Introduction to computing: A survey of courses in Greek higher education institutions

Nikolaos Avouris
Electrical &Computer Eng. Department, University of Patras, Greece
avouris@upatras.gr

ABSTRACT

This paper presents the landscape of introductory courses in computing across Greek higher education institutions. The survey is based on publicly available data on courses and textbooks, obtained by the eudoxus.gr higher education courses/textbooks website. Research questions addressed include: (a) the approaches used and technologies applied in different courses, derived from the main course textbooks used; (b) differences between programs of study that have computing as major subject and no-computing ones; (c) trends in teaching computer science over a period of the last six years, and (d) comparison of the findings to other recent surveys in other parts of the world. This survey, which is the first of its kind in such scale for Greece, allows the community of computer science educators to define current progress of the field and address challenges related to teaching introduction to computing courses.

CCS CONCEPTS

• Computing education→ Computing education programs→ Computer science education

KEYWORDS

Introduction to computing, computer languages, programming, higher education institutions

1 INTRODUCTION

The *Introduction to computing* course (in the literature, this is referred as CS1) is the first course in computing in the curriculum of a program of study. Its design and objectives have been a topic of intense discussion since the birth of Computer Science discipline. The first ACM recommendation (ACM Curriculum, 1968 [1]) noted that the first course should provide the student with "*the basic knowledge and experience necessary to use computers effectively in the solution of problems*", and suggested that the course could be used both for majors (students of computer science programs of study) and non-majors (students in other disciplines). Koffman et al., 1984 [8] made a recommendation for CS1 with the following learning objectives: *To introduce problem solving methods and algorithm development; to introduce abstraction in process and data, to teach software engineering methodology; to teach a high-level programming language; to provide familiarity with the evolution of computer hardware and software technology; to provide a foundation for further studies in computer science*.

These objectives have not changed drastically since, however the discussion on the contents and the approach of the introductory course continues. For instance, in ACM Curriculum 2001, six general models are proposed: Imperative-first, Objects-first, Functional-first, Breadth-first, Algorithms-first, and Hardware-first. This reflects the concern that the introductory course should either focus narrowly on some aspect of computing, e.g. programming, or be a breadth-first theoretical course. The most recent recommendation (ACM Curriculum 2013) acknowledges this, by stating that a key reason for which the content of such course remains a vigorous discussion topic after decades of debate, is that *"not everything relevant to a computer scientist (programming, software processes, algorithms, abstraction, performance, security, professionalism, etc.) can be taught from day one"*, so ultimately, choosing what to cover in an introductory course results in a number of informed tradeoffs.

One of the key decisions, especially if programming is the main focus of the introductory course, is the selection of an adequate programming language. Over the years, various trends where observed with respect to the first programming language. As Giangrande, 2007 [5] and Farooq et al., 2014 [3] have outlined through historic perspectives, different approaches have been adopted over the years, starting from Assembly (1960s), followed by Fortran (1970s), Pascal (early 1980s), C and Modula2 then C++ (1990s) and subsequently Java from 2000s until today, while Python seems to be the most prominent emerging new language since 2010. A representative example of this trend, for the last 10 years, is depicted in Figure 1, for Australian universities. In the plot, it is evident that Java and Python in 2013 were the most popular first programming languages, but the trend is that of decline of Java, emergent use of Python, while at the same time

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work
owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
PCI '18, November 29-December 1, 2018, Athens, Greece
© 2018 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6610-6/18/11…$15.00
https://doi.org/10.1145/3291533.3291549

we observe a decline of Basic, and an increase of the use of JavaScript. The reasons for paradigm shifts during these times have been reported as changes in technology, and adoption of new programming approaches like structured programming, object-oriented programming, etc.

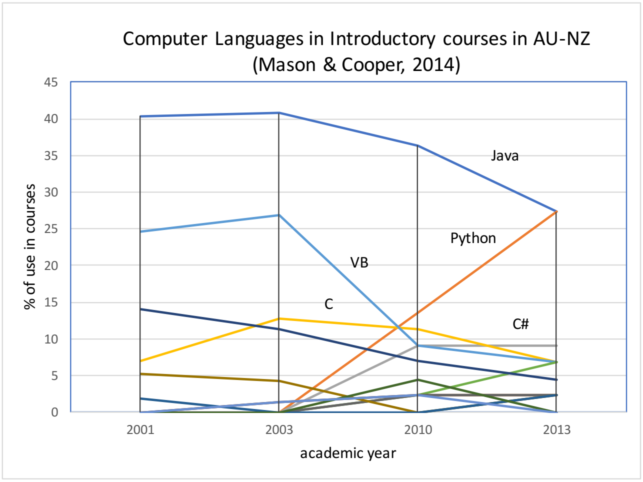

Figure 1: Trends in first programming language in Australian continent Mason et al. 2018 [10].

Given this background, it is interesting to map the landscape in a specific country, e.g. Greece, joining other studies that have been produced on this subject for countries like US, UK, Australia, Ecuador, etc. The main research questions to be addressed in the following, for the Greek higher education, are: RQ1: Which are the most widely used approaches in Introductory to Computing courses; RQ2: What are the differences between computing and not-computing disciplines; RQ3: How has teaching of the introductory course evolved over the years; RQ4: How the Greek higher education compares to other countries in this area? These questions will be addressed in the next sections.

2 METHOD OF STUDY

The data used in this study have been collected from the eudoxus.gr website that serves Greek higher education institutions, containing information on textbooks for the courses, allowing students to select textbooks distributed freely by Greek universities to their students. The eudoxus.gr database, contains information on programs of study, courses and recommended textbooks, since 2012, when this service went in operation. So, it is a valuable source of information on the Greek higher education system.

We used *scrape_eudoxus* an open source script [1] that allows downloading a relational database of universities, programs of study, courses and recommended textbooks for a selected academic year. For the purposes of this paper, the full eudoxus.gr database was built for the academic years 2011-2012 and 2017-2018, in order to compare the *Introduction to computing* courses of the Greek higher education institutions that are registered with the service. The downloaded database relates to the 44 higher education institutions of Greece (in Greece, according to the Constitution, all higher education institutions are public, supervised by the Ministry of Education, and use this service). The size of the database is shown in Table 1. As it can be seen, around 40,000 courses and 60,000 textbooks are contained in this site per academic year for over 500 programs of study that are offered by the Greek higher education system.

Using this database, we run a query that extracted the introductory to computing courses, that are taught in the first year of study of Greek higher education institutions.

Acad.year	2011-2012	2017-2018
Programs	558	557
Courses	40169	37884
Textbooks	58086	70151

Table 1: Size of databases used in the study

The query was as follows:

```
select courses.id, courses.semester,
courses.name, programs.id, programs.uni,
programs.name
from programs join courses on courses.prog_id =
programs.id
and courses.semester < 3 and courses.semester > 0
and (courses.name like "%ΠΛΗΡΟΦΟΡΙΚ%"
  or courses.name like "%ΥΠΟΛΟΓΙΣΤ%"
  or courses.name like "%ΠΡΟΓΡΑΜΜΑΤΙΣΜ%")
and programs.year = 'YEAR'
order by courses.id;
```

This query for `YEAR=2012` and `2018` returned all courses that contained stems *comput-*, *program-*, *informati-* in their title. Then we manually cleared off 1st year courses of no introductory character (e.g. computer architecture, computational mathematics, etc.), as well as other courses of first year of more advanced (CS2) character. Finally, 396 courses for 2011-2012 and 340 for 2017-2018 were identified, on which information on recommended textbooks was available. Then we processed their main textbooks (the first two in the textbook list were considered the most representative of the course content, in case a course reading list contained more textbooks) and parsed for keywords that may identify their main technology or approach used. The keywords that were searched in the textbook titles where the following:

```
'C', 'C++', 'BASIC', 'QBASIC', 'C#', 'PYTHON', 'MATLAB',
'PASCAL', 'FORTRAN', 'JAVA', 'OFFICE', 'EXCEL', 'R',
'MATHEMATICA', 'LINUX', 'UNIX', 'AUTOCAD', "HTML",
"PROLOG", "SCRATCH", "SCALA", "SCHEME", "JAVASCRIPT",
"CNC", "SPSS", "FLASH", "VBA", "BASIC.NET"
```

Subsequently, we grouped similar technologies under the same term, for instance `"VB"`, `"QBASIC"`, `"Visual Basic"`, under the term `"Basic"`. Finally, we classified technologies under the label *Programming*, *Skill-oriented* or *Theoretical*

[1] https://github.com/navouris/scrape_eudoxus

teaching approaches, the latter if neither of the first two labels applied.
Before proceeding further, we inspected the textbooks that were labeled as '*Theoretical*, in order to identify possible other technologies, not contained in the original list, or books of no theoretical character.
Through this process, courses have been identified that used a combination of approaches, e.g. multiple programming languages, as it will be discussed next.
Using this dataset, we attempt in the following to answer the research questions that were formulated in the Introduction.

Approaches	all		Computing		No Computing	
Programming	101	30%	30	63%	71	24%
Programming,Skills	8	2%		0%	8	3%
Programming,Skills,Theory	1	0%		0%	1	0%
Programming,Theory	11	3%	3	6%	8	3%
Skills	28	8%		0%	28	10%
Skills,Theory	24	7%		0%	24	8%
Theory	167	49%	15	31%	152	52%
total	**340**	100%	**48**	100%	**292**	100%

Table 2: Teaching approaches in Introductory courses (a) for all courses, (b) for computing programs, and (c) for non-computing programs

3 RQ1: WHICH ARE THE MOST WIDELY USED APPROACHES IN INTRODUCTORY TO COMPUTING COURSES

The first question to be answered is related to the approaches used in *Introductory to Computing* courses in the Greek higher education institutions. Following the data analysis described in the previous section, we derived the results, shown in Table 2(a).
As it can be seen, from the 340 courses of 2017-18, for which there was information on textbooks, half of them followed a purely *Theoretical* teaching approach (breadth-first), 30% followed a *Programming* approach, around 5% a programming approach combined with skills-based or theoretical one, around 15% a *Skill-oriented* approach, like use of personal productivity tools (e.g. MS Office suite), data analysis using Excel, introduction to web technologies, Linux shell, etc.
Next we focus on just the 121 courses that follow in some degree, a *Programming approach,* shown in Table 3 (b). In these courses, the language most widely used is C in 37% of the courses, followed by MATLAB (17%), then Basic (11%), C++ (6%), Fortran (6%), Java (5%), and Python (4%). There are a few courses with a combination of languages (e.g. C, C++), or other languages, like Pascal, R, Prolog, etc.
In terms of the programming paradigm used, if we assume that courses in which object-oriented languages are taught partially or fully, follow an object-first approach, just 17 out of the 121 courses (14%) follow an object-oriented approach (using Java, C++, C#, etc.). It should be however further observed that even this limited adoption may be in doubt, as in some cases use of C++ or Java for a course does not on its own mean adoption of objects-first approach, as often a subset of these languages may be used to follow an algorithms-first approach.

academic year	2011-2012		2017-2018		dif
programming language	courses	%	courses	%	%
C	57	42%	45	37%	-5%
MATLAB	14	10%	20	17%	6%
BASIC	23	17%	13	11%	-6%
C++	8	6%	7	6%	0%
FORTRAN	15	11%	7	6%	-5%
JAVA	5	4%	6	5%	1%
PYTHON		0%	5	4%	4%
PASCAL	6	4%	4	3%	-1%
R		0%	4	3%	3%
C,C++	1	1%	2	2%	1%
C,FORTRAN	1	1%	2	2%	1%
BASIC , FORTRAN	3	2%		0%	-2%
other	3	2%	6	5%	3%
total	136	100%	121	100%	

Table 3: Programming languages in introductory courses (CS ad no-CS) for academic years (a) 2011-12 and (b) 2017-18

4 RQ2: COMPARISON OF COMPUTING PROGRAMS TO NOT-COMPUTING PROGRAMS

The next research question relates to comparison of approaches and technologies used in courses that belong in *Computer Science* programs of study versus other programs of study. We separated the programs of study according to their title, using the following function that checks for keywords in the title:

```
def check_major(title):
  title = title.strip().split()
  words = [x for x in title if x.startswith("ΠΛΗΡΟΦΟΡΙΚΗ") or
       x.startswith("ΥΠΟΛΟΓΙΣΤ") or x.startswith("Η/Υ") or
       x.startswith("ΨΗΦΙΑΚ")]
  if words: return True
  else: return False
```

After inspecting the programs titles to confirm that the rule applies to the data set under study, we identified 33 computing programs and 236 non-computing ones. This means that in total 269 programs (out of 557 in total, 48%) have computing-related courses in their first year. Then we performed the analysis of the previous section, separately to courses of computing and no-computing programs of study. We thus derived Table 2 columns (b) *Computing* and (c) *No-Computing*.
It should be mentioned here, that traditionally in Greek universities, it is rare the same course to be offered to both major and non-major students in Computing, as courses are offered in large extend by the same department faculty, so the *No*

Computing departments tend to design and offer themselves the computing courses, tailoring them to their own needs and requirements.

As shown in Table 2, the *No Computing* programs of study (mostly programs in science, engineering, business etc.) tend to use a programming approach much less (24% of the courses), compared to *Computing* programs (63%). On the other hand, the skills-based approach is applied in 18% of the cases for *No Computing* programs, and does not appear at all in *Computing* programs of study.

In Table 4, more specific information on technologies that characterize the introductory to computing courses is shown.

In courses of *Computing* programs, 54% use C, and much less other technologies (6% C++, 6% Python, 4% Java). A theoretical approach characterizes 38% of the courses for these programs.

Technology	Courses	%	CS	%	No-CS	%
theory	203	60%	18	38%	185	63%
C	51	15%	26	54%	25	9%
OFFICE	41	12%	0	0%	41	14%
MATLAB	21	6%	0	0%	21	7%
EXCEL	19	6%	0	0%	19	7%
BASIC	13	4%	0	0%	13	4%
FORTRAN	10	3%	0	0%	10	3%
C++	9	3%	3	6%	6	2%
PYTHON	8	2%	3	6%	5	2%
JAVA	6	2%	2	4%	4	1%
PASCAL	4	1%	0	0%	4	1%
R	4	1%	0	0%	4	1%
MATHEMATICA	2	1%	0	0%	2	1%
Linux	2	1%	0	0%	2	1%
C#	2	1%	0	0%	2	1%
PROLOG	1	0%	1	2%	0	0%
AUTOCAD	1	0%	0	0%	1	0%
HTML	1	0%	0	0%	1	0%

Table 4: Frequency of occurrence of specific technologies and approaches (a) in all courses (340 courses), (b) in 48 courses of Computer Science programs and (c) in 292 courses of No-Computer Science programs.

The *No Computing* programs have quite different approaches. Theoretical approaches appear in almost two thirds of the courses, MS Office is used in 14% in the case of skill-oriented courses, Special reference to MS Excel appears in 7% of the courses, C is the most popular programming language with 9%, followed by MATLAB with 7%, Basic (4%) and Fortran (3%). It should be mentioned that the percentage in this Table do not add to 100% as there are courses in which more than one technologies appear.

It is also interesting that some technologies like MATLAB, Fortran, as well as the skill-related technologies appear in just no-Computer science-program courses and not at all in Computer Science ones.

5 RQ3: TRENDS IN TEACHING COMPUTER SCIENCE

The next research question is subject of a study identifying the trends in teaching *Introductory to Computing* courses over time.

Since our data source covers the period from academic year 2011-12 until today, we decided to compare the situation between then and 2017-18, in order to identify trends during this 6-year period.

In terms of teaching approaches (Table 2) there has been no significant changes during this period, with slight increase in programming approaches in computing programs. More specifically, in computing programs an increase in programming approaches by 7% was observed (from 56% of the courses in 2012 to 63% of the courses in 2018), with a decrease in theoretical courses (from 39% to 31%).

In terms of computing languages used, shown in Table 3, we observe that Fortran, C and Basic have decreased in use (by 5-6%), while MATLAB had the most increase (+6%), followed by Python (+4%). Increase of MATLAB should be attributed in its wide use in engineering programs, replacing gradually Fortran.

Finally, if we focus on just the courses of the *Computing* programs, (Table 5), in the 2011-12 courses, C was used in 82%, followed by C++ and Pascal that were used 6% each. So, comparing with the 2017-18 academic year, use of C fell by 10%, Python appeared in the list (+9%) Java increased by 3% and C/C++ increased by 6%, while Pascal disappeared (-6%).

academic year	2011-2012		2017-2018		dif
Programming Language	courses	%	courses	%	%
C	28	82%	24	73%	-10%
PYTHON		0%	3	9%	9%
JAVA	1	3%	2	6%	3%
C,C++		0%	2	6%	6%
C++	2	6%	1	3%	-3%
PROLOG		0%	1	3%	3%
PASCAL	2	6%		0%	-6%
BASIC	1	3%		0%	-3%
total	**34**	100%	**33**	100%	

Table 5: Trends in programming languages used in courses of CS Programs between academic years 2011-12 and 2017-18

6 RQ4: COMPARISON WITH OTHER COUNTRIES

An interesting question is to compare the landscape of the Greek higher education, as far as teaching introductory courses to computing to other systems. As already mentioned, there have been a number of studies related to teaching introductory to computing courses in different countries. Therefore, we will compare the findings of Table 3(b), that contained the programming languages used in 121 courses of Introduction to Computing (major and no-major) during 2017-2018, to other similar studies.

Mason & Cooper	Murphy et al.	Siegfried etal.	Guo	Arcos et al.	Avouris
2014	2017	2016	2014	2017	2018
Australia- NZ	UK	US	US	Ecuador	Greece
38 CS programs	80 CS teachers	387 CS programs	top 39 colleges	53 teachers	121 courses
Java (27%)	Java (46%)	Java (46%)	Python (69%)	Java (21%)	C (37%)
Python (27%)	Python (13%)	Python (20%)	Java(56%)	JavaScript(13%)	MATLAB (17%)
C# (9%)	C++ (12%)	C++ (19%)	MATLAB (20%)	C++ (12%)	Basic (11%)
C (7%)	C (5%)	C (6%)	C (18%)	PHP(10%)	C++ (6%)
JavaScript(7%)	JavaScript(3%)	Scheme(2%)	C++ (15%)	C# (9%)	Fortran (6%)

Table 6: Comparison of studies across higher education systems

Mason et al., 2018 [10] report on a survey of 38 introductory programming courses in Australian and New Zealand universities, conducted in the first half of 2013. The survey includes questions on programming language used, programming paradigm, pedagogical method used, reasons for choosing a specific language, etc. The most popular languages used according to the 2013 survey are: Java (27%), Python (27%), C# (9%) and C (7%), JavaScript (7%), Basic (7%), C++ (4%), etc.

A similar study was conducted in UK universities, reported by Murphy et al., 2017 [11]. This is the first survey of introductory programming courses (N = 80) taught at UK universities and colleges of higher education as part of their first-year computer science (or related) degree programs, conducted in the first half of 2016.

The survey investigated programming languages and tools used, as well as the underpinning rationale for these choices. The survey indicates a dominance of Java, despite the fact that Python is perceived, by the same respondents, to be both easier to teach as well as to learn. Java was the language mostly used (46% of language instances), with Python a runner up (13%), followed by C++ with 12% and C with 5% of the instances. The most popular reason given for choosing Java is that is "object-oriented language", while Python scores highest on pedagogical benefits.

Siegfried et al., 2016 [12] used the Reid list, created in 1990, showing the first programming language used in introductory programming courses taken by computer science and information systems majors in various US universities. The two most recent lists, that of 2015 and the pervious one (compiled in 2011) are compared in this report. In the more recent list, Java is the most popular language, followed by Python, with C++ and C in the third and fourth place. However, some significant changes in popularity occurred: Java's popularity declined from 197 out of 387 universities to 180 (46%), while Python's popularity grew significantly from 41 to 76 (20%). While C++ that moved from second to third place, is used in beginning programming courses in 6 fewer schools (74 colleges, 19%), while there are 3 more schools using C than four years before (22 colleges, 6%).

A survey concerning the introductory to computer science courses of the 39 top US universities by Guo, 2014 [7] revealed that Python was used in 27 universities (69%), followed by Java that was used in 22 colleges (56%), MATLAB in 8 colleges (20%), C in 7 colleges (18%) and C++ in 6 (15%).

A survey for a country of quite different economic and cultural conditions is the study of Arcos et al., 2017 [2] concerning teaching programming in universities of Ecuador. 53 teachers of 13 higher education institutions replied to questions concerning computing courses. On the question of programming languages used for teaching programming, it was found out that Java and JavaScript are the languages that are taught the most in institutions of higher education, 33.7% of the answers obtained include these languages. 29% of the answers correspond to C, C++ and C#, while PHP was preferred for Web programming, while languages like .Net, C# and Visual Basic were used by 13.1% of respondents. So, according to the survey, the 5 languages that are most taught in computer science programs, are: Java (21%), JavaScript (13%), C ++ (12%), PHP (17%), C# (9%), C (7%) and Python (4%). It should be noted that this survey concerns use of languages for teaching in all levels of courses, and not just introductory courses, so if we consider JavaScript and PHP as specific to web programming languages, the general-purpose languages are Java, C++, C#, C and Python in that order.

In summary, the findings of the reported surveys are shown in Table 6, together with the findings of the current study. In most of the reported studies Java is overall the most popular introductory language, while Python is rapidly gaining in popularity. In the trends section of our study, we observed a similar change, as both Java and Python gained in popularity between 2012 and 2018 in Greece, at a much slower pace, though. Comparing the findings of the current study to those in Table 6, the Greek higher education bears distinct characteristics. The most remarkable is that C is still widely used as introductory to computing language in the Greek higher education (37% of introductory courses, which grows to 73% for the Computing programs), Java and Python have very limited use, while Basic and Fortran are still in use (in No Computing programs), something that is not any longer the case in any other higher education system, from those included in the comparative studies.

7 CONCLUSIONS

In the reported research, following a descriptive statistics approach, we identified trends and teaching approaches relating to introductory to computing courses in Greek higher education. We identified that 35% of the introductory courses use a *programming first* approach, where C is the most popular language, 50% opted for a more *theoretical* approach, while 15% of the courses had as the main objective to develop specific *skills*, like use of personal productivity or data analysis tools.

The trend between 2012 and 2018 was of a small increase in programming courses and decrease of theoretical ones, while some slow changes have been observed in the programming languages used, with increase of MATLAB, Python and Java, and decrease of C, Basic, Fortran and Pascal, a trend observed widely in computing.

An interesting finding, when compared the Greek higher education system to those of other countries, including a central American case, is that in Greece, educators insist in wide adoption of C, while we observe a relatively slow increase in adoption of Java and Python that are much more popular in many other higher education institutions around the world. Further qualitative research needs to be performed in order to investigate the reason for this finding, as C, is a programming technology considered suitable mostly for systems programming, that has been judged not suitable for an introductory course (see for instance studies by Wainer & Xavier, 2018 [13], Koulouri et al. 2014 [9]).

On the other hand, the purely theoretical approach in introduction to computing, that was adopted in around 50% of courses overall, was used in 30% of the courses in *Computing* programs of study. It should be mentioned that the main textbook for this group of courses was Forouzan & Mosharraf, 2014 [4], that adopts a breadth-first approach to introduction to computing, with no programming involved. For *Computing* programs, this may be reasonable, as often there are other courses focusing in programming. However, in the *No Computing* programs, where often the course may be the only introductory in computer science in the curriculum, it should be the subject of future research, to study the effect of this lack of programming knowledge and skills to the students and the disciplines in general, as it may affect development of computational thinking and in general the development of the students' computing-related knowledge.

On the discussion on object-first vs no-objects or object-later approaches, the Greek case seems to differ considerably to most other cases, that have adopted already widely an object-first approach. Objects appear in just 14% of the introductory courses in Greek higher education. Introduction of objects at some point during the CS1 course can be facilitated by either Java objects-later approaches or by languages like Python that support multiple programming paradigms, as discussed by Goldwasser et al. [6].

Next, some limitations and future perspectives of the reported here research should be discussed. One limitation of the reported study is that it addresses only quantitative aspects of the problem, as more qualitative ones, like teaching approaches, pedagogy, resources used in the courses, are missing, as they were not available in the data sources used in this study. So, a future direction of this research could be that through instruments like questionnaires, focus groups, etc. we could combine the findings to more qualitative data. On the other hand, one clear strength of this approach, is its replicability, as it is based on an open data repository, demonstrating the potential of these data sources in research. A related strength is the fact that the whole Greek higher education system is included in the study (44 Universities, 40,000 courses), providing a comprehensive view of the situation. A possible future direction could be to perform follow up studies and establish a longitudinal study, based on this data set as in the case of the Reid list of first programming languages in the US [12].

Finally, we should observe that on the main issue of the computer language used for introduction to computing, a remarkable finding of this study is that the Greek higher education system seems to be lagging behind considerably, to many other systems. Investigation of the reasons for this situation and on measures to be taken to address this issue, is beyond the scope of this study.

REFERENCES

[1] ACM Curricula recommendations available online: www.acm.org/education/curricula-recommendations

[2] Arcos, G., Aguirre, G. L., Hidalgo, B., Rosero, R. H., Gómez, O. S. (2018). Current Trends of Teaching Computer Programming in Undergraduate CS Programs: A Survey from Ecuadorian Universities. KnE Eng., 1(2), 253-275.

[3] Farooq, M. S., Khan, S. A., Ahmad, F., Islam, S., & Abid, A. (2014). An evaluation framework and comparative analysis of the widely used first programming languages. PloS one, 9(2), e88941.

[4] Forouzan, B. A., & Mosharraf, F. (2008). Foundations of computer science. Cengage Learning EMEA

[5] Giangrande Jr, E. (2007). CS1 programming language options. Journal of Computing Sciences in Colleges, 22(3), 153-160.

[6] Goldwasser, M. H., & Letscher, D. (2008, June). Teaching an object-oriented CS1-: with Python. In Acm sigcse bulletin (Vol. 40, No. 3, pp. 42-46). ACM.

[7] Guo, P. (2014). Python is now the most popular introductory teaching language at top us universities. BLOG@ CACM, July, 47.

[8] Koffman, E. B., Miller, P. L., & Wardle, C. E. (1984). Recommended curriculum for CS1, 1984. Communications of the ACM, 27(10), 998-1001.

[9] Koulouri T., Lauria S., and Macredie, R.D. (2014). Teaching introductory programming: A quantitative evaluation of different approaches. ACM Transactions on Computing Education 14, 4 (2014), 26.

[10] Mason, R., Crick, T., Davenport, J. H., & Murphy, E. (2018). Language Choice in Introductory Programming Courses at Australasian and UK Universities. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (pp. 852-857). ACM.

[11] Murphy, E., Crick, T., & Davenport, J. H. (2016). An analysis of introductory programming courses at UK universities. The Art, Science, and Engineering of Programming, 1(2), 2017.

[12] Siegfried, R. M., Siegfried, J., & Alexandro, G. (2016). A Longitudinal Analysis of the Reid List of First Programming Languages. Information Systems Education Journal, 14(6), 47.

[13] Wainer, J., & Xavier, E. C. (2018). A Controlled Experiment on Python vs C for an Introductory Programming Course: Students' Outcomes. ACM Transactions on Computing Education (TOCE), 18(3), 12.