### NUMBER CHOOSING EACH RESPONSE

<table>
<thead>
<tr>
<th>Item</th>
<th>Vry Unsat</th>
<th>Unsat</th>
<th>Satisfact</th>
<th>Very Good</th>
<th>Excellent</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 OVERALL INSTRUCTOR RATING</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>16</td>
<td>23</td>
</tr>
<tr>
<td>8 OVERALL COURSE RATING</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>15</td>
<td>23</td>
</tr>
<tr>
<td>9 STUDENT RATING OF COURSE WORKLOAD</td>
<td>3</td>
<td>16</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>10 OVERALL UT GRADE POINT AVERAGE</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>6</td>
<td>3</td>
<td>23</td>
</tr>
</tbody>
</table>

**For the computation of averages, values were assigned on a 5-point scale so that the most favorable response was assigned a value of 5 and the least favorable response was assigned a value of 1.**
1. It would be helpful if the instructor would look up for frequently to see if students have questions. The tools in the course are
good, though the requirements are difficult to conform to sometimes...requiring 10 issues and only 5 commits on the early
projects which are very simple is difficult because we may not have 10 issues to add to the issue tracker. I strongly recommend a
code review tool be incorporated to larger group projects. At Intel, we do not do pair programming, but instead spendtime doing
code reviews with each other. This works really well with git or CodeCollaborator. I don't think these options are free, but
there probably is some free software out there.

2. I really think it would be a good idea to make sure the tests are really testing the core material of the class as opposed to re-
testing just a program's output. Otherwise we will end up learning a lot of new technologies, while familiarizing us with the group work environment. The lecture material is good as well - properly
familiarizes us with both Python and SQL.

3. The only comment I have is about the tests. The time allotted for the exams did not seem adequate to answer all the questions. I
always felt rushed while doing it, which will make it difficult to adopt the coding discipline.

4. This was one of my two favorite courses taken at USF so far. I am really glad that we were exposed to so many useful technologies.
I am glad that I got to learn Python, XML, and SQL, but I think we spent a little too much time on Python. I realize weneeded Python to complete the projects, but the focus should have been more on software engineering, XP, SCRUM, Kanban, Repos, etc. We
should cover the complexities of the language in a class like object oriented programming. The projects were great. I did not
like using the Z server for the projects because we have limited permissions on the server and this made it difficult to do anything. I liked Heroku and all of the readings. Only wish we could have spent more time on refactoring.

5. my one gripe about this class is the clearly insufficient lack of data available to us with regards to where i stand in terms of
grades in relation to others each of the quiz, project, etc. grades have average and median columns that the professor claims
are pieces of data that are available, but i don’t see them, which would potentially make this a technical error and not the
professor’s fault. however, if this is not the case, then the graders professor responsible for this information is not helping
get me a more accurate grade with regards to this class.

6. i think we could have used less lectures on xml and more on sql, especially db table design. i didn’t like being required to
use prezi. there are so many other good options, and i felt that prezi detracted from the actual content. i think your idea of making it interactive was a good one, but need to focus on the more interacting with the website. i liked the speakers with the exception of the one from paypal. he got side-tracked, so as a result he didn’t
have a cohesive idea, so the talk seemed like a bunch of random advice we could have just googled. the thing was i thought we
really could have benefitted from a good explanation of git branching workflows.

7. I understand the idea behind having us use new tools you and the TA’s weren’t familiar with, but I feel like it led to some
confusion when it came to things like test databases and the various versions of Django. If you’re going to insist that future
classes use heroku, I strongly suggest getting Django updated to version 1.5.1 updated on the CS machines. I’d also suggest
moving import and export to the second phase of the project. The WCDB project was too front loaded, and the first phase involved us
learning how to use Django rather than learning best practices. After using and getting these scripts to run and do correct work. Later phases were
just fine tuning of various things, as there wasn’t much left to do.

8. How a course should be taught varies widely across students and professors, but one thing that should hold either way is
that the instructor would look up frequently to see if students have questions. The tools in the course are

9. An excellent class. When the teacher gets genuinely excited about the language they’re teaching, you know it’s good. Suggestions

Emphasize to students that the course is more intense than they think even though they won’t believe you and that you
cannot slack off for a quiz or else it’ll snowball into a lot of bad quizzes. 2 Crack down or keep an eye on people who use
their phones in the class, secretly looked at it by phone most every lecture. 3 First project should be done alone i.e., keep as-is. Make pair programming mandatory for the second one because sometimes it’s the
kick in the rear some of us need to get real work done. 4 Paper surveys! The character limit on the eCIS is bad.

10. Testing for this course was very odd. It seemed that we were tested over issues that either were not covered extensively, or the
questions didn’t really test our comprehension of the subject. It seems that a lot of the questions asked on tests are
things that are available in public documentation that people look up normally.

11. I thought that this class focused too much on the Python specifics. I think that this class would be more true to if it’s name if
you had write code in Python that dealt with the concepts and have multiple choice test about Software Engineering from the
readings class discussion. The amount of required writing is not much to justify a writing flag. I’m a better programmer from
taking this class and I learned a lot about SWE, but I don’t like how little we were quizzed tested over the concepts that deal
with actual Software Engineering. We need more time on questions because the way that Dr. Downing’s questions are wordy and hard to find out what they’re asking of you.

12. The first phase of the wcdb project was very heavy. Please try to spread out that load more.

13. Pros Lectures and readings were very valuable Professor is one of the best I’ve ever had at relating concepts to students
Projects allow the students to see the working code. Cons I never felt like the multiple choice sections of the exams properly reflected the scope of the material of the course

14. Sir Downing, you are the best teacher in CS. Approachable, impartial, clever at Socratic questioning - how did you come up with
that method and how did you make it flexible, as the instructor? I think it is ridiculous to remove singleton, reflection, in-depth iterators. Otherwise you can pick 2 or 3 mini-problem and coach students to code the project from scratch during lectures, like you do for methods in TextPad, but on a bigger scale.
We desperately. God, out of survey space? come on, UT CTL. I will post my ideas in my blog and link on class site. Fellow
students, keep up, and do the readings! They give tips on how to succeed in the real world - it’s cold out there.