Synthesis and Application Essay #1
Barry Peddycord III

Responding to Student Writing: Encouraging Reflection and Revision	*Evaluation and Grading*	*TA'ing Courses with Computer-Intensive Assignments*
Molly Storment	Dr. Barbi Honeycutt	Dr. Sarah Heckman
Nov 7, 2011	Nov 7, 2011	September 7, 2012
230 Research Building III	230 Research Building III	135 Golden LEAF BTEC

As a TA, one of the most obvious and time-consuming responsibilities I have is grading. Most of the time, I consider grading as a "necessary evil" that takes away from the naturally more enjoyable and more visibly instructive time I spend interacting with my students in and out of class, which is why I've always preferred being a "lab instructor" OVER a "teaching assistant" or "grader". I recognize that as a student, one of the most vital components to learning at the undergraduate level is having access to timely and relevant feedback. As such, I decided to combine what I learned from FIT workshops on the topic of grading and feedback with my programming abilities to develop a tool to help make the process of giving students feedback even on the simplest of assignments just a little bit easier.

In the Evaluation and Grading workshop, the focus of the workshops lay squarely on rubrics. I've attended this workshop three times, each time coming in with a different opinion of the role of rubrics in education. In the workshop, we look at the positive and negative aspects of rubrics, but each time I am able generalize away from the more negative beliefs (that they limit creativity and the instructor's judgment) and focus on the positive elements (that they encourage transparency and enable reliable grading for students). Part of what makes rubrics so powerful is that they encourage the instructor to think about how they are evaluating whether or not students are meeting their learning objectives, so I see them as extremely valuable devices, but having a rubric isn't enough because they don't serve as good feedback on their own.

However, Molly Storment's workshop on feedback addressed this issue. She focused on how to provide thoughtful feedback that goes beyond saying that something is “good or bad” and rather how to cause students to think about their work and how they would improve it. We spent the workshop looking at what corrections we made while grading students work, and learned that the feedback that many instructors provide can often be as shallow as student writing. As time went on, I kept wondering that if it took that long to provide that kind of targeted feedback, it would take days to finish grading, even if I worked nonstop with a rubric. I realized that I would need some way to identify and address the most common and egregious mistakes made by students, but didn't have a clue as to how to do that.

It all came together when I attended a workshop offered through the College of Engineering a year later that was focused on helping TAs who would be teaching courses in computing-intensive environments, which definitely included me. The focus on this workshop was to deal with the obvious issues of teaching in such courses, such as dealing with students who have installation issues with required software on their personal machines, as well as the less-obvious issues, such as how to identify students who copy-and-paste code from the Internet and their classmates. As I was a Computer Science TA and Dr. Heckman was a Computer Science Professor, this workshop was very related to what I was doing.

Dr. Heckman, in this workshop, showed us how she used spreadsheets in her teaching and grading, and I particularly took notice of how she developed her own script in visual basic to export the grades from her gradebook and import them into NC State's Wolfware system. I decided to take it a step further and develop my own approach to grading at the assignment level using Google Docs (since I don't have Microsoft Excel) and Moodle (what my class uses). This turned me on to the idea that it would be relatively simple to create such a tool of my own, incorporating what I learned from FIT workshops to develop a way to speed up my grading while making my feedback more consistent and meaningful.

Last semester, I developed a very simple tool in the Python programming language that allows me to build a rubric in a spreadsheet (Google Docs, Excel, OpenOffice, etc) and then produce detailed feedback based on how I enter the student's scores. I start by putting my grading rubric in a Google Docs spreadsheet, where the rows represent the criteria and performance levels and the columns represent the individual students. The following table illustrates how the grading layout might appear for an example assignment.

	Barbi Honeycutt	Melissa Bostrom
* Three FIT Workshops (out of 50 points)		
Discusses 1 Workshop: 15		
Discusses 2 Workshops: 30	While your description of SpeedCon was excellent, it was not listed as an approved FIT workshop elective: 40	
Discusses 3 Workshops: 50		30
	You should have chosen Molly's workshop to discuss instead! :)	
* Application (out of 50 points)		
Connects what was learned across all workshops: 30	30	30
Introduces and explains a tool and how it was connected to workshops: 20	20	
The purpose of the assignment was to synthesize what was learned in the workshops and develop a tool. You did not introduce a tool: 0		0

Normally, when a student's work meets one of the criteria on the leftmost column, I simply enter the score under the student's name. When the number in the column and the number in the criteria match, then that is what will appear on the final report I give to the student (usually in the feedback portion of Moodle). If the criteria doesn't match the submission, I have two choices, I can either fill in the criteria myself an provide a new score (seen in row 4, column 2) or if the mistake is common among students, add a new criterion (row 10, column 1). I can also provide comments that don't have grades associated with them (row 6, column 2).

The advantage here is simple – if I see a large number of students making the same mistake, creating a new criterion to address that has a number of benefits. Firstly, I can write the feedback I want all students who make the same mistake can see, citing the part of the assignment that they misinterpreted or the course material that explains why their submission was incorrect. From what I've seen and heard from other instructors, most students get things right, and those who get things wrong tend to do so in very similar ways, meaning that most feedback can be made generically to address issues that many students are having. Secondly, I can ensure that students who make the same mistake get the same score as one another. When grading many assignments, the scores and penalties can become harsher or softer over time, causing inconsistency in grading – when students see that a peer received more points than they did for the same mistake, that gives them grounding for a regrade request that has nothing to do with the course content. This avoids that. Third and finally, this approach reinforces the idea that rubrics are a tool with the end goal of learning, and by grading students based on the feedback I want to give them, I keep the learning in mind even when I'm not in the classroom.

In order to turn the spreadsheet into feedback, I developed a python program that takes the spreadsheet and takes the feedback from the rubric and tallies up the student's final score. The output for Barbi Honeycutt from the above example would appear as follows and be copied directly into Moodle (or another learning management system).

```
* Three FIT Workshops (out of 50 points)
] While your description of SpeedCon was excellent, it was not
listed as an approved FIT workshop elective: 40
] You should have chosen Molly's workshop to discuss instead! :)

* Application (out of 50 points)
] Connects what was learned across all workshops: 30
Introduces and explains a tool and how it was connected to
workshops: 20

Total: 90
```

My students in Fall 2012 specifically told me that they appreciated the detailed feedback I gave them, and I attribute this feedback to the comments I made in my rubric tool. Normally I struggle giving my students feedback, since I take their mistakes personally, and get frustrated when I see the same mistake multiple times, often choosing not to write the entire comment for them like I would the first and second time I see the mistake. While handwriting feedback is often biased to the earliest students who get graded, the rubric approach inverts that as the latest students get the benefit from having the most common mistakes already in mind when their papers are reached.

Some of the problems associated with a rubric, discussed in the Evaluation and Grading workshop, is that rubrics often constrain the grader into trying to fit students into criteria that are ill-formed. This approach makes the rubric a dynamic and fluid document, since I can add new criteria at will, and even change their scores if I feel as though I've treated certain mistakes too softly or harshly. Developing the rubric on a just-in-time basis, I can enjoy its metacognitive benefits while not subjecting myself to its structural limitations.

```python
#! /usr/bin/env python
# Grading script. To use: `python grading.py rubric.csv`

import sys
import csv

# Read in the CSV file.
cdata = csv.reader(file(sys.argv[1]))
data = []
for x in cdata: data.append(x)

names = data[0][1:]
fields = data[1:]

# Process all of the student names.
i = 0
print "\n"
for n in names:
    i += 1
    t = 0

    # Print the student's name first.
    print n

    # Now go through the criteria.
    for x in fields:
        f = x[0]
        g = x[i]

        # If "f" is a criterion, print it.
        if f.startswith("*"):
            print "\n"+f

        # If "g" is a feedback, print it.
        elif g != "":
            if (g in f) and f != "": print " ] ",f
            else: print " ] ",g

            try: t += int(g.split()[-1])
            except: pass

    print "\nTOTAL = ",t

    # Block for input (you can comment this line out to get a single file of output).
    f = raw_input()
    print "\n"
```