

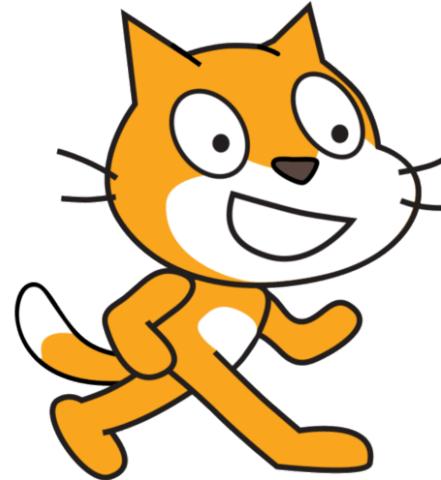
Die beste Einstiegsprogrammiersprache

ALEXANDER DOBRYNIN



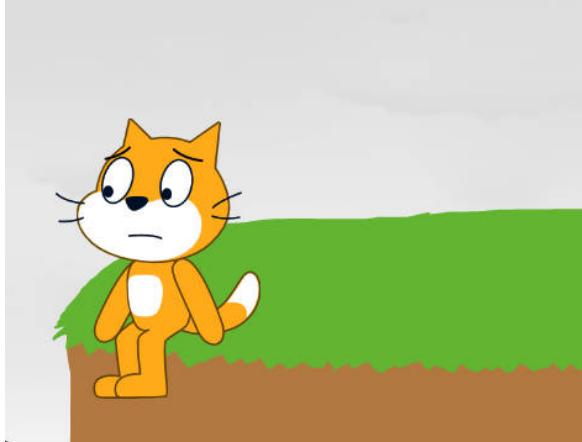
Educational Languages

- Wurden Explizit zum Lernen entwickelt
- Pädagogisch hohes Potenzial
- Erkenntnissen aus der Forschung fließen in die Entwicklung mit ein
- Die Programmiersprache **Scratch** hat die höchste Verbreitung
- Empirische Studien bestätigen den Erfolg von block-basierten Sprachen bei Programmieranfängern [1, 2, 3]
- Sind vor allem in Schulen verbreitet



Educational Languages

- Diese Sprachen werden verhältnismäßig wenig in Hochschulen eingesetzt [4]
- Keine Industrie Relevanz [5, 6]
- Industrie, Studierende und Eltern üben Druck auf die Hochschulen aus und beeinflussen damit die Wahl [6]
- Kein "richtiges" Programmieren, nichts für ältere Studierende und Feature limitiert [7]



Industrial Languages > Educational Languages

- Industrie Sprachen richten sich an professionelle Entwickler und nicht an Anfänger [8]
- Studien zeigen, dass Lehrende bei der Wahl der ersten Programmiersprache eher zu "Industrie Relevanz" als zu "pädagogische Vorteile" tendieren [9, 10, 11, 12]
- Das ist gefährlich, denn Pädagogik sollte an erster Stelle stehen [13]
- Daher ist die Frage nach der *besten Einstiegsprogrammiersprache* wichtig
- Die Entscheidung sollte auf Grundlage **evidenzbasierter Kriterien** getroffen werden [10]

The Programming Language Wars [14]

- Pseudowissenschaftliche Diskussionen schaden der Welt der Programmiersprachen
- Appell an einen Diskurs, wo Aussagen und Entscheidungen auf wissenschaftlicher Evidenz basieren
- Mehr Forschung die untersucht, ob z.B. bestimmte Sprachfeatures einen messbaren Effekt haben (kontrollierte Experimente)
- Programmiersprachen sind per se nicht gut oder schlecht, sondern für bestimmte Zwecke geeignet [15]

RESEARCH ARTICLE

X in 🐣 f 📧

The Programming Language Wars: Questions and Responsibilities for the Programming Language Community

Authors:  Andreas Stefik,  Stefan Hanenberg [Authors Info & Claims](#)

OnwardI 2014: Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software • October 2014 • Pages 283–299 • <https://doi.org/10.1145/2661136.2661156>

Published: 14 October 2014 [Publication History](#)

 Check for updates

44 1,875

ABSTRACT

The discipline of computer science has a long and complicated history with computer programming languages. Historically, inventors have created language products for a wide variety of reasons, from attempts at making domain specific tasks easier or technical achievements, to economic, social, or political reasons. As a consequence, the modern programming language industry now has a large variety of incompatible programming languages, each of which with unique syntax, semantics, toolsets, and often their own standard libraries, lifetimes, and costs. In this paper, we suggest that the programming language wars, a term which describes the broad divergence and impact of language designs, including often pseudo-scientific claims made that

Populäre Sprachen in der Wirtschaft

Populäre Sprachen in der Wirtschaft

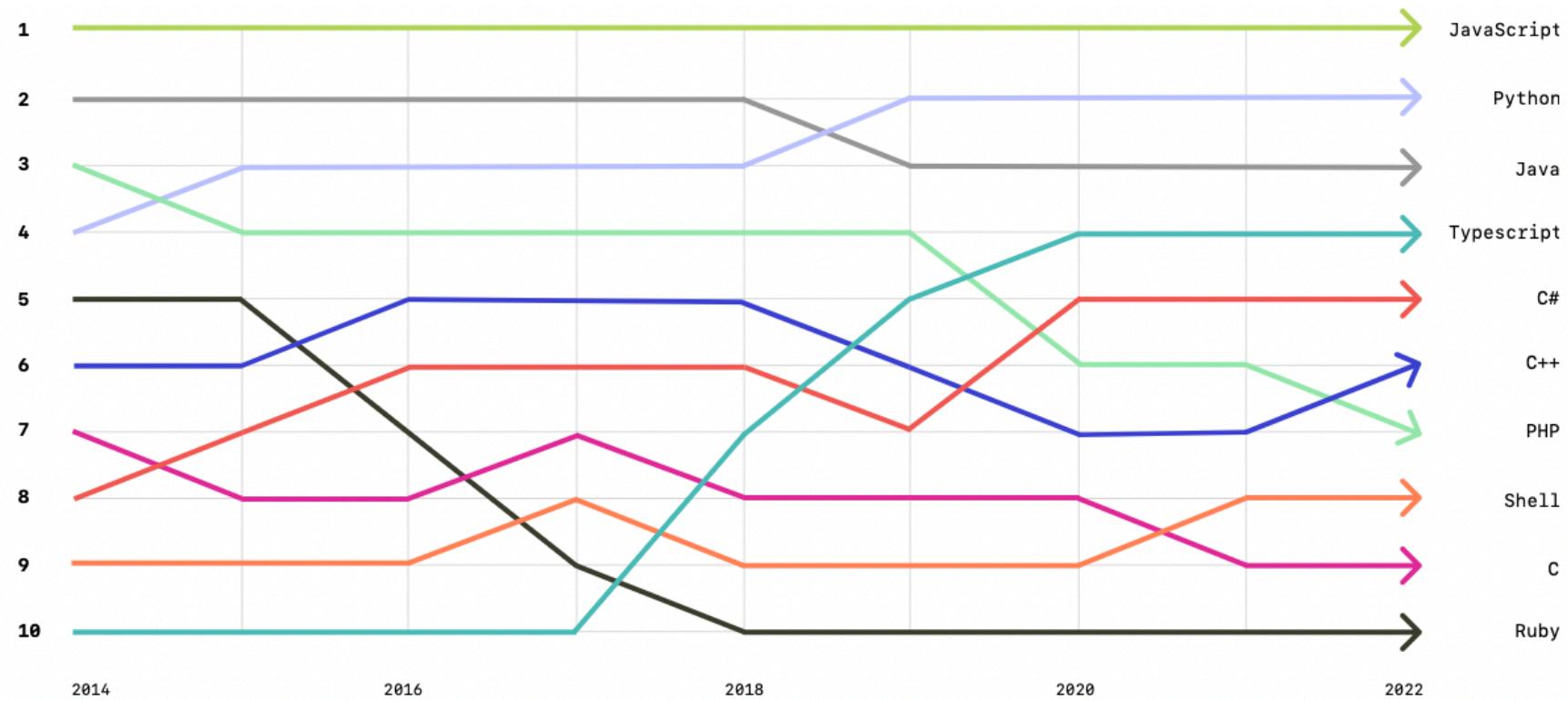
Stack Overflow Developer Survey: Die meist benutzten Programmiersprachen (2014 - 2023)* [16]

Platzierung	Programmiersprache	Bemerkung
1	JavaScript	Durchgehend auf #1
2	Python	Hat Java überholt
3	Java	Lange Zeit Konstant. Die letzten Jahre gesunken
4	C#	Mit der Zeit an Popularität verloren
5	C++	Mit der Zeit an Popularität verloren

*Vereinfachte Darstellung

Populäre Sprachen in der Wirtschaft

Octoverse: Dominante Programmiersprachen auf GitHub (2014 - 2022) [17]



Populäre Sprachen in der Wirtschaft

DevJobsScanner: Analyse der meistgefragten Programmiersprachen nach Job Beschreibung (2022 - 2023) [19]

Zusammenfassung

Industrie

1. JavaScript
2. Python
3. Java

Populäre Sprachen an Hochschulen

Populäre Sprachen an Hochschulen

Kummulierte und simplifizierte Auswertung von Umfragen (2017, 2018, 2019) oder der Analyse des Curriculums (2016, 2018, 2021) in USA [4, 22], Europa [8], Ireland [11], UK [12], Australasien [20] und Griechenland [21] von eingesetzten Programmiersprachen in CS1 (Introductory Programming)

Platzierung	Programmiersprache	Bemerkung
1	Java	In 5/7 Studien auf #1. Im Schnitt doppelter Abstand zu #2
2	Python	In 3/7 Studien auf #2
3	C++	In 2/7 Studien auf #3 und #2
4	C	In 2/7 Studien auf #1 (Europa und Griechenland)

Populäre Sprachen an Hochschulen

Gründe für die Wahl [4, 20, 11, 12]

Platzierung	Grund
1	Relevanz für Industrie
2	Pädagogische Gründe
3	Verfügbarkeit / Kosten für Studierende

- Der Abstand zwischen #1 und #2 ist klein
- Andere Gründe: Plattformunabhängigkeit, online Community / Hilfe, OO-Sprache, Verfügbarkeit von Bibliotheken / Literatur etc.

Populäre Sprachen an Hochschulen

Wahrgenommene Schwierigkeit für Studierende vs. wahrgenommene Nützlichkeit um Programmierkonzepte zu lehren

- C++ ist gleichermaßen schwierig und pädagogisch ungeeignet [20, 11]
- Java ist im Mittelfeld [11, 12]
- Python ist am einfachsten zu erlernen [20] und hat eine gute Ratio [12]



Platzierung	Programmiersprache
1	Java
2	Python
3	C++

main ↴ aoc-2022 / src / day07.rs ↴

alexlobry clippy cleanup

Code Blame 120 lines (111 loc) · 3.21 KB

```
1 use std::collections::HashMap;
2 use std::fs;
3
4 type Path = Vec<String>;
5
6 #[derive(Debug)]
7 enum FileOrDir {
8     File(i32),
9     Dir(String),
10 }
11
12 type FS = HashMap<Path, Vec<FileOrDir>>;
13
14 #[derive(Debug)]
15 enum Instruction {
16     CdUp,
17     CdDown(String),
18     Ls(Vec<FileOrDir>),
19 }
20
21 fn read_input() -> Vec<Instruction> {
22     let file_path = "inputs/day07.txt";
23     let input = fs::read_to_string(file_path).expect("Should have been able to read the file");
24     let mut cmds = input.split('$');
25     let mut instructions = vec![];
26     cmds.next();
27     for cmd in cmds {
28         let cmd = cmd.trim();
29         if let Some(dir) = cmd.strip_prefix("cd ") {
30             if dir == ".." {
31                 instructions.push(Instruction::CdUp);
32             } else {
33                 instructions.push(Instruction::CdDown(dir.to_string()));
34             }
35         } else if let Some(listing) = cmd.strip_prefix("ls\n") {
36             let mut entries = vec![];
37             for line in listing.lines() {
38                 let (lhs, rhs) = line.split_once(' ').unwrap();
39                 if lhs == "dir" {
40                     entries.push(FileOrDir::Dir(rhs.to_string()));
41                 } else {
42                     entries.push(FileOrDir::File(i32::from_str(rhs).unwrap()));
43                 }
44             }
45             instructions.push(Instruction::Ls(entries));
46         }
47     }
48     instructions
49 }
```

➤ `typeof NaN` ➤ `true==1`
↳ `"number"` ↳ `true`

➤ `9999999999999999` ➤ `true==1`
↳ `10000000000000000` ↳ `false`

➤ `0.5+0.1==0.6` ➤ `(!+[]+[]+![]).length`
↳ `true` ↳ `9`

➤ `0.1+0.2==0.3` ➤ `9+"1"`
↳ `false` ↳ `"91"`

➤ `Math.max()` ➤ `91-"1"`
↳ `-Infinity` ↳ `90`

➤ `Math.min()` ➤ `[]==0`
↳ `Infinity` ↳ `true`

➤ `[]+[]`
↳ `""`

➤ `[]+{}`
↳ `"[object Object]"`

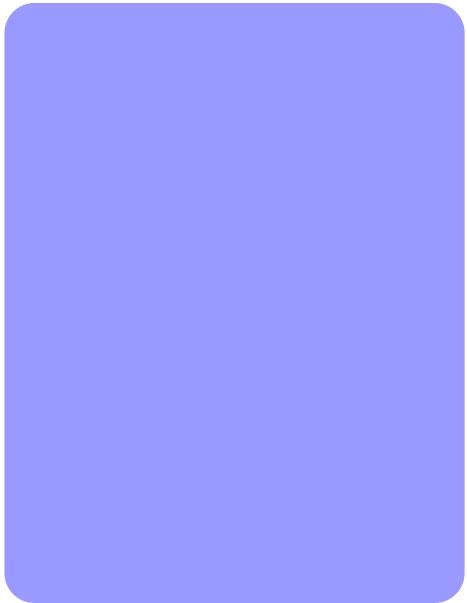
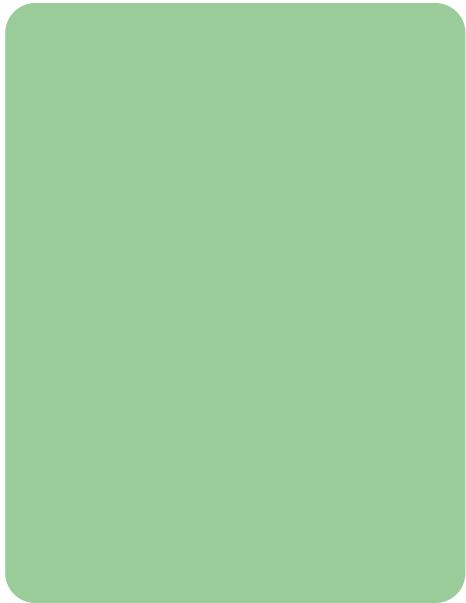
➤ `{ }+[]`
↳ `0`

➤ `true+true+true==3`
↳ `true`

➤ `true-true`
↳ `0`



Thanks for inventing Javascript



Curriculum

Gibt es eine empirisch fundierte Grundlage für ein Curriculum, in dem Hinweise oder Empfehlungen für Programmiersprachen gegeben werden?
Ja und Nein

CS Curriculum 2023

[23]

- Macht u.a. Vorschläge zu Learning Outcomes, Inhalten und Gewichtungen
- Die Vorschläge werden von internationalen Experten aus Akademikern (n=427) und Industrievertretern (n=865) überarbeitet
- Interaktiver Prozess. Kommunikation und Feedback auf der SIGCSE Konferenz und über SIG Mailing Listen
- Wird in wissenschaftlichen Publikationen zitiert [7, 11, 9]

Computer Science Curricula 2023

January 2024

The Joint Task Force on Computer Science Curricula

Association for Computing Machinery (ACM)

IEEE-Computer Society (IEEE-CS)

Association for the Advancement of Artificial Intelligence (AAAI)



Association for
Computing Machinery



IEEE COMPUTER
SOCIETY



Association for the
Advancement of
Artificial Intelligence

Core Hours

Knowledge Unit	CS Core	KA Core
Object-Oriented Programming	4 + 1 (SDF)	1
Functional Programming	4	3
Logic Programming		2 + 1 (MSF)
Shell Scripting	2	
Event-Driven and Reactive Programming	2	2
Parallel and Distributed Computing	2 + 1 (PDC)	2
Aspect-Oriented Programming		
Type Systems	3	3
Systems Execution and Memory Model	2 + 1 (AR and OS)	

FPL-OOP: Object-Oriented Programming

CS Core:

1. Imperative programming as a subset of object-oriented programming.
2. Object-oriented design:
 - a. Decomposition into objects carrying state and having behavior.
 - b. Class-hierarchy design for modeling.
3. Definition of classes: fields, methods, and constructors. (See also: [SDF-Fundamentals](#))
4. Subclasses, inheritance (including multiple inheritance), and method overriding.
5. Dynamic dispatch: definition of method-call.
6. Exception handling. (See also: [SDF-Fundamentals](#), [PDC-Coordination](#), [SE-Construction](#))
7. Object-oriented idioms for encapsulation:
 - a. Privacy, data hiding, and visibility of class members.
 - b. Interfaces revealing only method signatures.
 - c. Abstract base classes, traits and mixins.
8. Dynamic vs static properties.
9. Composition vs inheritance.
10. Subtyping:
 - a. Subtype polymorphism; implicit upcasts in typed languages.
 - b. Notion of behavioral replacement: subtypes acting like supertype.
 - c. Relationship between subtyping and inheritance.

Software Development Fundamentals (SDF)

Preamble

Fluency in the process of software development is fundamental to the study of computer science. To use computers to solve problems most effectively, students must be competent at reading and writing programs. Beyond programming skills, however, they must be able to select and use appropriate data structures and algorithms and use modern development and testing tools.

The SDF knowledge area brings together fundamental concepts and skills related to software development, focusing on concepts and skills that should be taught early in a computer science program, typically in the first year. This includes fundamental programming concepts and their effective use in writing programs, use of fundamental data structures which may be provided by the programming language, basics of programming practices for writing good quality programs, reading, and understanding programs, and some understanding of the impact of algorithms on the performance of the programs. The 43 hours of material in this knowledge area may be augmented with core material from other knowledge areas as students progress to mid- and upper-level courses.

This knowledge area assumes a contemporary programming language with built-in support for common data types including associative data types like dictionaries/maps as the vehicle for introducing students to programming (e.g., Python, Java). However, this is not to discourage the use of older or lower-level languages for SDF – the knowledge units below can be suitably adapted for the actual language used.

Curriculum

CS1

CS1 (Introductory Programming)

In der wissenschaftlichen Literatur diskutiert man seit 1970, welche Einstiegssprache geeignet ist [20, 10], allerdings ist man sich in 3 Dingen sehr einig:

1. Die Ziele von CS1 sind das Erlernen von **Problem Solving** und **fundamentaler Programmierkonzepte**, aber nicht das Lernen einer spezifischen Programmiersprache [5, 8, 9, 13, 24]. Wird von Umfrageergebnissen bestätigt [11, 12, 20]
2. Die Wahl der ersten Sprache prägt nicht nur den Erfolg von 1., sondern hat auch einen Einfluss auf die weitere Programmierausbildung [6, 8, 21]
3. Da verschiedene Faktoren die Wahl beeinflussen und Sprachen sich weiterentwickeln, sollte die Wahl regelmäßig überdacht werden [6, 20]. Zudem sollten die Kurse sich regelmäßig an die neue Generation von Studierenden anpassen, die mit neuen Technologien aufwachsen [8]

Curriculum

CS1

Kriterien

Quellen für Kriterien

1. Anforderungen an eine Programmiersprache (Eigenschaften) aus den empfohlenen Inhalten des CS Curriculum 2023 herleiten [23]
2. Studien über häufige Fehler von Studierenden beim Programmieren [25, 26, 27]
3. Design Entscheidung von *Educational Language Designer* [28]
4. Arbeiten, die Kriterien aus theoretischen Vorüberlegungen oder empirischen Daten aufstellen [4, 5, 6, 9, 10, 28, 29, 30, 31, 32]
5. Gründe für die Wahl einer Einstiegssprache [4, 11, 12, 20]

Kriterien

- 1. Größe
- 2. Popularität
- 3. Syntax
- 4. Semantik
- 5. Typisierung und Sicherheit
- 6. Verletzung von Erwartungen
- 7. Fehlermeldungen
- 8. Programmiergewohnheiten
- 9. Transition
- 10. Kompatibilität
- 11. Garbage Collection
- 12. Entwicklungsumgebung
- 13. Support und Verfügbarkeit
- 14. Unterstützung im Lernprozess

Größe

So groß wie nötig (lehrbar in CS1) aber so klein wie möglich [29]

- Großteil der Sprache wird nicht genutzt, obwohl dieser in Textbüchern oder Beispielen existiert [29, 30]
- Für alles was wir in der Sprache ausdrücken wollen, sollte es nur einen Weg geben [30]
- Psychologisch gesehen ist es wichtig, dass man eine Sprache vollständig gelernt hat [30]

$$\begin{array}{c}x \\ \lambda x. \, M \\ (M \, N)\end{array}$$

Syntax

Übereinstimmender Appell an lesbare und unkomplizierte Syntax [5, 9, 10, 24, 28, 30, 31, 32]

- Verständnis von Syntax steht dem Verständnis von Programmieren im Wege [8, 24, 33]
- Syntaktische Synonyme (Unterschiedliche Syntax für gleiche Semantik) vermeiden [29]
- Syntaktische Homonyme (Gleiche Syntax für das unterschiedliche Semantik, kontextabhängig) vermeiden [29]
- Codeblöcke mit Schlüsselwörter trennen (z.B. loop ... end loop) [5, 29], Schlüsselwörter anstelle von Symbolen [15], Code liest sich wie englischer Text [28, 31], etc.

```
// Java

import java.util.Scanner;

public class App {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Erste Zahl:");
        int x = scanner.nextInt();

        System.out.println("Zweite Zahl:");
        int y = scanner.nextInt();

        int sum = x + y;
        System.out.println("Ergebnis der Addition: " + sum);

        scanner.close();
    }
}
```

```
# Python

x = int(input("Erste Zahl:"))
y = int(input("Zweite Zahl:"))

sum = x + y
print("Ergebnis der Addition: ", sum)
```

```
// Java 21

import java.util.Scanner;

void main() {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Erste Zahl:");
    int x = scanner.nextInt();

    System.out.println("Zweite Zahl:");
    int y = scanner.nextInt();

    int sum = x + y;
    System.out.println("Ergebnis der Addition: " + sum);

    scanner.close();
}
```

```
# Python

x = int(input("Erste Zahl:"))
y = int(input("Zweite Zahl:"))

sum = x + y
print("Ergebnis der Addition: ", sum)
```

Verletzung von Erwartungen

Diskrepanz zwischen Erwartung (educated guess) und Realität [15, 29]

```
// C
#include<stdio.h>

main(void) {
    int x,y;
    if (x=5 && -10<y<10) {
        printf("ok");
    }
}
```

- `x=5`
 - Erwartung: Überprüfung auf Gleichheit
 - Realität: Wertet **immer** zu true aus, **egal** welchen Wert `x` hat
- `-10<y<10`
 - Erwartung: Überprüfung, ob `y` zwischen -10 und 10 liegt (Rangecheck)
 - Realität: Wertet **immer** zu true aus, **egal** welchen Wert `y` hat

Auswertung der Sprachen

- 1. Größe
- 2. Popularität
- 3. Syntax
- 4. Semantik
- 5. Typisierung und Sicherheit
- 6. Verletzung von Erwartungen
- 7. Fehlermeldungen
- 8. Programmiergewohnheiten
- 9. Transition
- 10. Kompatibilität
- 11. Garbage Collection
- 12. Entwicklungsumgebung
- 13. Support und Verfügbarkeit
- 14. Unterstützung im Lernprozess

Platzierung

Programmiersprache

1

Java

2

Python

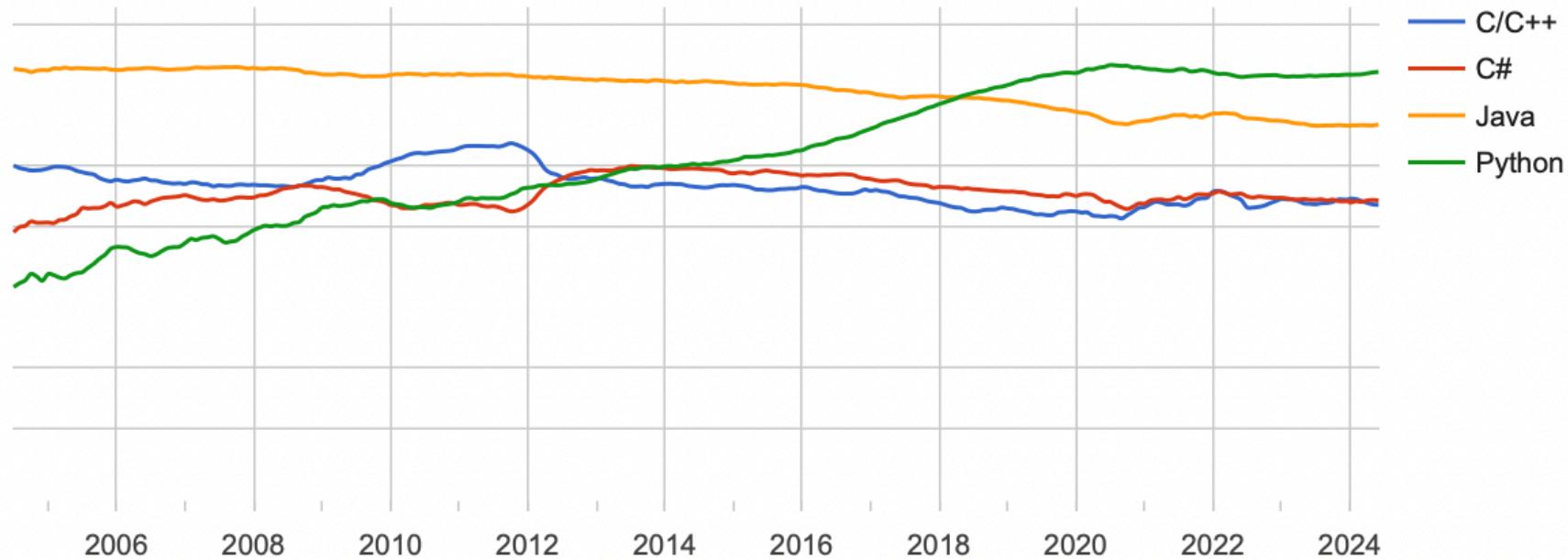
Auswertung der Sprachen

- C++
 - C++ ist für die Lehre eine schlechte Sprache [10, 15] → C++ ist schlecht
 - C++ auf Platz 7 / 9 [5]
- Java
 - Java ist ungeeignet [10, 31, 34]
 - Java ist OK, aber auch nicht gut [15] → Java ist OK
 - Java ist OK [28]
 - Java auf Platz 1 / 9 [5]
- Python
 - Python ist OK, aber auch nicht die Lösung [10]
 - Python hat im Vergleich am Besten abgeschnitten [28] → Python ist gut?
 - Python ist (signifikant) besser als Java [9]
 - Python auf Platz 2 / 9 [5]

Sollten wir Python machen?

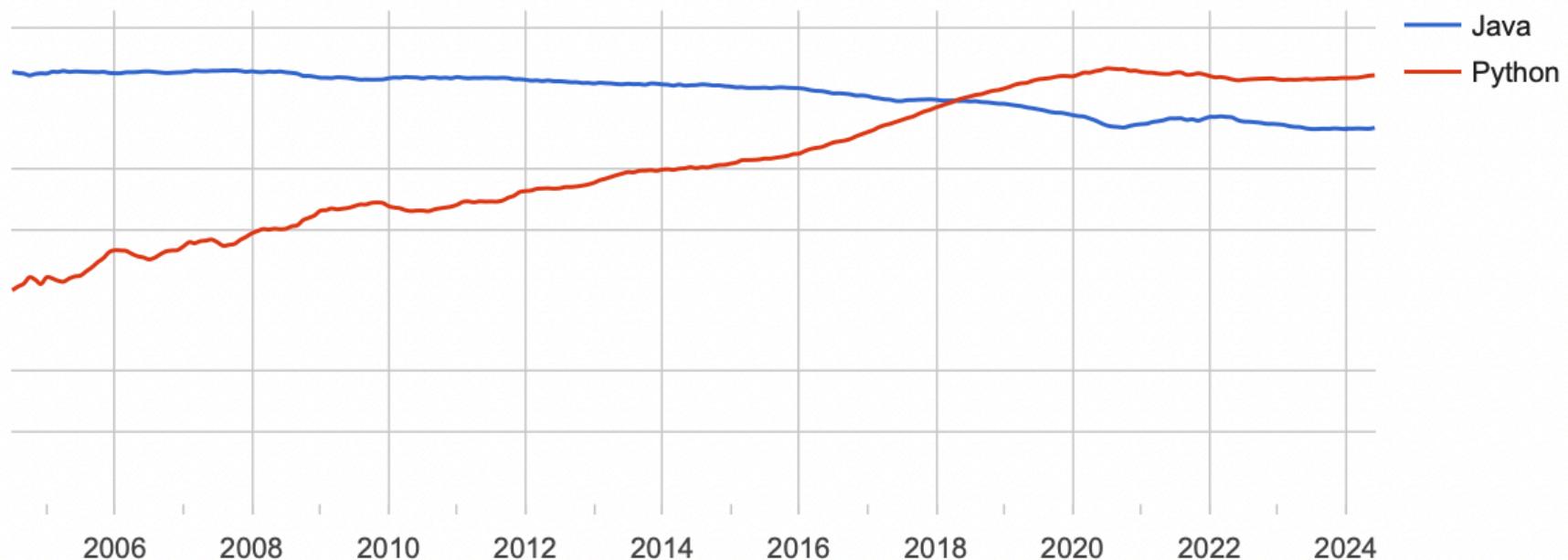
Trends von Sprachen

PYPL Index: Popularität von Programmiersprachen gemessen an Google Suche nach Tutorials
(2006 - 2024) [18]



Trends von Sprachen

PYPL Index: Popularität von Programmiersprachen gemessen an Google Suche nach Tutorials
(2006 - 2024) [18]



Trends von Sprachen in CS1 (USA)

Modifiziert übernommen aus [22]

Language	2011	2015	2019	Trend
C	18	21	17	
C++	85	76	69	
Java	193	179	159	
Python	41	75	107	

Sprachen in CS2 (USA)

Modifiziert übernommen aus [22]

Language	Number of schools (n = 353)	Fraction
Java	196	55 %
C++	90	25 %
Python	22	6 %
C	15	
Other	30	

Übergang von CS1 zu CS2 (USA)

Modifiziert übernommen aus [22]

CS1	CS2	Amount	Fraction
Java	Java	122	87 %
	Python	1	0,7 %
Python	Java	54	61 %
	Python	18	20 %
	C++	13	15 %

Müssen wir Python machen?

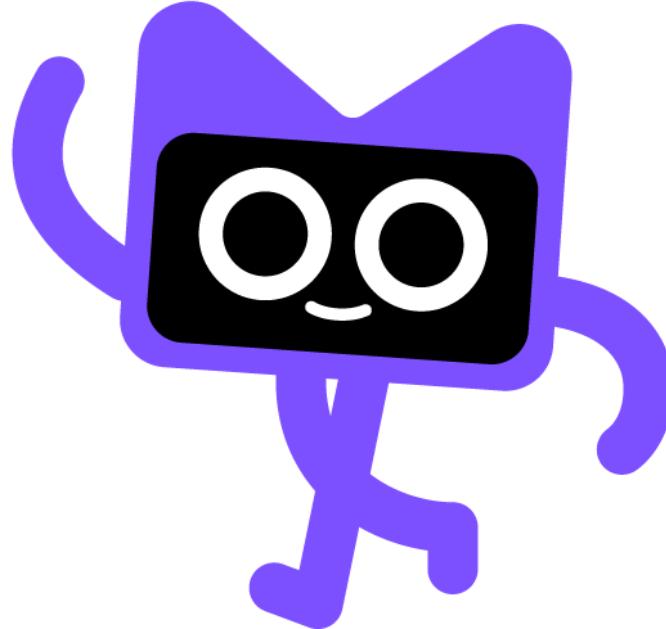
Gründe gegen Python (keine Sorge CK)

- Python ist nicht statisch getypt und hat keinen Compiler, obwohl beides als sehr wichtig erachtet wird [5, 10, 14, 15, 28, 30]
- Dynamische Typisierung ist schwieriger für Anfänger [4]
- Python erlaubt zwar Typ-Annotationen [35] und es gibt Type Checker [36, 37], allerdings verletzt das zum einen das Kernargument der "*simplicity*" und zum anderen nimmt man etwas ernster, wenn die Sprache es fest eingebaut hat, als wenn man sich auf Best Practice einigt [15]
- Python führt automatische Typ-Umwandlungen durch, die für Anfänger schwierig zu erkennen sind und zu unerwarteten Fehlern führen können [5]. Vor allem bei If-Abfragen.
- Start und End-Schlüsselwörter sind für Anfänger geeigneter als Klammern [5, 29]. Keine Studien gefunden, die etwas über Einrückungen sagen*

*Bython: Python with braces. Because Python is awesome, but whitespace is awful [38]

Fazit

- Die verbreitetsten Sprachen in Hochschulen sind aktuell C/C++, Java und Python
- Java ist seit mindestens einer Dekade die populärste Einstiegssprache
- Studien aus den USA, UK und Australasien geben Hinweise darauf, dass sich ein Trend zu Python entwickelt [20, 22]
- Python Popularität begründet sich hauptsächlich durch seine Einfachheit^[4], allerdings ist Python weit weg vom Ideal^[10, Folie von davor]
- Die Wahl einer Sprache ist ein Kompromiss zwischen verschiedenen Faktoren wie Ökologie, Politik und Pädagogik und hängt von Prioritäten und Zielen der Einrichtung und des Lehrenden ab [29]
- Einige Autoren sind der Meinung, dass die Qualität des Kurses und des Lehrenden^[6] und die Art wie gelehrt wird^[28] wichtiger als die Wahl der Einstiegssprache ist
- C taucht vermehrt dort auf, wo "Engineering" im Namen des Studiengangs oder der Fakultät auftaucht^[4]



Wo Kotlin?



Warum Kotlin voll gut ist

oder zumindest besser als Java

DISSERTATION

ausgearbeitet von

Alexander Dobrynin

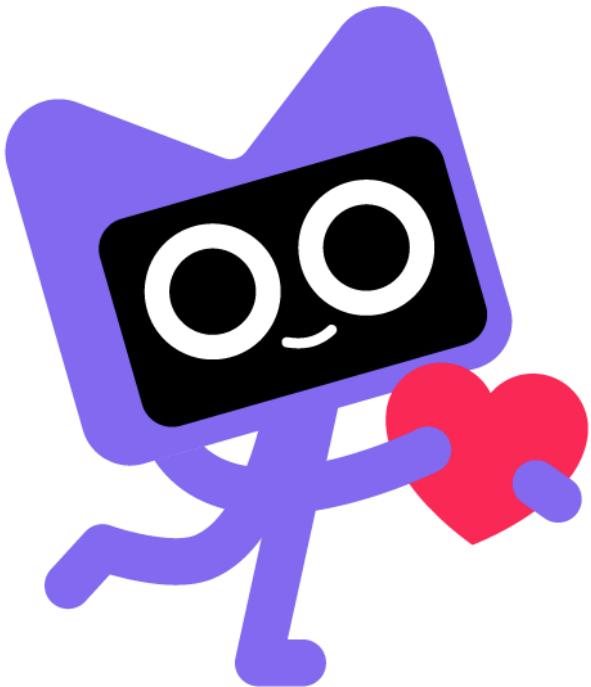
zur Erlangung des akademischen Grades

DR. RER. NAT.

vorgelegt beim

PROMOTIONS KOLLEG NRW
ABTEILUNG MEDIEN UND INTERAKTION

Erster Betreuer/in: Prof. Dr. Christian Kohls
Technische Hochschule Köln





DIGITAL XCHANGE
powered by Innovation Hub Bergisches RheinLand

Danke

<https://github.com/alexdobry/talks/blob/main/dix24.pdf>

Quellen

- [1] To block or not to block, that is the question: students' perceptions of blocks-based programming. <https://dl.acm.org/doi/10.1145/2771839.2771860>
- [2] Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. <https://dl.acm.org/doi/10.1145/3089799>
- [3] Using Learning Analytics in Understanding Students' Behavior in Block-based and Text-based Programming Modality. <https://ieeexplore.ieee.org/document/9694021>
- [4] What Language? - The Choice of an Introductory Programming Language.
<https://ieeexplore.ieee.org/abstract/document/8658592>
- [5] An Evaluation Framework and Comparative Analysis of the Widely Used First Programming Languages. <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0088941>
- [6] First Programming Language in Introductory Programming Courses, Role of.
https://link.springer.com/referenceworkentry/10.1007/978-3-319-60013-0_217-1

Quellen

- [7] Block-Based Object-Oriented Programming. <https://ieeexplore.ieee.org/document/9829246>
- [8] Introductory Programming Subject in European Higher Education.
<https://infedu.vu.lt/journal/INFEDU/article/371/info>
- [9] Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches.
<https://dl.acm.org/doi/10.1145/2662412>
- [10] Some thoughts on teaching introductory programming and the first language dilemma (Discussion Paper). <https://dl.acm.org/doi/10.1145/3631802.3631812>
- [11] A Survey of Introductory Programming Courses in Ireland.
<https://dl.acm.org/doi/10.1145/3304221.3319752>
- [12] An Analysis of Introductory Programming Courses at UK Universities.
<https://arxiv.org/abs/1609.06622>

Quellen

- [13] Java as first programming language: a critical evaluation.
<https://dl.acm.org/doi/abs/10.1145/292422.292440>
- [14] The Programming Language Wars: Questions and Responsibilities for the Programming Language Community. <https://dl.acm.org/doi/10.1145/2661136.2661156>
- [15] The Problem of Teaching Object-Oriented Programming, Part I: Languages.
https://www.researchgate.net/publication/220278819_The_Problem_of_Teaching_Object-Oriented_Programming_Part_I_Languages
- [16] Stack Overflow Developer Survey 2023. <https://survey.stackoverflow.co>
- [17] Octoverse: The top programming languages. <https://octoverse.github.com/2022/top-programming-languages>
- [18] PYPL PopularitY of Programming Language. <https://pypl.github.io/PYPL.html>

Quellen

- [19] Most Demanded Programming Languages. <https://www.devjobsscanner.com/blog/top-8-most-demanded-programming-languages/>
- [20] Language Choice in Introductory Programming Courses at Australasian and UK Universities. <https://dl.acm.org/doi/10.1145/3159450.3159547>
- [21] Introduction to computing: a survey of courses in Greek higher education institutions. <https://dl.acm.org/doi/10.1145/3291533.3291549>
- [22] Trends Of Commonly Used Programming Languages in CS1 And CS2 Learning. <https://ieeexplore.ieee.org/document/9569444>
- [23] Computer Science Curricula 2023. <https://ieeecs-media.computer.org/media/education/reports/CS2023.pdf>
- [24] A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. <https://ieeexplore.ieee.org/document/8447543>

Quellen

- [25] Blackbox Data Collection Project. <https://bluej.org/blackbox/>
- [26] Novice Use of the Java Programming Language. <https://dl.acm.org/doi/10.1145/3551393>
- [27] 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. <https://dl.acm.org/doi/10.1145/2676723.2677258>
- [28] An objective comparison of languages for teaching introductory programming.
<https://dl.acm.org/doi/abs/10.1145/1315803.1315811>
- [29] Seven deadly sins of introductory programming language design.
<https://ieeexplore.ieee.org/abstract/document/534015>
- [30] The Problem of Teaching Object-Oriented Programming, Part 1: Languages.
https://kar.kent.ac.uk/21879/2/the_problem_of_teaching_object-oriented_kolling_1.pdf
- [31] Java as first programming language: a critical evaluation.
<https://dl.acm.org/doi/abs/10.1145/292422.292440>

Quellen

- [32] Requirements for a first year object-oriented teaching language.
<https://dl.acm.org/doi/abs/10.1145/199688.199770>

- [33] Understanding the syntax barrier for novices.
<https://dl.acm.org/doi/10.1145/1999747.1999807>

- [34] Teaching Introductory Programming to IS Students: Java Problems and Pitfalls.
<https://eric.ed.gov/?id=EJ808971>

- [35] Python - Support for type hints. <https://docs.python.org/3/library/typing.html>

- [36] Mypy: Static Typing for Python. <https://github.com/python/mypy>

- [37] Pyre: Performant type-checking for Python. <https://github.com/facebook/pyre-check>

- [38] Bython - Python with braces. Because Python is awesome, but whitespace is awful.
<https://github.com/mathialo/bython>

Bilder

- <https://kotlinlang.org/docs/kotlin-brand-assets.html>
 - <https://en.scratch-wiki.info/w/images/ScratchCat-Small.png>
 - https://images-wixmp-ed30a86b8c4ca887773594c2.wixmp.com/f/d51354b6-b612-46e4-8e83-8584f38f035c/dfyt0sd-0f3df24e-763a-4bed-9a86-9ecd5ef2d5fb.png/v1/fit/w_474,h_358,q_70,strp/sad_scratch_cat_sitting_in_a_cliff__somewhere___by_375w-2x.jpg?token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1cm46YXBwOjdIMGQxODg5ODIyNjQzL2o-9mgc
 - https://res.cloudinary.com/practicaldev/image/fetch/s--ZDtqrBOj-/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https://github.com/damiancipolat/js_vs_m