De eerste stappen

Zie Scherp Scherper - Hoofdstuk 1

Tim Dams

Wat leer je in dit hoofdstuk?

Na dit hoofdstuk kan je:

- uitleggen wat **programmeren** en een **algoritme** zijn
- de rol van een **programmeertaal**, de **compiler** en **.NET** situeren
- een **console-project** aanmaken in Visual Studio
- in- en uitvoer verwerken met `WriteLine`, `Write` en `ReadLine`
- de meest voorkomende **beginnersfouten** herkennen en oplossen
- **kleur** gebruiken in de console

Eerst denken, dan typen

> First, solve the problem. Then, write the code.

- Leren programmeren is als een berg beklimmen waarvan je nooit de top bereikt

  → en dat is net het mooie: er valt altijd iets nieuws te leren

- De eerste stappen zijn nooit eenvoudig

Tip

Blijf volhouden. Blijf oefenen. Vloek gerust af en toe. En vooral: geniet van het ontdekken van nieuwe dingen.

Wat is programmeren?

- Computers zijn in essentie **ongelooflijk domme dingen**, hoe krachtig ze ook zijn
- Ze doen altijd **exact** wat jij zegt, niet wat je bedoelt

> **Programmeren** = leren praten met die domme computers zodat ze doen wat jij wilt.

Opmerking

Geef je hen de opdracht om te ontploffen, schrik dan niet dat je even later de 112 mag bellen.

Het algoritme

- Mythe: een programmeur schrijft de hele dag code
- Echt werk = **analyse van het probleem** + een goed **algoritme**

Een algoritme is het **recept** dat je aan de computer geeft: een **reeks instructies** in de juiste **volgorde**.

```
1 Haal dop van het ventiel.
2 Plaats pomp op ventiel.
3 Begin te pompen.
```

> ⚠ **Waarschuwing**
>
> Een andere volgorde geeft vreemde, en soms fatale, fouten. Logisch en analytisch denken is dus de echte kern.

Een programmeertaal

Een taal die de computer begrijpt is meestal:

- **ondubbelzinnig**: elke opdracht heeft exact één betekenis (anders dan “wat een koele kikker”)
- **woordenschat**: een vaste lijst woorden die je mag gebruiken
- **grammaticaregels**: een vaste volgorde

> **Tip**
>
> Enkel Yoda mag woorden in de verkeerde volgorde gebruiken. “bal rood jongen is gooit veel”?

C# (siesjarp)

- Eén van de vele programmeertalen, deel van **.NET** (“dotnet”)
- Een **hogere programmeertaal**: leesbaarder, dichter bij onze eigen taal
- C# lijkt als twee druppels water op **Java**, en stamt af van C en C++

> **Tip**
>
> Net als bij talen leren: ken je eenmaal Frans, dan leer je makkelijker Italiaans of Spaans. Zo ook met programmeertalen: na dit boek hebben Python en JavaScript weinig geheimen meer voor je.

Anders Hejlsberg

Anders Hejlsberg

De Deen die C# boven de doopvont hield bij Microsoft.

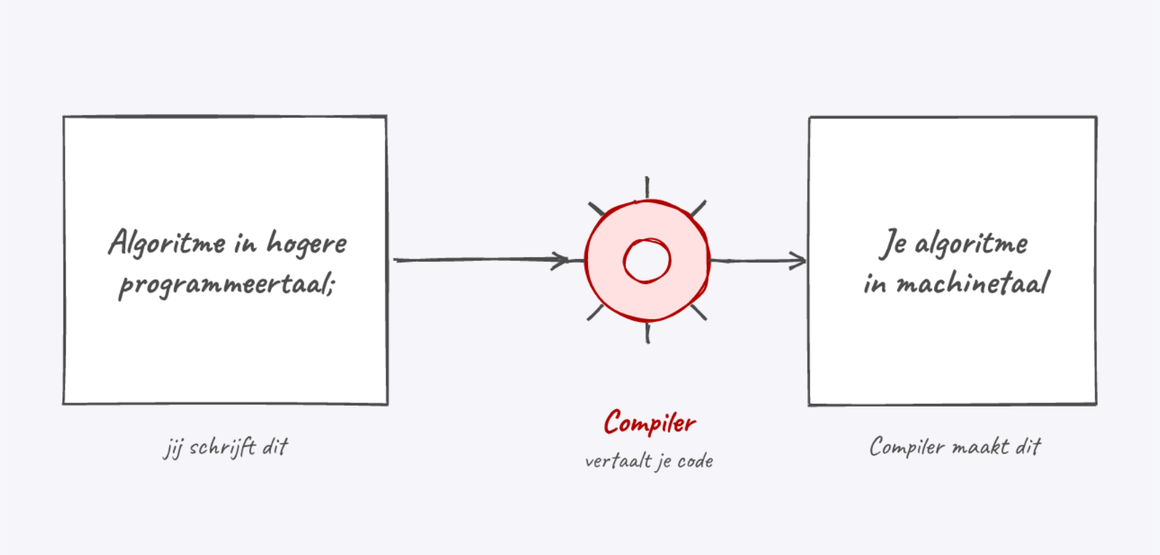
- Schreef eerder **Turbo Pascal**
- Was *chief architect* van **Delphi**
- Bedacht later ook **TypeScript**

Tip

Als je één poster in je slaapkamer hangt, zou het die van Anders moeten zijn.

De compiler

- Machinetaal rechtstreeks schrijven is loodzwaar: tientallen lijnen voor één letter op het scherm
- We hebben dus een **vertaler** nodig: de **compiler**



Vereenvoudigd compiler-overzicht.

De kern: **C#-code omzetten naar een uitvoerbaar bestand in machinetaal**.

.NET, kort

> **Opmerking**
>
> **.NET** is een **framework**: een grote verzameling bibliotheken plus de **Common Language Runtime (CLR)**.
>
> - Je C#-code wordt eerst omgezet naar **Intermediate Language (IL)**
> - Bij het uitvoeren zet de CLR die IL **Just-In-Time (JIT)** om naar machinetaal
> - Daardoor kunnen .NET-talen (C#, F#, VB.NET) samenwerken met diezelfde bibliotheken

Naamgeving van .NET en C

- Sinds 2020: gewoon **.NET gevolgd door een nummer** (.NET 5, 6, … en intussen hoger)
- Elk jaar in november een nieuwe versie; de **even** nummers zijn **LTS** (langer ondersteund)
- Dit boek werkt steeds met de **meest recente LTS-versie**
- Elke .NET-versie komt met een nieuwe **C#-versie** (features die je *kan*, niet *moet* gebruiken)

> **Tip**
>
> Waarom deze geschiedenisles? Online bots je vaak op oudere namen. Anders raak je het noorden kwijt bij het opzoeken.

Visual Studio: de IDE

- We werken met **Visual Studio Community 2026** (gratis)
- VS is een **IDE** (Integrated Development Environment): debugger, code-editor en compiler in één
- **Niet** Visual Studio *Code*: dat is een lichtere, aparte editor

> ⚠ **Waarschuwing**
>
> Sinds augustus 2024 bestaat VS Community niet meer voor **Mac**. Mac-gebruikers werken met VS Code of JetBrains Rider, of draaien Windows via een virtuele machine.

Installeren: één workload volstaat

- Download de **Community**-editie via visualstudio.microsoft.com/vs
- Kies bij de installatie zeker de workload **“.NET desktop development”**

Meer heb je voorlopig niet nodig.

Een nieuw project: Console App

Kies “Create a new project” en let op:

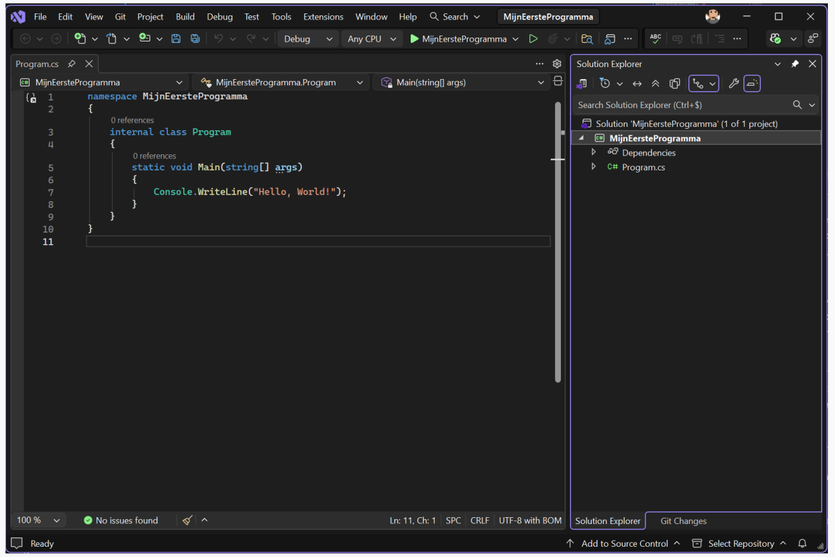
- Projecttype **Console App** met het **groene C#-icoon** (niet Visual Basic)
- Duid zeker **“Do not use top-level statements”** aan
- Laat **“Solution name”** met rust en vink **niet** “place solution and project in the same directory” aan

Belangrijk

Krijg je code te zien die er totaal anders uitziet? Dan koos je wellicht het verkeerde projecttype, de verkeerde taal, of vergat je de top-level-statements-checkbox.

De IDE-layout

- Elk open bestand krijgt een eigen **tab**
- De **Solution Explorer** toont alle bestanden van je project (niet wegklikken!)
- De **Copilot**-AI rechts mag je meteen wegklikken



VS zonder Copilot.

> **Tip**
>
> Layout om zeep? “Window” - “Reset window layout” zet alles terug.

A.I. even parkeren

- VS heeft krachtige AI ingebouwd: **IntelliCode** en **Copilot**
- Verbluffend goed, maar voor een beginner **af te raden**

> Je leert ook niet hoofdrekenen door vanaf dag 1 met een zakrekenmachine te werken.

Belangrijk

De programmeur van de toekomst moet **béter** zijn dan Copilot en ChatGPT. Wie enkel op AI leunt, beperkt z'n kansen. Zet de tools dus voorlopig **uit**.

Je eerste programma

```
1  namespace Demo1
2  {
3      internal class Program
4      {
5          static void Main(string[] args)
6          {
7              Console.WriteLine("Hello World!");
8          }
9      }
10 }
```

> **Belangrijk**
>
> **Voorlopig negeren.** De keywords `namespace`, `class` en `static void Main` lijken mysterieus. Dat geeft niet: schrijf voorlopig enkel **tussen de accolades van `Main`**. We pluizen ze uit vanaf hoofdstuk 7.

Even in het diepe

Ik gooi je heel even in het diepe deel van het zwembad, maar haal je er op tijd weer uit. Daarna spelen we rustig verder in het babybadje.

Onthoud nu alvast:

- Al je eigen code komt voorlopig **tussen de `Main`-accolades**
- Eindig elke lijn met een **puntkomma** (`;`)
- Code wordt **van boven naar onder** uitgevoerd

WriteLine: tekst op het scherm

- **`Console.WriteLine()`** toont alle tekst tussen de aanhalingstekens

```
1 Console.WriteLine("Hello World!");
2 Console.WriteLine("Hoi, ik ben het!");
3 Console.WriteLine("Wie ben jij?!");
```

De aanhalingstekens én de puntkomma achteraan zijn **cruciaal**.

ReadLine: input van de gebruiker

- **`Console.ReadLine()`** leest wat de gebruiker typt tot die op enter drukt

```
1 Console.WriteLine("Geef je naam?");
2 string naam = Console.ReadLine();
```

We maken een geheugenplekje (**variabele**) `naam` van het type `string` en bewaren daarin wat de gebruiker invoert.

> **Tip**
>
> Toekenning gebeurt **van rechts naar links**: `naam` (links) krijgt de waarde van het stuk rechts van de `=`.

Aanhalingsteken of niet?

```
1 Console.WriteLine("Dag " + naam + ", hoe gaat het?");
```

- **Mét** aanhalingstekens (`"Dag "`): letterlijk die tekst
- **Zonder** aanhalingstekens (`naam`): de **inhoud** van de variabele

`Console.WriteLine("naam");` toont dus letterlijk het woord *naam*, niet wat de gebruiker typte.

Write vs WriteLine

- **`WriteLine`** plaatst een *enter* aan het einde van de lijn
- **`Write`** doet dat **niet**: het volgende komt op **dezelfde lijn**

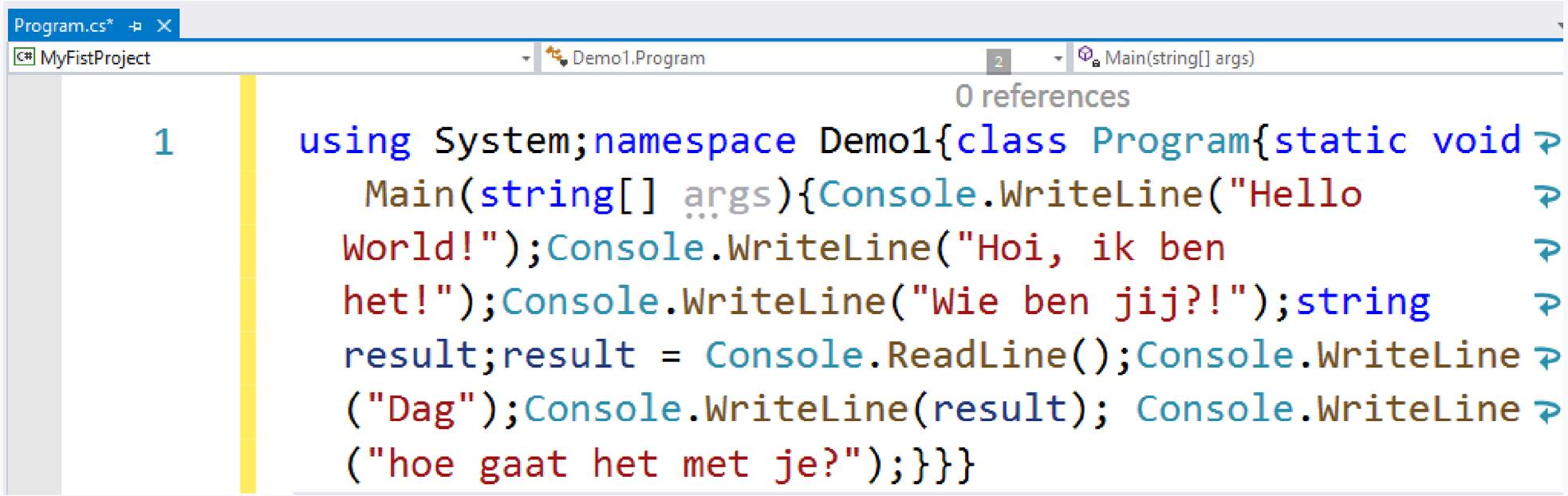
```
1 Console.Write("Dag");
2 Console.Write(naam);
3 Console.Write("hoe gaat het?");
```

Resultaat: `Dagtimhoe gaat het?`

> **Belangrijk**
>
> Spaties zijn ook tekens. Zet ze **binnen** de aanhalingstekens: `"Dag "`. Spaties buiten de quotes worden genegeerd.

C# negeert witregels

- Spaties, enters en tabs **buiten** aanhalingstekens worden genegeerd
- Je kan een heel programma op één lange lijn typen… maar doe dat niet



Eén lijn code. Cool? In sommige kringen. Onleesbaar? Zeker.

Zinnen aan elkaar plakken

De **+-operator** plakt tekst aan elkaar:

```
1  Console.WriteLine("Dag " + naam + ", hoe gaat het?");
```

> **Tip**
>
> `+` werkt prima, maar is niet de modernste manier. In het volgende hoofdstuk leer je **string interpolation**: veel leesbaarder.

Sneller typen met snippets

Je gaat **heel vaak** `Console.WriteLine` schrijven. Gelukkig zit er een snippet in VS:

> **`cw`** + tweemaal **`[tab]`**

En *automagisch* verschijnt er een verse lijn `Console.WriteLine();`.

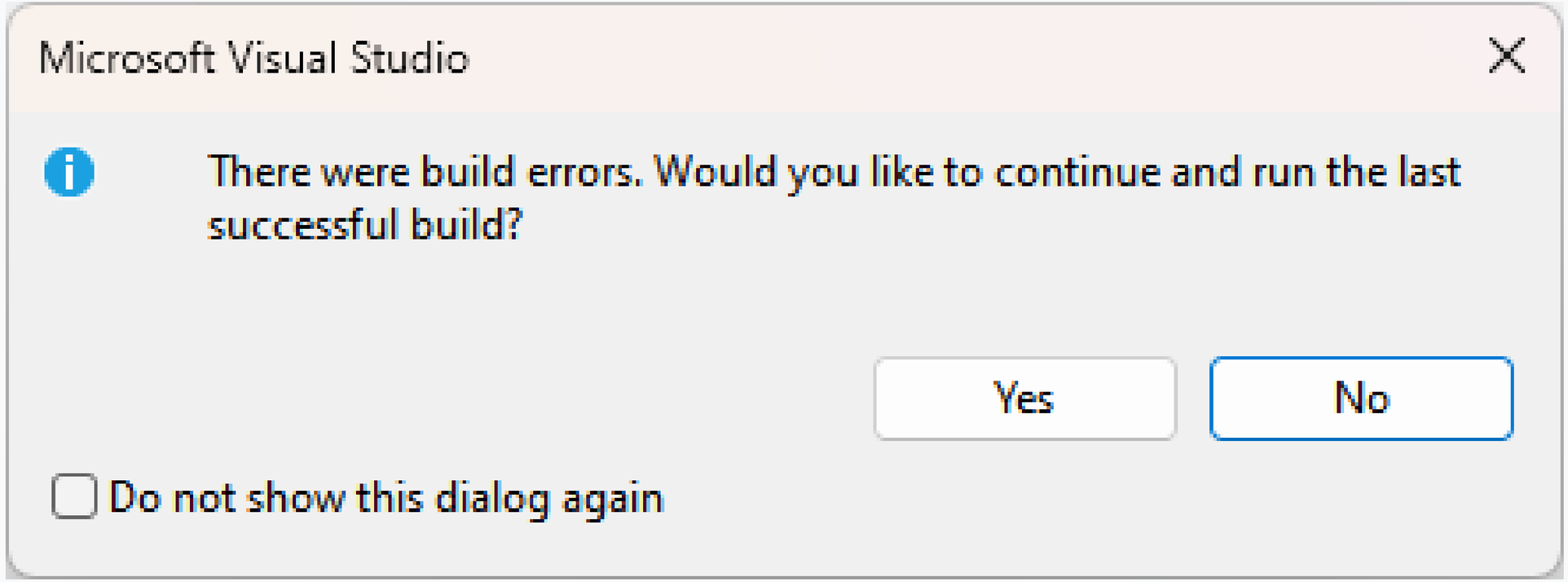
Fouten oplossen

Je code compileert pas als ze **100% foutloos** is. VS helpt:

- Fouten krijgen een **rode squiggly** onderlijning
- De **statusbalk** en de **error list** tonen al je fouten
- Klik op een fout om meteen naar de juiste lijn te springen

> **Tip**
>
> Veel fouten ineens? Kijk **net boven** de eerste fout. Vaak ontbreekt daar gewoon een puntkomma.

De gevaarlijke dialog



> ⚠ **Waarschuwing**
>
> Krijg je deze melding bij het starten? **Klik nooit op “Yes” en vink de checkbox nooit aan.**
>
> Anders draait VS de laatste **werkende** versie, niet je huidige code met fouten. Je zit dan uren naar een “spook” te kijken.

Het lampje

- Zet je cursor op een foutlijn, dan verschijnt soms een **geel lampje**
- Klik erop voor een mogelijke oplossing

Het lampje: brenger der oplossingen.

> **Belangrijk**
>
> Wees **kritisch**: VS is niet alwetend en kan niet altijd raden wat je bedoelt.

Top 5 beginnersfouten

1. **Puntkomma** vergeten
2. **Schrijffouten**, bv. `RaedLine` i.p.v. `ReadLine`
3. **Hoofdlettergevoeligheid**, bv. `Readline` i.p.v. `ReadLine`
4. Per ongeluk een **accolade** verwijderd
5. Code op een **plek waar dat niet mag** (enkel binnen `Main`)

Kleuren in de console

```
1  Console.BackgroundColor = ConsoleColor.Yellow;
2  Console.ForegroundColor = ConsoleColor.Black;
3  Console.WriteLine("Zwart op gele achtergrond");
4  Console.ResetColor();
```

- **`ForegroundColor`** = letterkleur, **`BackgroundColor`** = achtergrond
- Geldt voor alle tekst **hierna**; bestaande tekst verandert niet mee
- **`ResetColor()`** zet alles terug naar de standaard

Belangrijk

Hou het leesbaar en denk aan **daltonisme**: rood op groen las voor sommigen onleesbaar.

Conclusie

- **Programmeren** = een dom maar gehoorzaam toestel exact vertellen wat het moet doen
- Een goed **algoritme** (de juiste instructies in de juiste volgorde) komt vóór de code
- De **compiler** vertaalt je C# naar machinetaal; **.NET** levert het framework
- Met `WriteLine`, `Write` en `ReadLine` doe je in- en uitvoer
- Lees je **fouten**, wees kritisch op het lampje, en klik **nooit** op “Yes” bij die dialog

> **Tip**
>
> Volgende halte: **variabelen en datatypes**. Daar gaan we het woord “variabele” zo vaak gebruiken dat je oren ervan gaan bloeden.

Meer weten

Kennisclips

- Introductie tot C#
- Werken met VS
- Je eerste programma
- WriteLine, Write en ReadLine
- Fouten oplossen
- Kleuren in console

Oefeningen H1 · Quizlet flashcards

Zie verder: andere talen

Dezelfde concepten uit dit hoofdstuk, maar dan in andere talen. Handig om later code in Python of JavaScript te leren lezen.

Zie verder: tekst tonen in Python

Python doet dit zo:

```
1  print("Hello World!")
```

Geen `namespace`, `class` of `Main` en geen puntkomma: Python heeft veel minder ceremonie nodig. Die extra structuur in C# krijgt verderop stuk voor stuk betekenis.

Zie verder: input in Python

Python doet dit zo:

```
1  naam = input("Geef je naam?")
```

De vraag staat meteen in `input`, dus geen aparte `WriteLine` nodig. En je geeft geen type op: Python leidt dat zelf af, terwijl C# vooraf `string` vastlegt en dat laat controleren.

Zie verder: compiled vs interpreted

JavaScript wordt geinterpreteerd: het draait tot het op een fout botst.

```
1 console.log("Dit verschijnt nog wel");
2 consle.log("Pas hier crasht het");
```

De eerste lijn loopt, pas bij de typfout (`consle`) crasht het. C# compileert vooraf en weigert te starten zolang er een fout is: je vangt fouten dus vroeger.

Samenvattende poster

Een handige samenvattende poster (Opgelet: gemaakt m.b.v. ChatGPT Image Gen 2).