Python Programming in an IS Curriculum: Perceived Relevance and Outcomes

Jennifer Xu
jxu@bentley.edu

Mark Frydenberg
mfrydenberg@bentley.edu

Computer Information Systems Department
Bentley University
Waltham, MA 02452

Abstract

Recent years have witnessed a growing demand for business analytics-oriented curricula. This paper presents the implementation of an introductory Python course at a business university and the attempt to elevate the course's relevance by introducing data analytics topics. The results from a survey of 64 undergraduate students of the course are analyzed to understand their perceived relevance of having Python programming skills upon entering the workplace, and how course design and other student characteristics influenced the perceptions of their learning and performance. Results demonstrate that business students are highly motivated to take Python programming courses to better position themselves for future career opportunities in the growing field of data analytics. We also found that students with no prior programming experience performed better than students who had some prior programming experience, suggesting that Python is an appropriate choice for a first programming language in the Information Systems (IS) curriculum. The paper concludes with recommendations for offering an analytics-focused first programming course to bring added relevance to IS students learning programming skills.

Keywords: Python programming, business analytics, IS curriculum, relevance of IS

1. INTRODUCTION

Information Systems (IS) programs have continually incorporated contemporary concepts and technologies to prepare students for the complex business and technology environment (Bell, Mills, & Fadel, 2013; Topi, Valacich, Wright, Kaiser, Numamaker Jr. & Sipior, 2010). The advent of the big data era in recent years has spawned an increasing demand for business analytics skills and talents. To remain relevant and successful in this fast-growing market, many IS programs have adjusted their curricula to include more business analytics focused courses (Clayton & Clopton, 2019; Hilgers, Stanley, Elrod, & Flachsbart, 2015; Holoman, 2018; Sidorova, 2013; Wymbs, 2016). A common trend has been the shift from Java programming courses to Python programming courses. Because of its simplicity, flexibility, and availability of many libraries for data analysis, Python has become a widely used language for business analytics, marketing analytics, finance analytics, and many other application domains requiring data analytics. Python is ranked the world's most popular coding language by IEEE (Cass, 2018) and was named the "Language of the Year" of 2018 by the Application Development Trends Magazine (Ramel, 2019).

Although Python's popularity has been well known, it remains unclear how IS and business students perceive the importance of Python programming skills to their future careers and what factors influence their learning outcomes in Python courses. This paper is motivated by the IS discipline's dedication to maintaining relevance (Agarwal & Lucas, 2005; Baskerville & Myers, 2002; Benbasat & Zmud, 2003; Robey, 1996) and, more specifically, by the surging demand for including Python programming into the curricula of business schools or IS programs specializing in business analytics. We intend to address two research questions:

- *RQ1: How do IS and business students perceive the relevance of Python programming to their future career?*

- *RQ2: What factors impact the students' perceptions of learning outcomes and their actual performance?*

To address these research questions, we conducted a survey study and collected responses from 64 undergraduate students who took an introductory Python programming course at a business school of a northeastern U.S. university. We expect that students' positive perceptions of the relevance and learning outcomes may encourage enrollment in programming and other IS courses and help promote IS programs.

2. LITERATURE REVIEW

Business Analytics in IS Curriculum

To maintain relevance to the ever changing, increasingly complex technological and business environment, the Association for Information Systems (AIS) has developed a series of model curricula for graduate and undergraduate IS programs (Gorgone, Davis, Valacich, Topi, Feinstein & Longenecker Jr., 2002; Topi, Helfert, Ramesh & Wigand, 2011; Topi et al., 2010). However, because of various constraints, adhering to a standard model curriculum (Bell et al., 2013) is often difficult for all IS programs, and a more specialized curriculum may help IS programs seize market opportunities, maintain relevance, and address various challenges such as declining enrollment (Sidorova, 2013).

The emergence of data science and data analytics has brought about increased interest in introducing coding to non-computer science majors (Holoman, 2018; Silveyra, 2019; Wilder & Ozgur, 2015). Many IS programs have already implemented business analytics-oriented curricula in order to keep IS education relevant in a data-centric business environment. Among the business analytics skills recommended by prior studies (Gupta, Goul, & Dinter 2015; Hilgers et al., 2015; Wymbs, 2016), programming has repeatedly been identified as an essential skill. Although application development was removed from the core of IS 2010 model curriculum (Topi et al., 2010), over 80 percent of the IS programs surveyed still kept programming courses in their curriculum core (Bell et al., 2013). This reflects the belief of many IS educators that programming remains an important topic and that, although software development may no longer be a typical career choice for IS graduates, programming is a very useful skill for future business professionals.

Learning and Teaching Programming

Our RQ2 concerns learning outcomes. A large body of research that investigates the psychological and cognitive processes of learning to program and their impacts on learning outcomes (e.g., teaching effectiveness and individual performance) can be found in the literature of computer science (CS) education. The main finding in the literature is that learning outcomes can be affected by various factors including learners' cognitive development levels, learning styles, motivations, background, prior experience, and learning context and environment (Lau & Yuen, 2009; Robins, Rountree & Rountree, 2003; Shaw, 2012; Tie & Umar, 2010; White & Sivitanides, 2002). Roughly speaking, these factors can be grouped into three categories: individual characteristics, language characteristics, and context characteristics.

Individual Characteristics

Prior research has long studied the impact of individual characteristics on the outcomes of learning to program. One of the important characteristics is an individual's cognitive development level (Mayer, Dyck & Vilberg 1989). The cognitive development theory (Piaget, 1972) identifies three age-related development levels: pre-operational (2 − 7 years), concrete (7 − 12 years), and operational (12 years and above). The operational level requires abilities to abstract, form hypothesis, and solve complex problems (Biehler & Snowman, 1986). White and Sivitanides (2002) posited that an individual's cognitive development level predicts her programming performance and that different languages require different cognitive levels. For example, scripting and markup languages (e.g., HTML) require lower levels than object-oriented languages (e.g., Java and C++). Studies have found that an individual's learning style can also affect her performance in learning to program

(Lau & Yuen, 2009; Shaw, 2012; Tie & Umar, 2010). Perkins, Hancock, Hobbs, Martin & Simmons (1989) identified two types of learners, stoppers and movers, differentiated by their attitudes and behaviors when encountering a problem or a lack of direction to proceed. Stoppers often have a negative emotional reaction to errors and problems and cease to try; while movers continue to try, search, experiment, and revise. Research has also investigated the impacts of many other individual characteristics. For example, individuals with strong motivations often commit themselves to performing well and to acquiring the skill (Pendergast, 2006). Additionally, several studies have focused on the gender effect because of the dominance of male programmers in the information technology sector (Lau & Yuen, 2009; Underwood, G., McCaffrey, M., & Underwood 1990; Yau & Cheng, 2012) and reported mixed findings.

Although CS education research has accumulated a significant body of knowledge about learning and teaching programming, only a limited number of studies in the IS literature (Pendergast, 2006; Roussev, 2003; Urbaczewski & Wheeler, 2001; Zhang, Zhang, Stafford, & Zhang, 2013). can be found to focus on teaching business students how to program. Business students are different from CS students in many aspects (e.g., motivations, background, and perceptions of relevance). For this research question (RQ2), we propose our first hypothesis as

H1: A student's individual characteristics have a significant impact on the student's perceived learning outcomes and actual performance.

The literature has also shown that prior programming experience affects students' perceived learning and outcomes. (Bergin & Reilly, 2005; Bowman, Jarratt, Culver, and Segre, 2019). Students' perception of understanding a topic has the strongest correlation with their programming performance, and their own experience is related to how well they understood the concepts and their level of confidence their own work. Given this research, we propose our second hypothesis as

H2: A student's prior experience of programming has a significant impact on the student's perceived learning outcomes and actual performance.

Language Characteristics

In the history of programming languages, several types with different language characteristics have emerged, ranging from procedural (e.g., COBOL and C), OO (object-oriented, e.g., Java and C++), scripting (e.g., JavaScript), to visual (e.g., Visual Basic). Since the 1990s, Java has been a dominant language for teaching introductory programming (Shein, 2015). As a scripting language, Python is advantageous for its simplicity in syntax and flexibility in data structures, and has become more popular in recent years in introductory programming courses (Shein, 2015). Based on these findings, we posit in this research that

H3: A student's perceptions of the language characteristics of Python have a significant impact on the student's perceived learning outcomes and actual performance.

Context Characteristics

Learning context is a multi-faceted construct and may vary in terms of type (e.g., orienting context, instructional context, and transfer context) and level (learner, immediate environment, and organizational) (Tessmer & Richey, 1997). An individual's perception of the learning context may have a profound impact on his/her learning experience (Ramsden, 2005). The course design (e.g., lectures and topics) and study process can affect students' understanding of the concepts and performance in a Java programming course (Govender, 2009). Similarly, different teaching approaches (lecture + exercise vs. lecture-only) have been found to result in different student performance in an introductory C programming course in an IS program (Zhang et al., 2013). The literature has also reported the impacts of other contextual factors. This research focuses on the impact of course design in terms of topics and homework assignments. We hypothesize that

H4: A student's perception of the course design has a significant impact on the student's perceived learning outcomes and actual performance.

3. COURSE DESIGN

The introductory Python programming course is an elective for undergraduate CIS (Computer Information Systems) majors and minors at a northeastern U.S. business university. Undergraduate CIS majors are required to take a semester course in Java programming (Java I) and can choose to take an advanced Java programming course (Java II) as an elective.

Undergraduate CIS minors take a course in HTML and JavaScript, and may take Python to further their study of programming. This Python course has no prerequisites other than an introductory IT course required of all first-year students, and recently has become a prerequisite for the Introduction to Data Science course offered through the Mathematics department.

This course met for two 80-minute sessions each week in a 14-week semester. Each class session included instructor-led lectures or demonstrations, and often short, in-class exercises that reinforced the topics presented. One instructor taught two sections; the other taught one section. All sections used the same syllabus and shared common assignments and exams.

The evaluation of student performance consisted of seven programming assignments (40% of the grade), lab participation (5%), midterm exam (25%), and final exam (30%). Table 1 in Appendix 2 presents the topics covered and the seven homework assignments. Table 2 in Appendix 2 shows the grade distribution across three sections of the course. The average grade was 2.7/4.0.

This course presents basic programming concepts and techniques using Python 3, including loops and selection statements; data structures (e.g., lists and dictionaries); classes, and objects. Instructors omitted advanced topics such as higher order functions (e.g., map, reduce, filter, lambda), and other topics frequently taught in Java programming courses (e.g., graphics and user interface design), teaching instead, basic capabilities of several popular Python libraries for data analysis: NumPy, Matplotlib, and Pandas.

Including data analytics topics in an introductory Python programming course is a relatively new phenomenon, as evidenced by the lack of introductory textbooks from major publishers containing this content. Table 3 in Appendix 2 lists popular introductory Python textbooks from major publishers. These texts have a computer science focus and include advanced programming topics such as recursion, inheritance and polymorphism. Case studies or coding examples on graphical user interfaces, graphics processing, operating systems, or client server programming are less relevant to information systems students learning Python because of their interest in data science or data analytics. On the other hand, reference textbooks teaching data science topics generally assume prior programming experience. In addition to a standard Python textbook, we used online documentation for reference when teaching data analytics topics, and had students interact with examples and materials in the same way that professional developers might use these resources. This approach ensures the analytics examples use current versions of those modules and minimizes the burden on instructors to create a plethora of new materials. A homework assignment had students use functions from the three analytics libraries to perform simple text analysis of a sample of tweets.

4. RESEARCH METHODS

Survey Methodology

To answer our research questions, we used the survey methodology to collect data from the three sections of this course. A pre-course survey was administered in the first week of the semester and a post-course survey in the last week before the final exam. The pre-course questionnaire consists of questions about the student demographics, background, prior programming experience, and motivations to take the course. The post-course survey includes questions about students' attitudes and opinions about the course design (topics and homework assignments), their learning styles, and perceived outcomes of this course.

Assignments and exams were standardized across all instructors and sections. The 70 students who enrolled in the three sections were invited to take the pre- and post-course surveys. Both surveys were administered online where students' emails were captured by the survey website (Qualtrics) through individualized invitation links sent to students' email accounts. However, students were assured that their responses would not be accessed before their grades were posted to remove the social desirability bias (Campbell & Standley, 1963).

Among these students, 36 out of 70 (51.4%) are male and 34 (48.5%) are female. The mean age is 20.8 years. The majority of the students are seniors (n = 39, 55.7%) or juniors (n = 26, 37.1%) and only 6 (8.6%) students are sophomores and one student is a freshman. The large number of seniors is attributed to seniors having priority to register for the class first. Students majored in different business disciplines including CIS (n = 27, 38.6%), Finance (n = 15, 21.4%), Actuarial Science (n = 9, 12.9%), and other business majors such as Accounting, Marketing, Management, etc. (n = 17, 24.3%). Two students had not declared their majors by

the time of the pre-course survey. Fifty-six students (80%) indicated that they had prior experience of at least one programming language, including Scratch, VB, JavaScript, Java, C++, or C#. The numbers of students who had taken Java I and Java II are 37 (52.9%) and 11 (15.7%), respectively. In addition, 28 students (40%) had taken a web development course using HTML and JavaScript.

Six of the returned responses were incomplete with missing answers to some important questions and therefore removed from the study. The resulting sample consisted of 64 valid responses.

Variables and Measures

Independent variables used in this study are grouped into three categories: *individual characteristics*, *language characteristics*, and *course design (context) characteristics*. Tables 1 and 2 in Appendix 1 list the variables and corresponding items in the pre- and post- survey questionnaires.

Individual characteristic variables include gender, year, number of motivations (#motivs), number of prior programming languages (#prior_langs), and three dummy variables representing whether a student had taken Java I (Java_1), Java II (Java_2), and the Web Development (HTML) courses, respectively. As students grow and become more mature over their four years of college, we use a student's age and year (i.e., freshman, sophomore, junior, and senior) to approximately represent his/her cognitive development level. The number of motivations is captured by a pre-course survey item asking students their motivation to take the course with four non-exclusive options: "to increase my career opportunities", "I'm interested in the topic", "I will use Python in my own business in the future," and other (specify). The pre-course survey also asks students to check any programming languages they had learned previously (e.g., Scratch, VB, JavaScript, Java, etc.).

To assess individual learning style, a survey item asks students, when encountering a problem or an error, how frequently (sum to 100%) they would (a) ask the instructor for help, (b) visit the CIS learning center, (c) ask other classmates, (d) solve by themselves, or (e) search online. The total from (a)-(c) is calculated as an indicator (style_stopper) for the extent to which a student is a "stopper" (Perkins et al., 1989).

The language characteristic group includes two variables measured by two 5-point Likert scale questions in the post-course survey: perceived difficulty of Python syntax (syntax) and the perceived difficulty of programming logic (logic).

The independent variables in the context group regarding the course design are perceived usefulness of the topics (topics_useful), perceived relevance of the topics (topics_relevant), perceived helpfulness of the homework assignments (hw_helpful), and perceived difficulty of homework (hw_difficult). The averages of the scores for the topics and homework assignments by each student are used for the values of these variables.

The control variables include age, section, major, perceived overall difficulty of the course (course_difficult), and student self-reported average number of hours spent on this course in each week (hours_spent). Also used as a control variable, Python_relevant, is a student's perceptions of the overall relevance of Python measured by a group of six 5-point Likert scale questions (ranging from strongly disagree to strongly agree) in the post-course survey. The average of the six scores by each student is used for the value of this variable.

The two dependent variables are a student's perceived outcomes (outcomes) and the actual performance (grade). The perceived outcomes are captured in the post-course survey using a set of four 5-point Likert scale questions and measured using the average of the four scores (see Appendix). The student grade is a value between 0 and 100.

The post-semester survey also included open-ended questions on the delivery of this course and suggestions for future semesters.

4. ANALYSIS AND RESULTS

Perceived Relevance

To address the first research question (RQ1) about the perceived relevance of the Python programming course, we performed descriptive analysis of the responses from the surveys. The post-survey items regarding the overall relevance (Python_relevant) and the relevance of specific topics (topics_relevant) were used to assess students' perceptions (see Table 4 in Appendix 2). The scores range from 1 (strongly disagree) to 5 (strongly agree). The first data column in Table 4 presents the means (and standard deviations) of these variables for all students.

Appendix 2. Tables

#	Topics	Homework Assignments	Competencies Demonstrated
1	Display information using *print*()	About You: print information about you	Input and print functions
2	Expressions and Data Types	Restaurant: calculate totals of food orders based on unit price and quantity purchased	Built in functions, formatting
3	Control Structures (loops and selections)	Buzz Game: test for numbers containing or divisible by 7 (Offenholley, 2012)	For Loops, While Loops, If/Else, and If/Elif/Else statements, writing functions that return values
4	Strings and Text Files	User Account Management: store usernames, passwords, and allow users to add/edit/delete account information	Read and write text files, CSV files, use CSV reader and DictReader
5	Data Structures (List and Dictionary)	Donor Information Processing: maintain list of donors and donation amounts; determine most generous donors, and total donations	Read CSV files into a dictionary, list and dictionary methods
6	Classes and Objects	Battleship Game: create different classes (Grid, Ship, Game); enable communication and collaboration between objects	Create original classes and objects, constructors and methods
7	Introduction to Data Analytics	Tweet Analyzer: download and analyze a sample of tweets to determine most popular hashtags; create charts showing frequencies of hashtags and mentions	Test File Processing, Charts with Numpy and Matplotlib, filtering and sorting Pandas DataFrames, pie, bar, and other charts, with Pandas

Table 1. Topics and homework assignments.

Numeric Grade	Letter Grade	Instructor 1, Section 1	Instructor 1, Section 2	Instructor 2, Section 1
4.0	A	2	4	2
3.7	A-	5	3	3
3.3	B+	1	4	3
3.0	B	1	2	2
2.7	B-	5	3	3
2.3	C+	3	3	1
2.0	C	2	4	4
1.7	C-	1	0	1
1.3	D+	1	1	0
1.0	D	0	1	1
0.7	D-	0	0	1
F	F	2	0	1

Table 2. Grade distribution frequency across three sections offered.

Introductory Python Textbooks Considered.

- Downey, Allen. Think Python: How to Think like a Computer Scientist. O'Reilly Press, 2016.
- Lambert, Kenneth. Fundamentals of Python: First Programs 2nd Edition. Cengage, 2019.
- Liaing, Y. Daniel. Introduction to Programming Using Python. Pearson, 2013.
- Punch, William & Enbody, Richard. The Practice of Computing Using Python. Pearson. 2017.
- Zelle, John. Python Programming: An Introduction to Computer Science. Franklin, Beedle. 2017

Topic	Downey	Lambert	Liaing	Punch	Zelle
Computing Overview	1	1	1	0	1
Basic I/O and simple programs	2	2	1	1	2
Numeric Data Types	2	2	2		3
Graphics, Image Processing		7			4
Strings	8	4	3,8	4	5
Lists, Tuples, Dictionaries	10,11,12	5	10,11,14	7,9	5,11
Files and Exceptions	14	4	13	6, 14	5
Functions	3,6	6	6	5,8	6
If Statements and Booleans	5	3	4	2	7
Loops and Booleans	7	3	5	2	8
Program Development	4,9,20		7	10	9
Classes and Objects	15,16,17,18	9	12	11, 12, 13	10, 12
Algorithms	20	11		3	13
Recursion	5		15	15	13
Advanced Topics	19			16	
Windows-Based GUI		8	9		
Networking /Client Server		10			
Data Analytics Modules	NONE	NONE	NONE	NONE	NONE

Table 3. Python Textbooks and Contents. Numbers are corresponding chapter/modules covering each topic.

	All Majors	Individual Majors			
		CIS	Finance	Actuarial Science	Other Business Majors
Overall relevance (*Python_relevant*)	3.49 (1.21)	3.73 (1.50)	3.59 (0.82)*	3.11 (0.61) **	3.21 (1.10) **
Topic relevance (*topics_relevant*)	3.97 (0.82)	4.21 (0.87)	3.82 (0.95)*	3.89 (0.78) **	3.66 (0.64) **

** $p < 0.01$; *** $p < 0.001$

Table 4. Students' perceptions of the relevance of Python programming course.

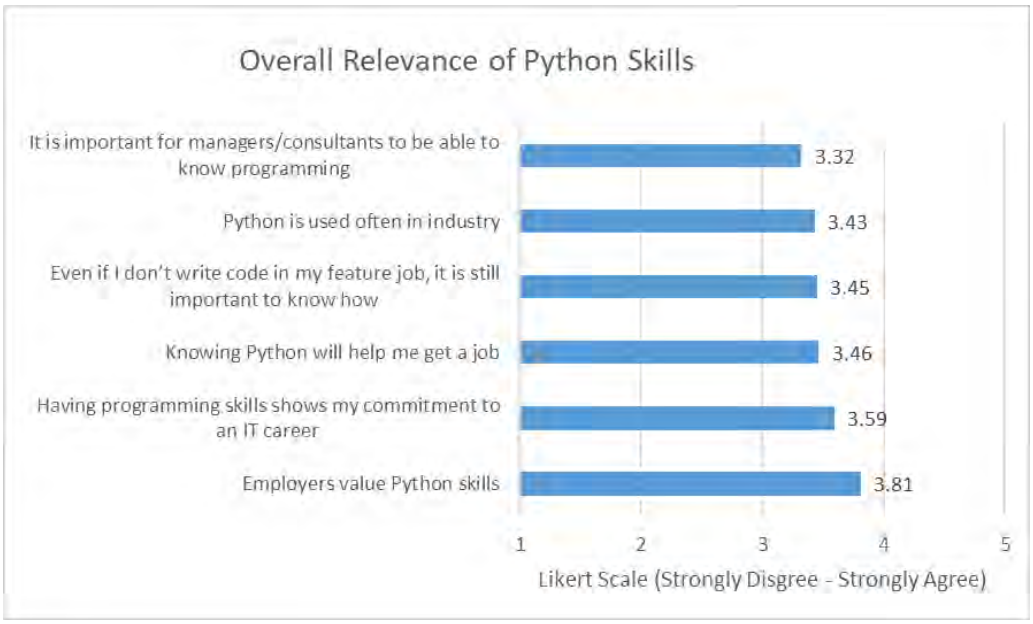
		Grade	Perceived Outcomes
Individual	*year*	-0.255	0.039
	gender (female)	0.405**	0.199
	#motivs	-0.281**	-0.015
	#prio_langs	0.165	0.166
	Java_1	-0.034	0.058
	Java_2	-0.138	0.115
	HTML	-0.009	-0.247
	style_stopper	-0.342*	0.049
Language	*syntax*	-0.146	0.059
	logic	-0.231*	-0.239*
Course Design	*topics_useful*	-0.056	0.325**
	topics_relevant	0.215	0.309**
	hw_helpful	0.048	0.173
	hw_difficult	-0.32*	-0.035
Control	*age*	0.114	-0.089
	section	0.054	-0.171
	Major (IS)	-0.356**	-0.041
	hours_spent	0.01	0.432**
	overall_difficult	-0.077	-0.191
	overall_ relevance	-0.031	0.068
R^2		0.67	0.64

***p < 0.01, *p < 0.05*

Table 5. Summary of regression analysis results.

Appendix 3. Charts and Visualizations

(a)



(b)

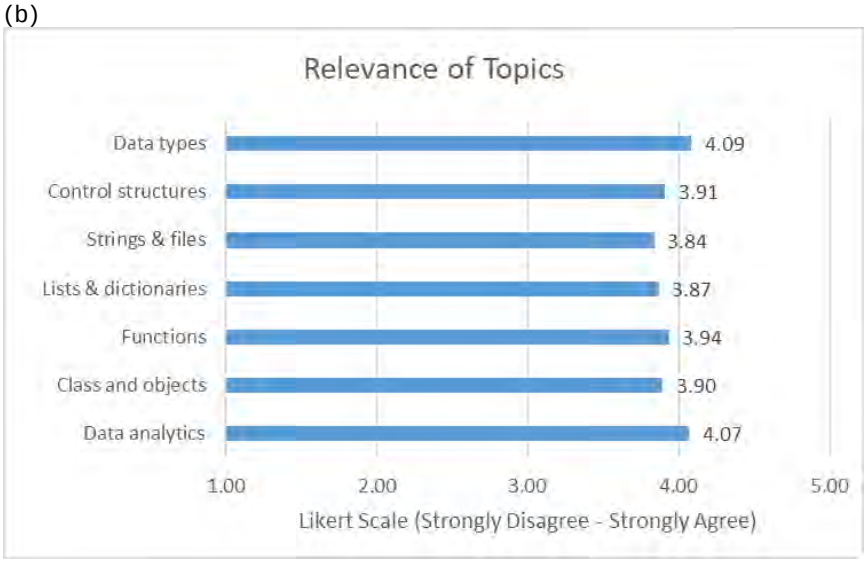


Figure 1. Students' opinions about the relevance of Python programming skills (a) and the relevance of individual topics (b).

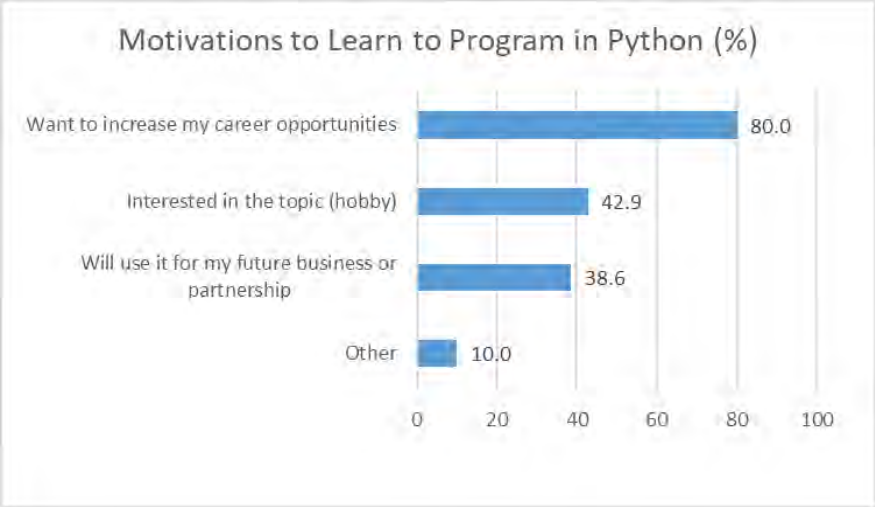


Figure 2. Students' motivations to take the Python programming course.

Figure 3. Students' perceptions of the learning outcomes.