Computer Science & the Core

Our Charge:

Explore the arguments for adding computer science or other data science courses to the core. Which course(s)?

<u>This presentation:</u>
(Part I) Background on CS & the core
(Part II) A recommendation to add CS1 as a core graduation requirement.

(Part I) Background

The history of CS & the core

1986	All students must complete 6 units of coursework in computing from a list that includes “Introduction to Computing (CS10)”
1988	Computing requirement removed.
1995	CS10 was redesigned and became CS1
2009	CS 1 was redesigned to be more accessible and applicable for students from options outside of CS.
2010	CS 1 was taken by >200 students for the first time and became unofficially “pseudo-core”
2017	Students argued for the addition of CS1 (and potentially CS2) to the core as part of the SFC.

CS & the core today

To understand the interaction of CS and the core today, we looked at data from the 468 students that graduated in the past two years (2018 & 2019).

Note: Interest in CS has continued to grow in the following classes.

How many students take at least one introductory programming course?

94% (439/468) of students took or passed out of an intro programming course

CS1, CS11, ChE15, BE/Bi103, Ph20

47 passed out of CS1

How many students take at least one introductory programming course?

94% (439/468) of students took or passed out of an intro programming course

ACM	14/14	EE	51/55
App Physics	4/4	EAS	6/6
Astrophysics	11/12	Geobiology	3/5
Bioengineering	22/22	Geology	1/2
Biology	19/20	Geophysics	3/4
BEM	2/3	Materials Sci	2/2
Chem Eng	34/34	Math	18/25 (72%)
Chem	22/27 (81%)	Mech E	48/49
CS	121/121	Physics	51/55
Econ	1/1	Planetary Sci	5/5

How many students take at least one introductory programming course?

94% (439/468) of students took or passed out of an intro programming course

72% (338/468) of students took or passed out of more than one intro programming course

What courses do students take?

CS1 was taken or passed out of by 418/468 (90%) of students

21 took a different intro programming course and did not take CS1

What courses do students take?

CS1 was taken or passed out of by 418/468 (90%) of students

ACM	14/14	EE	51/55
App Physics	4/4	EAS	5/6
Astrophysics	11/12	Geobiology	3/5
Bioengineering	22/22	Geology	1/2
Biology	19/20	Geophysics	2/4
BEM	2/3	Materials Sci	2/2
Chem Eng	27/34 (79%)	Math	18/25 (72%)
Chem	17/27 (63%)	Mech E	48/49
CS	121/121	Physics	38/55 (69%)
Econ	1/1	Planetary Sci	5/5

Students take option-specific courses.

What courses do students take?

CS1 was taken or passed out of by 418/468 (90%) of students
CS 11 was taken 462 times (students take this course multiple times)
ChE15 is taken by 35/468 (7.5%) → Nearly entirely ChemE majors
BE/Bi103a is taken by 46/468 (10%) → Mix of Bioengineering / CS / Chem majors
Ph 20 is taken by 53/468 (11%) → Nearly entirely Physics majors

CS1 background is assumed in upper-level computational courses.
Only 11 students took upper-level CS courses involving programming without first taking or passing out of CS1.

Learning programming without taking CS1 limits further study of CS.

Do students already know programming when they arrive?

CS 1 is passed out of at the same rate as Ph 1a (~20 students/year) and Ma 1a (~25-30 students/year).

Year	Attempted	Passed
2019	40	12
2018	51	16
2017	69	21
2016	50	13

Many students are borderline. Adding a second track to CS1 (like Ph1 and Ma1), would be helpful for teaching to students with different backgrounds.

When do students take introductory programing courses?

Course	Passed out	Freshman	Sophomore	Junior	Senior
CS1	47	323	30	10	8
CS11	-	197	121	55	89
ChE15	-	1	33	1	0
BE/Bi 103a	-	0	6	20	20
Ph 20	-	13	26	8	6

Could assume CS1 or CS11 as a prerequisite and build from there

Puts pressure on freshman fall core schedules

What intermediate CS courses put pressure on freshman core?

CS2 is taken by 225/468 (48%) of students.

The class was recently redesigned, current enrollment is 160 (+31%)

ACM	8/14	EE	32/55
App Physics	3/4	EAS	4/6
Astrophysics	2/12	Geobiology	0/5
Bioengineering	2/22	Geology	0/2
Biology	2/20	Geophysics	0/4
BEM	0/3	Materials Sci	2/2
Chem Eng	6/34	Math	10/25
Chem	5/27	Mech E	12/49
CS	121/121	Physics	19/55
Econ	0/1	Planetary Sci	0/5

What intermediate CS courses put pressure on freshman core?

CS2 is taken by 225/468 (48%) of students. Current enrollment is +31%.
CS21 is taken by 201/468 (43%) of students. Current enrollment is +17%.

ACM	6/14	EE	14/55
App Physics	2/4	EAS	2/6
Astrophysics	0/12	Geobiology	0/5
Bioengineering	3/22	Geology	0/2
Biology	3/20	Geophysics	0/4
BEM	0/3	Materials Sci	1/2
Chem Eng	4/34	Math	11/25
Chem	2/27	Mech E	10/49
CS	121/121	Physics	23/55
Econ	0/1	Planetary Sci	0/5

What intermediate CS courses put pressure on freshman core?

CS2 is taken by 225/468 (48%) of students. Current enrollment is +31%.
CS21 is taken by 201/468 (43%) of students. Current enrollment is +17%.

Course	Passed out	Freshman	Sophomore	Junior	Senior
CS1	47	323	30	10	8
CS2	-	119	58	32	16
CS11	-	197	121	55	89
CS21	-	98	53	36	14
ChE15	-	1	33	1	0
BE/Bi 103a	-	0	6	20	20
Ph 20	-	13	26	8	6

43 students (~40%) took both during winter term freshman year in the 2 years of the data. This year 47 students took both (>100% growth). Taking both courses means deferring core requirements.

Executive summary (Part I)

- 94% of students take or pass out of an intro programming course
- Options where students are the least likely to take an intro programming course are: Chemistry (81%) and Math (72%).
- CS1 is frequently taken freshman year, which puts extreme pressure on the fall term of the freshman core.
- Most students learn programming via CS1 but a few only take an option-specific course, which typically happens after freshman year.
- Students that do not take CS1 and learn programming via an option-specific course are not likely to continue to study computation.
- CS2 & CS21 are putting increasing pressure on the winter term of the freshman core.

Our Charge:

Explore the arguments for adding computer science or other data science courses to the core. Which course(s)?

Why add computer science to the core?

Philosophical Justification:

Caltech strives to create leaders in science and technology across industry & academia. Computational thinking is an essential perspective for such leaders today, regardless of their field. Caltech should ensure all students graduate with this skill.

Why add computer science to the core?

Pedagogical Justification:

Computational tools are fundamental across disciplines and are increasingly a part of intermediate & advanced courses in all areas. Having programming in the core ensures faculty can assume programming knowledge in every course, allowing courses to include modern computational topics without losing time to teaching introductory programming.

Why add computer science to the core?

Practical Justification:

The students have already decided that programming is a crucial piece of their training and voted with their feet. Currently, because CS1 is not recognized as part of the core, students and freshman advisers are often in conflict about course selection and parts of the core are deferred.

CCSC Recommendation: Add CS1 as a flexible core graduation requirement.

CS1 must be taken before graduation. Taking it freshman year is recommended but not required.

A placement exam must be available to allow students to pass out of CS1.

CS1 must be offered multiple terms (likely Fall & Spring if two terms).

EAS/CMS should be given resources to support this additional load.

Why require CS1 specifically (as opposed to a menu of programming courses)?

- CS1 provides coverage of introductory programming and some broader introduction to computational thinking/algorithms.

- Having a common requirement ensures a uniform background that can be assumed in courses across areas. Maintaining consistency is challenging across courses in different options/divisions.

- CS1 ensures students are prepared to continue to intermediate CS courses, if desired. Currently other courses do not provide this preparation.

- CS1 can be most easily integrated into freshman core schedules since it is already quasi-core.

If students already take CS1, why does it need to be in the core?

- Pedagogically, making CS1 a core requirement allows professors to assume knowledge of programming in all intermediate and upper classes on campus, making it easier to include computational perspectives.

- Practically, students take CS1 freshman year in overwhelming numbers and this negatively impacts the core and creates conflicts with freshman advisers. Including it in the core and offering it multiple terms allows for clear guidelines on scheduling for freshman year and eases pressure on the fall core term.

- Philosophically, it is important for Caltech to recognize the foundational role of computational thinking and ensure its students obtain that knowledge before graduation. The core is a strong signal internally and externally.

How does this address the challenges CS1 creates for the core currently?

- By offering CS1 multiple terms, it reduces pressure on the freshman core compared to the status quo dramatically.

  But, CS2 and CS21 provide additional pressure on the winter term. More discussion is needed in order to address that concern.

- By explicitly allowing CS1 to be taken any time before graduation (rather than requiring it during freshman year), it does not prevent taking major-required courses during freshman year.

  But, for other courses to be able to assume CS1 as background, it is important that students take it early. If needed, incentives around P/F could be used for this purpose (as done for other core courses).

What do students think about this proposal?

Adding CS1 to the core was a topic at the 2017 SFC. At that point students argued strongly in favor of adding CS1 to the core and student leaders have been actively engaging with faculty on this issue since then.

In considering this recommendation, additional discussions with students are needed.

What do freshman advisers think about this proposal?

Opinions vary widely but are largely positive. The bulk of responses are either:

- Strong support. "basic knowledge of programming is essential for functioning scientist/engineer" "great idea"
- Support but worry about increasing the size of the core. "enthusiastically support but ask whether something should be removed" "support the idea but sad to see something added without taking something away"

But, not all are supportive:

- Do not support & am worried about trends of growth in CS impacting those not in CS. "worry that we are becoming the California Institute of CS" "Are we happy with the change in nature of Caltech?" "It will make it harder for [non-CS] first years to get started [on] their chosen majors"

In considering this recommendation, additional discussions with faculty are needed.

Executive summary (Part II)

The CCSC recommends adding CS1 as a core graduation requirement as long as it can be offered multiple terms.

More discussion with faculty and students should be undertaken as this recommendation is considered.

Adopting this recommendation would not eliminate pressure on the core from enrollment in CS courses. Further consideration should be given to managing pressure on the winter term from CS2 & CS21.

Course descriptions

CS 1. Introduction to Computer Programming. *9 units (3-4-2); first term.* A course on computer programming emphasizing the program design process and pragmatic programming skills. It will use the Python programming language and will not assume previous programming experience. Material covered will include data types, variables, assignment, control structures, functions, scoping, compound data, string processing, modules, basic input/output (terminal and file), as well as more advanced topics such as recursion, exception handling and object-oriented programming. Program development and maintenance skills including debugging, testing, and documentation will also be taught. Assignments will include problems drawn from fields such as graphics, numerics, networking, and games. At the end of the course, students will be ready to learn other programming languages in courses such as CS 11, and will also be ready to take more in-depth courses such as CS 2 and CS 4. Instructor: Vanier.

CS 11. Computer Language Lab. *3 units (0-3-0); first, second, third terms. Prerequisites: CS 1 or instructor's permission.* A self-paced lab that provides students with extra practice and supervision in transferring their programming skills to a particular programming language. The course can be used for any language of the student's choosing, subject to approval by the i[illegible]cises guide the student t[illegible]r her familiarity, experien[illegible] students may propose their own programming project as the target demonstration of their new language skills. This course is available for undergraduate students only. Graduate students should register for CS 111. CS 11 may be repeated for credit of up to a total of nine units. Instructors: Blank, Pinkston, Vanier.

3 units, learn a new programming language (C, C++, Java, ...)

CS 2. Introduction to Programming Methods. *9 units (3-5-1); second term. Prerequisites: CS 1 or equivalent.* CS 2 is a demanding course in programming languages and computer science. Topics covered include data structures, including lists, trees, and graphs; implementation and performance analysis of fundamental algorithms; algorithm design principles, in particular recursion and dynamic programming; Heavy emphasis is placed on the use of compiled languages and development tools, including source control and debugging. The course includes weekly laboratory exercises and projects covering the lecture material and program design. The course is intended to establish a foundation for further work in many topics in the computer science option. Instructor: Blank.

CS 21. Decidability and Tractability. *9 units (3-0-6); second term. Prerequisite: CS 2 (may be taken concurrently).* This course introduces the formal foundations of computer science, the fundamental limits of computation, and the limits of efficient computation. Topics will include automata and Turing machines, decidability and undecidability, reductions between computational problems, and the theory of NP-completeness. Instructor: Umans.

ChE 15. Introduction to Chemical Engineering Computation. *9 units (1-4-4); first term. Prerequisites: Ma 2 (may be taken concurrently).* Introduction to the solution of engineering problems through the use of the computer. Elementary programming in Python is taught, and applied to solving chemical engineering problems in data analysis, process simulation, and optimization. No previous knowledge of computer programming is assumed. Instructor: Flagan.

Ph 20. Computational Physics Laboratory I. *6 units (0-6-0); first, second, third terms. Prerequisites: CS 1 or equivalent.* Introduction to the tools of scientific computing. Use of numerical algorithms and symbolic manipulation packages for solution of physical problems. Python for scientific programming, Mathematica for symbolic manipulation, Unix tools for software development. Instructors: Mach, Weinstein.

BE/Bi 103 a. Introduction to Data Analysis in the Biological Sciences. *9 units (1-3-5); first term. Prerequisites: Bi 1, Bi 1x, Bi 8, or equivalent; or instructor's permission.* This course covers tools needed to analyze quantitative data in biological systems. Students learn basic programming topics, data organization and wrangling, data display and presentation, basic image processing, and resampling-based statistical inference. Students analyze real data in class and in homework. Instructor: Bois.