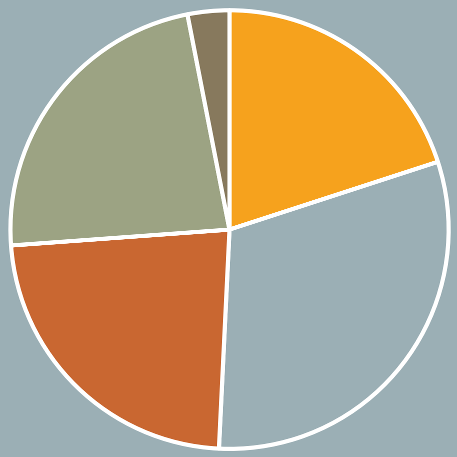
MORE FREQUENT AND GAME-IFIED FEEDBACK USING CODERUNNER TO TEACH INTRODUCTORY PROGRAMMING.

Stuart King & Chris Sangwin

School of Mathematics

PYTHON PROGRAMMING

Computing Background Survey



🟧 None ⬜ Minimal 🟥 Medium 🟩 High 🟫 Expert

A new course this year.

For MSc students (across all the MScs in the School of Mathematics).

Pitched at students who may have no programming experience.

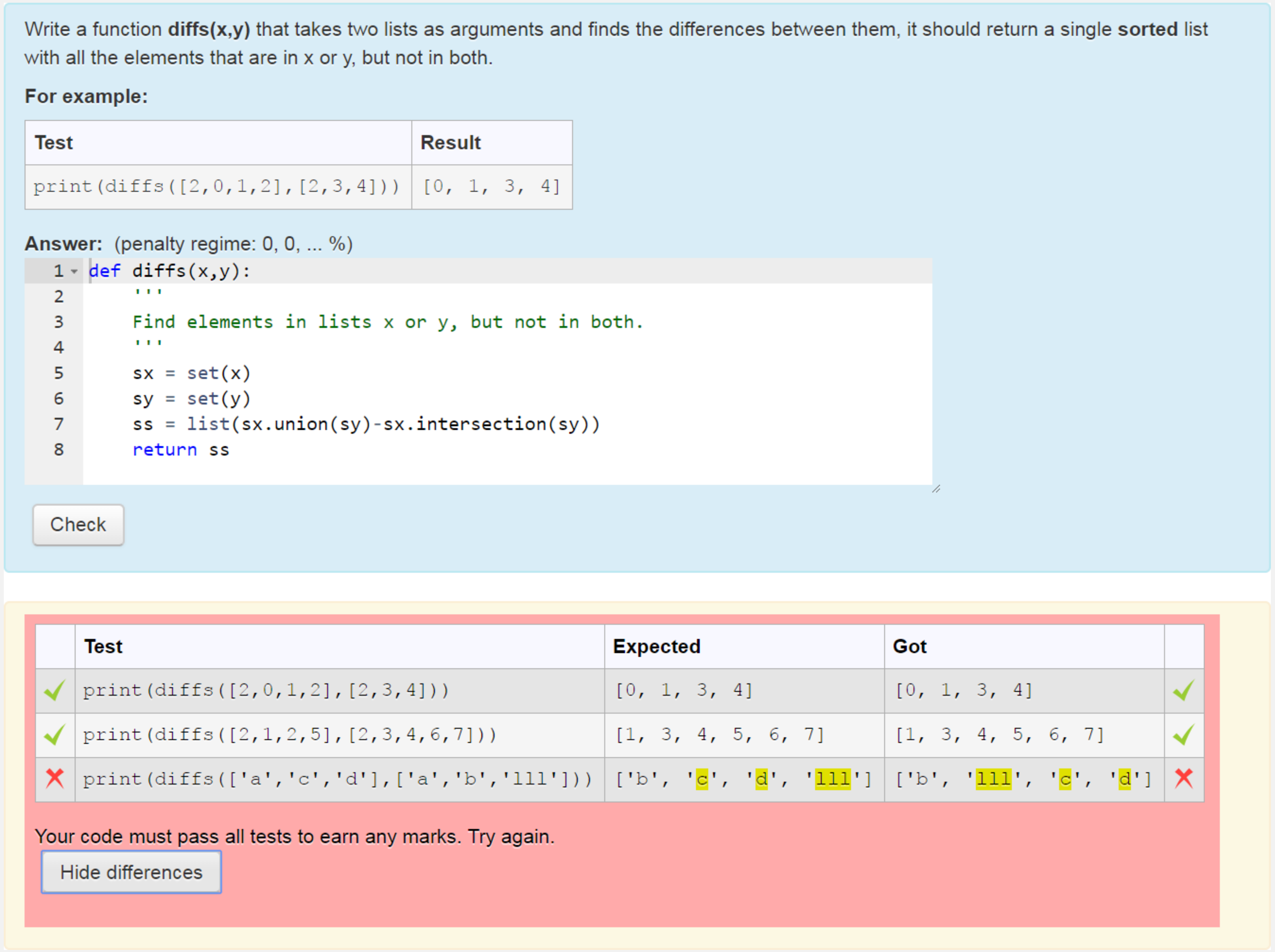
Opportunity to try some new ideas to engage students, and address the problem of the wide range of computing experience.

QUICKER FEEDBACK – INSTANT MARKING

- How can we provide feedback that meaningfully helps students?
- We learnt most about coding
  (a) by doing it: worked examples to problems
  (b) with human coach
  Hard to replicate at scale.
- Automatically mark students’ code providing them with immediate feedback.
- The aim: scaffold coding skills.
  Work up from simpler to more complex coding tasks without feeling ‘lost’.
- More feedback than can be provided by tutors.

CODERUNNER

- http://coderunner.org.nz
- Used for marking small chunks of code, in many languages.
- A plugin to the Moodle quiz.
- Linked as a tool through Learn to any UoE course.
- The student usually writes a function in python.
- Teacher specifies test cases – some released to the student and some kept back.
- Student's code is marked right if all of the test cases are passed, and wrong if one is failed.
- (Lots of options for assigning marks, multiple attempts etc. through Moodle's quiz)



Write a function **diffs(x,y)** that takes two lists as arguments and finds the differences between them, it should return a single **sorted** list with all the elements that are in x or y, but not in both.

For example:

Test	Result
`print(diffs([2,0,1,2],[2,3,4]))`	`[0, 1, 3, 4]`

Answer: (penalty regime: 0, 0, ... %)

```python
def diffs(x,y):
    '''
    Find elements in lists x or y, but not in both.
    '''
    sx = set(x)
    sy = set(y)
    ss = list(sx.union(sy)-sx.intersection(sy))
    return ss
```

Check

	Test	Expected	Got	
✓	`print(diffs([2,0,1,2],[2,3,4]))`	`[0, 1, 3, 4]`	`[0, 1, 3, 4]`	✓
✓	`print(diffs([2,1,2,5],[2,3,4,6,7]))`	`[1, 3, 4, 5, 6, 7]`	`[1, 3, 4, 5, 6, 7]`	✓
✗	`print(diffs(['a','c','d'],['a','b','lll']))`	`['b', 'c', 'd', 'lll']`	`['b', 'lll', 'c', 'd']`	✗

Your code must pass all tests to earn any marks. Try again.

Hide differences

BENEFITS FROM MY PERSPECTIVE AS COURSE ORGANISER

- Workshops were far more focused than in the previous (Java) course.
  Students grasped details of the language faster.
- Requires an exact answer rather than 'close enough'. Encouraged close reading of specifications in the question. This was **very hard** to get across in a traditional course.
- This method of teaching requires careful task design: I found this very helpful.
  Details of the wording are significant when you need a student to replicate an answer exactly.
- Performance on the final summative assessment was very good – I was hugely impressed by what all the students achieved, many started with little background in coding. The final task I set was much more complex than the assessment I used to set in Java.

STUDENT FEEDBACK

- Course was generally well-received in CEQ feedback.
- Coderunner was explicitly mentioned in the comments of a good number of the students.
  - it ‘forced’ them to program (a good thing!).
  - it was helpful in building their confidence as they progressed.
    “the code-runner stuff makes me feel successful when I complete one quiz. ”
- More ‘buzz’ than the more traditional Java course it replaced.
- Nominated a couple of times for teaching awards (one mentioning Coderunner).
- Coderunner didn’t suit everyone – ‘forcing’ some students through tasks was constraining. No easy predictor of who might dislike it.
- Students were completely addicted to completing the weekly Coderunner questions – the game-ifying effect. They were only formative!

CONCLUSIONS

- Game-ification didn’t require a very compelling game structure to get students hooked.
- Students cited CodeRunner in course feedback as being a positive.
  Endorsing the methodology of teaching skills in this way as much as this particular technology.
- Key features
  - instant feedback produced by automatic marking,
  - a graded set of tasks to work through to provide a consistent challenge.
- Tutors are also still key to discuss more difficult problems.
- Not everything can be marked using an automatic marker, e.g. style and efficiency. Coderunner worked really well for formative or low-stakes work.
- Coderunner is excellent at what it does – I can’t see myself going back to not using it (or something like it).
- Codrunner is available through Learn if you want to try it. But it is complex, so invest some time!