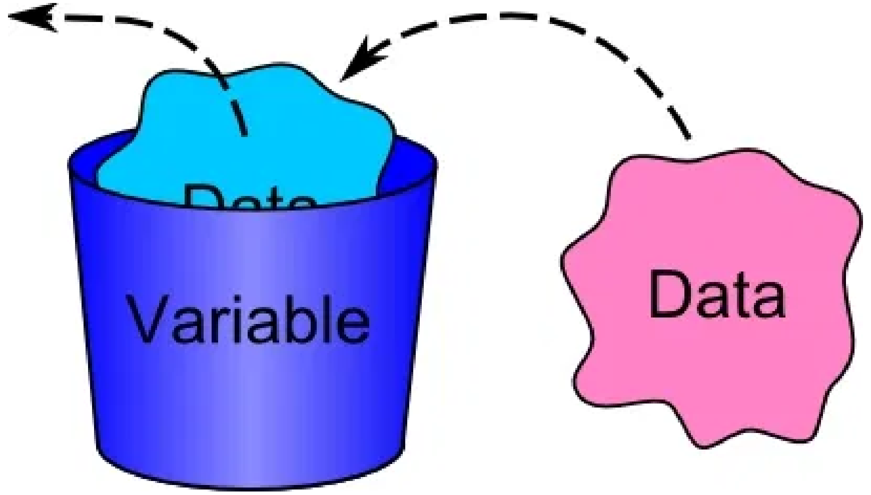
```
> cd <navigate to project folder>
> cd bin
> cd .\Debug\
> cd .\netcoreapp3.1\
> .\HelloWorld.exe
Hello World!
```

01 Hello world!

- ~~Why programming~~
- ~~Why C#~~
- ~~Hello World!~~
- Input/output
- CodeGrade

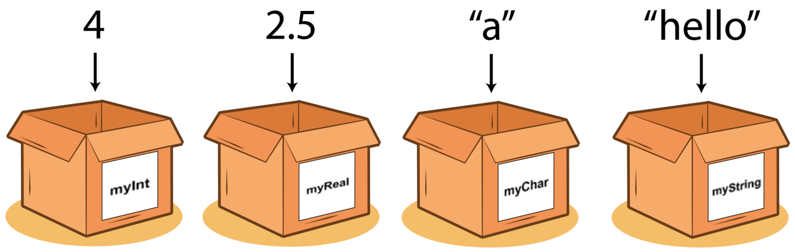
What is a variable?

- Container for **storing data** values
- Why do we need variables?
  - To **store information** given by the user as user input
  - To **calculate** and store intermediate results
  - To store information for **later use**
  - ...



Creating a variable

- C# strongly typed variables: variable must be **declared before using**
- What is **declaring a variable**?
  - give your variable **a name**
  - give your variable **a type**: C# has different primitive types!
- What is **initializing** a **variable**?
  - give your variable its **first value**
- Declaring and initializing can be done at the same time!

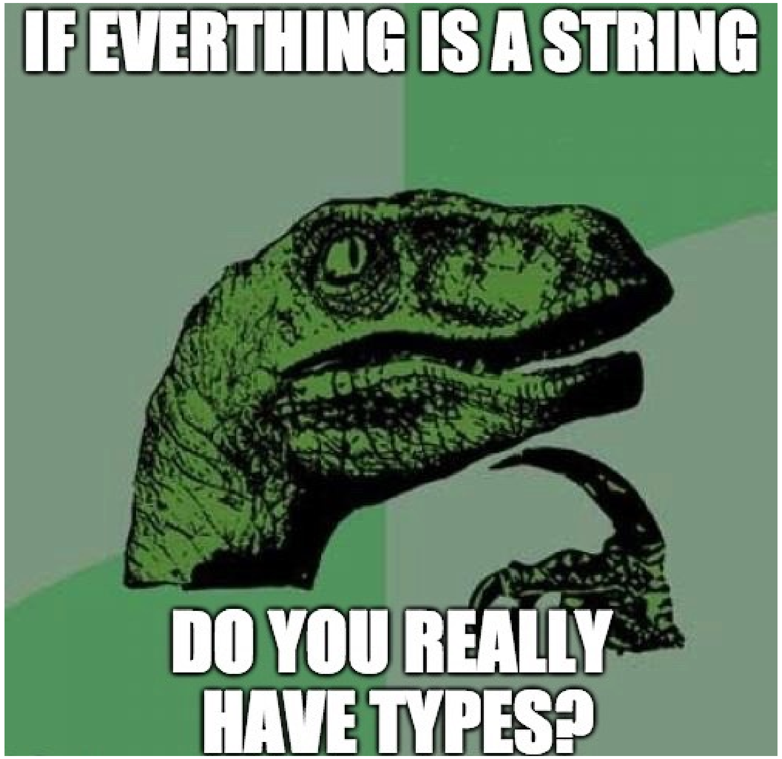


String

- **C# type** for **text** = **string**
  - text-value **between " "**
- Always declare variable before use
  - type + name
- Value of variable can change!

```
string text;
text = "some text";

string name = "Elke";
name = "Jan"
```

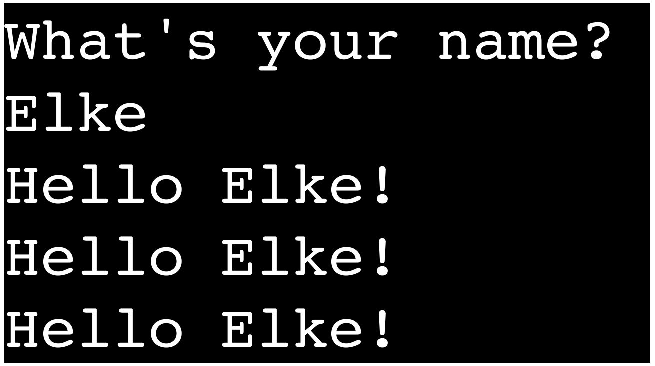


Everything in the console is a string!

- Everything on the command line is a piece of text = a string!
  - **write to console** = **Console**.**WriteLine**()
    - different ways of concatenating strings
  - **read user input from console** = **Console**.**ReadLine**()
    - store value given by the user in a variable for later use!

```csharp
Console.WriteLine("What's your name?");

string name = Console.ReadLine(); ;
Console.WriteLine($"Hello {name}!");
Console.WriteLine("Hello " + name + "!");
Console.WriteLine("Hello {0}!", name);
```



```
What's your name?
Elke
Hello Elke!
Hello Elke!
Hello Elke!
```

"No comment" is a comment

- A **comment** is an **explanation or annotation** in the code
- They are added to clarify code and are ignored by the compiler

```
//this is one line of comment

/*
 These are mulitple lines of comment!
*/
```

01 Hello world!

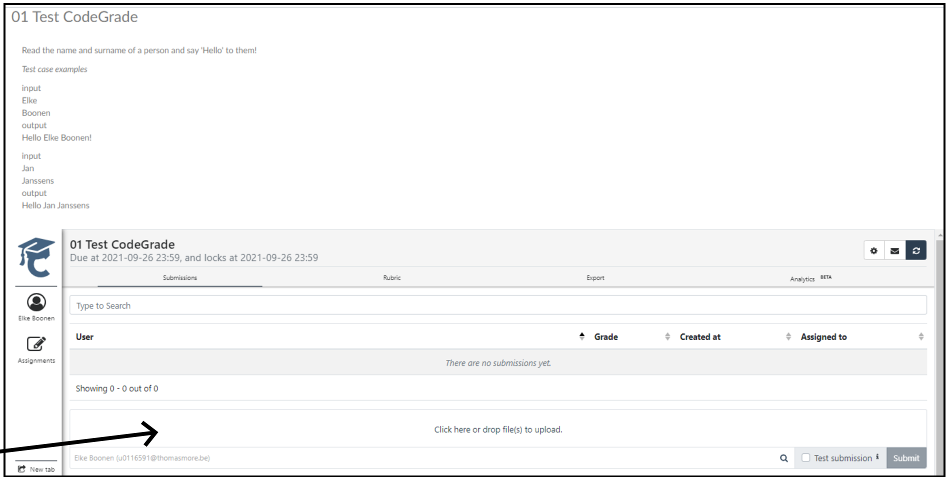
- ~~Why programming~~
- ~~Why C#~~
- ~~Hello World!~~
- ~~Input/output~~
- CodeGrade

We ❤ CodeGrade, yes we do!

- Find the assignments in Canvas
- Spend some hours to do them (plagiarism results in a 0!)
- Submit (only) your .cs-file
- Wait for your automatically generated result
- Hit a home run? Do a little dance ;)
- Not so successful? Tweak your solution and re-submit!
- You can keep practicing until the deadline to become better, but also to score higher points on your permanent evaluation

01 Test CodeGrade

- Go to **Modules** or **Assignments**
- Find **01 Test CodeGrade**
- **READ the assignment!**
- Create new project and solve assignment!
- **Upload program.cs-file of project**



01 Test CodeGrade

Read the name and surname of a person and say 'Hello' to them!

Test case examples

```
input
Elke
Boonen
output
Hello Elke Boonen!

input
Jan
Janssens
output
Hello Jan Janssens
```

01 Test CodeGrade
Due at 2021-09-26 23:59, and locks at 2021-09-26 23:59

User	Grade	Created at	Assigned to
There are no submissions yet.			

Showing 0 - 0 out of 0

Click here or drop file(s) to upload.

Elke Boonen (u0116591@thomasmore.be)

Practice makes perfect!

- Do your exercises, spend the hours!
- The better the exercises, the better the exam!

Say what? How many hours?

6 SP = 6 * 28 hours = 168 hours

Lessons = 12 * 4 hours = 48 hours

Exam = 2 hours

Exercise = 168-48-2 = 118 hours

codegrade