Teaching of Computing to Mathematics Students

Programming and Discrete Mathematics

Jack Betteridge, *James Davenport*, Melina Freitag, Willem Heijltjes, Stef Kynaston, Gregory Sankaran, Gunnar Traustason

Computer Science, Mathematical Sciences, postgraduates; University of Bath

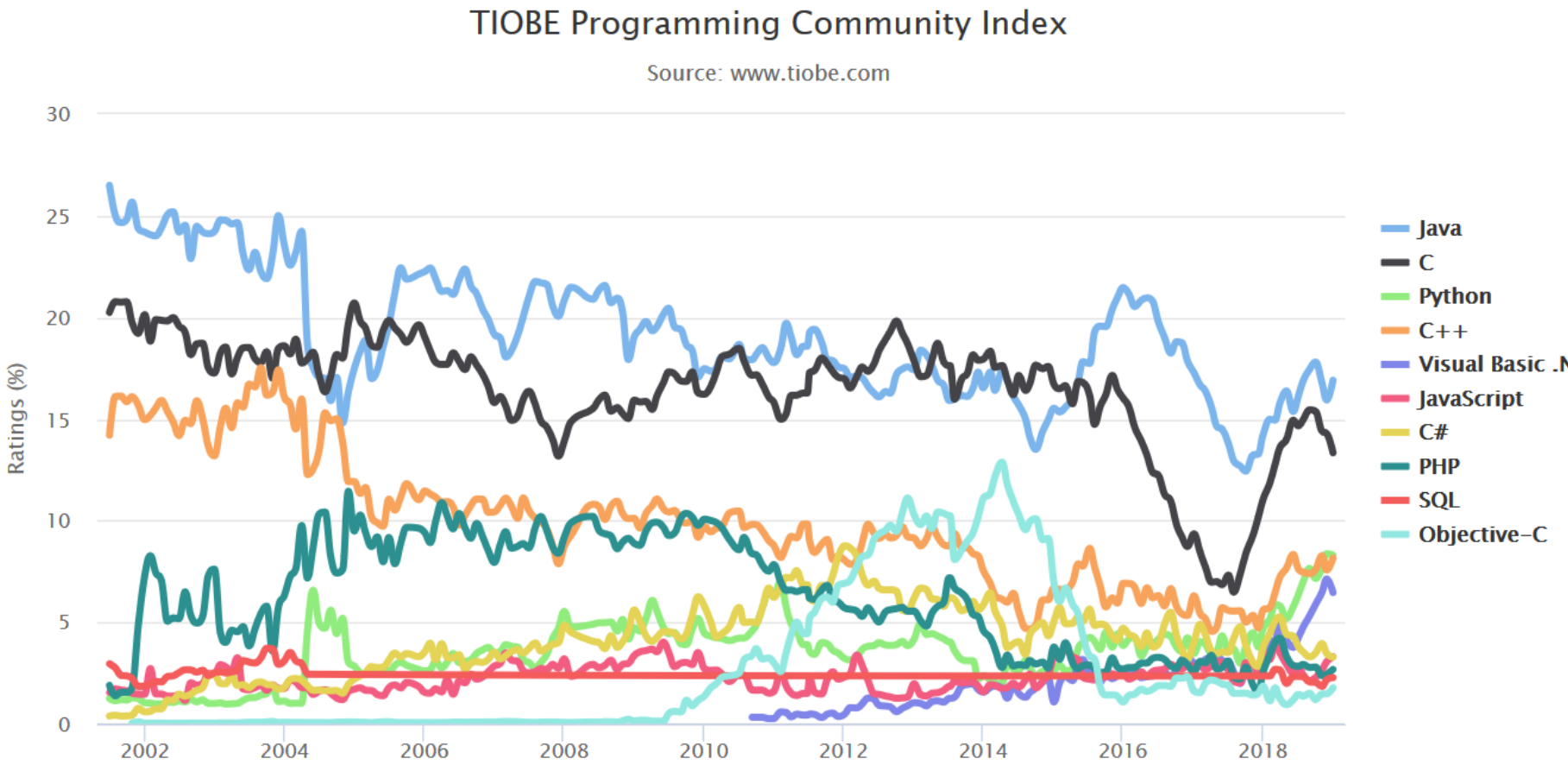
9 January 2019

Background

- Maths at Bath has always taught programming (Fortran, then C in 1984, . . .)
- In 2001 Computing split from Maths, and both went to Java (separate courses, different credit count)
- Java is a popular language for teaching [MCD17] and in industry [TIO18]
- Pretty traditional programming courses, assessed by major individual coursework and examination
- Significant overlap in the teaching and tutoring staff, very similar students in terms of grades, and many would do "thick sandwich" courses.

! Yet the Computing course was a success, and the Maths course was not

Language Popularity



[TIO18]; MATLAB is 11th at 1.502%

Reasons?

- The main weaknesses of the Mathematics course, frequently identified by students, were the lack of apparent relevance and the lack of connection with the syllabus.
- Java was used nowhere else in the Mathematics course (MATLAB or R were)
- Comparatively few sandwich students used Java
- The style of tutorial/assessment (big coursework done in open-plan laboratories) was very different from the weekly exercise sheets and personal feedback of the mathematics courses
- Students would leave the coursework to the last minute, find they couldn't actually program, and fail (sometimes a 40% failure rate)

Solutions?

- Replace "Programming" (6 ECTS credits in one semester) by "Programming and Discrete Mathematics" (12 over two) — this gives the programming longer, and makes it seem less separate
- Change the language to MATLAB — but retain the programming ethos: not package usage
- Split the open-plan lab into divisions, with one tutor to (fixed) students (initially 13:1, then 9:1)
- Pressurise timetabling into assigning all the labs on the same day, later in the week than the lectures
- Assign weekly pass/fail "tickable" exercises, and schedule the students' weekly work

What you should do this week (Week 6 of Semester1)

- **before** the lab, if you can, do the quiz and submit Tickable 5 in a ZIP as in the instructions in the quiz.
- If you didn't do this last time, get Tickable 4 ticked **by your divisional tutor**
- Get Tickable 5 ticked **by your divisional tutor**

even if you did the quiz successfully: the tutor still has to see it and comment on the code.

- Work on Tickable 6

Note that this is due in by **Wednesday** 14th: you are expected to do more work than just the labs

N.B. Catch-up classes for those few of you who need it:

Wed. 7 09:15 and 10:15 EB 0.7

Tue. 13 11:15 and 12:15 1E 3.9

Thu. 15 14:15 and 15:15 CB 5.13

Programming *and* Discrete Mathematics??

The aim is to teach an integrated set of materials:

1. Induction: taught as "mathematics", but used in correctness and complexity proofs of recursive algorithms.
2. $\mathcal{O}$-notation: taught as "mathematics", but used in complexity of algorithms.
3. Graphs: taught as "mathematics", but used as examples for data structures

All this is in Semester 1, and indeed the students comment that Semester 2 is less connected — we need to join up the error-correcting codes material better: reinforces Algebra, but the link to computing needs to be made.

Teaching Programming

1. Begin by writing functions, rather than scripts
2. Introduce error handling (`try...catch`)
3. Explicit marks (20%) for writing tests

! Due in two weeks before the actual code is due in

* This shifted where the misconceptions were discovered, and reduced their impact on the total marks

4. Teach (Semester 2) the object-oriented side of MATLAB — it's the most commonly-taught paradigm [MCD17]
5. Use automated grading (but style marks are manual, 20%)

Does it work?

- ? The complaints about relevance are still there: less strident, and final year students at SSLC tend to say "Oh yes it is".
- $+_1$ Failure rate is drastically down: in 2017/18 no student failed this module only.
- $+_2$ The first Numerical Analysis course no longer disappears under the weight of teaching programming. In particular the students have already met floating-point numbers.
- $+_3$ there's enough MATLAB in the first semester to allow the second semester statistics course to issue a one-page MATLAB/R conversion sheet, rather than spend several weeks explaining R.
- ? Sandwich student relevance seems to be unchanged.

Future

The University is forcing a major re-design of all programmes.

+ It is inconceivable that Maths will stop teaching programming, the question debated is *how* to teach it.
* The general feeling is "we are fifty years ahead of [Bon18, Recommendation 8]"

But the language to be taught has implications for Numerical Analysis ($+_2$) and Statistics ($+_3$).

- Python is being mooted as a plausible option (it probably wouldn't have been in 2009).

P. Bond.
An Independent Review of Knowledge Exchange in the Mathematical Sciences.
https://epsrc.ukri.org/newsevents/news/mathsciencereview/, 2018.

E. Murphy, T. Crick, and J.H. Davenport.
An Analysis of Introductory Programming Courses at UK Universities.
The Art, Science and Engineering of Programming, 1(2):18–1–18–23, 2017.

TIOBE.
TIOBE Index for December 2018.
http://www.tiobe.com/tiobe_index, 2018.