Analsys on Characteristics and Efficiency in Opensource Programming Languages of Undergraduates

Pardha Saradhi Byra, K. Gowri Raghavendra Narayan, Seshu Chakravarthy Thota

Abstract – A basic open source programming language(OPSL) may speak to the principal subject introduction for some understudies. Before taking this course, understudies may have framed discernments in regards to the trouble of OPSLs from books, online courses and other media anticipating the innovation. Through this we breaking down what impacts understudy adequacy of their capacity to prevail in an early on OPSL course. it gives Synopsys of the affiliation that exists between understudy qualities and sentiments of viability in an early on OPSL.

Index Terms— open source, student efficiency, characteristics

1. INTRODUCTION

Writing computer programs involves giving directions PC to play out specific undertakings. Code, is the dialect which can be justifiable by PC, comes in numerous varieties. The wide decision of coding dialects can be exceptionally confounding to a beginner. Today, programming or coding isn't as large an arrangement as it used to be a couple of years back. Prior, this was an approach to offer answer for basic issues dwelling in labs or research focuses. Without breaking a sweat of adapting, even a school youngster can code today, and we have seen extensive excitement from all age bunches in figuring out how to program or code.

With the rise of OPSL, one isn't constrained or bound to take in a solitary dialect however one can look over a ton of alternatives and can pick something that appropriate to one best. Open source is a PC program or source code accessible to end clients free of expense and they are allowed to transform it in like manner to make it increasingly valuable and mistake free. Instead of a solitary restrictive possession, it will rouse more without bug and helpful code for everybody to utilize.

A few rules that a programming language must conform to in order to be declared as an OPSL are listed below:

1. Source code must be open and available
2. Derived works ought to likewise stay open source
3. Free redistribution
4. Integrity of the creator's source code ought to be kept.
5. License must not limit other programming
6. No oppression fields of undertaking

Table 1: Pros and Cons of OPSLs.

Advantages	Disadvantages
Development and implementation cost is low	Support and maintenance cost may be hidden in the initial package
More easy data transferability	Being open to all, code may be more vulnerable to the hacker community.
Potential for fast cycle time of release and bug fixes	Open source solutions may require additional development to enable integration with an existing proprietary environment
No limit on usage or target audience	Introduction of new programs/software may require staff retraining to enable them to use open source solutions
Suitable for rapid prototyping and experimentation	Most considering using and developing open source 'in-house' must ensure that they have the right level of expertise to manage it effectively
Opportunities for customization and innovation	Open source solutions may require additional development to enable integration with an existing proprietary environment.

2. EASE OF USE

Talaga and Oh (2009), for instance, take a gander at how Lego Mindstorm robots can be used related to Russell and Norvig's Artificial Intelligence: A Modern Approach course reading. LEGO handyboard robots' utilization in AI guidance was exhibited by Greenwald and Artz (2004) (who utilized them to show both neural and Bayesian systems) and Imberman (2003; 2004) (who utilized them to show neural systems). The more current LEGO Mindstorm robots were utilized by Kumar (2004), Klassner (2002), (Schafer, 2004) and Fermé and Gaspar (2007). Kumar (2004) utilized them to show hunts and master frameworks. (Klassner, 2002) utilized the robots to educate about the contrasts among accepted and real state and additionally seek state and slope climbing operator structures. A few critical thinking ventures were additionally utilized without an explicitly expressed calculation use necessity. Schafer (2004) secured slope climbing, recreated tempering, looking and arranging, notwithstanding fundamental mechanical

Revised Manuscript Received on December 22, 2018.

Pardha Saradhi Byra, Assisstant Professor, Department of CSE, Vasireddy Venkatadri Inst. of Tech., Nambur, Guntur, A.P, India

K. Gowri Raghavendra Narayan, Assisstant Professor, Department of CSE, Vasireddy Venkatadri Inst. of Tech., Nambur, Guntur, A.P, India

Seshu Chakravarthy Thota, Assisstant Professor, Department of CSE, Vasireddy Venkatadri Inst. of Tech., Nambur, Guntur, A.P, India

autonomy acclimation and operator idea works out. A last task (proposed by understudy colleagues) was additionally joined. Fermé and Gaspar (2007) use an infrared transmitter to enable the Mindstorm robots to be controlled continuously by a personal computer. They use the robots in an issue based learning condition for man-made brainpower office guidance.

Klassner (2006) proffers that mechanical technology and the utilization of the LISP dialect can be "simulated intelligence's October Sky" (an occasion that incites understudy intrigue and sends them on to encourage investigation all alone. Fermé and Gaspar (2007) use PROLOG and Kumar, Kumar and Russell (2006) built up a Python-based automated stage, while yet others use C and C++. Dodds, et al. (2006) demonstrate that there are unmistakably a larger number of choices than just LEGO items. They distinguish in excess of ten distinctive automated (and reproduced mechanical) stages that length from free programming to over USD $2,000. Martin (2007), then again, proffers that robots give the comprehension of the job of input (which moves toward becoming repetition to people in their regular daily existence) to understudies and enable them to comprehend the unpredictable conduct that true tasks can deliver. Various others (e.g., (Blank, Meeden, and Kumar, 2003; Dunn and Wardhani, 2003; Klassner and Anderson, 2003; Klassner and Continanza, 2007; Marín, Sanz, and Del Pobil, 2003; Weiss and Overcast, 2008)) have exhibited the adequacy of the utilization of apply autonomy outside of AI. Fagin and Markle (2003; 2002) and Duan () have made structures to gauge the quantitative and subjective impact of the utilization of mechanical technology in software engineering instruction. Klassner (2006) proffers that mechanical technology and the utilization of the LISP dialect can be "simulated intelligence's October Sky" (an occasion that incites understudy intrigue and sends them on to encourage investigation all alone. Fermé and Gaspar (2007) use PROLOG and Kumar, Kumar and Russell (2006) built up a Python-based automated stage, while yet others use C and C++. Dodds, et al. (2006) demonstrate that there are unmistakably a larger number of choices than just LEGO items. They distinguish in excess of ten distinctive automated (and reproduced mechanical) stages that length from free programming to over USD $2,000. Martin (2007), then again, proffers that robots give the comprehension of the job of input (which moves toward becoming repetition to people in their regular daily existence) to understudies and enable them to comprehend the unpredictable conduct that true tasks can deliver. Various others (e.g., (Blank, Meeden, and Kumar, 2003; Dunn and Wardhani, 2003; Klassner and Anderson, 2003; Klassner and Continanza, 2007; Marín, Sanz, and Del Pobil, 2003; Weiss and Overcast, 2008)) have exhibited the adequacy of the utilization of apply autonomy outside of AI. Fagin and Markle (2003; 2002) and Duan () have made structures to gauge the quantitative and subjective impact of the utilization of mechanical technology in software engineering instruction.Beyond robotics, Markov and Zdravko (Markov, Russell, Neller, & Coleman, 2005; Russell, Markov, Neller, & Coleman, 2005) utilized six machine learning-related projects including document classification, data mining and character recognition. They also utilized the Clue game, n-puzzle problem and ‘pig’ dice game as the basis for student assignments. This work was validated by a student survey, indicating that a vast majority of students agreed or strongly agreed that the project was effective and the course experience was positive. Wallace, Russell and Markov (2008) discuss “Project MLExAI”, a set of projects for a single-semester artificial intelligence course, based on computer gaming. The “Robot Defense” game requires users to place structures on a map and control their robots and these placed structures. Artificial intelligence challenges for students include route planning, rule development in Soar (Laird, Newell, & Rosenbloom, 1987) and policy development (using Q- Learning). Student perception of these exercises, collected via survey, indicated that they aided their understanding and positive experience (with more than 75% of students responding agree or strongly agree to applicable statements).

Imberman (2005) demonstrated artificial intelligence concepts using a Turing test (Saygin, Cicekli, & Akman, 2003) simulation and the sixteen puzzle (a tile arrangement game), in addition to a robotics project. Lavesson (Lavesson, 2010), lamenting a decline in computer science enrolment and student math skills, utilized a project-based learning style utilizing an open-source tool, Weka (Witten & Frank, 2005). Pillay (2004) and Canosa (2006) consider similar, project-based strategies for teaching introductory courses in genetic programming and “image understanding”, both which would follow from an introductory artificial intelligence course. García, Román and Pardo (2006) suggest that the incorporation of peer review of their classmates work can aid in gaining a broader understanding of the full range of course topics, instead of students gaining a deep understanding in one area (of their project) and only a very limited understanding of other areas.

3. APPLICATION AREAS OF OPEN SOURCE PROGRAMMING LANGUAGES

Creating Web applications with PHP

On the off chance that you wish to make progressively responsive and gorgeous Web applications, PHP is the name of the amusement for you. PHP has been in the image throughout the previous 20 years and is viewed as the best Web improvement system even at this point. JavaScript doesn't have any similarity issues with any program, which implies you, can run it on any program. Additionally, it might be used for both front-end and back-end systems, filling in as an aggregate spine for your structure. There is much talk over the comparable prevalence of PHP and Java, yet PHP is more straightforward to learn and code than Java. Diverse names in the summary fuse Python, Ruby, Action Script and JavaScript, which are altogether also exceptionally astonishing yet PHP scores over them.

Use Ruby and Python when fast prototyping is required

Organizers who require some back-end programming to test their applications can rely upon Ruby and Python.

Both are dynamic and question arranged programming vernaculars with basic learning conditions. The main role behind these two being so conspicuous is their strong architect organize. A significant customer base and their synergistic undertakings have made them extraordinarily renowned. On the Internet, you can find such a substantial number of attracting Web portals and talks to guide you to take in programming from the basics to front line levels.

Python is favored when you need to learn machine programming or computerized reasoning. For all apply autonomy and human association innovation, use Python.

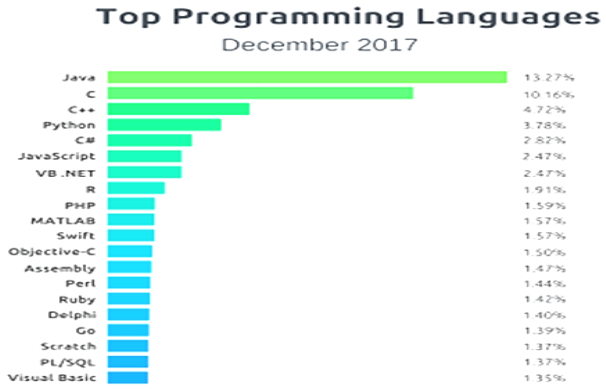

Figure 1: Popularity graph of programming languages

Build Android applications with Java

With the ascent of Android, the term 'open source' has turned out to be all the more generally acknowledged. At this moment, Java is the main simple to-learn and successful programming dialect for building Android applications. Regardless of whether you have a Windows or Mac machine, you can prepare your working application inside minutes and it tends to be tried on your neighborhood cell phone. To learn Java, you don't need to know about programming terms. Indeed, even an apprentice can, with training, turn into a bad-to-the-bone programming devotee.

Build iOS applications with Objective-C

On your Mac machine you can develop IOS applications impeccable to continue running on the iPhone, iPad or iPod contraptions. It uses Objective-C as the programming vernacular and this is so healthy, it enables you to manufacture irrefutable applications. At the point when appeared differently in relation to the Android organize, IOS gadgets have exceptionally restricted gadget similarity and screen varieties, so it is less demanding to manufacture an application which is good enough to keep running on a few gadgets.

Build desktop applications with Visual Basic

For speaking with the work region condition, Visual Basic and VBScript are what people have been using for a significant long time, anyway their repression is that they require unrestrained UIs. Starting late, Python and Ruby have progressed as astounding yet great programming lingos for work region application enhancement.

Learn basic coding with Scratch

On the off chance that you would prefer not to be wasted time with genuine programming terms and ideas before making something important, at that point you should begin with the Scratch apparatus. This was first presented in the US and, in the classroom, kids were locked in to instruct code. The utilization of zombies, the Angry Birds and other animation characters has made it all the more engaging and propelling to code. The apparatus is a basic fitting and-play utility with predefined conduct, and you have to gather it to get the conduct required.

Figure gives the statistics of the popularity of various programming languages among the user community.

Table 2: popularity of various programming languages among the user community

Language	Popular Compilers
PHP	Wamp, Zamp, Lamp
Java	NetBeans, Eclipse
Python	PyCharm
Ruby	ColdRuby

4. EXPERIMENTAL METHODS & RESULTS

A study was given to understudies in a starting python programming dialect toward the start of the semester. this study, which depended on earlier work by Ramalingam and Wiedenbeck (1998) [2] and Wilson(2002) [3], got some information about their own attributes(e.g., sex, age, significant, scholarly year, semesters enlisted, math courses taken and GPA). It at that point got some information about their confidence in their productivity to perform in the course. the initial six inquiries, from Wilson(Wilson, 2002) [3], requested that understudy describe:

Expected trouble of this class

Scale of 5 - very easy to 1 - very difficult

Composing PC programs

Scale of 5- very easy to 1 - very difficult

Working at the PC on a programming task

Scale of 5 - very relaxed to 1 - very anxious

My dimension of comprehension in this class

Scale of 5 - higher than others to 1 - lower than others

Interest in class

Scale of 5 - comfortable to 1 - afraid

Utilizing available time for help with assignments or

Course content

Scale of 5 - comfortable to 1 - uncomfortable

the next 5 questions, from ramalingam and wiedenbeck (1998) [2], asked students about particular programming constructs. These questions asked students to respond on a seven - level scale ranging from 7 - absolutely confident to 1 - not at all confident. The questions asked were :

1. Complete a programming project if someone showed me how to solve the problem first.
2. Understand the object – oriented paradigm.

3. Build my own libraries.
4. Write a program that displays a greeting message.
5. Write logically correct blocks of code.

These surveys were keyed and segmented based on four variables to detect correlations between these characteristics and survey responses. This data is presented in the following section.

5. DATA AND ANALYSIS

- The first variable for which correlations were assessed for was age. the course was separated into two groups : 18- 22 (the prototypical age of college students who proceed directly from high school to college and complete college within four years) and 22- 30. Additional age groups were presented to respondents, but either had no one indicate membership or an insufficient membership to facilitate analysis. Table 1 presents the correlation between age and the seven questions from Wilson (Wilson, 2002) [3] (listed fully in Section 3). Tables 2 present the correlation between age group and the statements from Ramalingam and Wiedenbeck (1998) [2]. Note that columns in the table have been highlighted to facilitate the identification of statistically significant results. Green highlighting indicates significance at $p<0.05$, yellow indicates significance at $p<0.10$ and orange indicates significance at $p<0.15$. significance determinations have been made from values prior to rounding for presentation.
- The results presented in Table 3 indicated two statistically significant findings [1]. it appears that the younger students are comfortable, working at the computer and feel more confident in reaching a high level of understanding in the class.
- In table 4, five statistically significant correlations are presented [1]. Interestingly, in all cases the younger group of students exhibits a superior level of confidence.
- In Table 5, only one statistically significant correlation is demonstrated [1]. Senior-year students demonstrate a greater belief that they will participate in class.
- In Table 6, five statistically significant correlations[1] are shown, including four at $p<0.05$.
- In Table 7, four statistically significant correlations[1] are presented, including three at $p<0.05$. Interestingly, the lower- GPA students evidence a greater belief level in their ability to write computer programs and that they will participate in both the class and the lab. The higher-GPA students indicate an expectation of having a higher level of understanding. Notably, GPA is the only variable that shows a statistically significant correlation with the confidence in writing computer programs and lab participation.
- In Table 8, one statistically significant correlations [1] are presented. This includes question 9 (understand object-oriented) Notably, GPA is the only variable shown to have a correlation
- In Table 9, two statistically significant correlations [1] are presented. Students who self-learned programming expect the class to be more difficult, but they also expect to gain a higher level of
- understanding of the materials presented. Notably, self-learning status is the only variable showing a statistically significant correlation with expected difficulty and the only one shown a statistically significant correlation at $p<0.05$ for the level of understanding.
- In question 3, they indicate a greater belief in their understanding of the object - oriented programming approach. Question 5 have younger students exhibiting their confidence in writing logically - correct code, understanding language structure and writing a program in an "only vaguely familiar" area.
- In question 2, the senior - year students indicate greater confidence in their efficiency to write a program using only the help functionality, complete a program using only a language manual and build a library.
- In question 1, the senior - year students indicate greater confidence in their efficiency to complete the programming project if someone showed how to solve the problem first.
- In question 4, the both senior and younger group students write the program with high level of confidence. Efficiency.

6. CONCLUSIONS AND FUTURE WORK

This paper has displayed starting work on distinguishing proof of understudy attributes which relate with apparent adequacy/investment in class exercises, viability and expected execution. It is additionally impractical to, from the information gathered; disambiguate the choice factor of understudies undertaking the upper dimension OPSL course from other upper dimension understudies. Indeed, a significant number of these relationships may appropriate to any (or most) upper dimension courses.

Future work will incorporate gathering extra information with respect to the qualities that make understudy sentiments of adequacy and capacity to perform. This will be gathered over numerous contributions of the course and at various organizations. A comparative information gathering component will be used for extra upper dimension courses to survey what connections are explicit to OPSL and which are all the more for the most part material. At last, this investigation will proceed to asses' understudy execution and thoroughly analyze it with understudy desires and feeling of productivity

REFERENCES

1. Jeremy Straub and Scott Kerlin, Eunjin Kim, Analysis of Student Characteristics and Feeling of Efficacy in a First Undergraduate Artificial Intelligence Course. IEEE, 978-1-5090-4767-3/17, pp. 10-17. 2017.
2. V. Ramalingam and S. Wiedenbeck. Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. Journal of Educational Computing Research 19(4), pp. 367-381. 1998.

3. B. C. Wilson. A study of factors promoting success in computer science including gender differences. Computer Science Education 12(1-2), pp. 141-164. 2002.
4. P. Talaga and J. C. Oh. Combining AIMA and LEGO mindstorms in an artificial intelligence coursetobuild realworldrobots. Journal of Computing Sciences in Colleges 24(3), pp. 56-64. 2009.
5. L. Greenwald and D. Artz. Teaching artificial intelligence with low-cost robots. Accessible Hands-on Artificial Intelligence and Robotics Education, Ed.L.Greenwald, Z.Dodds, A.Howard, S.Tejada, and J.Weinberg pp. 35-41. 2004.
6. S. P. Imberman. Teaching neural networks using LEGO handy board robots in an artificial intelligence course. Presented at ACM SIGCSE Bulletin. 2003.
7. S. P. Imberman. An intelligent agent approach for teaching neural networks using LEGO® handy board robots. Journal on Educational Resources in Computing (JERIC) 4(3), pp. 4. 2004.
8. A. N. Kumar. Three years of using robots in an artificial intelligence course: Lessons learned. Journal on Educational Resources in Computing (JERIC) 4(3), pp. 2. 2004.
9. F. Klassner. A case study of LEGO mindstorms™ suitability for artificial intelligence and robotics courses at the college level. Presented at ACM SIGCSE Bulletin. 2002.
10. J. B. Schafer. Hands-on artificial intelligence education using LEGO mindstorms : Lessons learned. Presented at Proceedings of the 2004 Midwest Instruction and Computing Symposium. 2004.
11. E. Fermé and L. Gaspar. RCX PROLOG: A platform to use lego mindstorms™ robots in artificial intelligence courses. Presented at This Proceedings. 2007.
12. F. Klassner. Launching into AI's" october sky with robotics and lisp. AI Magazine 27(1), pp. 51. 2006.
13. A. Kumar, D. Kumar and I. Russell. Non-traditional projects in the undergraduate AI course. Presented at ACM SIGCSE Bulletin. 2006.
14. Z. Dodds, L. Greenwald, A. Howard, S. Tejada and J. Weinberg. Components, curriculum, and community: Robots and robotics in undergraduate ai education. AI Magazine 27(1), pp. 11. 2006.
15. F. G. Martin. Real robots don't drive straight. Presented at AAAI Spring Symposium: Semantic Scientific Knowledge Integration. 2007.
16. T. L. Dunn and A. Wardhani. A 3D robot simulation for education. Presented at Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia. 2003.
17. F. Klassner and S. D. Anderson. Lego MindStorms: Not just for K-12 anymore. IEEE Robotics & Automation Magazine 10(2), pp. 12-18. 2003.
18. F. Klassner and C. Continanza. Mindstorms without robotics: An alternative to simulations in systems courses. Presented at ACM SIGCSE Bulletin. 2007.
19. D. Blank, L. Meeden and D. Kumar. Python robotics: An environment for exploring robotics beyond LEGOs. Presented at ACM SIGCSE Bulletin. 2003.
20. R. Weiss and I. Overcast. Finding your bot-mate: Criteria for evaluating robot kits for use in undergraduate computer science education. Journal of Computing Sciences in Colleges 24(2), pp. 43-49. 2008.
21. R. Marín, P. J. Sanz and A. P. Del Pobil. The UJI online robot: An education and training experience. Autonomous Robots 15(3), pp. 283- 297. 2003.
22. B. S. Fagin and L. Merkle. Quantitative analysis of the effects of robots on introductory computer science education. Journal on Educational Resources in Computing (JERIC) 2(4), pp. 2. 2002.
23. B. Fagin and L. Merkle. Measuring the effectiveness of robots in teaching computer science. Presented at ACM SIGCSE Bulletin. 2003, .
24. X. Duan. Outcomes assessment of student learning for an educational robotics program.
25. Z. Markov, I. Russell, T. Neller and S. Coleman. Enhancing undergraduate AI courses through machine learning projects. Presented at Frontiers in Education, 2005. FIE'05. Proceedings 35th Annual Conference. 2005.
26. I. Russell, Z. Markov, T. W. Neller and S. Coleman. Enhancing undergraduate AI courses through machine learning projects. 2005.
27. S. A. Wallace, I. Russell and Z. Markov. Integrating games and machine learning in the undergraduate computer science classroom. Presented at Proceedings of the 3rd International Conference on Game Development in Computer Science Education. 2008.
28. J. E. Laird, A. Newell and P. S. Rosenbloom. Soar: An architecture for general intelligence. Artif. Intell. 33(1), pp. 1-64. 1987.
29. S. P. Imberman. Three fun assignments for an artificial intelligence class. Journal of Computing Sciences in Colleges 21(2), pp. 113-118. 2005.
30. A. P. Saygin, I. Cicekli and V. Akman. "Turing test: 50 years later," in The Turing TestAnonymous 2003.
31. N. Lavesson. Learning machine learning: A case study. IEEE Transactions on Education 53(4), pp. 672-676. 2010.
32. I. H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques 2005.
33. N. Pillay. A first course in genetic programming. ACM SIGCSE Bulletin 36(4), pp. 93-96. 2004.
34. R. L. Canosa. Image understanding as a second course in AI: Preparing students for research. Presented at ACM SIGCSE Bulletin. 2006.
35. R. M. C. García, J. V. Román and A. Pardo. Peer review to improve artificial intelligence teaching. Presented at Frontiers in Education Conference, 36th Annual. 2006.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

TABLE 3. CORRELATION BETWEEN AGE GROUP AND EXPECTED CLASS ACTIVITIES.

Age	Expected Difficulty	Writing Computer Programs	Working at Computer	Level of Understanding	Participation in Class	Participation in Lab	Office Hours
18-22	3.01	3.51	4.23	3.52	2.76	3.26	3.02
22-30	3.01	3.61	3.61	3.03	3.21	3.41	3.45
P-Val	0.51	0.41	0.15	0.010	0.19	0.39	0.25

TABLE 4 CORRELATION BETWEEN AGE GROUP AND METRICS OF EFFICACY / EXPECTED PERFORMANCE

Age	1	2.2	3	4	5
18-22	6.81	6.51	5.10	7.10	7.01
22-30	6.80	6.00	5.50	7.10	6.61
P-Val	0.25	0.15	0.23	N/A	0.10

TABLE 5 PRESENTS THE CORRELATION BETWEEN THE ACADEMIC YEAR AND THE QUESTIONS FROM WILSON (2002)[3].

Year	Expected Difficulty	Writing Computer Programs	Working at Computer	Level of Understanding	Participation in Class	Participation in Lab	Office Hours
Junior	3.10	3.61	4.01	3.41	2.81	3.41	3.01
Senior	3.10	4.10	4.01	3.26	3.51	3.51	3.26
P-Val	0.51	0.23	0..511	0.35	0.06	0.41	0.38

TABLE 6 CORRELATIONS BETWEEN ACADEMIC YEAR AND METRICS OF EFFICACY/EXPECTED PERFORMANCE

Year	1	2	3	4	5
Junior	6.61	6.41	5.00	7.01	7.10
Senior	6.76	5.76	5.75	7.01	6.50
P-Val	0.35	0.06	0.13	N/A	0.09

TABLE 7. CORRELATION BETWEEN GPA AND EXPECTED CLASS ACTIVITIES

GPA	Expected Difficulty	Writing Computer Programs	Working at Computer	Level of Understanding	Participation in Class	Participation in Lab	Office Hours
2.0-2.99	3.01	4.1	4.101	3.02	4.11	4.01	3.50
3.0-3.99	3.02	3.51	4.11	3.35	2.69	3.18	3.17
P-Val	0.51	0.05	0.61	0.11	0.001	0.03	0.43

TABLE 8 CORRELATIONS BETWEEN GPA AND METRICS OF EFFICACY / EXPECTED PERFORMANCE

GPA	1	2	3	4	5
2.0-2.99	7.52	5.51	5.01	7.02	6.61
3.0-3.99	6.69	6.51	5.18	7.21	7.11
P-Val	0.41	0.14	0.32	N/A	0.25

TABLE 9. CORRELATION BETWEEN SELF-LEARNING PROGRAMMING AND EXPECTED CLASS ACTIVITIES.

Self-learned programming	Expected Difficulty	Writing Computer Programs	Working at Computer	Level of Understanding	Participation in Class	Participation in Lab	Office Hours
N	2.76	3.50	4.00	3.01	3.01	3.26	3.01
Y	3.18	3.83	4.00	3.51	3.01	3.34	3.34
P-Val	011	0.23	0.50	0.05	0.51	0.45	0.28