Title Development and use of Learner Assessment

Theme: How Learning through making mistakes promotes stronger problem solving, student engagement and confidence at assessment

Keywords: Confidence Engagement performance at Assessment

Short Abstract: Tutor shows his own fallibility to demonstrate problem solving and promote confidence in learners in how they approach problems and are active in their own learning.

Case Study

Connor Duignan – Tutorial Approach

Programme – L6 Certificate, Information Technology for process digitisation

Modules. Introduction to programming and Computer Architecture

Background

I run tutorials for a set of Level 6 computer science modules to help students tackle the practical and code-based aspects of the module such as programming languages. All of my students are adult learners and many of them have a basic understanding of computing, but some have never even approached a computer-sciened related topic before, thus there can some trepidation around the topic and assessments.

Tutorials and tumbleweeds

The tutorials are conducted online via Zoom, so when I log-in I see an ocean of names and no faces; at this point it is usually very quiet and the students will hesitate to even say “hello”. I share my screen when we’re ready so that the students can see what I’m doing.

To start a tutorial, I will first give a brief summary in my own words of the topics that were covered in the preceding lecture, laying out the sometimes very long lesson into a few topics of discussion. Depending on the engagement level of the class, I may get question at this point to go over a particular topic, but most of the time I need to coax interaction out of the class by simply beginning and then asking them questions.

I do this by first stating what I’m trying to accomplish with a piece of computer code, then I start to write the code from memory, stating exactly what I’m doing at every point and what each snippet of code is meant to do. I find that students can follow along more successfully if I explain as I go along, and I get engagement out of the students by asking for suggestions in either the layout of the code or the names of the variables, often making jokes about the prevalence of fruit in the examples the main lecturer uses in the course notes.

Imperfection and problem solving

I often intentionally do not look at my reference material so that my code will be imperfect (and often not functional), so that I need to work it out in front of the students. Through this I demonstrate the problem-solving process that is the core to being a successful programmer. This allows the students know that I am a human who makes mistakes and not an infallible expert that they cannot keep up with or relate to.

I ask the students what the outcome of what the code will be, and at this stage the class is much more comfortable in giving answers. I then run the code and see the result, which sometimes contains an error or a warning which gives me the perfect opportunity to examine the code and its intended results more closely. I go through the error in the code with the class and I ask them what they think the problem may be, I then often use my reference material to discover my mistake. I fix the error while explaining what went wrong, and the theory behind what went wrong. Most of my best teaching moments have come from me making a mistake. In computers, doing something right can almost seem like magic to someone who doesn't know how the process works, but getting something wrong and then fixing it allows them to understand that it is a logical system of rules and interactions that are consistent.

Eager and Engaged

This style of tutoring helps build a rapport with my students, and often by assessment time they are much more comfortable engaging with the tutorial and asking questions. So for the assessment, I adjust my approach to letting the students control more of the flow of the tutorials and using their questions to guide to us particular topics. At this stage, the students are familiar with my approach and are able to articulate their questions more effectively. Sometimes this results in almost too much engagement, so I split my tutorials in two: one section to go over sample assessment questions in detail, and one to answer questions.

Preparation for assessment via problem solving.

To prepare to for assessment questions, I will let the students answer the questions themselves. I begin by writing the set-up code, and I write the proper syntax, but the students are the ones giving suggestions on how to solve the problem. If there is a problem to which they cannot find the solution, I will explain the theory behind it and decrypt the question from computer code into logic that the students can quickly solve. I don't answer the questions for them because that is not going to help the student in an assessment situation, but I do help them find a solution. At the end of the day, programming is mostly about problem solving and that's the skill I aim to teach and is a key learning outcome.

For the question-and-answer portion, it often consists of students who are stuck on a single piece of a topic that just doesn't make sense to them. I will then write some code and explain it word by word, showing how code works and then changes as I add to it. Again, I will sometimes intentionally introduce an error into the code so that I can demonstrate the kind of mistakes that students often make when writing that particular piece of code and how it affects the surrounding programme.

The key when dealing with students who are struggling is patience, and once they know that they can take their time, they relax and can begin learning at their own pace.

Summary

Overall making mistakes and having the confidence in your ability to solve the problem created a level of confidence in the students to approach assessment and examination briefs and questions with curiosity and creativity which has let to stronger performance in their assignments and examinations.