Programming as a mathematical activity

Peter Rowlett

Sheffield Hallam University

@peterrowlett

p.rowlett@shu.ac.uk

27th June 2019

Programming in the Undergraduate Mathematics Curriculum, Middlesex University

Background

- I took over a second-year programming elective in 2018/19 from two people who were retiring.

Background (Sept 2018)

Background (Oct 2018)

Apologies for any Cross Posting

Dear All,

I am delighted to announce that the IMA has launched the second call in its *Higher Education Teaching and Learning Initiative*. The call aims to support the development of a national network for all involved in teaching and supporting learning within the mathematical sciences in higher education with the aim of further encouraging collaboration and facilitating the sharing of effective practice. Building upon the IMA's Academic Representatives Scheme, the *Higher Education Teaching and Learning Series* provides an opportunity for higher education mathematical sciences departments to request supporting funds for facilitating a teaching and learning related workshop. Whilst any well-argued theme for a workshop related to the learning and teaching of mathematics within UK higher education will be considered, we are particularly keen to encourage proposals that share good practice in the following areas:

- Developing highly employable graduates within the mathematical sciences.
- Enhancing feedback practices in the mathematical sciences.
- Developing undergraduate programming and coding skills in the mathematical sciences.
- Moving beyond the examination – alternative approaches to assessment in the mathematical sciences.

The module I inherited

The module I inherited

- 'Mathematical Programming for Excel and the Web'.
- 20 credits, optional in the second year of BSc Maths.
- Ran for many years (mostly as two 10-credit modules).
- Followed on from first year:
  - compulsory section of a core module on Excel-VBA programming;
  - optional 10-credit 'Basic Computer Programming' on Visual Basic (which was not a prerequisite module).

Excel section

- VBA programming. Intended learning outcomes:
  - use features of Excel, including macros and software objects, to carry out a range of mathematical and/or scientific data processing tasks and to develop, test and implement a substantial program in Excel.
- Started with a refresher of first year topics.
- Then half a term of programming exercises.
- Then half a term of individual project.

Web section

- PHP, HTML, databases. Intended learning outcomes:
  - create a variety of web applications both client-side using appropriate scripting tools, and server-side in PHP, using advanced HTML and typical web server-based tools to generate dynamic web pages.
- Half a term of programming exercises.
- Then half a term of individual project.

Assessment

- Excel:
  - complete set of exercises (10%);
  - individual project (40%).
- Web:
  - complete set of exercises (10%);
  - individual project (40%).

What I changed

What I changed

- First year: we retired ‘Basic Computer Programming’.
- Title: ‘Programming with Mathematical Applications’.
- Learning outcomes:
  1. select and use appropriate programming techniques to solve mathematical problems in various application contexts;
  2. develop, test and implement a substantial mathematically-based programming project.
- De-emphasise particular technologies.
- Increase taught content, de-emphasise individual projects.

Topics 2018/19

- Semester 1:
  - Programming fundamentals and mathematical functions.
  - Excel/graphics.
- Semester 2:
  - Web, databases, etc.
  - Individual project.

Programming fundamentals and mathematical functions

Programming fundamentals and mathematical functions

- To teach from scratch the basics of programming, in a new language.
- Also to emphasise mathematical aspects.
- To lay foundations for the rest of the module.

Programming vs. coding

- *Programming* uses program constructs to express an algorithm for a computer to interpret.
- There are lots of common program constructs (if statements, loops, functions, and so on).

Programming vs. coding

- *Programming* uses program constructs to express an algorithm for a computer to interpret.
- There are lots of common program constructs (if statements, loops, functions, and so on).
- *Coding* is the business of writing a program in the syntax of a particular language.

Programming vs. coding

- *Programming* uses program constructs to express an algorithm for a computer to interpret.
- There are lots of common program constructs (if statements, loops, functions, and so on).
- *Coding* is the business of writing a program in the syntax of a particular language.
- The purpose is to learn programming; coding in a particular language is necessary but secondary.

“A secret” (from my module notes)

- “In some sense, no one really knows how to code.
- “If you have learned a few programming languages and are using one you haven’t used in a while, it is very common to make syntax mistakes. These essentially don’t matter, because they are the sort of mistakes that are easily fixed by looking things up. . . .
- “So please don’t feel like you are failing to progress if you have to search for help with syntax.
- “From the point of view of this module (at least), learning how a for loop works and what it can be used for is very important, but remembering the precise syntax to make one happen in [a particular language] is secondary.
- “If you were regularly working in a particular programming language, you would be much more capable of remembering the syntax. If you stopped for a while, this would degrade, but what is important – how and why program constructs work – would remain in your understanding and be usefully transferred to other programming languages.”

Teaching directed to programming that isn't coding

Propositional logic

- Propositions and predicates, NOT, AND, OR, XOR, truth tables.

Propositional logic

- Propositions and predicates, NOT, AND, OR, XOR, truth tables.
- Example exercise:
  - Consider the following conversation.

    WAITER: Would you like tea or coffee?

    CUSTOMER: Tea, please.

    WAITER: Would you like milk or sugar?

    CUSTOMER: Both, please.

    Which type of 'or', OR or XOR, is being used by the waiter in each of the two questions?

Propositional logic

- Propositions and predicates, NOT, AND, OR, XOR, truth tables.
- Example exercise:
  - Computers do what you tell them to do, not what you really *meant* them to do, so when programming it helps to think strictly logically. In natural language, it is often irritating to communicate in this way. Consider the following unhelpful dialogue. Is person 2 right?

    PERSON 1: Is the window open or closed?

    PERSON 2: Yes.

Algorithms

- Algorithms as a sequence of precise instructions.
- Example task: “Write algorithms for the following in order. Once you have written them, show them to someone else. When you are shown someone else’s, try to be as pedantic as possible and willfully misinterpret the instructions. The point is to think about how to really tightly specify a sequence of actions.

Algorithms

- Algorithms as a sequence of precise instructions.
- Example task: “Write algorithms for the following in order. Once you have written them, show them to someone else. When you are shown someone else’s, try to be as pedantic as possible and willfully misinterpret the instructions. The point is to think about how to really tightly specify a sequence of actions.

    1. “Putting on a t-shirt.

Algorithms

- Algorithms as a sequence of precise instructions.
- Example task: “Write algorithms for the following in order. Once you have written them, show them to someone else. When you are shown someone else’s, try to be as pedantic as possible and willfully misinterpret the instructions. The point is to think about how to really tightly specify a sequence of actions.

    1. “Putting on a t-shirt.
    2. “Making a cup of tea.

Algorithms

- Algorithms as a sequence of precise instructions.
- Example task: "Write algorithms for the following in order. Once you have written them, show them to someone else. When you are shown someone else's, try to be as pedantic as possible and willfully misinterpret the instructions. The point is to think about how to really tightly specify a sequence of actions.

    1. "Putting on a t-shirt.
    2. "Making a cup of tea.
    3. "Differentiating the product of two functions, $f(x) = u(x)v(x)$.

Algorithms

- Algorithms as a sequence of precise instructions.
- Example task: "Write algorithms for the following in order. Once you have written them, show them to someone else. When you are shown someone else's, try to be as pedantic as possible and willfully misinterpret the instructions. The point is to think about how to really tightly specify a sequence of actions.

    1. "Putting on a t-shirt.
    2. "Making a cup of tea.
    3. "Differentiating the product of two functions, $f(x) = u(x)v(x)$.
    4. "Deciding whether a number is prime."

More logic: implication

- $(p \implies q)$.

More logic: implication

- $(p \implies q)$.
- Example

Suppose a train has a safety system where pulling an alarm handle puts the brakes on. Two relevant propositions are

p : "The alarm handle is pulled"

q : "The train stops"

The condition which describes the system working properly is

$p \implies q$: "If the alarm handle is pulled, then the train stops."

More logic: implication

- $(p \implies q)$.
- Example

Suppose a train has a safety system where pulling an alarm handle puts the brakes on. Two relevant propositions are

p : "The alarm handle is pulled"

q : "The train stops"

The condition which describes the system working properly is

$p \implies q$: "If the alarm handle is pulled, then the train stops."

- What do the following situations tell us about the truth of $p \implies q$?
  (a) The handle has been pulled but the train has not stopped.
  (b) The handle has not been pulled and the train has not stopped.
  (c) The handle has not been pulled but the train has stopped anyway.

More logic: implication

- Lewis Carroll sorites (crocodile-managing babies, teachable kittens, and so forth).
- Then: if statements in Python.

Why Python?

- It's well-used.
- It is a nice, clean language in which to teach programming basics without hangups from VBA.
- I've not used it, but I've heard good things.
- It does some fab things, but I tried to focus my teaching on the fundamentals that are common with other languages.

Rest of topic

Defining variables and using them in if statements	variables-and-if.html	
Strings	strings.html	Defining variables
Tuples, Lists and Dictionaries	lists-etc.html	Defining variables
For loops	loops-for.html	While loops
While loops	loops2-while.html	For loops
Commenting and documentation	commenting.html	
Functions	fn.html	Modules
Modules	modules.html	Functions
Recursion	recursion.html	Recursion
Error handling	errors.html	
User input	uinput.html	
File input and output	fileio.html	

End of topic

- 'Python in practice'.
- Examples – Write a program that:
  - "asks the user to input a string and outputs all permutations of the characters in that string."
  - "takes as input a matrix and outputs its eigenvalues."
  - "asks the user to input a complex number in Cartesian form and outputs the same number in polar coordinates."
  - "asks the user to input a number and outputs a message indicating whether that number is adundant or not."
  - "that simulates flipping a coin 1000 times and outputs a count of how many heads occurred and what was the longest chain of heads in a row."
  - "generates random passwords that include a mixture of lowercase letters, uppercase letters, numbers and symbols."
  - "plays Paper-Scissor-Stone against the user."

Coursework 1

- A really tightly-specified list of functions.
- e.g. "Write a function called 'question3' that takes as its input a number x and returns the value of $\frac{x^2+3}{x-4}$."
- I wrote a script that looped through the student files querying the functions and producing feedback.

Coursework 1

- Marking script for $\frac{x^2+3}{x-4}$:
  - Checks no syntax errors, that 'question3' is provided, then runs `question3(9)` and `question3(-4)`.
  - Then checks it:
    - doesn't throw an exception;
    - returns a value rather than printing;
    - returns a number rather than a string;
    - returns the right values.

Coursework 1

- Marking script for $\frac{x^2+3}{x-4}$:
  - Checks no syntax errors, that 'question3' is provided, then runs `question3(9)` and `question3(-4)`.
  - Then checks it:
    - doesn't throw an exception;
    - returns a value rather than printing;
    - returns a number rather than a string;
    - returns the right values.
  - Then runs `question3(4)`. Checks that it raises an exception and whether this is the standard Python exception or something more user-friendly.

Coursework 1

- Marking script for $\frac{x^2+3}{x-4}$:
  - Checks no syntax errors, that 'question3' is provided, then runs `question3(9)` and `question3(-4)`.
  - Then checks it:
    - doesn't throw an exception;
    - returns a value rather than printing;
    - returns a number rather than a string;
    - returns the right values.
  - Then runs `question3(4)`. Checks that it raises an exception and whether this is the standard Python exception or something more user-friendly.
  - Then runs `question3("Peter")`.

Coursework 1

- There's an old joke that goes something like this:

  A software tester walks into a bar: orders a beer, orders 9999 beers, orders AAAA beers, orders -1 beers, orders 0 beers, orders a cat, ...

Excel/graphics

Excel/graphics

- Excel-VBA refresher.
- Euler's Method in VBA.
- Data import and analysis (involving UserForms).
- Generating graphics using LaTeX/TikZ/PGF.

Why Excel-VBA?

- I don't have experience with or particularly like it, but I understand it to be used.
- It's used elsewhere in the course.
- It has immediate access to graphical interface aspects.

Why LaTeX/TikZ/PGF?

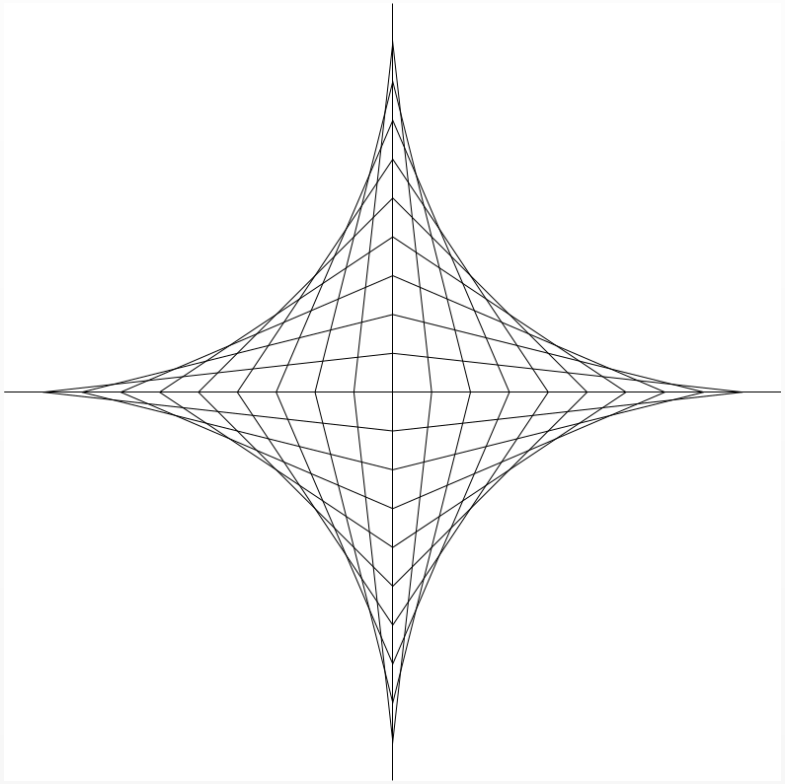
- Well.

Why LaTeX/TikZ/PGF?

- Well.
- Fun to see programming in an unusual context?

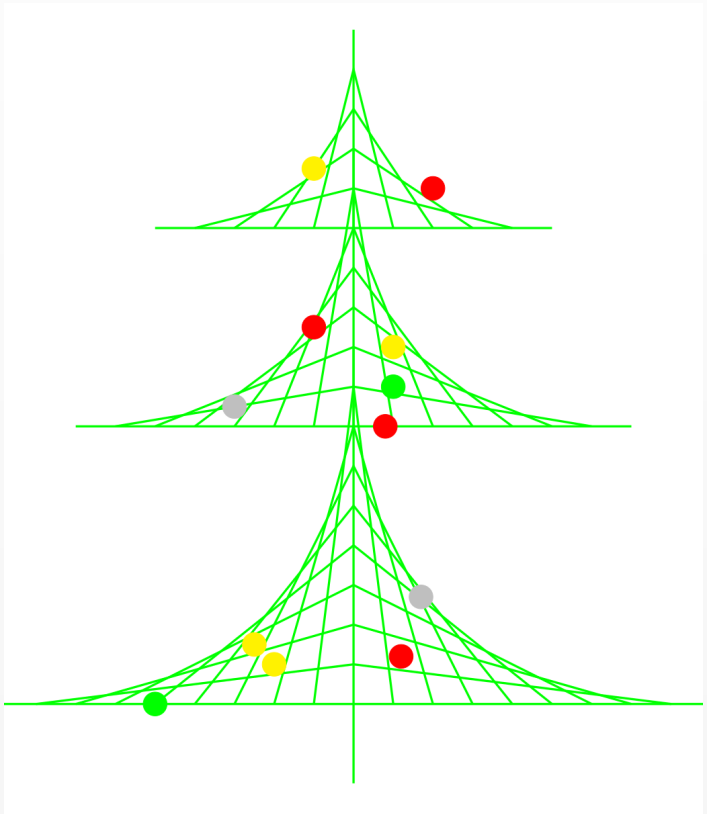
LaTeX/TikZ/PGF

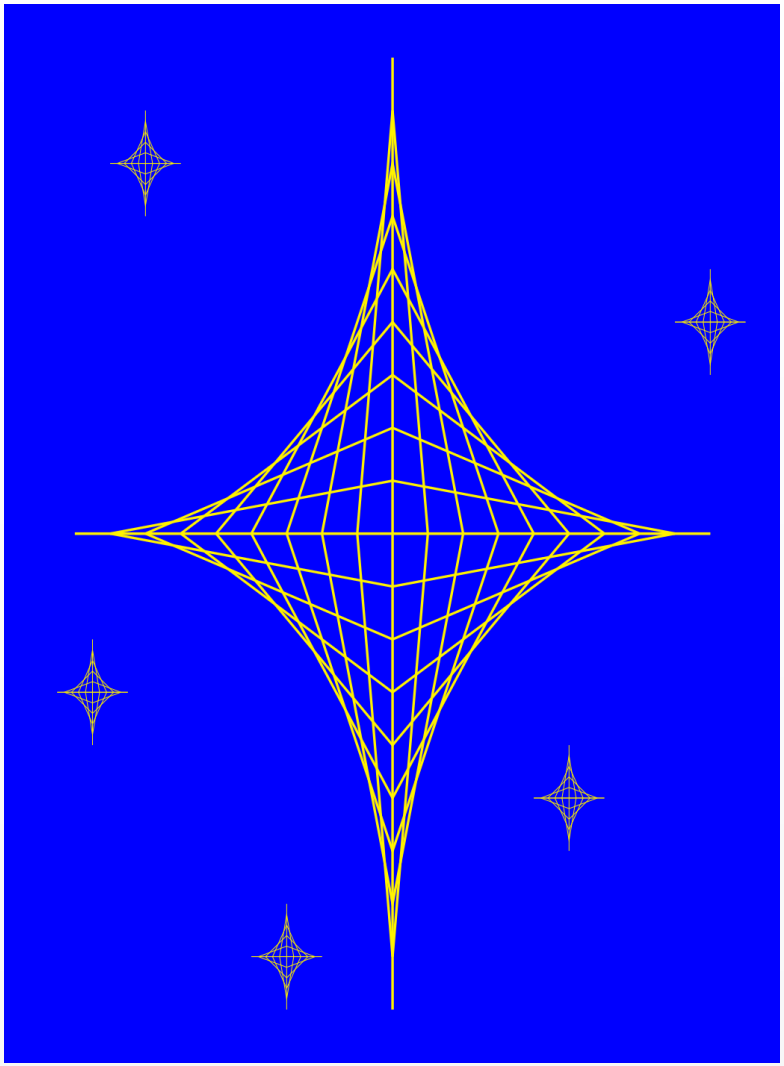
1. The following figure is obtained by connecting (0,1) to (10,0) and (0,2) to (9,0) and so on. Try drawing it by hand to get a feel for how it works. Then create it using a TikZ for loop.
2. By adjusting the step length on your code, make a version with more lines for a smoother curve.



LaTeX/TikZ/PGF

(Then we drew some Christmas-y pictures with this method.)





Web, databases, etc.

Web, databases, etc.

- HTML.
- CSS.
- MathJax.
- Generating HTML+MathJax using Python.
- SQL (SELECT, CREATE, INSERT, UPDATE).

Why SQL?

- Well-used.
- Seems a good idea to be familiar with.

Why HTML, CSS, MathJax?

- Ditto?

Coursework 2

- A program specification, but higher-level than coursework 1.
  - A Python script that would generate 10 sets of HTML+CSS+MathJax pseudo-randomised question and answer files.
  - An Excel-based marking aid, that imports student data from a text file, offers a user interface for input of marks and outputs SQL code to enter marks into a database.

Individual project

Individual project

- 6 weeks individual project, with 12 hours in-class time.
- Hand in:
  - a program (no particular language);
  - a HTML page explaining:
    - the mathematical, statistical or technical content;
    - how the program works;
    - a critical review of the work.

Individual project

- A choice of around 30 different topics.
  - programming topics not covered in the module (e.g. object-oriented programming);
  - extension of topic covered (e.g. numerical methods or statistical analysis in SciPy);
  - Python module not covered (e.g. NetworkX);
  - simulation/modelling (e.g. cellular automata, league table, game, iterative prisoner's dilemma);
  - mathematical investigation (e.g. fractals, machine learning, brute force number theory);
  - historical investigation (what did Ada Lovelace's program do?, calculating π, early computer emulators).

Individual project

- Example project brief:

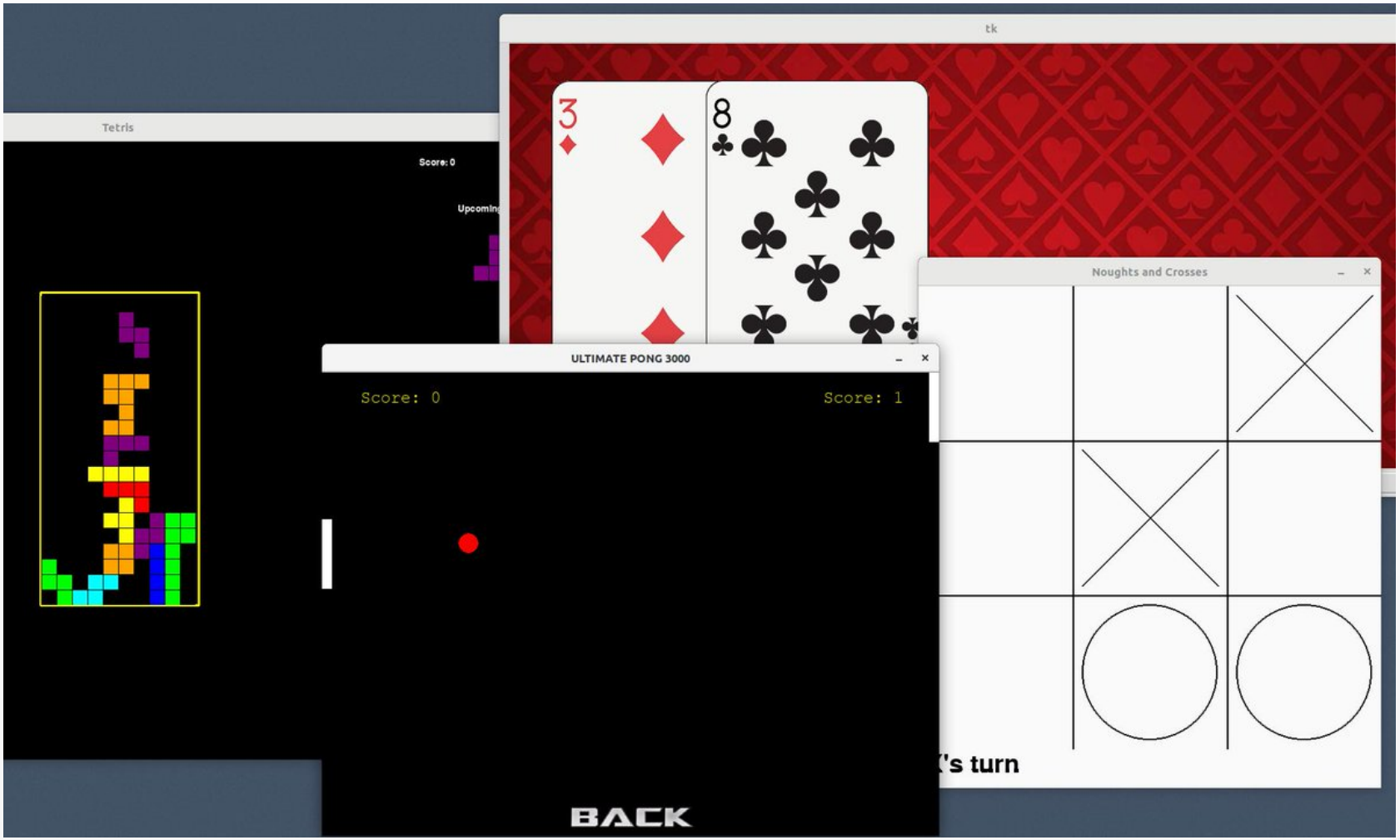
  A genetic algorithm is a method for solving optimisation problems based on natural selection, the process that drives evolution in biology. The genetic algorithm takes a set of input data and modifies this. At each step, it selects individual data from the population to be 'parents' and uses them to create 'children' for the next generation. Over successive generations, the population evolves towards an optimal solution. This project will investigate genetic algorithms, providing a description and example application to demonstrate what you have learned.

Individual project

- Project briefs backed up by in-class discussion, including two scheduled meetings with me.

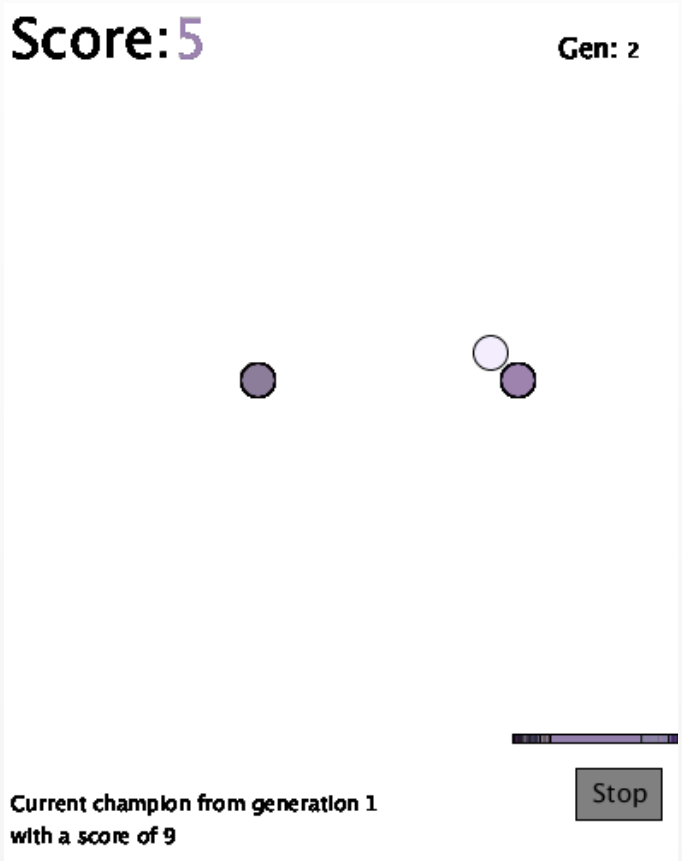
Individual project

- Most popular projects were scheduling a tournament and programming a game.



Individual project

- Some really nice work:
  - a neural network for hand-writing recognition (7×7 pixel A, B or C);
  - a genetic algorithm to place breakout.



Individual project

- Some really nice work:
  - an object-oriented banking system;
  - an Excel sheet to explore several cellular automata by changing rules.

Individual project

- Languages chosen:
  - Python 13 students;
  - VBA 8;
  - Matlab 1;
  - Java 1.

How did it go?

Student feedback

- Limited.

Student feedback

- Limited.
- But almost-all positive.

The riddle of prior knowledge

- Students all experienced programming in a few classes of VBA in the first year.
- Some also had some combination of:
  - taken the first year option;
  - arrived at university with a qualification in computing;
  - a serious casual interest in programming.
- "If you are finished, help someone else, or do something with it to entertain yourself."
  - Mixed success.

Building independence

- Coursework 1: tightly-specified, low-level functionality.
- Coursework 2: tight program specification, but higher-level and more flexibility to choose how to approach.
- Individual project: vague brief, lots of freedom to design own approach.

Building independence

- Coursework 1 (10%): tightly-specified, low-level functionality.
- Coursework 2 (40%): tight program specification, but higher-level and more flexibility to choose how to approach.
- Individual project (50%): vague brief, lots of freedom to design own approach.

Changes for next year

- Adjust weightings of first and second coursework, and probably reduce number of questions in first coursework.

Changes for next year

- Adjust weightings of first and second coursework, and probably reduce number of questions in first coursework.
- Release 'Python in practice' problems early, for those who are already ahead of the material.

Changes for next year

- Adjust weightings of first and second coursework, and probably reduce number of questions in first coursework.
- Release 'Python in practice' problems early, for those who are already ahead of the material.
- Anything useful I pick up from today!

Thanks for listening!

Programming as a mathematical activity

Peter Rowlett

Sheffield Hallam University

@peterrowlett

p.rowlett@shu.ac.uk

27th June 2019

Programming in the Undergraduate Mathematics Curriculum, Middlesex University