Timeless Lessons from Teaching Programming, or Not

Paul Dickson
In The Beginning...

There was Charles Babbage and the Analytical Engine which led to computers
In The Next Beginning...

There was Alan Turing with algorithms and computations which led to computer science.
Who Deserves the Blame?
Opening Limerick

➔ There once was a group led by McCracken
➔ That found intro students not good at hackin
➔ One third of them failed
➔ Our tears they all trailed
➔ Despite this our efforts do not slacken
66.7% Pass rate in introductory CS

After introductory CS, students should be able to:
- Abstract the problem from its description
- Generate sub-problems
- Transform sub-problems into sub-solutions
- Re-compose the sub-solutions into a working program
- Evaluate and iterate

Most students could not
A Brief and Inaccurate History of Programming
Grace Hopper - Bugs

Photograph courtesy of Naval Surface Warfare Center, Dahlgren, Virginia, USA
Grace Hopper actually introduced the concept of RAID to computer science for reducing errors when she said “someone use the damn Raid to kill those bugs!”.
RAID

- Redundant
- Array of
- Independent
- Disks
RAID

- Redundant
- Array of
- Independent
- Disks
To the Programming Languages!

🔗 Fortran
To the Programming Languages!

- Fortran
  - DITRAN (DIagnostic FORTRAN)
  - WATFOR (WATERloo FORTRAN)
  - WATFIV (WATFOR’s successor)
  - PUFFT (Purdue University Fast FORTRAN Translator)
To the Programming Languages!

- Fortran
- Logo
To the Programming Languages!

- Fortran
- Logo

Logo is to blame for Turtle Graphics
To the Programming Languages!

- Fortran
- Logo
- Basic
- Pascal

Teaching Languages
To the Programming Languages!

- Fortran
- Logo
- Basic
- Pascal
- C

Teaching pointers in CS1 is a great idea!
- no one ever
To the Programming Languages!

- Fortran
- Logo
- Basic
- Pascal
- C
- C++
- Java
- Python
To the Programming Languages!

- Fortran
- Logo
- Basic
- Pascal
- C
- C++
- Java
- Python
- Scratch
Scratch
Programming Trends

🔗 https://www.youtube.com/watch?v=Og847HVwRSI&feature=youtu.be
After introductory CS, students should be able to:

- Abstract the problem from its description
- Generate sub-problems
- Transform sub-problems into sub-solutions
- Re-compose the sub-solutions into a working program
- Evaluate and iterate

Most students could not
Teach Yourself Programming in Ten Years

- Peter Norvig
- https://norvig.com/21-days.html
Learning to Program is Easy

Andrew Luxton-Reilly
Department of Computer Science
University of Auckland
Auckland, New Zealand
andrew@cs.auckland.ac.nz

ABSTRACT
The orthodox view that "programming is difficult to learn" leads to uncritical teaching practices and poor student outcomes. It may also impact negatively on diversity and equity within the Computer Science discipline. But learning to program is easy — so easy that children can do it. We make our introductory courses difficult by establishing unrealistic expectations for novice programming students. By revisiting the expected norms for introductory programming we may be able to substantially improve outcomes for novice programmers, address negative impressions of disciplinary practices and create a more equitable environment.

Keywords
novice; programming; computer science education; standards; expectations; learning outcomes; curriculum; CS1

1. INTRODUCTION
Learning to program is easy. Almost everyone can learn how to write their first computer program with minimal effort. Although the syntax of each programming language differs, producing a simple program that can print out a string of letters (such as that presented in Figure 1) is trivial to write, understand, reproduce and modify.

```python
print('Hello World!')
```

Figure 1: A simple Python program

Learning to program is hard however. Novice programmers suffer from a wide range of difficulties and deficits. Programming courses are generally regarded as difficult, and often have the highest dropout rates. [26]

This view of computer programming as a difficult skill to learn may have implications for equity and diversity. It may encourage students to engage in activities that are not conducive to learning, it is potentially unfair to students and may also lead to research practices that focus on student shortcomings rather than curriculum deficiencies.

This paper challenges the orthodox view that computer programming is innately difficult to learn. Instead, an alternative explanation is proposed to explain the challenges faced by students as they learn to program.

2. STUDENT PERFORMANCE
The belief that learning to program is difficult appears to be widespread among teachers and researchers involved in Computer Science Education (e.g. see [5, 10, 26, 35]). Despite the evidence, the orthodoxy is firmly held with the...
Expectations

- Algebra 6+ years of math
- Calculus 10+ years of math
- Physics 9+ years of math
Expectations

- Algebra 6+ years of math
- Calculus 10+ years of math
- Physics 9+ years of math
- Programming 0 years algorithms and computation
What to do

- Accept 33% failure rate
- Lower expectations and reduce knowledge upon graduation
- Start earlier
What to do

- Accept 33% failure rate
- Lower expectations and reduce knowledge upon graduation
- Start earlier Ireland is already starting
Timeless Lessons Learned

- Number your punch cards
- Check for insects
- Alias `rm` to `rm -i`
- Don’t use tiff images when swapping between big and little endian computers
- Restart your Windows 95 machine daily
Actual Thoughts

➔ What you do matters