To: Dr. Titus Brown
From: STEM E.D., LLC
Re: Software Carpentry Workshop Evaluation
Date: June 19, 2012. FINAL.

A pre-workshop evaluation of the Software Carpentry Workshop was conducted on May 7, with a post-workshop evaluation occurring May 9, 2012. In all, 56 participants completed one or both of these surveys at either Michigan State University or University of Texas-Austin (Table 1).

Table 1. Participant response rates.

Institutions	Pre only	Post only	Both Pre and Post
Michigan State University	36	34	30
University of Texas-Austin	10	14	7

Overall, participants were very satisfied with the workshop. Scales used to measure participant understanding and ability also indicate statistically significant improvement.

EXECUTIVE SUMMARY OF RESULTS

We summarize evaluation results below. Following this summary, we provide: 1) A description of scales used to measure participant characteristics, including indices of validity and reliability, and 2) results for qualitative and close-ended questions related to participant impressions of the workshop. In summary, we found that:

1. Scores on the Perception of Computational Ability scale were calculated for the pre-workshop survey. Overall scale scores were calculated as averages across all six items. *Prior to instruction, participants expressed either no or low ability in computational ability*, with a scale average of 1.73 ± 0.49. We note that this scale was not given post-workshop; we encourage its use as a pre-post measure of change in the future.
2. Scores on the Computational Understanding scale were calculated for both the pre- and post-workshop surveys. Results of the t-test indicate that pre- and post-workshop results are statistically different, $t(36) = 10.2$, $p < .001$, with post-workshop results higher than pre-workshop. *This indicates that participants perceived greater understanding after engagement in the workshop.*
3. Scores on the Python Coding Ability scale were calculated for both the pre- and post-workshop surveys. Results of the t-test indicate that pre- and post-workshop results are statistically different, $t(36) = 8.93$, $p < .001$, with post-workshop results higher than pre-workshop. *This indicates that participants perceived greater coding ability after engagement in the workshop.*
4. *Participants were generally very satisfied with the workshop (Table 6). On average, participants rated the workshop components as Good-Very Good.*
5. *Participants generally felt the workshop met their needs and would overwhelmingly recommend it to others.*
6. Participants were generally positive about the workshop in their open-ended comments. Suggestions for improvement include: more introductions and basic tutorials, PDF of commands diagrams, a reading list for Python and SQL, and videos of specific codes and related processes.

WORKSHOP EVALUATION SCALES AND RESULTS

Software Carpentry participants completed a pre-workshop survey containing three scales: Perception of Computational Ability, Computational Understanding, and Python Coding Ability. Respondents also completed several demographic questions and responded to an open-ended question about workshop expectations. The post-workshop survey contained two scales, Computational Understanding and Python Coding Ability, as well as questions about overall perceptions of the workshop.

Demographics

Demographic data are reported for pre-workshop respondents. Two pre-workshop survey respondents did not complete any demographic questions; this provided response data for n=44 participants. Participants ranged in age from 21-61, with an average age of 28.8 ± 7.7 years. Participants were predominantly male (n=26), with the remainder female (n=18) and no transgendered participants. Participants were also two-thirds Caucasian (n=28), and one-third Asian, with non-response from an additional two participants. Finally, the academic status of participants included undergraduates (n=2), graduate students (n=29), and Ph.D.-holding professionals (n=13).

Perception of Computational Ability

A new scale measuring perception of computational ability and containing six Likert-type items corresponded to the various abilities taught in the workshop. Participants were asked to rate their ability in each as No Ability, Low Ability, Intermediate Ability, or High Ability. A score of 1 implies No Ability, while a score of 4 implies High Ability.

Factor Analysis. Confirmatory factor analysis of responses to Perceptions of Computational Ability items was undertaken to confirm one ability scale. Note that this analysis is preliminary given the small sample size; small sample size is reflected in the Kaiser-Meyer-Olkin measure of sampling adequacy of 0.51, below the 0.6 value recommended for factor analysis. The data set met other minimum conditions necessary for factor analysis. First, the majority of items correlated with at least one other item at a level over 0.3, indicating that a factor structure could be expected to emerge. The Bartlett's Test of sphericity was significant (χ^2 (15) = 100.1, p<0.001). Communalities for 5 of the 6 items were above 0.3, indicating shared variance with other items. Given these data, factor analysis was performed on all six items.

A single scale explains 41.7% of the variance in the data and all items yielded factor loadings over 0.32, suggesting the presence of a single scale. We note that the "Databases" item has low communality and loading (Table 2), perhaps reflecting a misunderstanding about the meaning of "Databases" in this context.

Table 2. Factor loadings, communalities, and item response averages for Perception of Computational Ability Scale

Items	Factor Loadings	Communalities	Response Average	Response Meaning
Python coding	.729	.531	1.99 ± 0.91	No-Low Ability
Python debugging	.720	.519	1.72 ± 0.78	No-Low Ability

Unix scripting	.705	.498	2.00 ± 0.84	Low Ability
Unit testing	.681	.464	1.43 ± 0.62	No-Low Ability
Version control	.584	.341	1.48 ± 0.69	No-Low Ability
Databases	.384	.147	1.76 ± 0.79	No-Low Ability

Reliability analysis. Cronbach's alpha was calculated to establish internal consistency of items. The scale has high reliability, with a calculated alpha for the sample of 0.703. Despite the small sample, this is above the recommended minimum of 0.70 for analysis of individual scores.

Results. Scores on the Perception of Computational Ability scale were calculated for the pre-workshop survey. Overall scale scores were calculated as averages across all six items. Prior to instruction, participants expressed either no or low ability across the scale, with a scale average of 1.73 ± 0.49 (Table 3). This is similar to responses in each of the six areas measured by the survey (Table 2). We note that this scale was not given post-workshop; we encourage its use as a pre-post measure of change in the future.

Table 3. Pre and Post Workshop Scale Averages

Scale	Pre	Post
Perception of Computational Ability	1.73 ± 0.49	NA
Computational Understanding	2.26 ± 0.61	2.91 ± 0.40
Python Coding Ability	2.48 ± 0.63	3.00 ± 0.45

Computational Understanding

A new scale measuring computational understanding and containing 12 Likert-type items corresponded to the types of understanding expected of participants post-workshop. Participants were asked to indicate the extent to which they understood specific concepts, with ratings of Strongly Disagree, Disagree, Agree, and Strongly Agree. A score of 1 implies low understanding (Strong Disagreement), while a score of 4 implies high understanding (Strong Agreement).

Factor Analysis. Confirmatory factor analysis of responses to post-instruction Conceptual Understanding items was undertaken to confirm one understanding scale. Note that this analysis is preliminary given the small sample size, although the Kaiser-Meyer-Olkin measure of sampling adequacy of 0.71, above the 0.6 value recommended for factor analysis. The data set met other minimum conditions necessary for factor analysis. First, the majority of items correlated with at least one other item at a level over 0.3, indicating that a factor structure could be expected to emerge. The Bartlett's Test of sphericity was significant (χ^2 (66) = 195.8, p<0.001). Communalities for many items were above 0.3, indicating shared variance with other items. Given these data, factor analysis was performed on all twelve items.

A single scale explains 35.8% of the variance in the data and all items yielded factor loadings over 0.32, suggesting the presence of a single scale. Given low communalities (Table 4) and

explained variance, however, we suggest that the workshop facilitators review items and work with evaluators to determine if these define two separate understanding scales.

Table 4. Factor loadings and communalities for Computational Understanding Scale

Items	Factor Loadings	Communalities
I understand what "$ cat hashbang.py" means.	.498	.248
I understand what "cp" means.	.357	.127
I know what grep does.	.471	.222
I understand what "str[1:-1]" means.	.675	.455
I know how to define a function in Python.	.656	.430
I know how to raise an exception in Python.	.506	.256
I understand what "git merge" means.	.806	.650
I am familiar with the SVN version control system.	.504	.254
I understand what "git checkout -f" means.	.636	.405
I know what SQL is.	.779	.607
I know how to program a join.	.527	.277
I recognize when a database is designed adequately for my needs.	.604	.365

Reliability analysis. Cronbach's alpha was calculated to establish internal consistency of items. The scale has high reliability, with a calculated alpha for the sample of 0.815. Despite the small sample, this is above the recommended minimum of 0.70 for analysis of individual scores.

Results. Scores on the Computational Understanding scale were calculated for both the pre- and post-workshop surveys. Data appeared appropriate for parametric tests, so paired samples t-tests were run on matched data (n=37). Results of the t-test indicate that pre- and post-workshop results are statistically different, $t(36) = 10.2$, $p < .001$, with post-workshop results higher than pre-workshop (Table 3). This indicates that participants perceived greater understanding after engagement in the workshop.

Python Coding Ability

A new scale measuring computational understanding and containing 26 Likert-type items was modified from the scale of Askar and Davenport (2009) to focus on Python and to reflect concepts taught in the workshop. Participants were asked to indicate the extent to which they agreed with statements about their Python coding ability, with ratings of Strongly Disagree, Disagree, Agree, and Strongly Agree. A score of 1 implies low ability (Strong Disagreement), while a score of 4 implies high ability (Strong Agreement).

Factor Analysis. Confirmatory factor analysis of responses to post-instruction Python Coding Ability items was undertaken to confirm one ability scale. Note that this analysis is preliminary given the small sample size, although the Kaiser-Meyer-Olkin measure of sampling adequacy of 0.82, above the 0.6 value recommended for factor analysis. The data set met other minimum conditions necessary for factor analysis. First, the majority of items correlated with at least one other item at a level over 0.3, indicating that a factor structure could be expected to emerge. The Bartlett's Test of sphericity was significant (χ^2 (325) = 1095, p<0.001). Communalities for most items were above 0.5, indicating shared variance with other items. Given these data, factor analysis was performed on all items. Results indicate that one item related to motivation had low factor loading, prompting its removal. Analysis was then performed on the remaining 25 items.

A single scale explains 53.3% of the variance in the data and all items yielded factor loadings over 0.32, suggesting the presence of a single scale (Table 5).

Table 5. Factor loadings and communalities for Python Coding Ability Scale

Items	Factor Loadings	Communalities
I can write syntactically correct Python statements.	.821	.674
I understand the language structure of Python.	.749	.562
I can write logically correct blocks of code using Python	.836	.699
I can write a Python program that displays a greeting message.	.519	.270
I can write a Python program that computes the average of three numbers.	.652	.424
I can write a Python program that computes the average of any set of numbers.	.743	.552
I can write a small Python program to solve a problem that is familiar to me.	.834	.695
I can write a complex Python program to solve any given problem as long as the specifications are clearly defined.	.836	.699
I can design a Python program in a modular manner.	.799	.638
I understand the object-oriented paradigm.	.778	.606
I can make use of a pre-written function if given a clearly labeled declaration of the function.	.737	.543
I cannot complete a programming project unless someone else helps me get started.	-.802	.643
I can debug a long and complex program that I have written.	.800	.639
I can comprehend a long, complex multi-file program.	.820	.672
I cannot complete a programming project unless someone shows me how to solve the problem first.	-.729	.531

I cannot complete a programming project unless I have the language reference manual.	-.346	.120
I can complete a programming project if I have a lot of time to complete the program.	.703	.494
I can complete a programming project if I have just the built-in help facility for assistance.	.432	.186
I can find ways of overcoming the problem if I get stuck at a point on a programming project.	.800	.641
I can come up with a suitable strategy for a given programming project in a short time.	.821	.675
I cannot complete a programming project unless I can call someone for help if I get stuck.	-.711	.506
I can mentally trace through the execution of a complex multi-file program developed by someone else.	.696	.485
I can rewrite confusing portions of code to be more readable.	.777	.603
I can find a way to concentrate on my program, even when there are many distractions around me.	.433	.187
I can find ways of motivating myself to program, even if the problem area is of no interest to me.	.757	.573

Reliability analysis. Cronbach's alpha was calculated to establish internal consistency of items. The scale has a very high reliability, with a calculated alpha for the sample of 0.96. Despite the small sample, this is above the recommended minimum of 0.70 for analysis of individual scores.

Results. Scores on the Python Coding Ability scale were calculated for both the pre- and post-workshop surveys. Data appeared appropriate for parametric tests, so paired samples t-tests were run on matched data (n=37). Results of the t-test indicate that pre- and post-workshop results are statistically different, $t(36) = 8.93$, $p < .001$, with post-workshop results higher than pre-workshop (Table 3). This indicates that participants perceived greater coding ability after engagement in the workshop.

Overall Workshop Impression

Participants were asked to respond to several close-ended questions related to overall impressions of the workshop (Table 6). Participants also responded to open-ended questions about their expectations for the workshop (pre-survey) and their perceptions of the workshop (post-survey).

Close-Ended Questions. Five questions asked participants to rate components of the workshop, as well as the overall workshop on a 5-point Likert scale of Very Poor-Poor-Adequate-Good-Very Good. A 1 corresponds to a Very Poor rating and a 5 corresponds to a Very Good Rating. Participants were also asked if the workshop met their needs on a 4-point scale (4=very well), if they learned as expected from the workshop, if understanding of

computational science changed, and if they would recommend the workshop.

Table 6. Overall Workshop Impressions

Workshop Components	Average Score (ideal score)	% Yes
Shell Training	4.44 (5)	NA
Python Training	4.04 (5)	NA
Subversion Training*	3.62 (5)	NA
Workshop Overall	4.16 (5)	NA
Meet Needs?	3.3 (4)	NA
Learn What Hoped to Learn?	NA	85%
Computational Understanding Change?	NA	81%
Recommend Workshop?	NA	96%

*Ten participants indicated that they did not know what "subversion" was.

Results. Post-workshop responses to questions about the efficacy of the workshop indicate that participants were generally very satisfied with the workshop (Table 6). On average, participants rated the workshop components as Good-Very Good. The one exception (Subversion Training) was rated Adequate-Good; however, ten participants indicated that they did not know what "subversion" meant and some rated the "git" training instead. This suggests that participants may not have understood this item, and it should not be interpreted.

Participants generally felt the workshop met their needs and would overwhelmingly recommend it to others. Participants gave the workshop a rating of 3.3 out of 4, indicating the workshop met their needs well to very well. Eighty-one percent (n=39) of participants felt their computational understanding changed, and 85% (n=41) felt they learned what they hoped to learn. Finally, a full 95% (n=46) indicated that they would recommend the workshop to others.

Qualitative Data: Participant Expectations and Perceptions. On the pre-survey, participants responded to a prompt: "*Please provide any additional comments about your expectations for the workshop below*." Participants also responded to a similar post-survey question: "*Please provide any additional suggestions or comments about the workshop below.*", and were given opportunities to comment to each of the three yes-no questions in Table 6.

Results. Eleven participants provided comments about expectations on the pre-survey (Table 7). Of these, seven explicitly mentioned a desire to learn basics about, or improve ability in, computing, Python, Unix, plotting, and/or version control. One participant indicated a desire to increase existing proficiency in Python. Three responses were not coded (two participants wrote snippets of code, and one asked for coffee).

Post-survey responses align with quantitative results, indicating overall satisfaction with the workshop. Qualitative comments related to whether or not participants learned what they hoped to learn suggest that attendees felt the workshop helped them with coding design and programming (n=5), while three participants valued the opportunity to use iPython. Participants indicated that they would specifically value more real world examples and opportunities to test

code (n=4) and would appreciate more help with Python (n=7).

The background of the participant impacts their perceptions of whether or not their computational abilities changed as a result of the workshop. While some of the workshop attendees indicated that they were generally unfamiliar with computational science or the specific programs taught in the workshop, others indicated that they were already competent. For example, one attendee noted that "I am a CS degree holder I just took the workshop for specific tools, not general CS understanding" while another indicated that s/he "I did not know much before about computational science". These comments are indicative of the generally wide distribution of computational abilities of attendees.

Sixteen participants made a wide range of suggestions for the workshop. In general, comments were positive (e.g., "Great work! You made it easy to follow a complex subject and improve my code!"; "Wonderful workshop. Learned a lot. Instructors explain things very well! Thank you."). Several participants commented on the fact that installation problems inhibited their ability to engage in the material on the first day. Two participants felt the workshop was too general for those who know Python and SQL, although the majority of other comments related to the complexity of the workshop suggest the material was too advanced for some students. Participants were interested in receiving more introductions and basic tutorials. Suggestions for additional supporting materials included PDF of commands diagrams to clarify structure of git/version control, a reading list for Python and SQL provided before workshop, and videos of specific codes and related processes. One participant specifically indicated that the material was too advanced for novices, and would have appreciated instruction that spent "more time demonstrating how common problems that many of us face can be aided through use of the software". Finally, one participant complained about the logistics of the workshop at UT (e.g., "our TA was out of the room half of the time"), one complained that the room at MSU was too small, and three complained about the start time (8:30 a.m.) and lack of coffee.

Table 7. Qualitative Response Rates

Prompt/Question	N responses
Please provide any additional comments about your expectations for the workshop. (pre)	11
Did you learn what you had hoped to learn from participating in this workshop? (post)	21
Did your understanding of computational science change because of your participation in this workshop? (post)	8
Would you recommend this workshop to your colleagues? (post)	11
Please provide any additional suggestions or comments about the workshop. (post)	16

REFERENCES

Askar, P. & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java programming among Engineering Students. Turkish Online Journal of Educational Technology, 8(1), 26-32.