

252-0027

Einführung in die Programmierung

Herbstsemester 2023/24

Thomas R. Gross

**Department Informatik
ETH Zürich**

©Thomas Gross 2020 - 2023

Vorlesung+

- **Im Hörsaal Präsensvorlesung**
 - Stellen Sie Fragen!
 - Bitte keine persönlichen Gespräche
- **Remote Übertragung (ML E12 und LiveStream)**
 - Stellen Sie Fragen, melden Sie Probleme
 - Aufzeichnung in ETH Video Portal
- **Fragen & Kommentare via EduApp Course Channel**

Diese Veranstaltung

- **Vorlesungen (ML D 28)**
 - Dienstag 10 – 12
 - Freitag 8 – 10 (ausserdem Do 28. 9. & 5. 10., 14 – 15, ETF F5)
- **Übungen (diverse Räume)**
 - Mittwoch 16 – 18
- **Ein Angebot: keine Anwesenheitspflicht**
 - Sie sind für Ihre (Aus)Bildung verantwortlich

Informationen

- **Web Seiten der Gruppe (Laboratory for Software Technology)**

www.lst.inf.ethz.ch

Im ETH Content Management System – nicht immer sofort online

- **Vorlesung und Übungen sind ein Angebot : keine Anwesenheitspflicht**
 - Sie sind für Ihre (Aus)Bildung verantwortlich

Auf unserer Web Seite finden Sie

- **Folien (wenn möglich vor der Vorlesung aber keine Garantie)**
- **Achtung: 1 Seite/Slide**
 - Drucken Sie 2, 4, 6 Seiten pro Blatt A4 Papier
 - Besser: drucken Sie nicht ...
- **In der Vorlesung geschriebene Folien (Auswahl)**
 - Vielleicht 24h-48h nach der Vorlesung
- **Diverse Links (zu Video Portal, Material der Übungsgruppen, Aufgabenstellungen)**

Informationen

- **Auf dem Video Portal der ETH**
 - Aufzeichnung des übertragenen Livestreams
 - Nur Hauptprojektor

- **Die Vorlesung wird aufgezeichnet**
 - *Daher* bittet unsere Rechtsabteilung Sie zur Kenntnis zu nehmen, dass diese Vorlesung aufgezeichnet wird. Wenn Sie nicht auf einem Video erscheinen wollen, dann setzen Sie sich bitte nicht in den Blickwinkel der Kamera im D 28.

Wichtig:

Bitte belegen Sie die Vorlesung in *myStudies*.

Sonst können wir Sie nicht erreichen und Sie haben nicht Zugang zu Aufgaben, usw.

Unbedingt bis Freitag den 22. 9., 12 Uhr mittags!

Das Programm für heute

«Educators, generals, dieticians, psychologists, and parents program. Armies, students, and some societies are programmed.»

Alan Perlis (Foreword to «Structure and Interpretation of Computer Programs», H. Abelson and G. J. Sussman, 1985)

Für uns: Allgemeinste Form des Programmieren:

«Computer Programming»

Das Programm für heute

- Was ist Programmieren?
- (Soll man) Programmieren lernen?
- Bezug zum Informatikstudium an der ETH
- Wie wir die Lernziele erreichen ...

Was ist überhaupt «Programmieren»

- **Programmieren → Programm**
 - Programm: von griechisch *prógramma*
 - Schriftliche Bekanntmachung, Aufruf; Tagesordnung [Duden]
 - Anweisungen
- **Informatik: Programm wird *ausgeführt***
- **Programm(ausführung) manipuliert Symbole**
 - Text, Zahlen, Bilder, ...

Was ist überhaupt «Programmieren»

- **Programmieren: Erstellen eines Programms**
- **Programmieren (Zusammenfassend): Software Entwicklung**
 - Programmierung behandelt alle Aspekte – von Entwurf bis Installation
 - Programm realisiert einen Algorithmus
 - Algorithmus: beschreibt Schritt-für-Schritt wie eine Aufgabe gelöst wird

(Soll man) Programmieren lernen?

Heute Thomas Herbst hat für eine Einladung war schon leuchtet

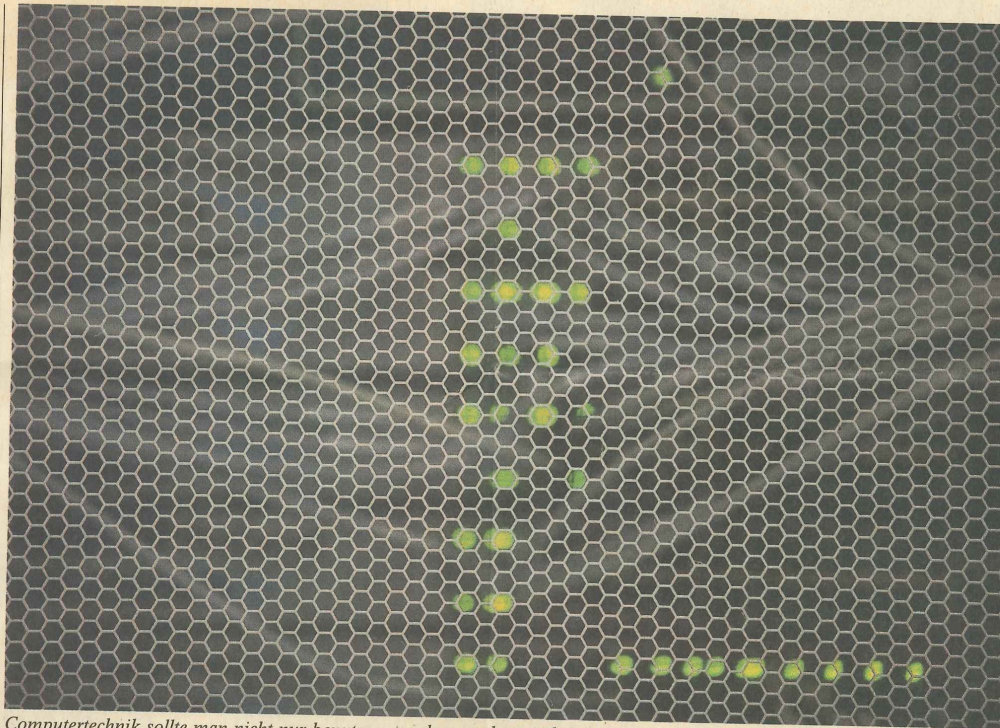
Programmieren oder programmiert werden

Die Grundlagen der Informatik sollten an den Schulen in einem eigenständigen, obligatorischen Fach unterrichtet werden

Der Informationsgesellschaft fehlen Fachleute mit Informatikkenntnissen, Menschen, die Möglichkeiten und Grenzen der maschinellen Informationsverarbeitung zu beurteilen wissen. Kann ein neuer Stundenplan am Gymnasium helfen?

Stefan Betschon

Einer steigt in eine Zeitmaschine, fliegt weit in die Zukunft. Er trifft auf feingliedrige, hellhäutige Menschen, die im Überfluss leben, ohne einer Beschäftigung nachzugehen. Nur abends, wenn die Dunkelheit kommt, werden sie unruhig. Es gibt noch eine zweite Art von Menschen, dunkelhäutige, stark behaarte Kreaturen, die unter der Erde leben und immer nur arbeiten. Sie halten Maschinen am Laufen, die offenbar für die Wohlfahrt aller wichtig sind. Sind diese hart arbeitenden Maschinisten unter der Erde die Sklaven der Oberirdischen? Oder sind die kindhaften, grossäugigen Menschen oben am Licht nur die Haustiere der Arbeiter unten im Maschinenraum? – Wer sich mit dem Stellenwert der Informatik in der Informationsgesellschaft beschäftigt, dem kann es passieren, dass er sich an die «Zeitmaschine» von H. G. Wells erinnert fühlt. Wir sind voll und ganz von informationsverarbeitenden Maschinen abhängig. Aber nur wenige von uns fühlen sich berufen, in der Maschine



Computertechnik sollte man nicht nur benutzen, sondern auch verstehen können. Im Bild: ein Server.

MARTIN RÜTSCHI / KEYSTONE

komplexität, die Grenzen der Automatisierbarkeit, sichere Kommunikation, Modellbildung, Problemlösungsmethodik. Es gehe nicht darum, aus Gymnasiasten Informatikexperten zu machen, sondern ihnen Einblicke zu vermitteln, die zur verantwortungsvollen Nutzung und zur Vermeidung grösseren Schadens nötig seien. Es ginge also darum, «computational thinking» einzuüben. «Wer konstruktives Informatikdenken nicht kennt, kann die moderne Informationswelt, in der wir leben, nicht verstehen und wird zum Bürger zweiter Klasse.»

Ein Taxi ohne Fenster

Bildungsdefizite im Bereich Informatik sind nicht nur in der Schweiz ein Problem. Auch in den USA wird darüber diskutiert. Prominente Computerunternehmer, darunter Bill Gates und Mark Zuckerberg, haben eine Stiftung gegründet – code.org – um junge Menschen für eine vertiefte Auseinandersetzung mit Informatik zu gewinnen. Die Website ziert eine Aussage von Steve Jobs: «Ich glaube, alle in diesem Land sollten programmieren lernen, würde sie denken lehren.»

Der New Yorker Autor Douglas Rushkoff hat diese Aussage, die nach Steve Jobs' Tod in einem verlogenen Interview aufgetaucht, vermutlich nicht gekannt. Aber sie net sich gut, um den Inhalt seines kleinen Buchs zu veranschaulichen.

werden Explosionen, Transformationen, Revolutionen vermeldet. Von Al-

Die Geschichte, wie wir auf der dritten Welle surfend in eine bessere

Drei emeritierte Hochschulprofessoren haben nun ein Buch herausge-

25. April 2017, 16:36 Tech-Jobs

Wer nicht programmiert, muss putzen!

THE HILL

31. Dez. 2019

Enrichment > Education

Biden tells coal miners to “learn to code”

By Alexandra Kelley | Dec. 31, 2019

(Soll man) Programmieren lernen?

VIEWPOINT

The End of Programming

By Matt Welsh

Communications of the ACM, January 2023, Vol. 66 No. 1
40 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

RESEARCH-ARTICLE **OPEN ACCESS** • 

Programming Is Hard - Opportunities and Challenges of AI Code Generation

Authors:  Brett A. Becker,  Paul Denny,  James Finnie-Ansley,  Andrew Luxton-Reilly,  James Prather,  Eddie Antonio Santos [Authors Info & Claims](#)

SIGCSE 2023: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 • March 2023 •

Pages 500–506 • <https://doi.org/10.1145/3545945.3569759>



Large Language Models and the End of Programming

Date & Time May 9, 2023 12:00 PM Eastern Time (US and Canada)

Webinar ID 942 7707 2999



Open Source Product

Introducing GitHub Copilot: your AI pair programmer

Today, we're launching a technical preview of GitHub Copilot, a new AI pair programmer that helps you write better code.

- **Idee:**
 - Benutzer gibt *Spezifikation* ein
 - Spezifikation: Aufgabenstellung
 - Copilot liefert ein korrektes Programm

Count «code»: Spezifikation

```
5 // Return the number of times that the
6 // string "code" appears anywhere in the
7 // given string, except we'll accept any
8 // letter for the 'd', so "cope" and "cooe"
9 // count.
```

Count «coXe»: Vorschlag für Name

```
5 // Return the number of times that the
6 // string "code" appears anywhere in the
7 // given string, except we'll accept any
8 // letter for the 'd', so "cope" and "cooe"
9 // count.
10 // countCode("aaacodebbb") → 1
```

countCode

```
// count.
```

```
public static int countCode("aaacodebbb")
```

```
public static int countCode("aaacodebbb"){  
    int count = 0;  
    for(int i = 0; i < str.length()-3; i++){  
        if(str.substring(i, i+2).equals("co")  
            count++;  
        }  
    }  
    return count;  
}
```


countCode

```
public static int countCode("aaacodebbb"){  
    int count = 0;  
    for(int i = 0; i < str.length() - 3; i+  
        if(str.substring(i, i + 2).equals("a"  
            count++;  
        }  
    }  
    return count;  
}
```

countCode

```
// Return the number of times that the
// string "code" appears anywhere in the
// given string, except we'll accept any
// letter for the 'd', so "cope" and "cooe"
// count.
public static int countCode(String str){
    int count = 0;
    for(int i = 0; i < str.length() - 3; i++){
        if(str.substring(i, i + 2).equals("co") &&
            str.charAt(i + 3) == 'e'){
            count++;
        }
    }
    return count;
}
```

Copilot

- Wählte guten Namen für das Programm
- Brauchte (etwas) Hilfe mit dem Parameter
- Zusammenfassung: 

Copilot mit Aufgaben auf Deutsch ...

String-1 > middleThree

prev | next | chance

Given a string of odd length, return the string length 3 from its middle, so "Candy" yields "and". The string length will be at least 3.

middleThree("Candy") → "and"

middleThree("and") → "and"

middleThree("solving") → "lvi"

```
21 // Gegeben sei ein String mit ungerader Laenge. Finden Sie den
22 // String der Laenge 3, der in der Mitte ist. Also fuer "Candy" ist
23 // die Antwort "and". Der Input String hat immer mindestens
24 // die Laenge 3.
```


Copilot w/ German

```
23 // Gegeben sei ein String mit ungerader Laenge.  
24 // String der Laenge 3, der in der Mitte ist. A  
25 // die Antwort"and". Der Input String hat immer  
26 // die Laenge 3.  
27     public static String middleThree(String str  
28         int mid = str.length() / 2;  
29         return str.substring(mid - 1, mid + 2);  
30     }  
31
```

- **Kein Problem**
 - Copilot kennt sogar einen guten Namen für das Programm ...

Copilot mit Aufgaben auf Italienisch

- **Danke an Giacomo Ponti!**
- Data una stringa di lunghezza dispari, ritorna la stringa di lunghezza 3 dal centro della stringa originale. La lunghezza della stringa originale sarà almeno di 3 caratteri.
 - Auch die italienische Aufgabenstellung für middleThree und andere sind kein Problem
 - (Alle Aufgaben, die wir probierten ...)

- **Andere mach(t)en ähnliche Erfahrungen**
 - Copilot recht gut für solche und ähnliche Probleme
- **Ist Programmieren kein Problem mehr?**

(Soll man) Programmieren lernen?

- Programmieren – das Latein des 21. Jahrhunderts?
- Stärkt logisches Denken ...



Lehrplan Gymnasium
2022,
Kanton Zürich

Regelunterricht > Fächer

Latein



(Soll man) Programmieren lernen?

- **Programmieren – das Latein des 21. Jahrhunderts?**
- **Stärkt logisches Denken**
 - Angeblich
 - Aber nicht wirklich wichtig
 - Für die meisten Studierenden
- **Gilt das auch für Programmieren ?**
 - Github Copilot macht die Arbeit für uns??

“ As far as I can tell,
Copilot was
specifically trained
on all the intro
programming
assignments ever

E Berger, Coping w/ Copilot

- **Copilot «lernte» mit öffentlichen Repositories und Programmier Aufgaben**
 - Copilot kennt nicht nur die Lösungen sondern weiss auch was das nächste Problem ist ...
 - Genauso kennt Copilot «beliebte» Aufgaben wie «Türme von Hanoi», «Fibonacci Zahlen»,
- **Aber wer will Lösungen zu diesen Aufgaben??**

Prüfungsvorbereitung 2018

Im 1. Schritt



Week	Size (Words)	Topic	Success
4	197	String-Addition: loops, arrays	0
5	456	Tool Rental: classes, arrays, iteration, JUnit	0
6	344	Valleys & Hills: arrays, data analysis, I/O	0
7	146	String Interleaving: recursion	0
8	151	List Reversal: references, working w/ classes	0
9	163	Class Puzzle: inheritance (<i>w/o classes or driver</i>)	0
10	600	Desk Calculator: inheritance, recursion	0
11	302	Data Analysis (FIFA): ArrayList<..>, Map<..>	0
12	194	Sublist Palindrome: Set<List<..>>, exceptions	100

Copilot

Your AI pair programmer

Copilot – Erwartet Hinweise

- **Mögliche Hinweise**
 - Einfügen von Variablen
 - Änderung des Kontrollflusses
 - Hilfe bei Implementation einer Datenstruktur
 - Attribute und Methoden von Objekten
 - Definition von Vergleich(Prädikat)en
- **Vorläufige Ergebnisse für ausgewählte Aufgaben**
 - Experimente von Giacomo Ponti

Copilot – mit Hinweisen (2018)

Anzahl Hinweise



Week	Size (Words)	Total	Success
4	197	4	100
5	456	3	100
6	344	4	100
7	146	2	0
8	151	1	100
9	163	12	100
10	600	5	100
11	302	3	100
12	194	0	100

Copilot – Pair Programmer nicht Ersatz

- **Mögliche Hinweise**

- Einfügen von Variablen
- Änderung des Kontrollflusses
- Hilfe bei Implementation von Datenstruktur
- Attribute und Methoden von Objekten
- Definition von Vergleich(Prädikat)en
- ... und weitere

- **Sie haben dazu Fragen?**

←

←

←

←

←

- **Dann sind Sie hier richtig**

Programmierung und Informatik

- Ein Thema der Informatik – es gibt auch andere
 - Zentral wenn Sie lernen wollen, wie die Informatik ein Problem angeht
 - Fokus auf «Möglichkeiten und Grenzen der maschinellen Informationsverarbeitung»
 - Es gibt «unmögliche» Probleme
 - Kosten der Berechnung (einer Lösung) sind sehr wichtig

Programmierung

**«*Programming as universal activity*» by Vinton Cerf, CACM
March 2016, vol 59(3) p 7**

- **analyzing problems**
- **breaking them down into manageable parts**
- **finding solutions**
- **integrating the results**

Programmierung

**«*Programming as universal activity*» by Vinton Cerf, CACM
March 2016, vol 59(3) p 7**

- Probleme analysieren
- Probleme in (beherrschbare) Teilprobleme zerlegen
- Lösungen finden
- Ergebnisse zusammenfügen/kombinieren

Vinton Cerf (*1943)

Ph.D. UCLA (1972)

Assistant Professor Stanford (1972-76)

Program Manager (D)ARPA (1976-82)

**Zusammen mit Bob Kahn massgeblich
an der Entwicklung von TCP/IP beteiligt
(Internet Protokoll)**

**Nach 1982 diverse Positionen in
Industrie & Verbänden, seit 2005 VP
Google**

Dr. h.c. ETH 1998



Schlagfertig beantwortete Vint Cerf die teilweise ziemlich kritischen Fragen des ETH-Publikums.(2006)

<http://web.ethlife.ethz.ch/articles/campuslife/vintcerf.html>

Programmierung

- **Lösungen finden: für andere Menschen**
 - Beschreiben wie eine Lösung aussehen soll
- **Lösungen finden: für eine Maschine**
 - Anweisungen für eine Maschine/Computer

Programmierung

- **Lösungen finden: für andere Menschen**
 - Beschreiben wie eine Lösung aussehen soll
- **Lösungen finden: für eine Maschine**
 - Anweisungen für eine Maschine/Computer
- **Beschreibung, Anweisung: in einer «Sprache»**

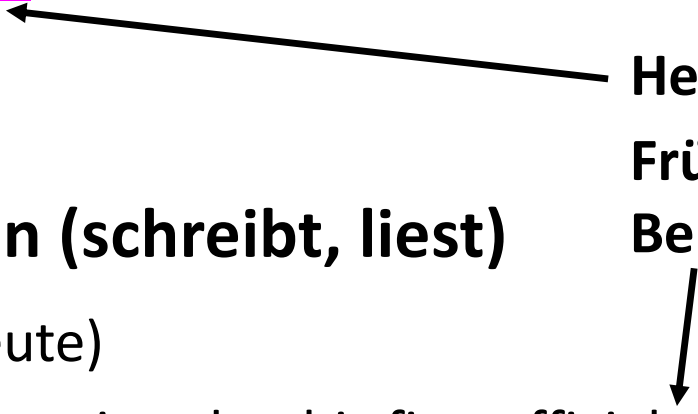
Wie Programmieren lernen?

- **Programm: Folge von Anweisungen, die von einem Computer ausgeführt werden (können)**
- **Wir müssen verstehen welche Anweisungen der Computer ausführen kann**
 - Dann Erstellen der Folge von Anweisungen
- **Mögliche Anweisungen: Programmiersprache**

Einführung in die Programmierung

- **Müssen eine Programmiersprache verwenden**
- **Sprache für Computer (führt aus)**
 - ... schreibt
- **Sprache für Menschen (schreibt, liest)**
 - ... selten führt aus (heute)

Einführung in die Programmierung

- Müssen eine Programmiersprache verwenden
 - Sprache für **Computer** (führt aus)
 - ... schreibt
 - Sprache für Menschen (schreibt, liest)
 - ... selten führt aus (heute)
 - Charles [Peirce] was appointed to his first official position in the Coast Guard Survey [..] in July 1861, as an *assistant computer* at \$ 35 per month. (J. Brent, Charles Sanders Peirce: A Life. 1993. p 61)
- Heute: Maschine
Früher:
Berufsbezeichnung
- 

Einführung in die Programmierung

- **Müssen eine Programmiersprache verwenden**
- **Wir verwenden Java™**
 - «Industrial strength» Sprache
 - Viele Konzepte
 - Nicht alle werden in «Einf. in die Programmierung» vorgestellt/verwendet
 - ... diese werden auch nicht für die Prüfung erwartet
 - Mehr Themen/Konzepte in weiteren Vorlesungen

Programmieren

- **Wie kann man Programmieren lernen?**
 - Braucht man eine besondere Begabung?
 - Gibt es nur Naturtalente (und der Rest kann zuschauen)?
- **Wie kann man XXXX lernen?**

Jede(r) kann programmieren lernen

- **Ziel der Vorlesung: Kompetenz**
 - Lernziel: Sie können korrekte Programme systematisch erstellen
- **Programmieren ist zentrales Thema der Informatik**
 - Aber nicht das einzige!
- **Wichtig sind:**
 - Aufmerksamkeit
 - Imagination, Phantasie
 - Übung Übung Übung Übung Übung

Veranstaltung = Vorlesung + Übung

- **Programmieren erfordert Übung**
- **Nur Übung macht perfekt ...**
- **Sie lernen nicht programmieren wenn Sie sich nur die Vorlesung anhören**
 - Sie müssen die Übungsaufgaben lösen (oder es zumindest versuchen)

Basisprüfungstipps

Einführung in die Programmierung

EProg gilt als relativ einfaches Fach, mit welchem man gut kompensieren kann. Jedoch sollte es nicht unterschätzt werden, denn es benötigt genügend Übung. Dies gilt auch, wenn Du schon programmieren konntest, denn Themen wie Vererbung und Polymorphismus sind komplex.

Programmieren ist vor allem Übungssache. Investiere also genügend Zeit in die Übungen.

Basisprüfungstipps



Visionen Dez 2020

Programmieren zu können zählt sich aus im Informatikstudium.

Übungen

- **Aufgabenblätter**

- Werden über Web Seite publiziert
- In der Übungsgruppe bei Bedarf vor-besprochen

- **Eine Gelegenheit zu lernen!**

- Praxis Übungen
- Bonus Übungen

- **Teilnahme an Übungsgruppe nicht verbindlich aber sehr empfohlen**

- Es gibt selten nur eine Lösung
- Üben Sie das Diskutieren und Vergleichen verschiedener Lösungen

Praxis und Bonus Übungen

- **Werden teilweise automatisch korrigiert**
- **«Automatisch» – durch einen Computer**
 - d.h. durch ein Programm, das von einem Computer *ausgeführt* wird
- **Hinweis: Ihr Java Programm wird nicht direkt ausgeführt, die Java Anweisungen werden *übersetzt* (durch einen Compiler)**
 - Analyse des Programms
 - Selbe Technologie hilft bei der Bewertung von Programmen
 - Thema der Forschung

Praxis Übungen

- **Sie können die Lösungen im Internet (wahrscheinlich) finden**
 - Musterlösung aus früheren Jahren
 - Langweilen die Assistierenden und Copilot
- **Eine Gelegenheit zu lernen!**
 - Besprechung in der Gruppe, Feedback, Überarbeiten

Bonus Übungen

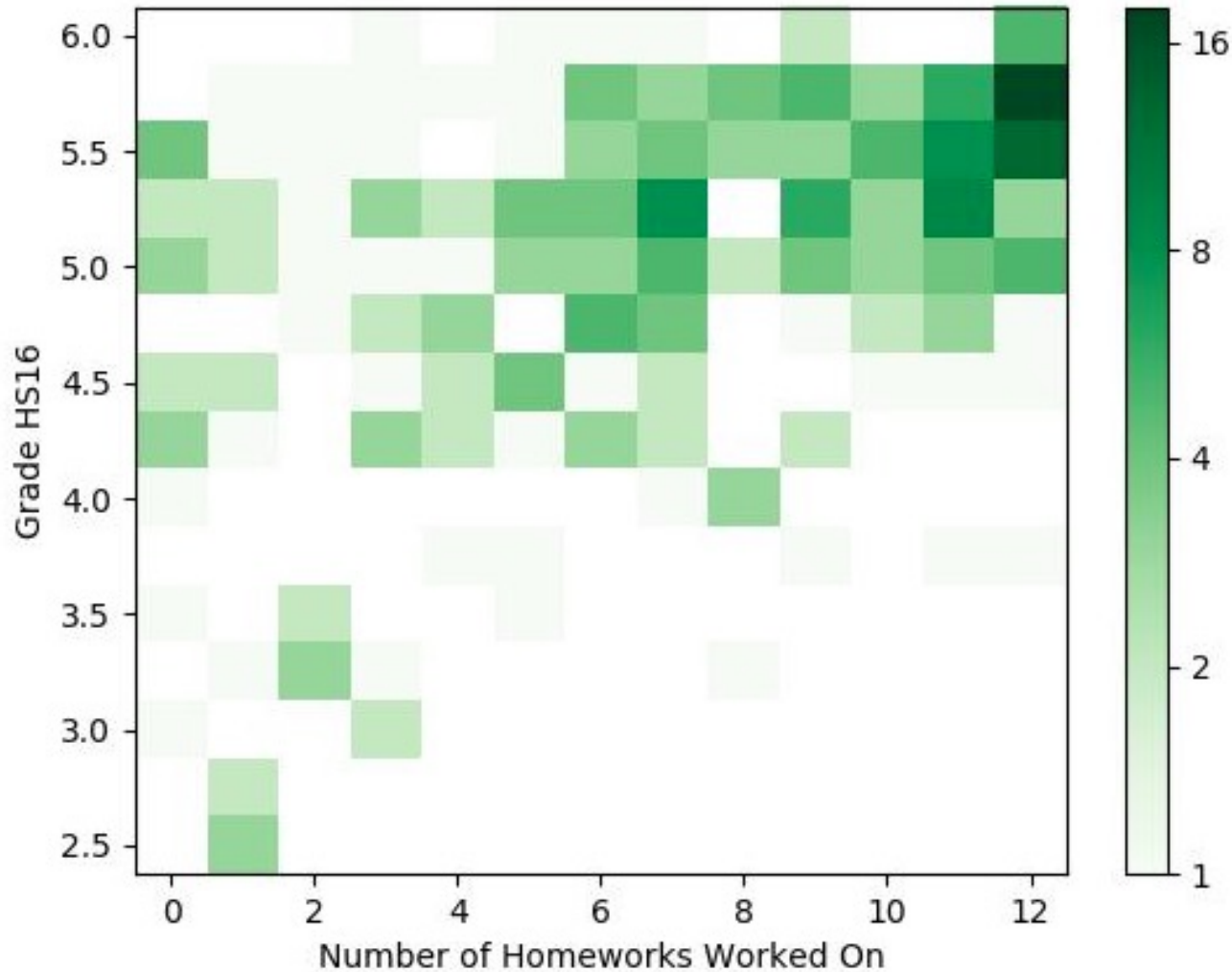
- **Sie können (ab 4. oder 5. Aufgabenblatt) «Bonuspunkte» für die Basisprüfung sammeln**
 - Bonuspunkte helfen die Note anzuheben – maximal 0.25 Notenbonus
 - Maximalnote auch ohne Bonus erreichbar
- **Programmieren ist Teil der Basisprüfung**
 - Details später.

Bonus Übungen

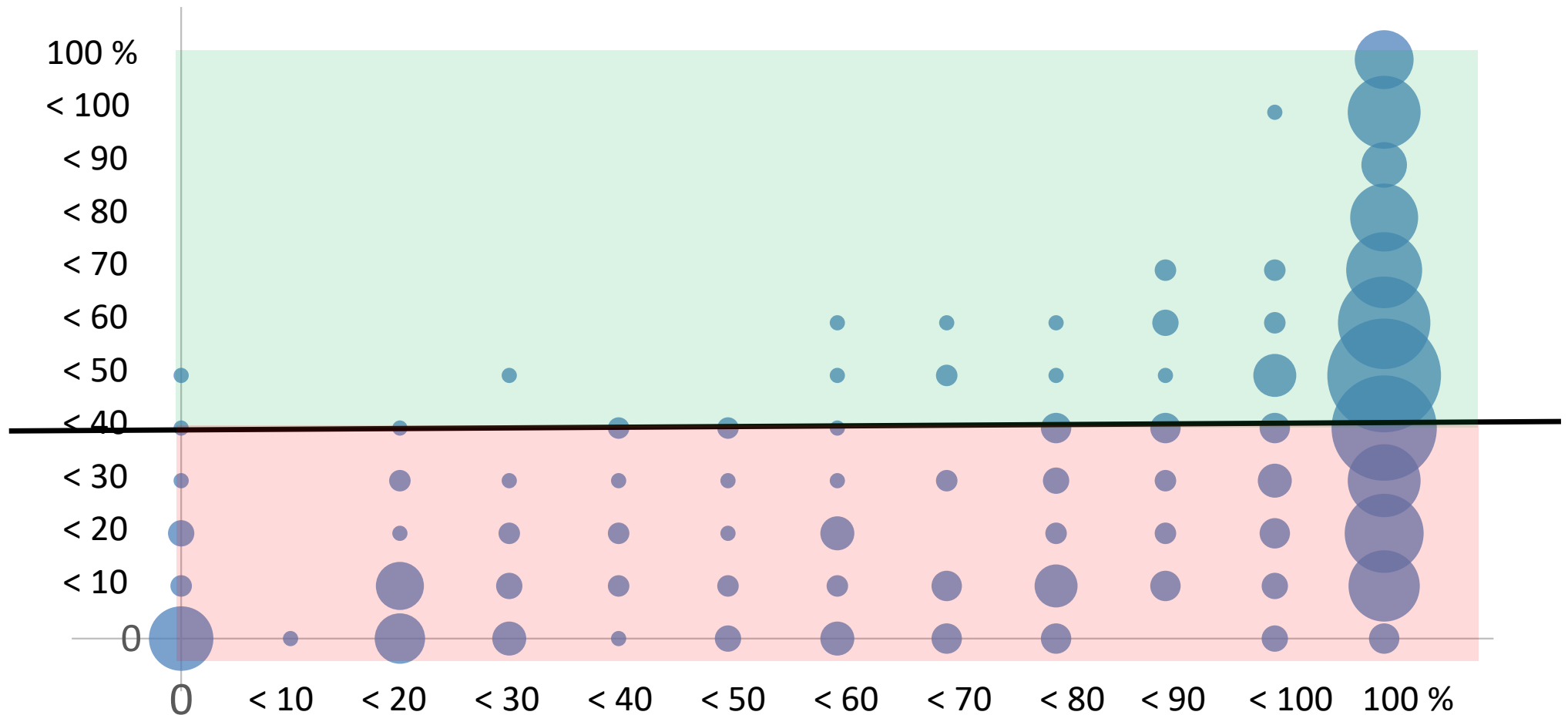
- **Anreiz**
- **Sie müssen die Aufgaben *selber* lösen.**
- **Abschreiben (oder abschreiben lassen) ist unehrliches Verhalten und wird nach der ETH Disziplinarverordnung geahndet.**

Übungen vs Prüfungserfolg (WS16/17)

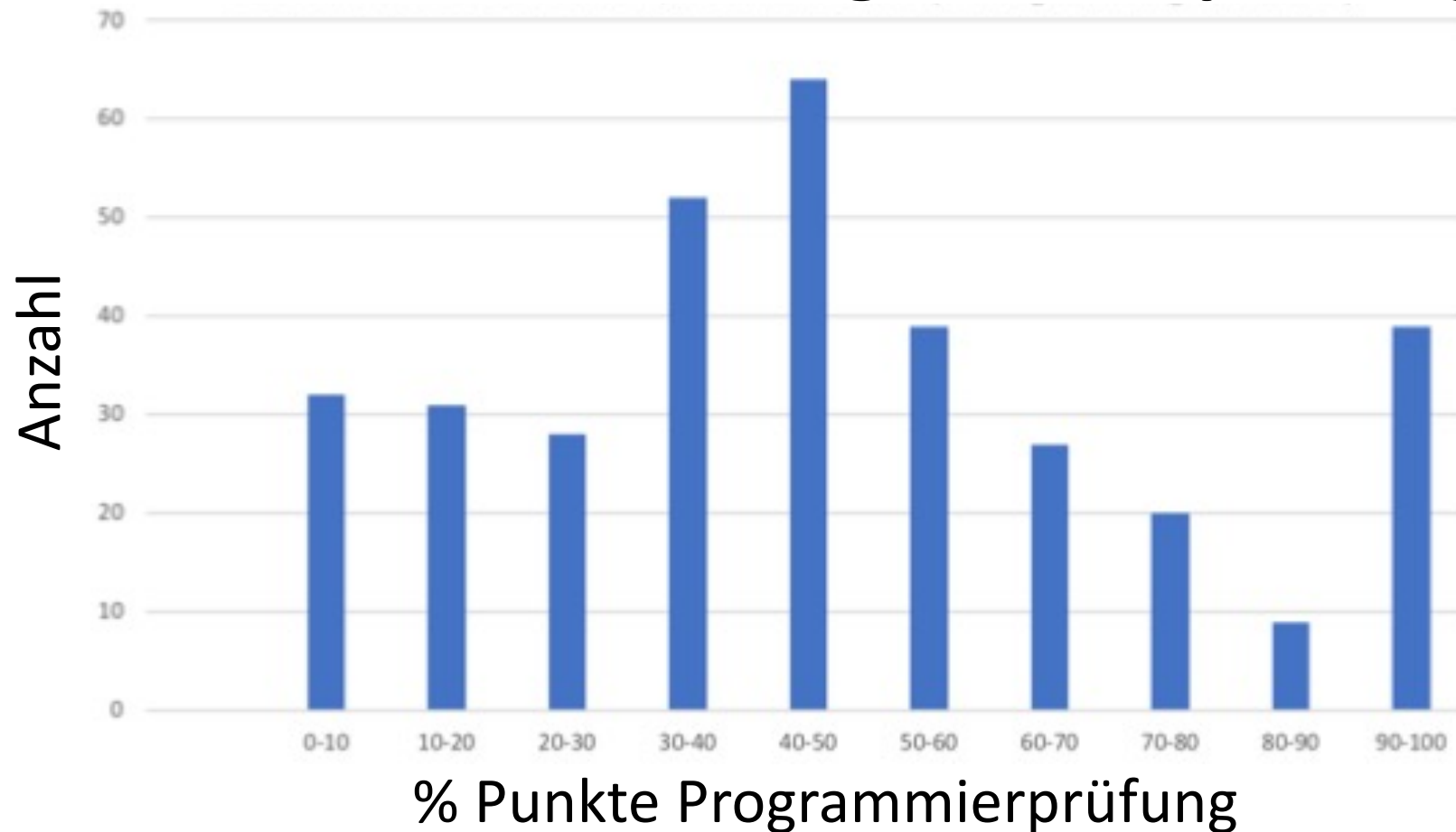
Dank an R Meier & M Faes für die
Graphik. (Session Herbst 16/17)



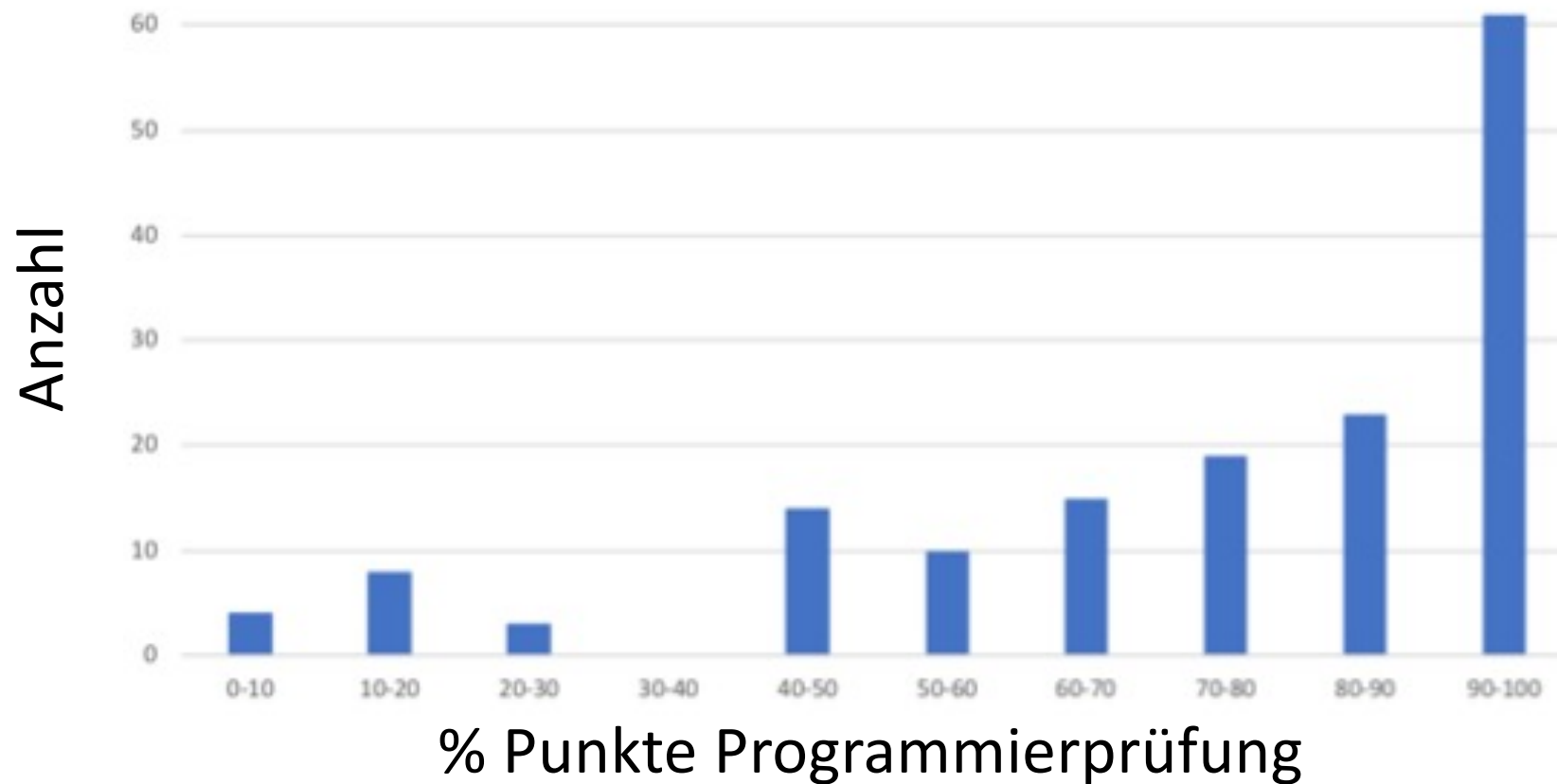
% Bonus vs. % Programmieren (HS21)



Anzahl Studierende (100 % LE Bonus *insgesamt*)
→ X % Punkte Programmierprüfung (HS21)



Anzahl Studierende (100 % LE *Timed* Bonus) → X % Punkte Programmierprüfung (HS22)



Bonus Übungen

- **Anreiz**
 - Stichwort: Eigenverantwortung
- **Nur die Bonusaufgaben lösen? Riskant wenn Sie wenig/keine Programmiererfahrung haben.**
- **Und: Sie müssen die Aufgaben *selber* lösen um zu lernen.**

Übungsgruppen

- **Wir bieten drei Arten von Gruppen an:**
- ***Fokusgruppen***
 - Fokus auf Studierende ohne Vorkenntnisse und ohne vorherige Programmiererfahrung (egal welche Programmiersprache)
- ***Gemischte Gruppen***
 - Für alle Vorkenntnisstufen
- ***Repetierenden Gruppe (schneller Einstieg)***

Übungsgruppen

- **Wir bieten drei Arten von Gruppen an:**
- ***Fokusgruppen***
 - Fokus auf Studierende ohne Vorkenntnisse und ohne vorherige Programmiererfahrung (egal welche Programmiersprache)
- ***Gemischte Gruppen***
 - Für alle Vorkenntnisstufen
- **Freie Wahl -- Alle Gruppen behandeln den selben Stoff/die selben Aufgaben**

Einschreibung in Übungsgruppen

- **Ab heute Nachmittag in myStudies möglich**
- **Wenn die bevorzugte Art von Gruppe nicht verfügbar ist:
Warten!**
 - Wir passen Anzahl und Grösse der Gruppen an
- **Wenn Sie mit anderen Studierenden in eine Gruppe wollen:**
 - Zusammen (zeitgleich) einschreiben
- **Bitte bis Freitag (22. 9.) einschreiben!**
 - Nachzügler später möglich, aber evtl. mit Einschränkungen
 - Nur wer in Gruppe eingeschrieben ist kann Lösungen einreichen

Bei Problemen

- **Erste Anlaufstelle Treffen der Übungsgruppe**
- **Wenn nicht möglich: «Study Center»**
 - Coaches können 1-1 helfen
 - Study Center nicht Ersatz für Übungsgruppe
- **Weitere Informationen auf dem Web**

Wöchentlicher Ablauf (im Normalfall)

- bis **Mittwoch Morgen: Praxis Übungen und (später) Bonus Aufgaben auf dem Web publiziert**
 - Übung 0: [Heute] Einrichten der Arbeitsumgebung
- **Mittwoch Nachmittag: Treffen Übungsgruppe**
 - Besprechung Aufgaben (alte, neue, extra) und Vorlesungsthemen
 - *Diese Woche nur Study Center* für Probleme mit Arbeitsumgebung
- **Mittwoch Folgewoche: Abgabe der Lösungen (Praxis Übungen und Bonus Aufgaben)**

Besondere Regeln für 9. & 11. Woche

- **Bonus Aufgabe erst am 21. 11. (5. 12.) publiziert und muss innerhalb von 2 Stunden abgegeben werden**
- **Weitere Details auf dem Web und später in Vorlesung**

Weiterer Ablauf (Plan)

- 22. 9.: 8:15 (Fr): Vorlesung
- 26. 9.: 10:15 (Di): Vorlesung
- **27. 9.: 1. Treffen der Gruppen (Mittwoch nächste Woche)**
 - Thema: Arbeitsumgebung einrichten
- **28. 9.: 14:15 (Do): Vorlesung [ETA F5, Keine «Algorithmen und Datenstrukturen» zu dieser Zeit]**
- 29. 9.: 8:15 (Fr): Vorlesung
- 3.10.: 10:15 (Di): Vorlesung
- **5. 10.: 14:15 (Do): Vorlesung [ETA F5]**
- 6. 10.: 8:15 (Fr): Vorlesung

Vorschau

- **Übung U0: (19. 9. -> 27. 9.): Einrichten der Arbeitsumgebung**
 - Keine Abgabe
 - Wir wollen Ihnen helfen schnell arbeiten zu können: jetzt anfangen!!
 - Sprechstunden (Study Center) nutzen (**Mi 20.9. 16-18, Fr 22.9. 12-14**)
- **Übung U1 (27. 9. -> 4. 10.): «Abgabe» durchspielen, einfache Aufgabe**
 - So erhalten Sie auch Feedback, ggf. in der Gruppe einreichen
- **Übung U2 (4. 10. -> 11. 10.): einfachste Programme**
- **Ab Übung U4: Bonuspunkte möglich (später mehr)**