CS1
V2.0

CS1

V2.1

Motivation

Shifting from Teacher-centered to Student-centered learning

Identifying variation within the student cohort

Implementing programming exercises to foster theory

Increasing focus on imperative programming

Introductory Programming

166 Students

15 ECTs

7 Weekly assignments

2 Assignments

3 Tests

1 Four-week project

(Objects-first with Java)

Introductory Programming

Stakeholders question students basic imperative competencies.

Are they capable of composing basic constructs?

[Nicolajsen, S. M., *Understanding Programming Languages as an Advanced Beginner*.]

Introductory Programming

Increase opportunities of *learn-by-doing*

(Shifting from Teacher-centered to Student-centered learning)

Understand student diversity in terms of experience

(Identifying variation within the student cohort)

Systematically design exercises to strengthen understanding of PL constructs

Increasing focus on imperative programming

Shifting from Teacher-centered to Student-centered learning

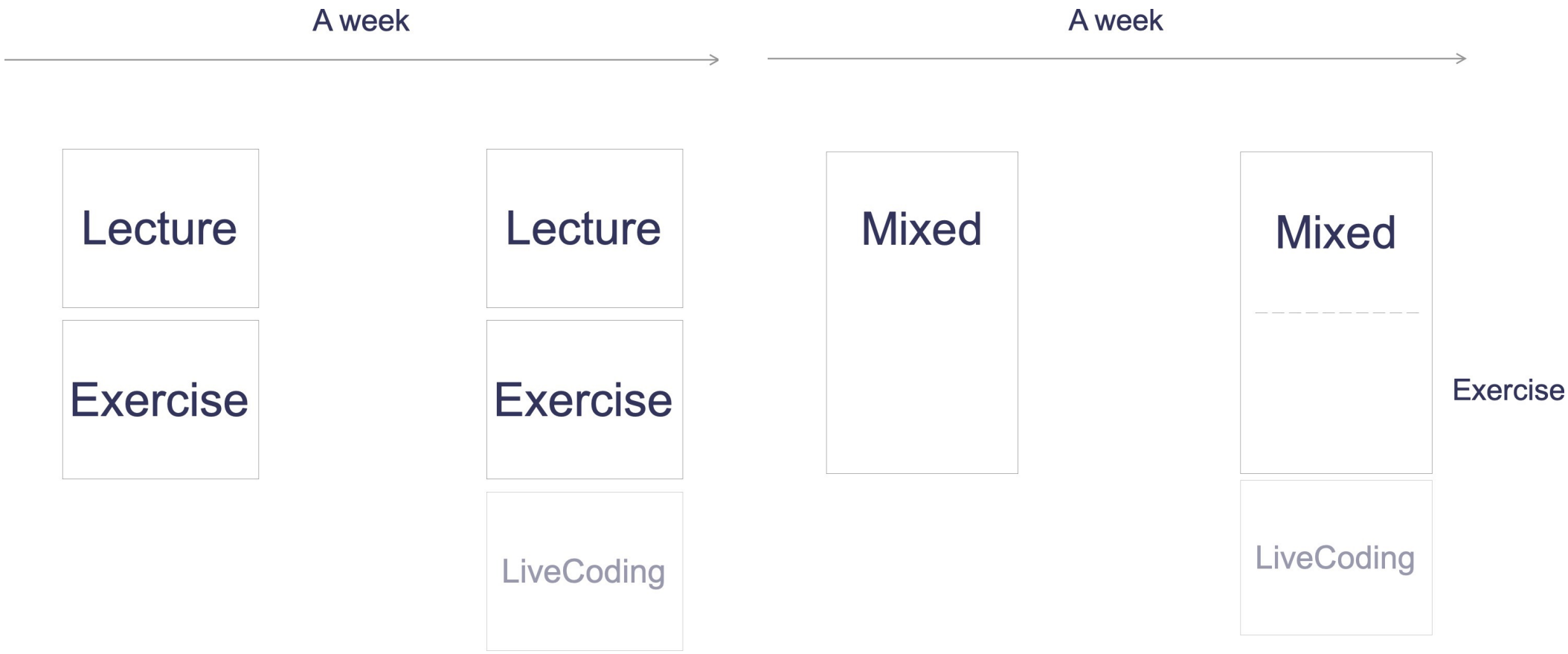
Lectures contain multiple (different) techniques and theory.

(T_A, T_B , … , T_n)

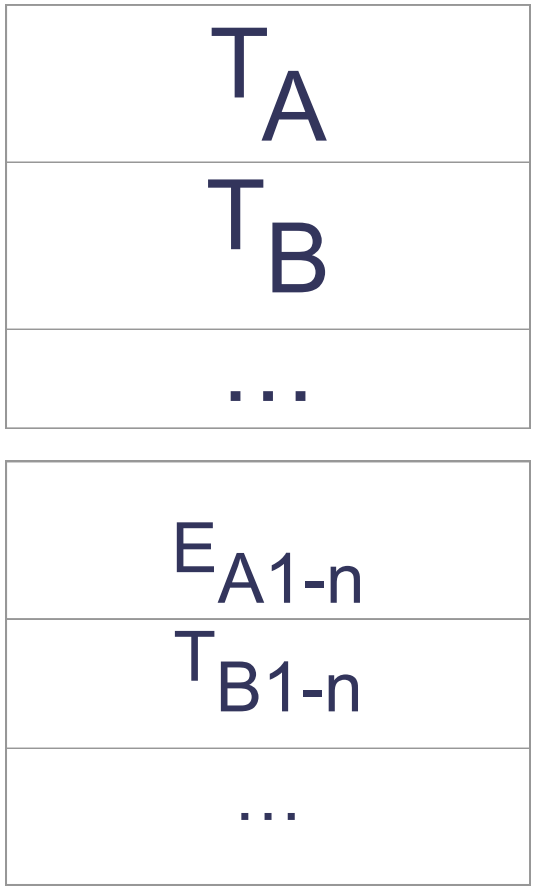
Labs (or exercises) following iterates these.

($E_{A1\text{-}X}$, $E_{B1\text{-}Y}$, … , $E_{n\text{-}Z}$)

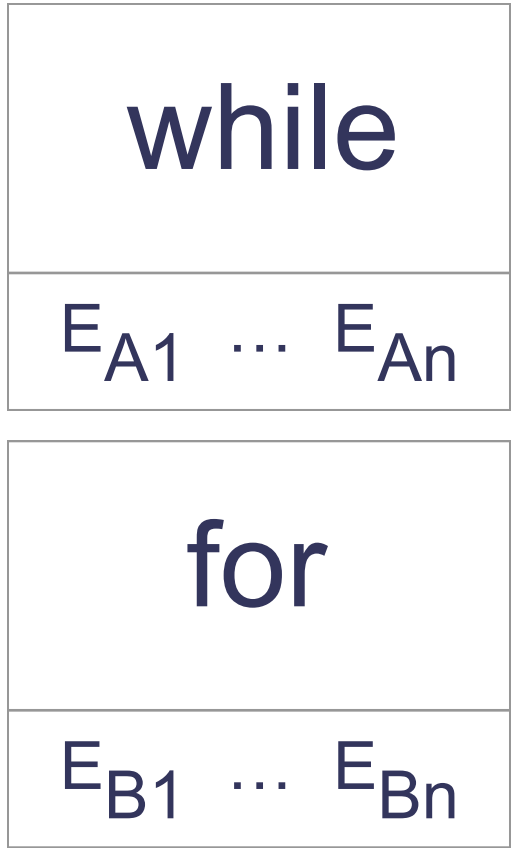
Shifting from Teacher-centered to Student-centered learning



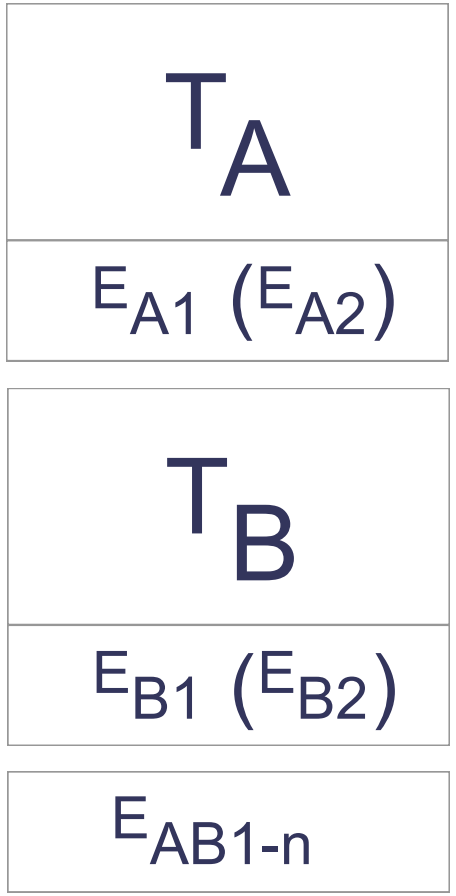
Shifting from Teacher-centered to Student-centered learning



Shifting from Teacher-centered to Student-centered learning



Shifting from Teacher-centered to Student-centered learning (in practice)

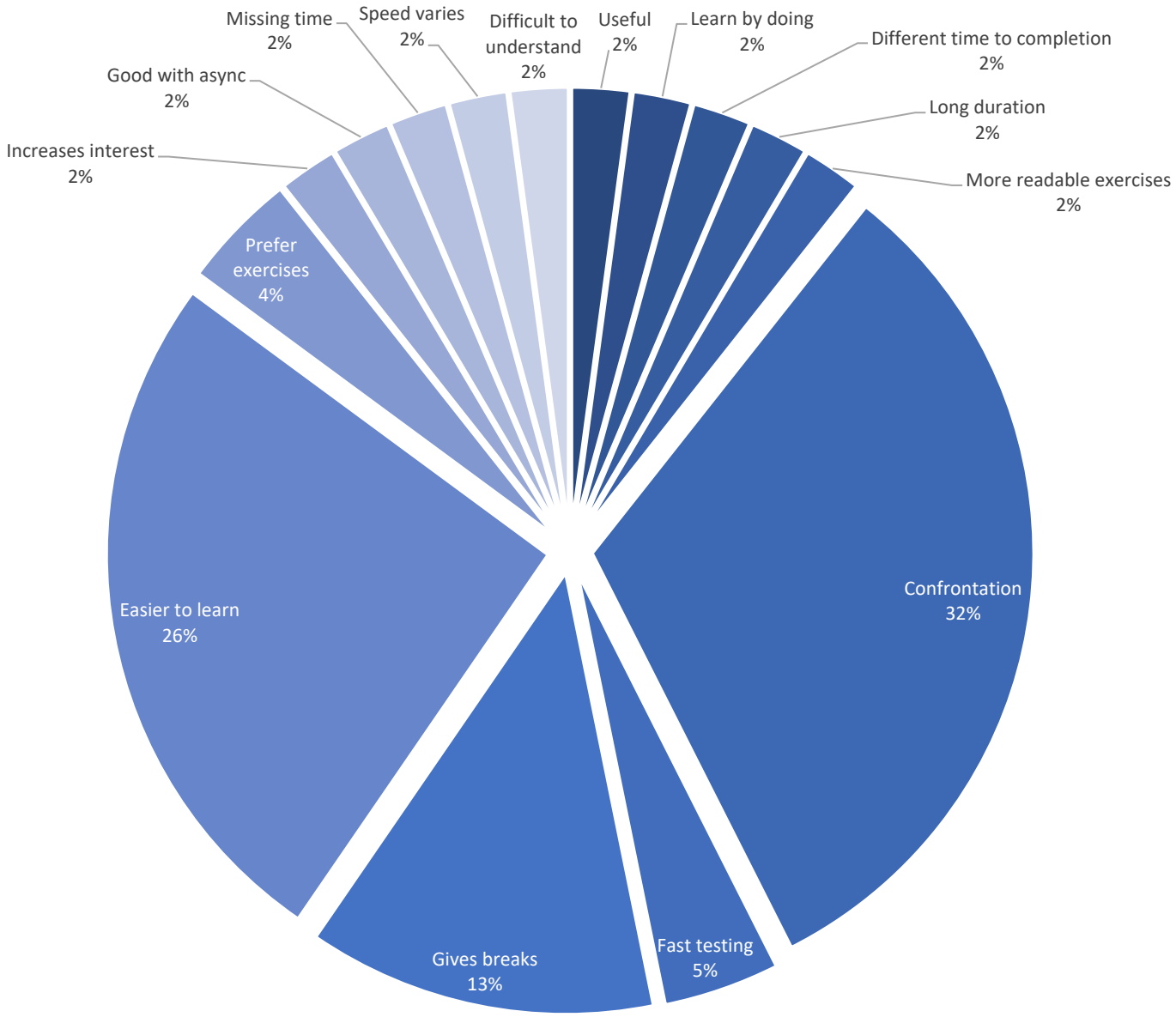


Typically 1-2 exercises
Alone and in combination

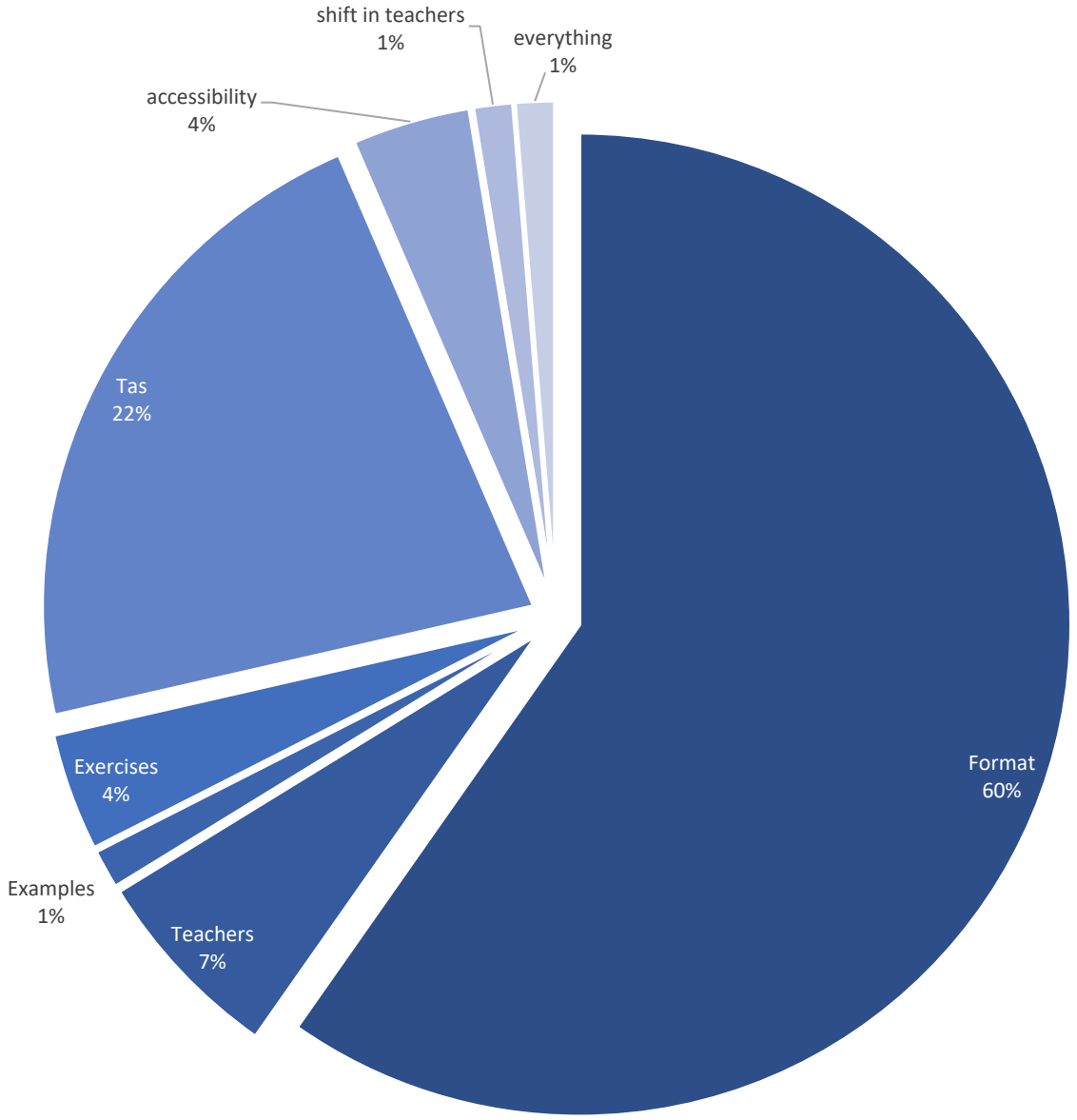
Use most fitting
Or combine



What are the effects?



*What do you feel about the format compared to other courses? (N=50)



* Anything about the teaching you find particular good (please expand)? (N=79)

Sometimes yes, however, other times I feel like the teaching is going too fast.

Yes!! – There is relatively many successes. The exercises during teaching helps a lot with understanding the concepts. It is super nice that there is so much focus on letting the students getting "hands-on". A little like learning a music instrument, it makes sense that you do not have a teacher who runs theoretical one-way communication to their students but that the teacher allows their students to try things in practice before moving on with more theory.

Overall, I benefited from the course

5.61

The course was organized in a way that helped me learn

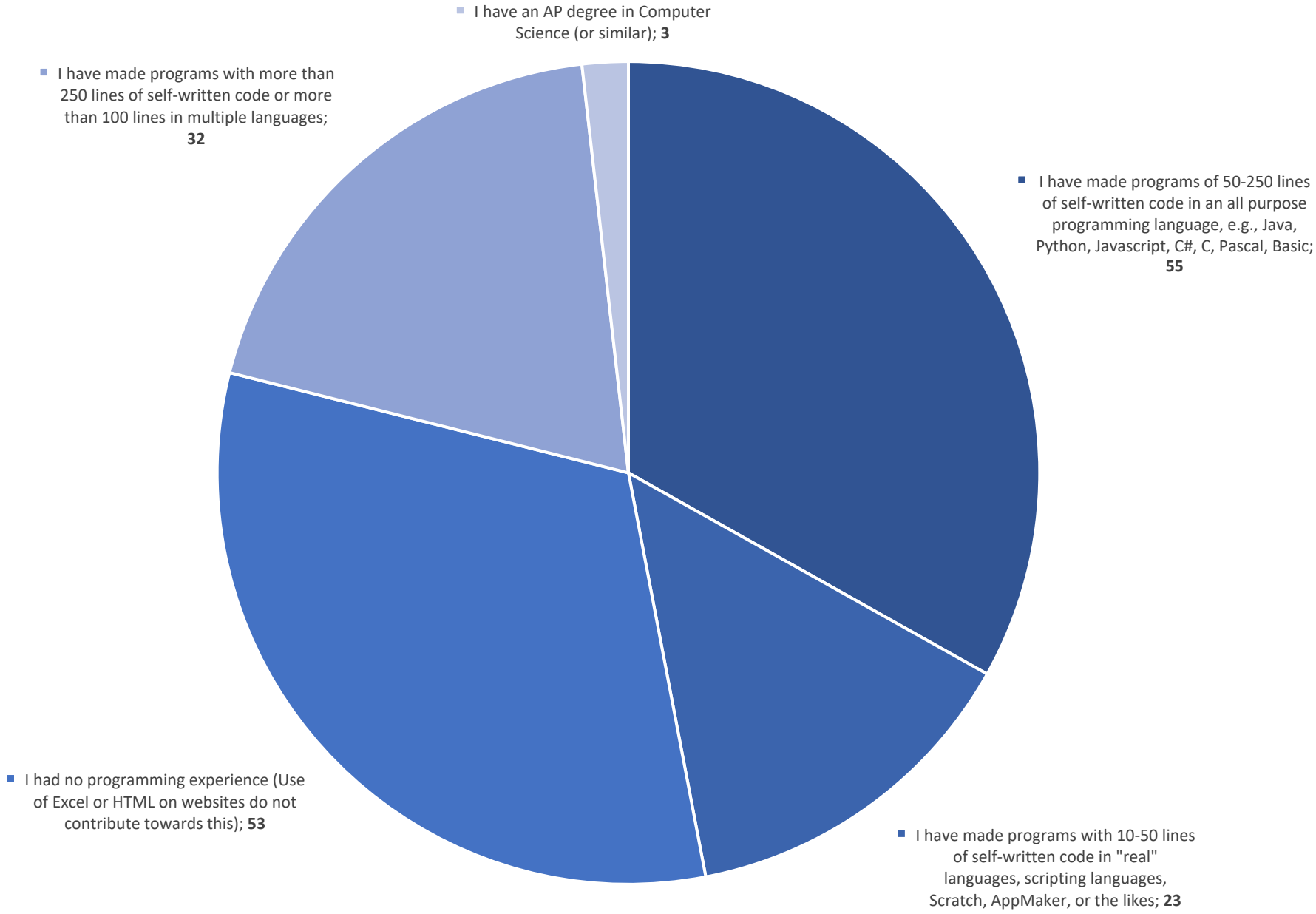
5.57

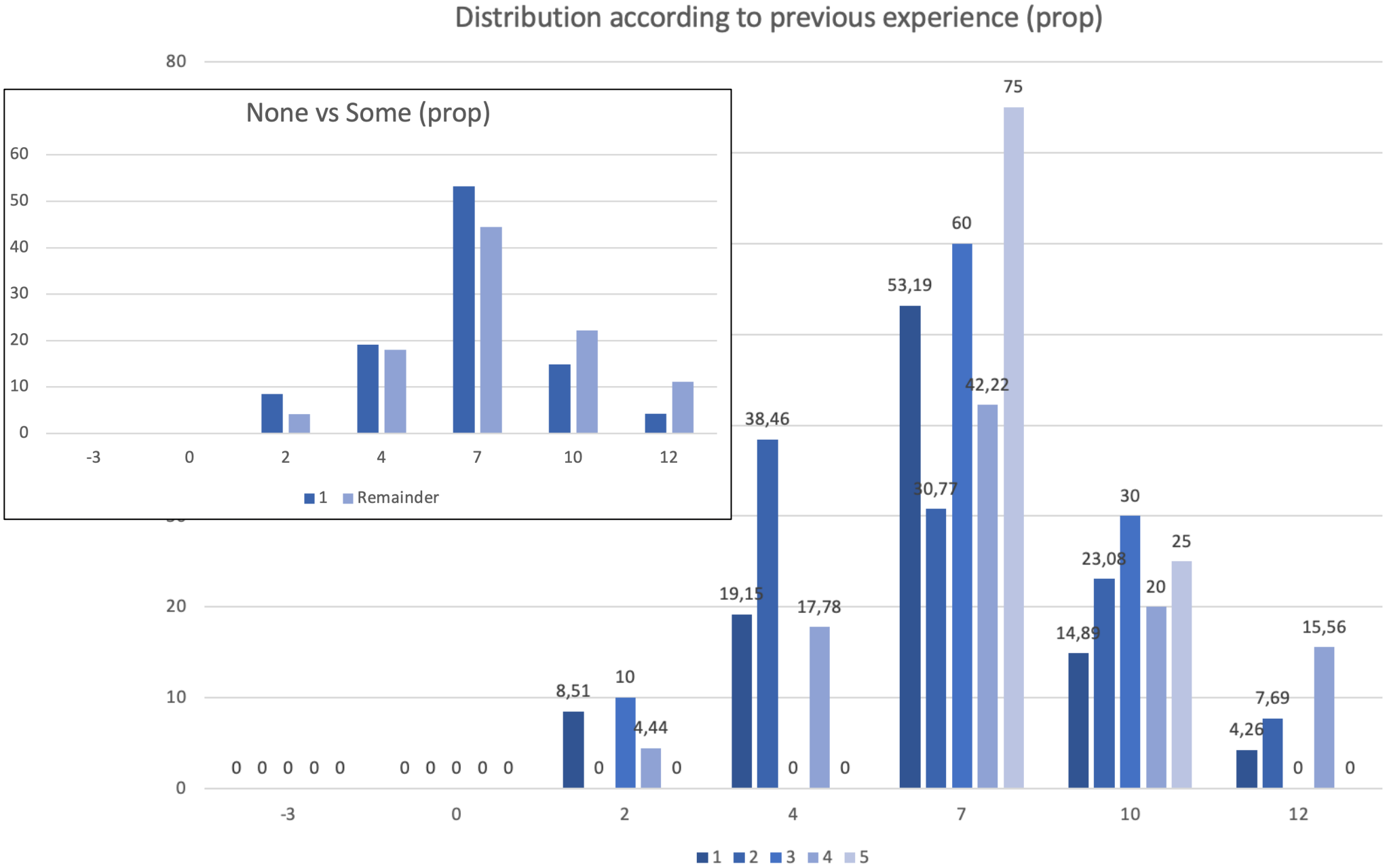
Identifying variation within the student cohort

Student experience impacts:

- Performance
- Self-efficacy (also of others)
- Retention

And potentially course design.


I have an AP degree in Computer Science (or similar); 3
I have made programs with more than 250 lines of self-written code or more than 100 lines in multiple languages; 32
I have made programs of 50-250 lines of self-written code in an all purpose programming language, e.g., Java, Python, Javascript, C#, C, Pascal, Basic; 55
I had no programming experience (Use of Excel or HTML on websites do not contribute towards this); 53
I have made programs with 10-50 lines of self-written code in "real" languages, scripting languages, Scratch, AppMaker, or the likes; 23


Distribution according to previous experience (prop)
None vs Some (prop)
1
Remainder
1 2 3 4 5

Affecting CS1 courses…

What is the focus of CS1?

[Misconceptions reconceived: A constructivist analysis of knowledge in transition, Smith III, J. P., DiSessa, A. A., and Roschelle, J.]

Extracurricular or mandatory?

Implementing programming exercises to foster theory

Systematically implement programming exercises.

Focusing on Tasks, Techniques, Technology, and Theory

Implementing programming exercises to foster theory

03

03-1: funktioner (statiske metoder)

03-1-T1	Introduktion til procedurer og deres værdi (og at de kan genbruges)	Theory/technology
03-1-A1	At lave en ny procedure til at printe et-eller-andet (hello world?)	Task/technique
03-1-T2	Introduktion til funktioner (i.e., parametre og at der kan være flere)	Theory/technology
03-1-A2	Øvelse med at lave en funktion der tager parametre (kan udvides til flere param)	Task
03-1-T3	Introduktion til retur værdier	Theory/technology
03-1-A3	Øvelse med at anvende funktioner der returnerer noget med parametre	Task/technique
03-1-T4	Funktioner kan nestes, og vi kan anvende dem flere gange	Technique/technology
03-1-A4	**Mere open-ended opgave som kræver at man sammensætter flere funktioner ellers vil der være gentaget kode.** **03-1-A4 bør indeholde flere opgaver således at de rigtigt bliver smidt rundt i tingene. Gerne med genanvendelser af ting fra blok 01-2 (Iteration) samt 02-1 (if-statements)**	**Task**
03-1T5	Opsummering.	

*Tilføjet opgaver hvori man skal trace kald i et stykke kode
*Tilføjet en A før 03-1-A3 som bare arbejder med retur værdi.

05

05-1: Introduktion ArrayList

05-1-T1	ArrayListers formål forklaret syntaks + semantik (med integers, for-loop)	Theory
05-1-A1	En opgave med at oprette en liste og tilføje nogle elementer.	Task/technique
05-1-T2	Iteration: hvordan tilgår vi indholdet.	Technique/technology
05-1-A2	Opret og gennemløb en liste med integers.	Task
05-1-T3	Forklaring på hvorfor vi skriver Integers med stort (boxing), og hvordan vi kan anvende vilkårlige typer heri.	Theory
05-1-A3	En opgave med en liste af andre elementer	Task
05-1-T4	Snak om generiske typer i Java. (Genforklar primitiv + ikke-primitive)	Theory
05-1-T5	Vi kan også give disse lister med i funktions kald …	Technique
05-1-A5	**Opgave i at passe en arrayliste til en funktion som behandler indholdet.**	**Task**

05-2: Typer af iteration over ArrayList

05-2-T1	Vi kan både bruge for (og while) som vi gjorde i 05-1, men der findes en separat måde at løbe lister igennem på! Vis for-each	Technique/technology
05-2-A1	En opgave hvor de studerende skal anvende for-each	Task
05-2-A2	**Eventuelt en større opgave i stil med 05-1-A5, hvor den studerende skal reflektere på forskellen mellem de to metoder.**	**Task**

*Berører kort overloading

Implementing programming exercises to foster theory

1) Introduce (part of) technique
2) Task to learn-by-doing

} **Repeat**
until technique is covered
for all related techniques.

3) Attempt **technology** generation through **epistemological obstacles**.
4) Generate **theory** by institutionalising differences between **techniques**.

Implementing programming exercises to foster theory (in practice)

1) Introduce `Sets`
2) Task in `using Sets`

} **Repeat**
For `reading`, `modification`
For `Sets` and `Maps`.

3) `Set` and `Map` "*choice-and-implement*" exercises.
4) Outline `benefits` and `downsides` of structures, and `use cases`.

Implementing programming exercises to foster theory

Current (anecdotal) observations from this exercise design:

1) Students are more capable of keyword identification

[Nielsen, S. K., *Obstacles and strategies of Novice programmers*]

2) Students still lack strategies for translating from problem to code.

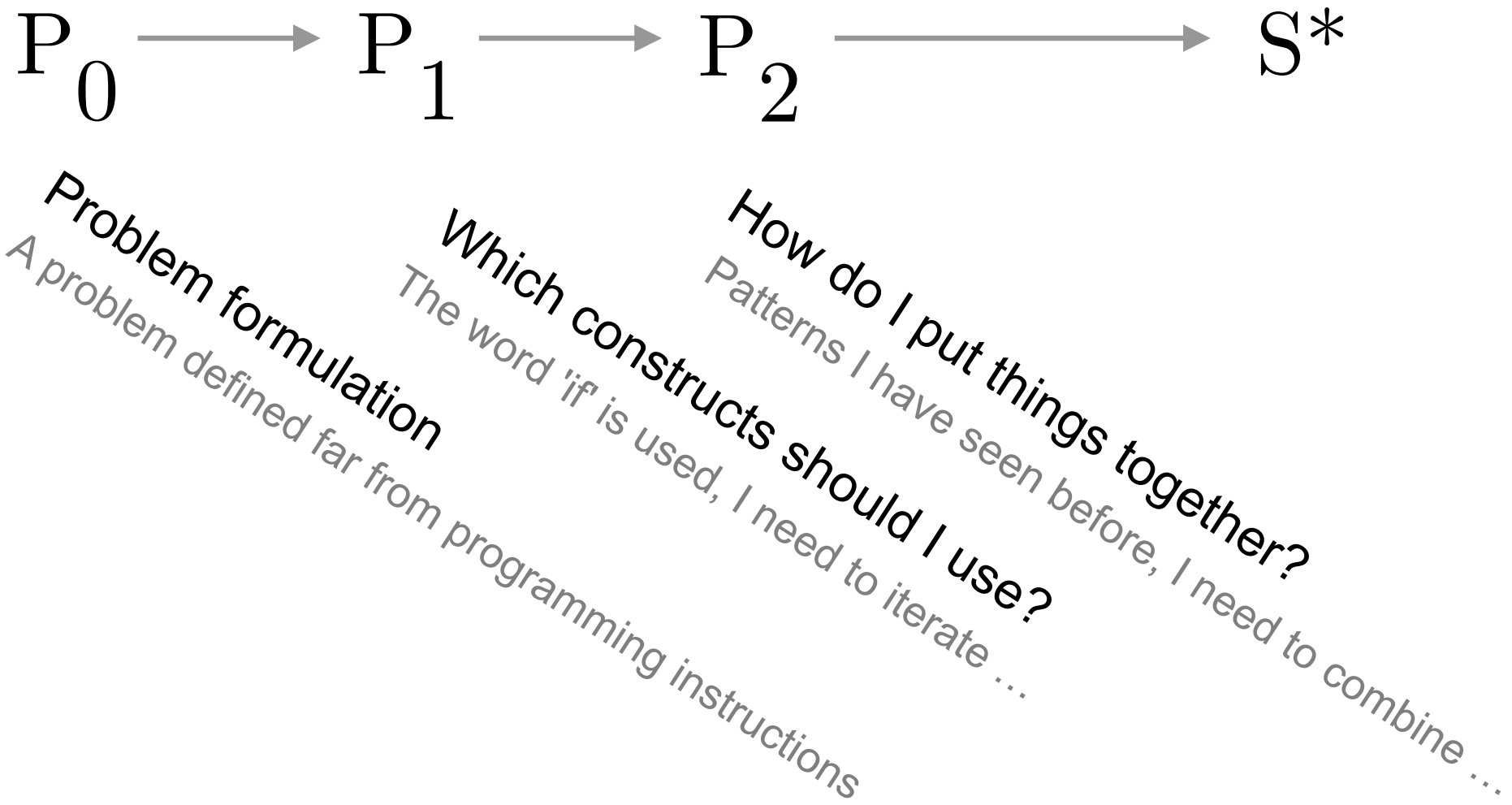
[Nielsen, S. K., *Obstacles and strategies of Novice programmers*]

Lack of strategies

Lack of strategies

Lack of strategies

Lack of strategies



Increasing focus on imperative programming

UGE	##	DATO	EMNE	AFLEVERINGER
01 35	01	Aug 30	Introduktion & Sekventielle konstruktioner	(Høsten begynder)
	02	Sep 01	Kontrol Strukturer: forgrening & Iteration	
02 36	03	Sep 06	Objekt-Orienteret programmering (Introduktion)	
	04	Sep 08	Funktioner	
03 37	05	Sep 13	Datastrukturer I	
	06	Sep 15	Datastrukturer II	
04 38	07	Sep 20	Datastrukturer III & Standardbiblioteker	
	08	Sep 22	Arrays	
05 39	09	Sep 27	Objekt-Orienteret programmering I	
	10	Sep 29	Objekt-Orienteret programmering II	
06 40	--	Okt 04	Obl. programmeringsprøve 1 kl 14:00 @ AUD1	
	11	Okt 06	Input/Output & Regulære udtryk	
07 41	12	Okt 11	Nedarvning I	
	13	Okt 13	Nedarvning II	(Høsten slutter)
42			Efterårsferie	

Increasing focus on imperative programming

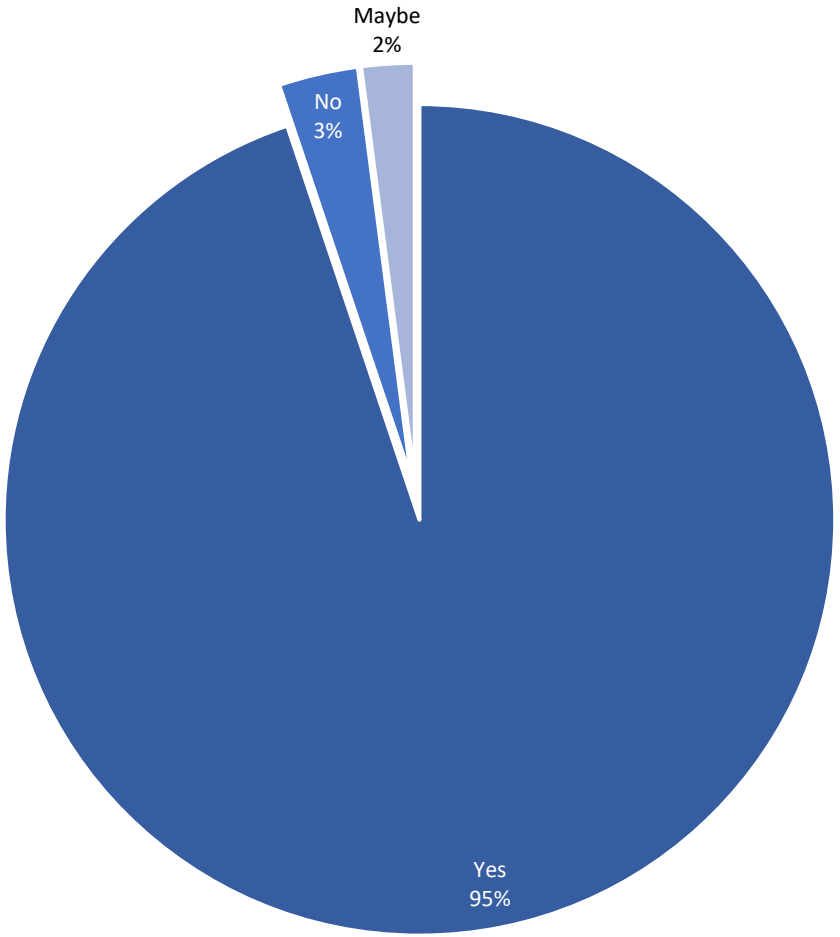
Current (anecdotal) observations from this change:

(And based on results of mandatory tests and exams)

1) Students are *better* at understanding program flow

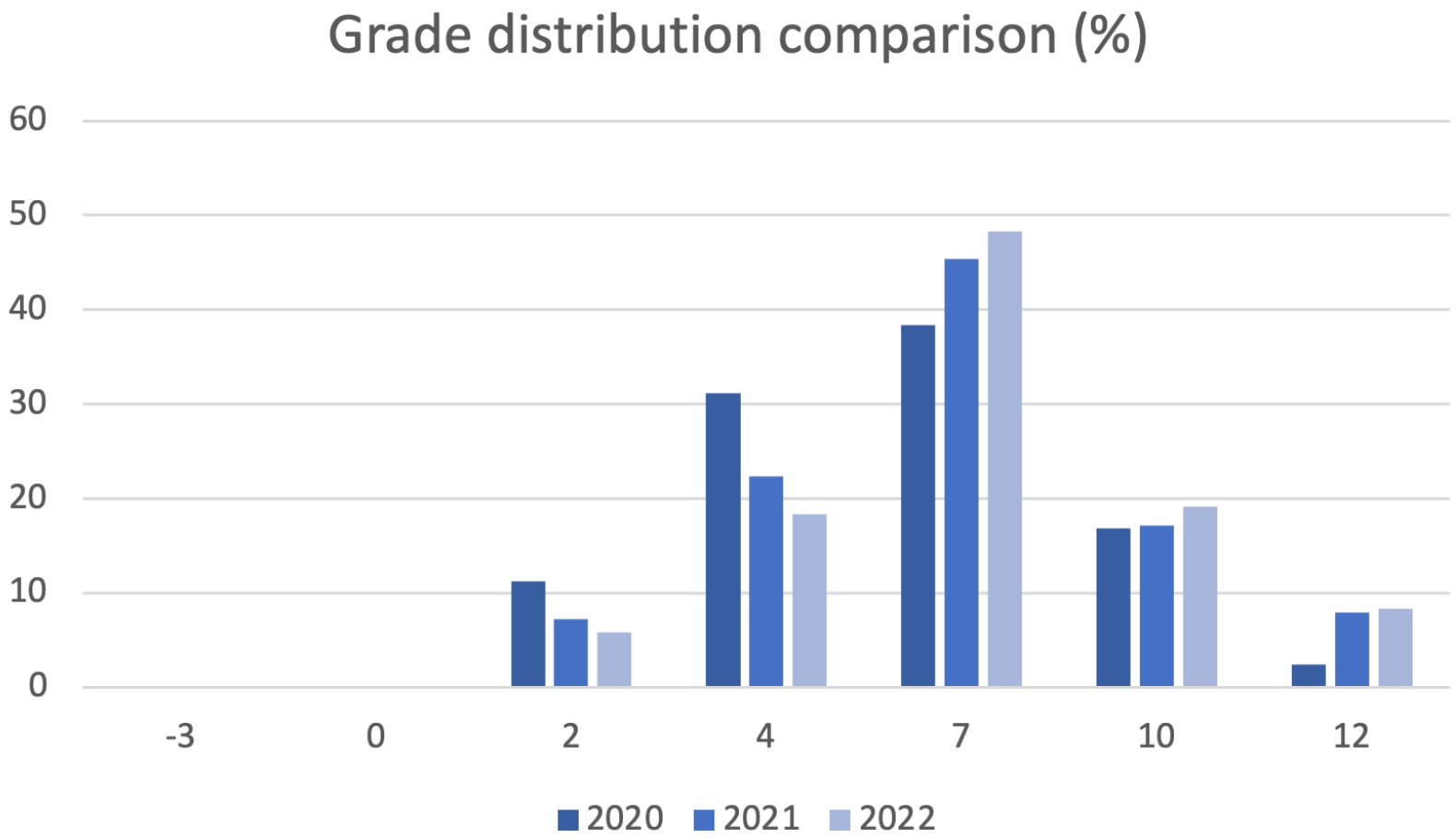
2) Some still lack basic imperative (algorithmic) understanding

Are the students learning to code?



*Do you feel like you are learning to code? (N=98)

Are the students learning to code?



(N_0=125, N_1=152, N_2=120)

Takeaways & Future work

- The mixed format encourages confrontation and training
- Obtaining data on student experience is essential for design
- We need to train strategies for programming more.

Questions?

Not approved for exam

N	*28*
1	8
2	4
3	8
4	6
5	2

Recording use (hours spent)