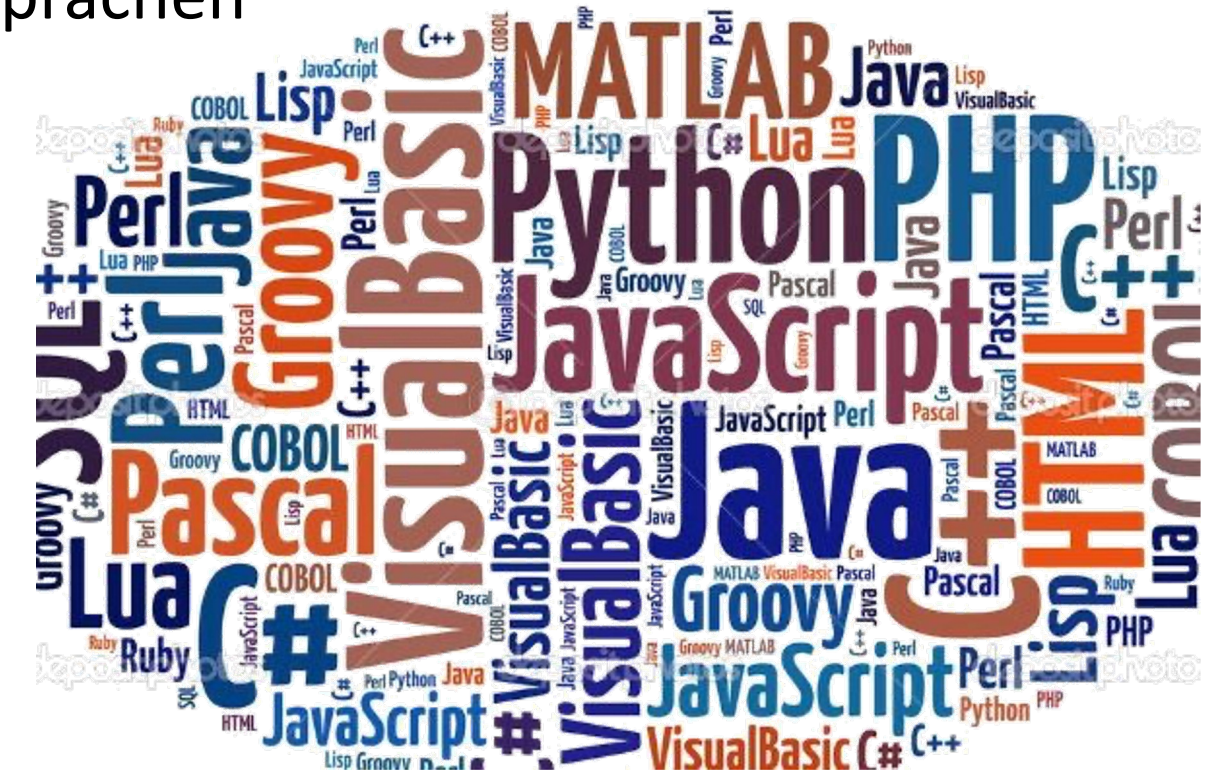
EF IF

Übersicht EF IF

- **Jahr A: Programmierjahr**
  - Hauptziel: Programmierprofi in C# werden
  - Gleich mehr dazu
- **Jahr B: Hardware-Jahr**
  - Grundlagen der Elektrotechnik (im Labor)
  - Mikrokontroller (Arduino) programmieren
  - Website programmieren und auf Webserver hosten
  - uvm.
- Disclaimer: Abweichungen möglich

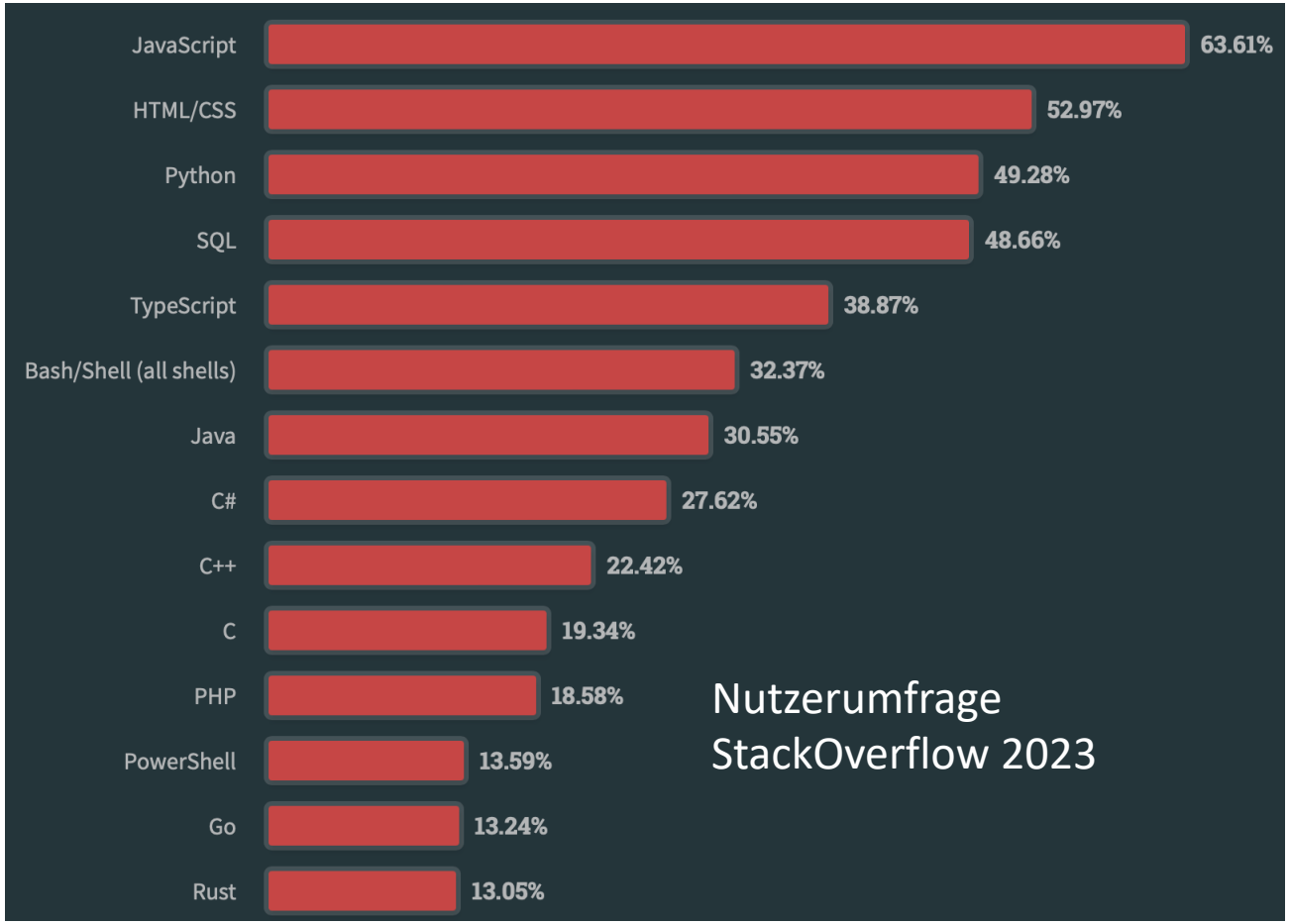
Programmiersprachen

- Gibt knapp 10’000 Programmiersprachen
- Welche sollte man lernen?



Welche Programmiersprache lernen?

1. **Anwendungsbereich?**
   - Wissenschaftliche Arbeiten
   - Websites
   - Games
   - Desktopapplikationen (z.B. Word, Photoshop, …)
   - …
2. **Relevanz heute?**
   - Community
   - Verfügbare Materialien (Tutorials, Videos, …)
3. **Schwierigkeitsgrad?**



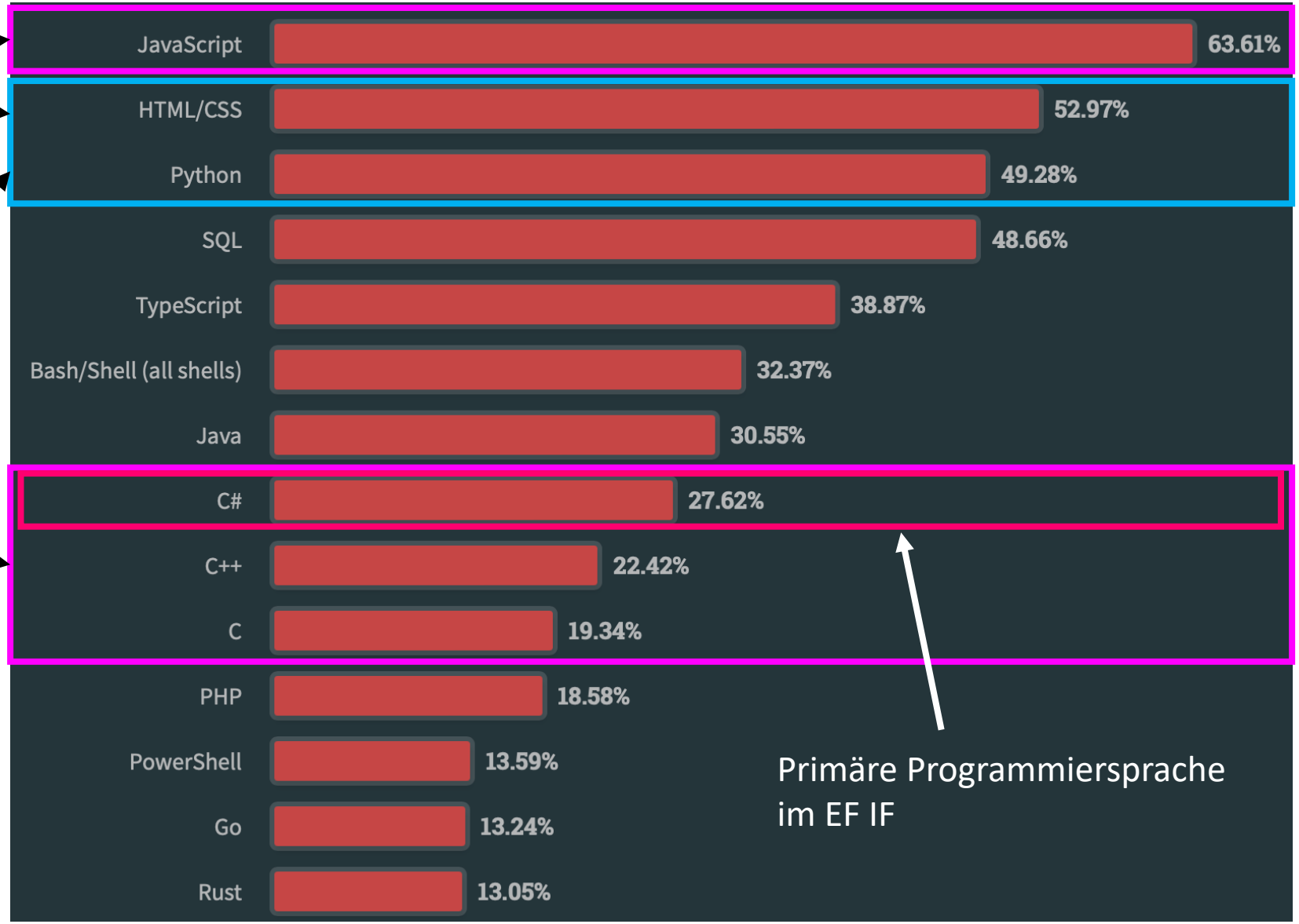
Grundlagenfach Informatik

Ergänzungsfach Informatik

Web

Wissenschaftliche Arbeiten, Algorithmen, AI, Daten

Desktop-Applikationen, Games, Hardware

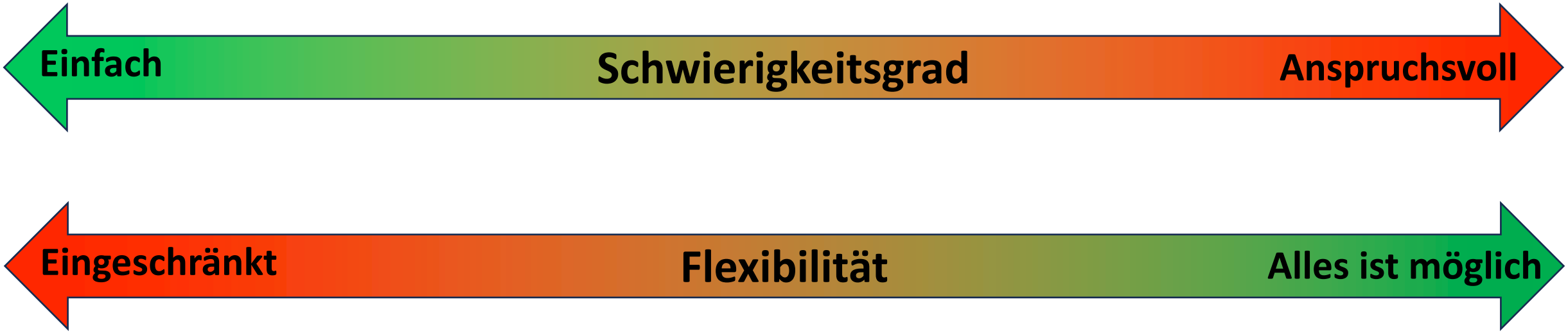


Primäre Programmiersprache im EF IF

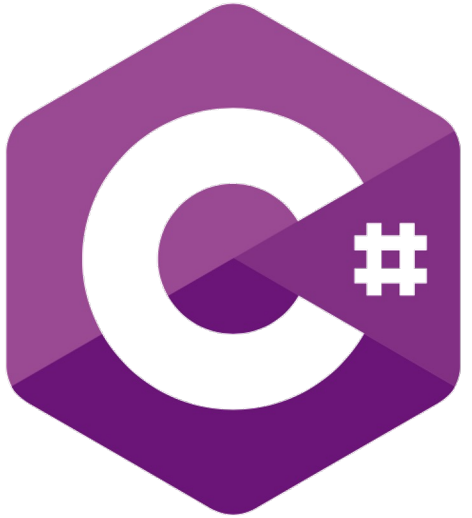
Schwierigkeitsgrad

Python **C#** **C++** **Assemblersprache**

JavaScript **Maschinensprache**



Warum C#?



- C#, sprich «C sharp»
- Gute Ergänzung zu Python aus GF IF
- Viele andere Anwendungsgebiete als Python
- Greift tiefer als Python -> tieferes Verständnis für Programmierung
- Code typischerweise robuster (weniger fehleranfällig)
- Objektorientierte Programmierung lernen

Übersicht Jahr A (Programmierjahr)

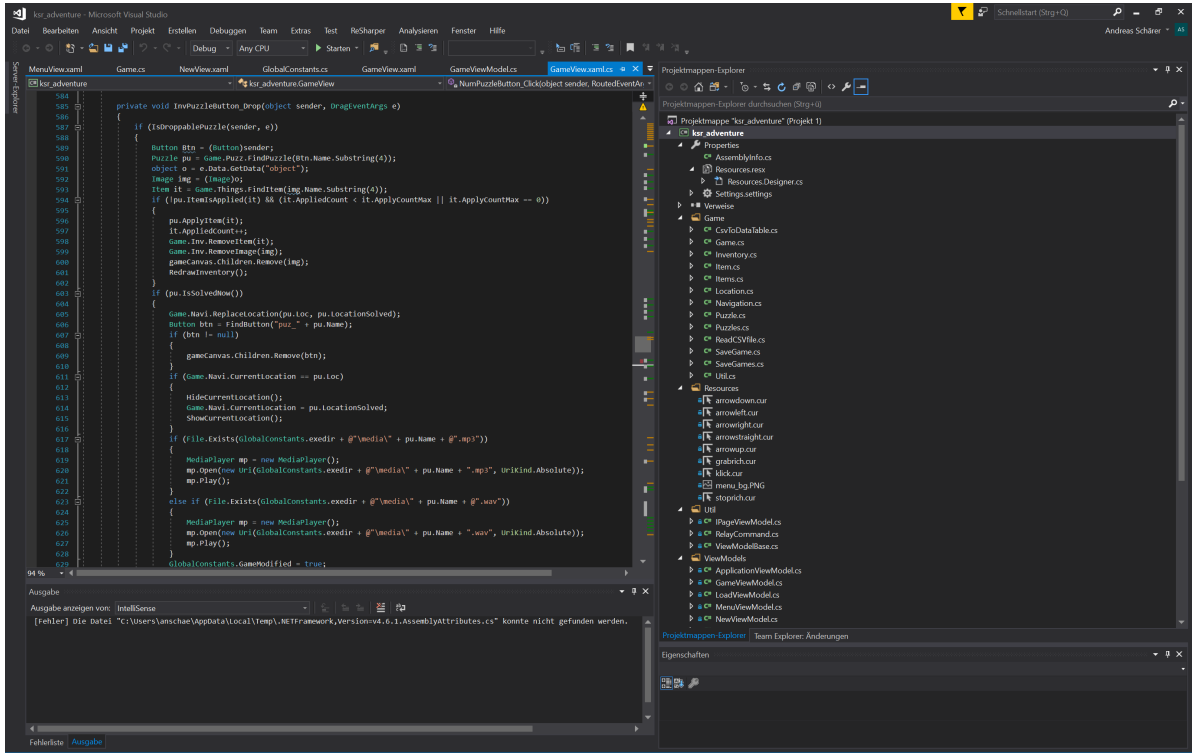
- 1. Semester (HS):
  - 1. Quartal: C# lernen
  - 2. Quartal: C# anwenden, kleine Projekte
- 2. Semester (FS):
  - Noch offen, versch. Möglichkeiten
  - Möglichkeit 1: C# weiter vertiefen
  - Möglichkeit 2: gemeinsames Projekt
  - Möglichkeit 3: …
- Benotung: siehe Wiki

C# lernen schwierig?

- Nein!
- Allgemein gilt: Kann man eine Programmiersprache gut, kann man die meisten anderen in kurzer Zeit lernen
- Grundelemente gibt es in meisten Sprachen:
  - Variablen
  - Listen / Arrays
  - Verzweigungen: if-else
  - Schleifen: while, for
- Kleine Unterschiede im Syntax
- Jede Programmiersprache hat ihre Sonderheiten …
- … und Bibliotheken

Wie C# programmieren?

- IDE **«Visual Studio»**
- Achtung: *nicht* «Visual Studio Code»
- **IDE**: **I**ntegrated **D**evelopment **E**nvironment
  - Code-Editor mit Features
  - Kann Code direkt ausführen
- Nachher: herunterladen & installieren



Datentypen in Python

- Haben mit Python verschiedene **Datentypen** angetroffen:
  - `type(42)` → `<type 'int'>`
  - `type(3.14159)` → `<type 'float'>`
  - `type(True)` → `<type 'bool'>`
  - `type("EF Informatik")` → `<type 'str'>`
  - `type([2,3,5,7,11,13])` → `<type 'list'>`
- Unterschiedliche Datentypen werden unterschiedlich im RAM gespeichert …
- … und benötigen unterschiedlich viel Speicherplatz

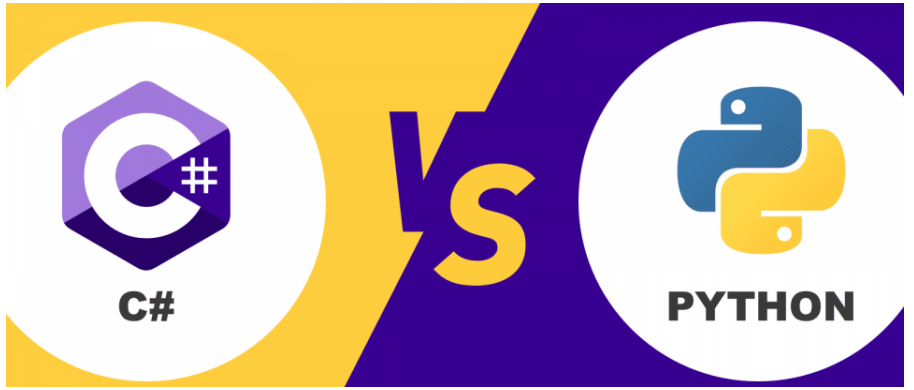
Datentypen in Python

- Python: muss sich nicht gross um versch. Datentypen kümmern
- Python-Code

```
a = 42 # hier ist a ein int
a = 3.14159 # hier wird a zu einem float umdeklariert
a = True # ... und hier zu einem Boolean
a = "And now I am a string!" # ... und hier zu einem String
a = [2,3,5,7,11,13] # ... und jetzt noch zu einer Liste
```

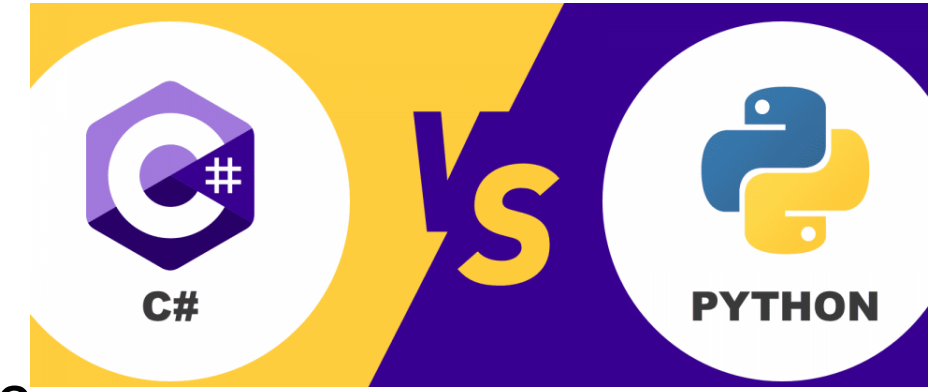
- Variable kann den Datentyp *beliebig ändern*
- Im Hintergrund kümmert sich Python darum, dass richtig im RAM gespeichert wird
- Ist grosse Stärke und Schwäche von Python zugleich:
  - Stärke: einfach, schnell programmieren
  - Schwäche: fehleranfällig

C# vs. Python: Datentypen



- In C# muss Variable zuerst **deklariert** werden:
  - Name festlegen
  - Datentyp festlegen
- Beispiel:
  - Deklaration: `int a;`
  - Variable a kann jetzt Wert zugewiesen werden, aber nur Integer (int):
    ```
    a = 42;
    a = 13;
    ```
  - ~~`a = 3.14159;`~~ -> Fehler! Weil 3.14159 kein int
- Deklaration & Wertzuweisung in einer Zeile möglich: `int b = 7;`
- C# nennt man deshalb **typsichere Programmiersprache**
- Dies macht C#-Code robuster als Python-Code

C# vs. Python: Verschiedenes



- Beispiel Python vs. C#: Countdown 10, 9, … , 0

```python
x = 10

while x >= 0:
    print(x)
    x = x - 1
```
Python

```csharp
int x = 10;

while (x >= 0)
{
    Console.WriteLine(x);
    x = x - 1;
}
```
C#

Deklaration notwendig

Runde Klammern, kein Doppelpunkt

Ausgabe in Konsole

Geschwungene Klammern anstelle Einrückungen

Einrückungen irrelevant, machen aber Code lesbar

Codezeilen mit Semikolon ; abschliessen

Auftrag

- Siehe zusammen auf Wiki