UNIVERSITY
INSTITUTE
OF LISBON

Learning to code in class with MOOCs: process, factors and outcomes

João Pedro Grangeia Gomes

Master in Computer Science and Business Management

Supervisor:
Doctor Cláudia Maria Lima Werner, Full Professor,
COPPE/UFRJ

Co-Supervisor:
Doctor Fernando Brito e Abreu, Associate Professor,
Iscte

November, 2020

[This page has been intentionally left blank]

TECHNOLOGY
AND ARCHITECTURE

Learning to code in class with MOOCs: process, factors and outcomes

João Pedro Grangeia Gomes

Master in Computer Science and Business Management

Supervisor:
Doctor Cláudia Maria Lima Werner, Full Professor,
COPPE/UFRJ

Co-Supervisor:
Doctor Fernando Brito e Abreu, Associate Professor,
Iscte

November, 2020

[This page has been intentionally left blank]

Learning to code in class with MOOCs:
process, factors and outcomes

[This page has been intentionally left blank]

To my parents and my sister.

[This page has been intentionally left blank]

Acknowledgements

I would like to thank Professors Fernando Brito e Abreu and Claudia Werner for giving me this opportunity to work with them. As advisers, your mentoring and support were crucial to keep me focused and motivated to achieve this goal in my life.

I would also like to thank Professor João Caldeira for the work on this template that allowed me to write this dissertation.

To my friends, who I met five years ago at the beginning of this journey, thank you for all the time spent in *"sala de estudo"* and all the parties and moments that we been through, it would not have been the same without you.

To my closest friends, thank you for always being present in the most important moments of my life, for helping me to be a better person and for all the support during this journey.

A special thank you to my family for giving me the opportunity and conditions to get where I am at.

To my girlfriend, I would like to thank you for all the support, comprehension, love and for always believing in me.

Lisbon, November 2020

João Pedro Grangeia Gomes

[This page has been intentionally left blank]

Abstract

Problem: Python became the most popular programming language in recent years, beating Java, the programming language still widely used as the main programming language in many undergraduate degrees on computer science related areas. Students from those degrees often do not get Python in their syllabus, but the job market is demanding it increasingly.
Objective: To assess if learning a new programming language by following a MOOC is feasible in a fully dedicated mode and allows achieving a learning outcome comparable to the traditional in-class learning process.
Proposal: Students from undergraduate degrees lacking Python skills followed a dedicated and intensive learning process on that language based on an in-class MOOC. The latter is suitable for students with some background in programming, as is the case, allowing a faster learning pace. Participants' subjective perception of the corresponding workload was monitored.
Validation: A programming contest, using an automatic judge, was used as a validation for this proposal. Two groups of students participated: those from three degrees lacking Python, which followed the proposed MOOC (experimental group), and those from the degree that includes Python programming, which had a traditional in-class learning process (control group).
Conclusions: The experiment results were analysed and it was inferred that the proposed in-class MOOC learning approach is as effective as the traditional learning approach. Furthermore, it was identified that the students' average grades obtained in the previous programming courses taken as part of their degree's syllabus and the number of MOOC modules finished in the context of this experiment directly influence the number of points obtained in the contest.

Keywords: learn programming, MOOC, Python, programming contest, hackathon

[This page has been intentionally left blank]

Resumo

Problema: Nos últimos anos, Python tornou-se a linguagem de programação mais popular, ultrapassando o Java, que continua a ser muito usada como principal linguagem de programação em muitas licenciaturas relacionadas com informática. Estas licenciaturas acabam muitas vezes por não oferecer esta competência aos estudantes, no entanto o mercado de trabalho procura-a cada vez mais.
Objectivo: Avaliar a possibilidade de aprender uma nova linguagem de programação através de um MOOC num regime de total dedicação. E por fim, perceber se este permite obter resultados comparáveis ao ensino tradicional.
Proposta: Os estudantes com falta de conhecimentos de Python realizaram um processo de aprendizagem intensivo desta linguagem através de um MOOC em sala de aula. Este último é adequado a estudantes com alguns conhecimentos de programação, permitindo assim um ritmo mais rápido de aprendizagem. A perceção subjetiva dos participantes sobre a respetiva carga de trabalho foi monitorizada.
Validação: Realização de um concurso de programação recorrendo a um juiz automático. Dois grupos de estudantes participaram neste concurso: estudantes das 3 licenciaturas sem conhecimentos de Python, que realizaram o MOOC (grupo experimental), e os estudantes da licenciatura que inclui Python e que teve uma aprendizagem tradicional (grupo de controlo).
Conclusões: Os resultados deste experimento foram analisados e inferiu-se que a aprendizagem de um MOOC em sala de aula é tão eficaz quanto o ensino tradicional. Para além disso, foi também verificado que a média de notas dos estudantes obtida nas unidades curriculares de programação que já frequentaram no seu curso e o número de módulos feitos no MOOC no contexto desta experiência influenciam diretamente os pontos obtidos no concurso de programação.

Palavras-chave: aprender programação, MOOC, Python, concurso de programação, hackathon

[This page has been intentionally left blank]

Contents

[This page has been intentionally left blank]

List of Figures

List of Tables

[This page has been intentionally left blank]

Acronyms

APA	American Psychological Association.
BPMN	Business Process Model and Notation.
DCTI	Department of Information Science and Technology.
ECTS	European Credit Transfer and Accumulation System.
ICPC	International Collegiate Programming Contest.
ICT	Information and Communications Technologies.
IT	Information Technology.
MOOC	Massive Open Online Course.
NASA	National Aeronautics and Space Administration.
RR	Rapid Review.
SPOC	Small Private Online Course.
SRL	Self-Regulated Learning.
TLX	Task Load Index.

[This page has been intentionally left blank]

Chapter 1

Introduction

Contents

This chapter introduces the motivation of this work, describes the problem faced by the majority of the Information Technology (IT) undergraduate students at Iscte, reviews some of the existing learning approaches and proposes a learning approach to overcome this. Finally, it describes the research objective and questions, and summarizes the main contributions of this dissertation.

[This page has been intentionally left blank]

Chapter 1
Introduction

1.1 Motivation

1.1.1 The rise of Python and its education gap

According to the PYPL index (PopularitY of Programming Language), Python has grown for the last five years and now leads the popularity race. This is confirmed by a comparative search in Google Trends (Figure 1.1). The big data field is becoming increasingly important for business, helping in identifying trends, checking out the competition, improving operations, recruiting and managing talent. Since Python is specially suited for data analysis and visualization fields [31], many companies are offering job opportunities for Python programmers (see Table 1.1).

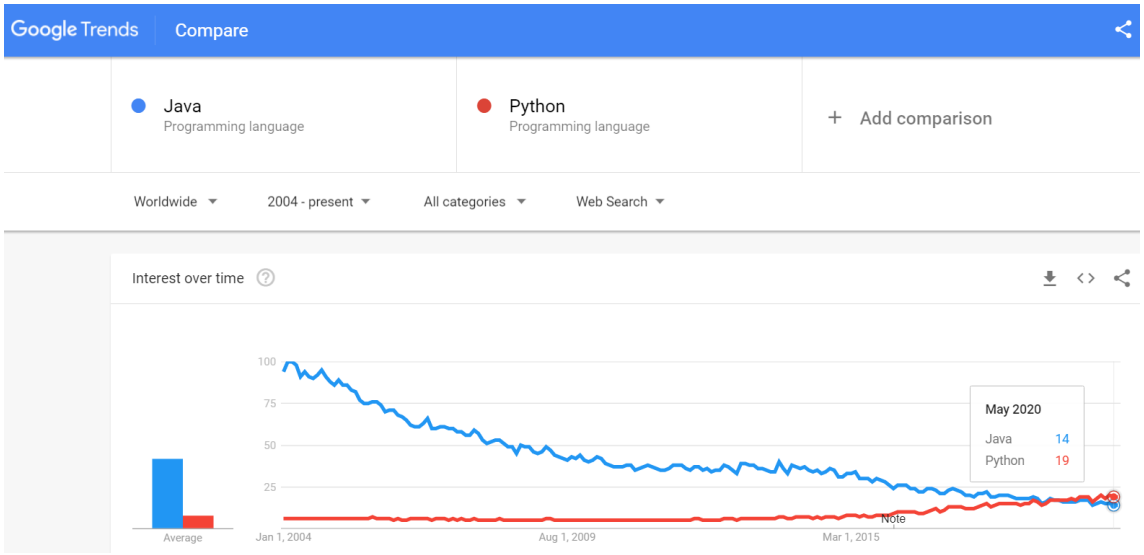

Figure 1.1: Java vs. Python - Google trends

Table 1.1: Number of jobs offered regarding Python (20/07/09)

Website	Number of jobs
Dice.com	7995
Monster.com	24577

Students pursuing an IT degree, when looking for the job market, are then motivated to learn Python. However, it is often the case that that language is not included in the degree's regular syllabus. This "Python gap" currently happens in 3 out of our 4 undergraduate degrees[1] at the Department of Information Science and Technology (DCTI) in Iscte, where Java is the main programming language taught, and Python is not included in their syllabus.

Changes in undergraduate degree syllabus often require a long bureaucratic approval cycle where a national accreditation board is involved. This renewal process is often not agile enough

[1] Those 3 are: Computer Science and Business Management degree, Computer Engineering degree and Telecommunications and Computer Engineering degree. The exception is the recent (started in 2019) Data Science degree.

to follow the market demands which are continuously changing. For those who are in charge of adapting undergraduate degree syllabus to the latter, one apparent simple solution would be offering optional courses with the traditional teaching approach. For students, this traditional approach means reading, watching the learning material in class, and complete tasks at home. For the faculty staff, this option could be easier due to the experience in teaching, however they are often unavailable for preparing and teaching extra optional courses, due to their filled schedules. Furthermore, it is unfeasible to overload the students' weekly schedule with optional courses and exceed the usual 30 European Credit Transfer and Accumulation System (ECTS) credits that they already get on each term. Therefore, other learning alternatives should be looked for.

1.1.2 MOOCs for the rescue

Some years ago, e-learning platforms popped-in, allowing to make available online, for asynchronous usage, a diversity of learning and self-assessment materials, such as slides, videos or written tutorials, podcasts, quizzes and assignments [1].

Given the increasing availability of data storage and network bandwidth, those platforms were improved to support a new educational vehicle, the so-called **Massive Open Online Courses (MOOCs)**, which are led by subject experts from higher education or industry. MOOCs are "massive" courses contrarily to the traditional courses which are limited by the physical dimensions of in-class rooms, since they are offered "online" on any internet connected device with a browser. They are "open" to everyone without entry qualifications.

MOOC participation is for free or could have a fee, usually when participants desire to obtain a certificate. Those fees became a new business model for some universities and other organizations, based on the large potential audience of these MOOCs who wants to obtain valid certificates with academic or professional training. As a result, there are now multiple MOOC platforms offering hundreds of courses, some of which have had hundreds of thousands of students enrolled [4].

Despite the popularity of MOOCs, several studies across the last decade have reported completion rates[2] below 10% [11, 12, 17, 18, 21]. Several reasons have been pointed out for this failure in MOOCs completion. Among the more recurrent ones are:

- students lack the skill to independently lead their own learning process, especially regarding time management [41];
- students face sentiments of isolation and disconnection, similar to those experienced in distance learning environments [15].

Even though MOOCs are usually conceived for Self-Regulated Learning (SRL), some solutions to mitigate those problems have been proposed in the literature for MOOC platforms' creators. One of the solutions is merging peer-to-peer learning features in MOOCs, to allow interaction among participants. That discussion may improve participants' motivation, thereby reducing the isolation issue [14]. To mitigate the time management issue, some authors have proposed to add some pedagogical strategies found to be beneficial in the traditional classroom

[2]Instead, "attrition rates" are sometimes reported. They stand for the rate of students that stop participating in a course (aka dropouts)

setting, such as *retrieval practice* and *study planning* in MOOCs [18]. The existence of these mitigating features for reducing dropout should be taken into account when choosing a MOOC. A review of recent studies about academic engagement in MOOCs can be found in [24].

1.1.3 Other learning approaches

Other learning approaches have gained strength in latest years and are being applied in computer programming. They have been facilitated by the widespread availability of the same type of online learning and self-assessment resources found in MOOCs.

Blended Learning is a combination of online learning (e.g. MOOCs) with the traditional face-to-face learning. It does not replace traditional learning, instead the two methods of learning are used complementary together. Blended learning can be particularly useful for part-time students, due to the availability of self-paced learning features and the option to frequently assess knowledge [7].

Flipped Classroom is a new learning model, rated as one of the top trends in educational technology [39]. This learning model refers to readjusting the time in and out of class, transferring the learning driver from teachers to students. In brief, teachers no longer deliver new subjects in class, and students must pursue self-study by using online materials at home, on their own pace, before class. Those materials were previously prepared or selected by the teacher in charge of the course. The time in class is spent on working through the concepts being delivered, with the guidance of a teacher that helps students with problems encountered in self-study [35]. An e-learning platform complements this learning model by allowing discussion between participants.

Peer-to-peer Learning is a teacher-less approach where students are expected to search for solutions by themselves, using online materials. Learners proceed through a sequence of activities that include group discussion about learner responses and may shift from a role of tutee to tutor. A peer tutor is anyone who has a similar status as the person being tutored that is not expected to teach, but mainly acts as a facilitator of the learning of their peers in solving tasks [48]. It has been observed that the discussion of challenging problems leads to better outcomes than working individually. Additionally, incentivizing people to help each other generate still better results. [14].

1.2 The proposed learning approach

In this dissertation we propose a learning approach that we named **In-class MOOC**[3] which consists in using a MOOC in the context of a scheduled class. It is not a *blended learning* approach because it is not blended with *traditional learning* contents, neither is followed by it, such as in the *Flipped Classroom* approach.

There were several reasons to choose this learning approach: (i) unavailability of faculty staff, (ii) avoiding excessive workload for students during each term, (iii) the will to reuse the most suitable MOOC on Python among the many existing ones, while (iv) trying to reduce the issue of high attrition rates.

[3] In opposition to an *Off-class MOOC*, the usual delivery mode.

Although most undergraduate curricula propose learning multiple topics (i.e. different courses) in parallel like in all our aforementioned degrees, learning a new programming language on such short notice requires exclusive dedication. The first reason discarded the traditional learning, blended learning, and flipped classroom approaches since they all require faculty staff. To account for the second reason, we scheduled this learning approach to be offered during the interval that occurs between two consecutive academic semesters. University students are often invited to participate in extracurricular activities during these academic breaks and their adherence is usually greater for short-term activities. That is why we decided that the application of this learning approach should not exceed more than a week. Furthermore, during each term, students have multiple different courses and learning a new programming language on such short time requires exclusive dedication. This option is aligned with findings that suggest that understanding one topic at a time may improve learning, due to the concentrated focus on just one topic [10]. In order to fulfill the third reason (choice of a suitable MOOC for Python) a research was undertaken and described in Chapter 3. Last, but not least, the search for engaging students in the experiment for our learning approach, while trying to mitigate the issue of low MOOCs' retention, led us to the following steps:

- make students aware of the importance of mastering Python, for their future competitiveness in the job market, with evidences from the field (job offers);
- valuing certificates given by a trusted university for the MOOC completed;
- participants were together in the same room, each in a separate table, doing the MOOC independently in their laptops, using headphones, not to disturb colleagues; interactions were allowed during the coffee-breaks, to fight the sentiment of disconnection [14];
- a defined schedule with 4 half-days (5 hours each), with a coffee-break in-between, offered to all participants;
- students' presence and evolution in MOOC's completion were monitored.

Summing up, as seen in Table 1.2, there are several pedagogical approaches that mainly depend on student's autonomy and availability of faculty staff. Discussing which is the best approach for learning a new programming language is not consensual, due to the complex nature of programming and individual differences, including study approaches, thinking styles, and the focus of supervision [51]. The recognition of those differences among students, with more or less proficiency in programming with another programming language, will be taken into consideration during the MOOC selection process covered in Chapter 3.

Table 1.2: Comparison of learning approaches

Characterization / **Approach**	Scheduled classes or sessions	Participants interaction	Faculty staff	Learning drivers
Traditional learning	Yes	Yes	Yes	Faculty or expert in-class
Blended learning	Yes	Yes	Yes	Same as above, complemented with online materials
Flipped Classroom	Yes	Yes	Yes	Students alone first, then in-class clarification of doubts
Peer2Peer learning	Yes	Yes	No	Students collectively
Off-class MOOC	No	Usually not	No	Students alone
In-class MOOC	Yes	Limited	No	Students individually, but not alone

1.3 Research objective and questions

In this dissertation, we aim to mitigate the "Python gap" problem, described in the previous section, faced by the majority of the IT undergraduate students at the DCTI. Our **main research objective** is to assess if the *In-class MOOC* approach is adequate for learning a new programming language (Python in this case).

Based on the aforementioned research objective, we want to analyse the effectiveness of the proposed learning approach and which factors have an impact on this effectiveness. Furthermore, we also want to identify which factors influenced the learning resilience of students during the proposed learning approach. Last, but not least, we want to know if the *In-class MOOC* approach allows achieving a learning outcome comparable to the traditional in-class learning approach. To clarify the previous concerns, we formulate the following **research questions** in the context of using the *In-class MOOC* approach for learning Python as a second programming language:

- **RQ1** - How to select an adequate MOOC?
- **RQ2** - Which factors influence learning resilience?
- **RQ3** - Which factors influence learning effectiveness?
- **RQ4** - Is this approach as effective as traditional in-class learning?

1.4 Main contributions

The main contributions of this dissertation are:

- the proposal of the *In-class MOOC* learning approach and the confirmation of its adequacy for learning Python as a second programming language;
- the use of Business Process Model and Notation (BPMN) for producing a graphical representation of the learning process in three different MOOC platforms (Coursera, Udemy and Codecademy); this BPMN modelling was used to present the features of each platform to an expert panel, in order to make them decide which was the most suitable MOOC for our purpose;
- the identification and discussion of the factors that do, or do not, influence learning resilience and effectiveness in this context, by means of a controlled experiment (including a programming contest) with a control group (placebo) and an experimental group;
- the promotion of the use of an automatic judge (Mooshak v2[4]) within DCTI's students.

1.5 Organization

The remainder of this dissertation is organized as follows. Chapter 2 proposes and applies a literature review protocol and a mapping taxonomy to the related work, to get a picture of the state of the art. Chapter 3 proposes a step-wise method for selecting a MOOC course on Python, that includes a set of criteria, and applies it to reach three final candidates, that are then discussed by an expert panel for reaching a consensus on the final choice. Chapter 4 describes

[4] Mooshak is a system for managing programming contests on the web, that has been used as well for automatic correction of programming assignments, either individually or in groups

a controlled experiment, named Pythacon, that allowed to validate the *In-class MOOC* learning approach, using the previously selected MOOC. Last, but not least, conclusions are drawn and provisions for future work are outlined in chapter 5.

Chapter 2

State of the Art

Contents

This chapter aims at outlining the state of the art, by means of a rapid review based on a review protocol and a proposed taxonomy for classifying the related work found. Finally, discusses some gaps that have been found in the literature and the opportunities that can be explored in this dissertation.

[This page has been intentionally left blank]

Chapter 2
State of the Art

2.1 Introduction

The purpose of this chapter is to outline the state of the art in the area of alternative approaches to learn programming using a Rapid Review (RR), also known as rapid systematic review. According to [32], a RR is *"a type of knowledge synthesis in which components of the systematic review process are simplified or omitted to produce information in a short period of time"*. In [9], a comparison between the characteristics of RRs and systematic reviews is conducted. From this study, we can deduct that the RR must have a protocol, may use few or just one database and can be conducted by a single person. Additionally, the selection is based on inclusion/exclusion criteria and descriptive summaries are used as a synthesis procedure. Therefore, the RR protocol is described in the following sections.

2.2 Rapid review protocol

This RR is organized into five sections. This section focuses on the protocol used, i.e., the research objectives, inclusion and exclusion criteria, the search strategy, the results of the search string execution and the validity threats of the undertaken RR. In Section 2.3, it is proposed a taxonomy with some criteria for the related work. In Section 2.4, the chosen articles for this review are synthesized and classified based on the proposed taxonomy. Section 2.5 covers the summary with the final considerations.

2.2.1 Research objectives

The research objectives for this review are:

1. Undertake a rapid systematic review of empirical research on the learning of a programming language using several approaches.
2. Select the most relevant studies to review according to the choice criteria.
3. Analyse and synthesize the selected studies.
4. Find gaps in order to suggest new fields for further investigation.

2.2.2 Choice criteria

For this literature review it was given preference to empirical studies whose participants were from areas related to computer science. In general, all the studies included in this review had to discuss at least one of the learning approaches presented in Chapter 1, exclusively in the programming field.

2.2.3 Inclusion and exclusion criteria

Inclusion Criteria:

- Written in English.
- Articles based on empirical methods, e.g., questionnaires, formal experiments, case studies.
- Papers in journals or conference proceedings and book chapters are considered.

Exclusion criteria:

- Studies that were published before 2016.
- Studies whose title or abstract are not relevant or related to the theme.

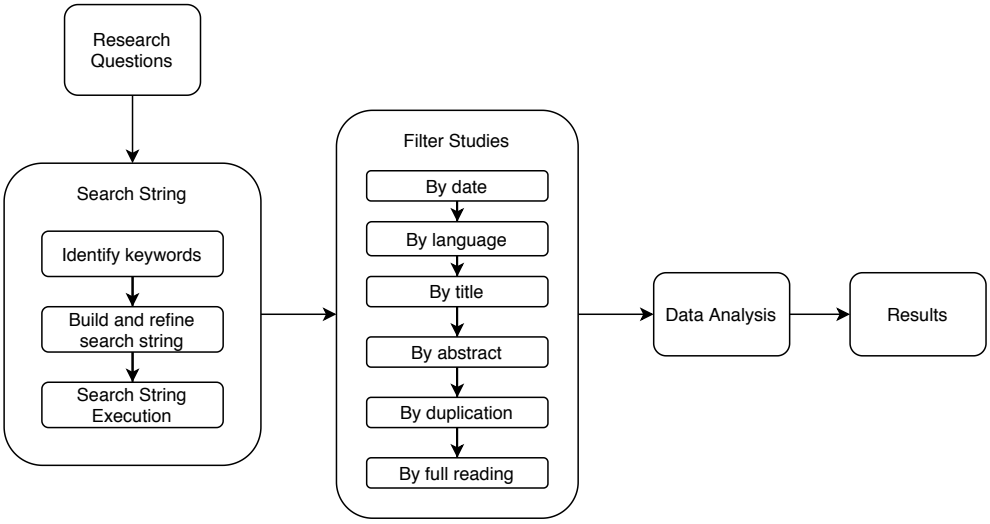

Figure 2.1: Summary of the rapid review protocol

2.2.4 Search strategy

The first step of this search was to choose the following two databases in accordance with its importance to the theme of this dissertation (IEEE Xplore and Scopus). As some searches were done on these databases, the search string was refined using the logical operator "AND" to join both sides of the string, where the left side is the scope and the right side is the learning approach. The logical operator "OR" was used to join all the keywords from the same domain. As a final result, the search string was as follows: ("learn programming" OR "computer programming") AND ("programming challenge" OR "flipped classroom" OR "blended learning" OR MOOC OR hackathon). After the execution of the search string on the selected databases, it was applied the criteria in the following order: date, language title, abstract, duplicates detection and full reading.

2.2.5 Results

The results of the search string execution in the chosen databases with the respective criteria applied are presented in Table 2.1.

Table 2.1: Search string execution

	Scopus	**IEEE Xplore**
("learn programming"OR "computer programming") AND ("programming challenge"OR "flipped classroom"OR "blended learning"OR MOOC OR hackathon)	267	38
Removed by date	74	13
Removed by language	2	0
Removed by title	97	14
Removed by abstract	74	5
Removed by duplication	0	1
Removed by full reading	12	2
Final	8	3

2.2.6 Validity threats

As reported in [8], the RR conducted in this chapter may present some validity threats. The main threats to the validity of this RR are:

- the reduced number of databases, which can lead to not discovering studies which could be relevant to this dissertation.
- the inadequate size and number of samples used in the related work (Section 2.4), which could lead to a low reliability of the results.
- the entire selection procedure being conducted by a single researcher, which may introduce selection bias.

2.3 Taxonomy

A taxonomy was developed to serve as a better classification of the articles presented in the related work. This taxonomy will also contribute to an understanding of what are the commonalities and differences among the studies. The criteria chosen for this taxonomy was developed as the review of the articles was being made. Therefore, the taxonomy used in this review contains the criteria presented in Table 2.2.

Table 2.2: Taxonomy for related work classification

Criterion	**Abbreviation**
Domain	D
Learning Approach	LA
Learning Motivation	LM
Learning Results	LR

2.3.1 Domain (D)

This criterion allows to identify and classify the participants described in the studies from the related work. In order to do that, a nominal scale is used.

2.3.2 Learning approach (LA)

By using this criterion it is possible to identify which was the alternative learning approach used on the related work studies. To classify this criterion it is adopted a nominal scale with the following options:

- Flipped Classroom - The learning approach used was the flipped classroom method.
- Hackathon - The learning approach used was a hackathon.
- MOOC - The learning approach used was a MOOC.
- Blended Learning - The learning approach used was the blended learning method.
- Hybrid Learning - Two from the above learning approaches are combined.

2.3.3 Learning motivation (LM)

Using a Likert-scale, this criterion allows to understand the students motivation in the adopted alternative learning approach.

0. Not applicable (not covered in the study).
1. The students motivation worsen significantly.
2. The students motivation worsen slightly.
3. The students maintained their learning motivation.
4. The students motivation improved slightly.
5. The students motivation improved significantly.

2.3.4 Learning results (LR)

With this criterion, it is possible to understand if the students improved their learning results when other learning approach was taken instead of traditional learning. This criterion is only applied when there is a comparison between the adopted learning approach and the traditional one. This criterion is defined using a Likert-scale:

0. Not applicable (not covered in the study).
1. The students' results were much better in traditional learning.
2. The students' results were better in traditional learning.
3. The students' results did not have significant differences.
4. The students' results were better in the learning approach defined in the study.
5. The students' results were much better in the learning approach defined in the study.

2.4 Related work

2.4.1 MOOC´s Integration Approach: Assessment and Comparative Studies of all Moroccan Universities [43]

Objective: Integrate distance learning platforms MOOC into the Moroccan Universities.

Description: In this paper a problem faced in some Moroccan Universities is described and the implementation of MOOCs is the solution proposed to combat this problem. According to the authors, Moroccan Universities are facing an obstacle to the education, which is a huge number of students entering the university cycle each year, while the number of professors considering retirement increased and the recruitment of professors decreased. The aforementioned leads to a low professor-student ratio threatening the quality of the education. Another reason mentioned is the appearance of epidemics, as an example the coronavirus that limited the traditional face-to-face learning. As a solution to those problems, the authors mentioned the MOOC platforms as one of the most effective solution. Furthermore, a study on the types of distance learning in Moroccan Universities is made and a study on MOOC's implementation in the concerned universities is undertaken. To the extent of understanding which would be the best manner of MOOC integration in Moroccan Universities, the authors described an experiment composed of 130 learners divided into 5 groups. Each group followed the MOOC in different modes: MOOC alone, MOOC with teacher support, MOOC with other external resources, MOOC with weekly report and collaborative MOOC. The results of this experiment showed that: the learners of the five groups were very motivated and appreciated this MOOC integration, they were ready to recommend it to other learners, and the interaction among learners (collaborative mode) reinforced their motivation to follow the MOOC. As a conclusion of this work and based on the experiment, the authors mentioned some recommendations that must be taken into account to integrate MOOCs in higher education.

Relation to our work: Despite the problem found in this study being different from the problem described in this dissertation, the solution proposed is similar, which is the integration of a MOOC to solve the problem. In this study, students were divided in groups and each group followed a MOOC in a different mode. The MOOC mode that is more relatable to this dissertation is the collaborative mode where students had the possibility to interact (although remotely) with each other following the MOOC. Furthermore, this MOOC mode showed that this interaction reinforced their motivation. These results support one of the reasons of our proposed learning approach that intends to fight the disconnection and dropouts by students following a MOOC.

Table 2.3: Taxonomy instantiation for [43]

D	LA	LM	LR
Undergraduate students from a Moroccan University	MOOC	5	0

2.4.2 Flipped Classroom Approaches in Computer Programming Courses in Japan [29]

Objective: Explore the effectiveness of flipped classroom approach in an undergraduate computer programming course.

Description: In this study the authors referred an issue found in the traditional learning approach which is a wide variety of students with different skills and proficiency levels. This problem is faced especially in traditional uniform and structured courses, such as computer programming courses where a culture of lecture-type classes exists. In order to solve this problem, the authors proposed an implementation of the flipped classroom method, described as very effective method across various courses but under-explored with respect to computer programming. The implementation of this learning approach counts with 53 full-time undergraduate engineering students majors at a Japanese university. None of those participants had previous experience in this method. The students were tasked to watch a 10 minutes video lesson with detailed explanations of Java basics before each of the first five classes. In class, students were expected to apply the knowledge acquired working on practical tasks. At the end of the course, the students were asked to fill a questionnaire about the flipped classroom method. As results of this implementation, the authors took into account the answers of the questionnaire that showed that 75% of the participants highly valued their flipped learning experiences. Furthermore, the authors mentioned that students also highly rated the effectiveness of video lessons, which enhanced their motivation and comprehension. Regarding the benefits and disadvantages of flipped classroom to teachers, the authors mentioned *"Although preparing videos was time consuming for the instructors, the preparation process helped them to reconsider how to teach computer programming, (...) including designing in-class tasks, developing their pedagogical skills, and examining student preparedness for programming"*.

Relation to our work: The learning approach implemented in this study was the flipped classroom which is somewhat similar to the learning approach proposed in this dissertation. The video lessons explaining the concepts instead of lecture-type classes are a common point in these two approaches. Therefore, the main conclusion that can be extracted from this study to consider in ours, is the motivation and comprehension increase showed by these students when the video lessons were applied as a method to explain the concepts of programming.

Table 2.4: Taxonomy instantiation for [29]

D	LA	LM	LR
Undergraduate engineering students from a Japanese university	Flipped Classroom	5	0

2.4.3 Using a Programming MOOC as an Admission Mechanism for CS [44]

Objective: Investigate differences in study outcomes between students who applied to study Computer Science (CS) using a programming MOOC and students who used the traditional

application method.

Description: In this study, it is mentioned some of the possibilities of using MOOCs in a university context such as teaching purposes, attract and retain students, give credit points, prepare students for university and as an admission mechanism. The latter is the one adopted in the University of Tartu in Estonia. Therefore, to reach the objective of this study, the authors analysed a sample with 366 students, 66 applied via MOOC and 300 were admitted via other admission procedure. The results of this work showed that there was no difference among MOOC and non-MOOC students in the study outcomes and in retention.

Relation to our work: The relation found in this study was the effectiveness of a programming MOOC to give the expertise needed for a computer science degree. In this work, it is not very perceptible which is the background in terms of programming of the students who were admitted via the traditional application approach. The results showed that students who were admitted via programming MOOC and non-MOOC application method did not have difference in their performance on CS courses during the first year, meaning that both admission mechanisms are on equal terms regarding the preparation for the CS course. However, the comparison made in this study does not attend the purpose of comparing traditional learning with alternative approaches regarding how to teach programming as we want for our work.

Table 2.5: Taxonomy instantiation for [44]

D	LA	LM	LR
Candidate students from the University of Tartu in Estonia	MOOC	0	0

2.4.4 Admitting Students through an Open Online Course in Programming: A Multi-year Analysis of Study Success [33]

Objective: Compare study success between students accepted through MOOC and students accepted through the traditional entrance exam and high school matriculation exam.

Description: In this paper, the authors mentioned that, since 2012 the University of Helsinki had piloted a new admission process where, in addition to traditional admission, prospective students could apply for a study right into the computer science degree using an introductory programming MOOC as an admission process. To the extent of giving context, the authors discuss university admission policies around the world. However, in this case it is only important to refer that the admissions to the university specified in the paper are usually based on a national enrollment exam, which is referred by the authors as a merit-based process. This new admission process aimed to solve a problem of retention and dropout in computer science degrees, discussed in several studies. Considering this, the authors study how students admitted through MOOC managed in their first year studies are compared to students admitted through traditional admission. The factors assessed in this comparison were the completed credits and weighted grade point average (GPA), and the proportion of students who completed their studies in time. The data analysed by the authors were collected between 2012 and 2015.

The sample was composed of 981 students, 225 of MOOC admission and 756 of traditional admission. The test results of this analysis showed that MOOC admission may yield students who were more committed to their studies and consequently more likely to start their studies. Furthermore, students from the MOOC admission performed better in their CS studies and were more likely to complete their degree.

Relation to our work: The relation found in this work that is relatable to our work is mainly the preparation given by the MOOC in terms of programming for a computer science degree. In this case, students who were admitted via MOOC performed better in their CS studies than the students who were admitted via national exam. Despite not knowing the entirety of the programming background of the students who were admitted via traditional, the students admitted via MOOC were apparently more prepared when enrolling in the CS course than the other students. Considering all the aforementioned, the comparison that has been made in this study was not considered because it does not attain our comparison purpose.

Table 2.6: Taxonomy instantiation for [33]

D	LA	LM	LR
Candidate students from the University of Helsinki in Finland	MOOC	0	0

2.4.5 Secondary Students' Views on Using Flipped Classroom to Learn Computer Programming: Lessons Learned in a Mixed Methods Study [13]

Objective: Explore how secondary students perceive the use of flipped classroom for learning computer programming.

Description: This study described the adoption of the flipped classroom approach in a Hong Kong secondary school where 40 students from two Information and Communications Technologies (ICT) classes have participated. During 4 weeks, this learning approach was adopted to teach computer programming topics such as conditional and, repetition control flow constructs and arrays. Before each class, participants were asked to self-study parts of the programming unit named "Intro to Programming" at *Code.org*. In class, the teacher gave a quick review of the online learning materials but the main objective of these classes was that students solved exercises on an individual basis. After those 4 weeks, the participants were categorized into high- and low-performing groups based on their performance in class exercises. Eight out of forty participants were interviewed in order to understand their opinion about flipped classroom effectiveness. The results of the interview and the questionnaire administered to all participants at the beginning and at the end of this experiment showed that generally students had positive attitudes towards learning computer programming through flipped classroom. Furthermore, the authors concluded that "*When compared with traditional classroom instruction, it is found that the use of flipped classroom is effective in supporting students to learn computer programming at their own pace, promoting class interactions between teacher and students, as well as sustaining student engagement in learning computer programming.*" On the other hand, the authors mentioned that this approach could have an adverse effect on programming education because learning at home

may be affected by online distractions.

Relation to our work: The flipped classroom has the component of learning the concepts of a programming language (in this case) through video lessons and this is the point in common found between this study and our work. As the authors concluded, the video lessons allow students to learn in their own pace. This conclusion is important to our study because we will discuss later in this dissertation whether or not the video lessons in a MOOC context are as effective as the traditional lecture-type classes.

Table 2.7: Taxonomy instantiation for [13]

D	LA	LM	LR
ICT students from a secondary school in Hong Kong	Flipped Classroom	4	4

2.4.6 Student Perception of the Contribution of Hackathon and Collaborative Learning Approach on Computer Programming Pass Rate [45]

Objective: Investigate students' perceptions about the contribution of collaborative learning and Hackathon to improve computer programming pass rate.

Description: In the present work, the authors described a problem faced by the students in computer programming. According to the author, it is due to the lack of problem solving skills amongst students. Consequently, they cannot obtain their qualifications without the programming subjects and, as a result, student's dropout increases. The proposed solution is a hackathon and a collaborative approach to improve computer programming skills as well as academic results. Therefore, this study was conducted at the Durban University of Technology (South Africa) in 2016 with 80 IT participating students. The students' opinion regarding the collaborative learning activity and hackathon was collected using a questionnaire. The results of the questionnaire analysis showed that students overwhelmingly agree that: collaborative learning would improve their computer programming pass rate and hackathons should be part of the IT curriculum because it would improve their abilities to work with a team on computer programming tasks and improve programming skills. As a conclusion, and based on the students' opinion, the authors mentioned that the hackathon approach could help students solidify their problem solving skills, programming and communication skills and their academic results, because they will apply the knowledge acquired during the semester in real-life scenarios.

Relation to our work: In this dissertation, it is also proposed a hackathon to consolidate and improve the programming skills acquired before. However, there is one main difference between this study and our work, which is the students are assessed on an individual basis and not as a team like in the hackathon proposed in this study. In our case, this will prevent the collaborative learning among students. Regardless, the "codefest" environment and the competition can still motivate students to improve their programming skills, as expected in this study.

Table 2.8: Taxonomy instantiation for [45]

D	LA	LM	LR
Undergraduate IT students from Durban University of Technology in South Africa	Hackathon	0	0

2.4.7 Research on the Reform of Flipped Classroom in Computer Science of University Based on SPOC [3]

Objective: Propose a new flipped classroom model based on Small Private Online Course (SPOC) for students in regular university.

Description: In this paper it is described the implementation of a new flipped classroom model based on SPOC in a basic course (C programming) of computer science and technology in university. The methodology followed to implement this new model was divided in three parts: guidance before class, traditional classroom and the improvement after class. The SPOC platform was used in the first part (guidance before class) where students had videos, PowerPoints and other materials for online learning. In the traditional classroom part, students spent most of the time solving problems and receiving feedback from the colleagues. In the after class part, students were encouraged to peer discussion, and deeply reflect the content of classroom. In order to measure students' satisfaction towards this new model, the authors undertook a questionnaire that 180 students answered. The results showed that learners were satisfied with SPOC platform. Furthermore, the students' answers in the questionnaire showed that, when comparing the traditional classroom with flipped classroom based on SPOC, they were satisfied with their learning efficiency and their ability to problem-solving improvement.

Relation to our work: A SPOC platform is like a MOOC platform. However, there is a main difference, which is that the MOOC is open to the masses and SPOC offers custom-made courses to a small and specific group of learners. Considering this and knowing that SPOC and MOOC follow the same principles regarding the learning process, we can extract from this study that this SPOC (applied to a flipped classroom model) had a good impact on students' motivation, abilities and learning efficiency.

Table 2.9: Taxonomy instantiation for [3]

D	LA	LM	LR
Students enrolled in a computer science and technology course at university	Hybrid Learning (Flipped Classroom based on MOOC[1])	4	4

2.4.8 Study Effort and Student Success: A MOOC Case Study [46]

Objective: Examine the relationship between student effort over time and student success in an introductory level MOOC for programming and computer science.

[1] In this case, it was considered the SPOC as a MOOC due to its similarities

Description: In this paper, the authors undertake a examination of the relationship between student effort over time and students success in a MOOC. MITx 6.00x was the introductory course to computer science and programming offered by MIT and hosted on edX MOOC platform that was used in this study. The content of this course included video lessons, homework questions, assignments, three exams and a forum. Additionally, the course offered a certificate, at no cost, and based on getting a final grade of at least 55%. The HarvardX-MITx Person-Course was the dataset used that contains aggregated information of each individual that participated in MOOCs from Harvard and MIT on the edX MOOC platform. Focusing only on the aforementioned MOOC, the authors analysed records from 32621 participants using logistic regression and correlation tests. The results showed that there is almost a linear positive relationship between student effort over time and student success in a MOOC. However, this almost linear positive relationship weakens eventually. With this in mind, the authors concluded that those who exerted effort over the longest amount of time actually had a lower probability of obtaining a certificate than others who exerted effort over somewhat less time. A given explanation for this curvilinear relationship is *that some students may just need more time to learn and develop competence in introductory programming and computer science than others, for instance, based on their prerequisite knowledge.* Another possible explanation is that there are two different types of achievement goals (people who wanted to prove to others their competences and people who were more concerned in self-improvement). Furthermore, the authors also found other variables that affected the certification ratio: increasing age somewhat negatively influences the student success, females had a higher probability of earning the certificate than males and students with less than secondary education had lower odds for obtaining the certificate than students with more education.

Relation to our work: In this study the success in a MOOC is explored, and useful information can be extracted to take into account in our work. For example, the influence of effort over time and previous education on the MOOC success are certainly factors to be assessed in our experiment too.

Table 2.10: Taxonomy instantiation for [46]

D	LA	LM	LR
Students enrolled in a MOOC on edX platform	MOOC	0	0

2.4.9 Introducing Basic Programming to Pre-University Students: A Successful Initiative in Singapore [40]

Objective: Undertake an intensive 3-week basic programming course that aimed to formally expose pre-university students in Singapore to programming.

Description: In this paper, the authors noticed that most students in Singapore were not formally exposed to programming before entering the university. In 2015, a course named "Lets Code" was conducted in a blended learning format and included video lessons, quizzes, video conferences, meet-up tutorial and take-homes programming assignments. The purpose was to

expose pre-university students to programming. Ruby was the programming language chosen for this course. The instructors of this course agreed that *"this blended learning would have more passive lectures to be done at home delivered in the form of video lessons and the limited classroom time during tutorial meet-up sessions would be reserved for active learning activities such as quizzes, in-class programming exercises and code criticism."* Mentors, who were undergraduate students, were responsible for giving feedback on the submitted assignment solutions while the participants were enrolled in video lessons. A total of 535 coders were enrolled in this course and 80.6% completed the course and were awarded with certificates. The attrition rate was 19.4%, which was considered low by the authors because there was no obligation to finish the course. The reasons given by coders who withdrew were the huge amount of time required, co-curricular activity commitments during the holiday period, loss of interest, inability to follow the lessons, etc. The authors conclude that the purpose of this study was attained considering that *"412 participants who had never written a line of code before, wrote their "Hello World!" program through this project. 78.9% of these first-time coders managed to complete the course, and more than one-third of them gained enough skills over the three weeks to be recognized as competent beginning programmers."*

Relation to our work: The course "Let's code" mentioned in this study has one important point in common to the proposed learning approach of the present dissertation. This point in common is the similarity of the course presented in this study and a MOOC because the concepts of a programming language are taught through video lessons and assessed via assignment and quizzes. Moreover, the reasons given in this study to explain the students' dropout can be taken as lessons to our experiment. As a conclusion, the authors mentioned that the main objective of the study was attained, which make us believe that this can be a effective learning approach to teach and improve programming skills.

Table 2.11: Taxonomy instantiation for [40]

D	LA	LM	LR
Pre-university students in Singapore	Blended Learning	4	0

2.4.10 Using Flipped Classroom Approach to Teach Computer Programming [2]

Objective: Discuss the suitability of the flipped classroom to teach computer programming and report the pilot experience of using this approach at Qatar University

Description: In this paper, the author begins to explain why flipped classroom might suit the programming subject. The first reason given is the difficulty of many students to learn computer programming that leads to high rate of dropout. These difficulties are categorized in three types: related to the nature of the subject, related to students and related to teaching methods. As an example of these types of difficulties the author described one of them which is the lack of problem-solving skills because teachers spend much of the course time focusing on syntactic details that leads to a students' focus on reading textbooks and understanding language syntax instead of practicing the development of new programs. After that, the author explains why flipped classroom is suitable to teach programming. It is mentioned by the author

that the main advantage of this model over the traditional one is that it maximizes the time spent in class to develop programming skills by teaching the programming syntax out of class. Additionally, in-class activities offer an opportunity to increase students interaction with the instructor, so they can get more feedback about their learning process, which improves their awareness of deficiencies. Considering all of the above mentioned, it is undertaken a pilot experiment to investigate the effectiveness of using the flipped classroom to teach computer program. This experiment was conducted during the fall 2015 at Qatar University. 41 students, without any programming background participated in this study which was applied in only one topic ("Arrays") of the programming course. The experiment was conducted through three phases:

- online activities, aiming to build students' knowledge of the syntax related to arrays in C++ using recorded videos and quizzes;
- in-class activities, aiming to improve knowledge acquired in online activities through solving exercises and group discussion;
- in-lab activities, aiming to develop programming skills.

The evaluation of this pilot experiment focused on students' feedback to assess their attitude in this experience and students´ performance to evaluate the effectiveness of the flipped classroom on their learning. The results showed that students had positive attitudes toward using this learning approach and their performance in the area of programming language syntax and structure improved. On the other hand, the author mentioned some of the challenges that should be considered in the future which are: how to encourage students to study the subject in advance and come prepared to in-class activities and apply this learning approach not only on the "Arrays" topic but in the whole course.

Relation to our work: The "online activities" mentioned in this study are similar to our proposed learning approach, meaning that students acquire their programming knowledge through video lessons at their own pace as in MOOCs. The results attained in this experiment are important to our work because demonstrate us that learning programming concepts through video lessons could improve their learning effectiveness.

Table 2.12: Taxonomy instantiation for [2]

D	LA	LM	LR
Students enrolled in a programming course at Qatar University	Flipped Classroom	4	0

2.4.11 Teaching Computer Programming using MOOCs in multiple campuses: Challenges and Solutions [42]

Objective: Describe the challenges faced in implementing a MOOC, the solutions attempted, and other tools required to make learning of a computer programming course, a better experience.

Description: In this paper it is described the experience gained over two consecutive years of running a computer programming course on a MOOCs platform at Birla Institute of Technology and Science in India. First, the authors explained that the reasons why they thought that MOOC was a better choice were to reduce the workload of faculty (700 students registered in the course), for a better student engagement in large courses and also to mitigate the problems in the traditional method. The implementation used the edX platform to deliver the videos which were made available to students one week before to give them time to go through them. To guarantee that students watched the videos online quizzes were used so they could keep pace with the delivered content. Moreover, there were labs where Mooshak[2] was one of the experimented platforms to evaluate the former. Two tutorial sessions per week were also part of the program where students had the opportunity to solve problems and to clear and discuss doubts. Furthermore, the evaluation components of this new method were mid semester test, online quizzes, two online tests, lab attendance, evaluated labs and comprehensive exam. After the implementation, some challenges faced during the MOOC are discussed in this paper[3]. The results of this implementation (2014-15 and 2015-16) compared to the traditional approach (2013-14) showed an overall increase in grades and in the learning of students. The findings from student feedback indicated that those with prior programming background benefited most from this model and irregular students suffered in their learning despite of having the content at hand all the time.

Relation to our work: As in our proposed learning approach, this study also proposed an implementation of a MOOC. In this paper, the MOOC is implemented in a programming course which is different from our work. However, the video lessons and online quizzes mentioned in this study seem to had a great contribute to the students' grades increase. Moreover, the authors mentioned that students with prior programming background benefited most from this model which is a important conclusion to us, because in our work the proposed learning approach will be applied to students with some knowledge in programming.

Table 2.13: Taxonomy instantiation for [42]

D	LA	LM	LR
Students enrolled in a programming course at Birla Institute of Technology and Science in India	MOOC	0	4

2.5 Summary

A summary of the categorized related work aiming to clarify and compare more easily all the studies reviewed is presented in Table 2.14. It is noticeable that MOOCs have been studied in latest years. In [33] and [44], MOOCs were proposed to serve as admission mechanisms to the university. In [43], [3] and [42], MOOCs were applied to programming courses (or similar) in order to solve problems found in the traditional learning [43] and to try to improve students'

[2] the same automatic judge as the one used in this dissertation

[3] However there was no relevance in those challenges to our work, so it was decided to not include them in this description.

learning effectiveness [3],[42]. In [46], the students' success in a MOOC considering their effort time was explored. Considering this, all of these studies showed that MOOCs had a great contribution towards the learning effectiveness in programming.

The flipped classroom method, which has the video lessons to teach the programming concepts in common to MOOCs, showed great results when compared to the traditional learning [13]. Additionally, this method motivates students in learning programming [29], [13], [2].

The Hackathon method seems to have a lack of studies exploring it. However, in [45] the students' perception about this method to serve as an approach to consolidate and improve learning skills was very positive.

In [40], which is the study more relatable to our work, the 3-week initiative undertaken to teach programming using video lessons, quizzes and assignments to pre-university students showed great results in their learning effectiveness.

Considering the aforementioned, it is noticeable a lack of studies using MOOCs to teach a programming language and comparing the results with the traditional learning. Additionally, the Hackathon method has been under-explored, so we find an opportunity to use it as a method to consolidate the knowledge acquired in the MOOC approach, while improving the programming skills.

Table 2.14: Summary of related work

Article	D	LA	LM	LR
[43]	Undergraduate students from a Moroccan University	MOOC	5	0
[29]	Undergraduate engineering students from a Japanese university	Flipped Classroom	5	0
[44]	Candidate students from the University of Tartu	MOOC	0	0
[33]	Candidate students from the University of Helsinki	MOOC	0	0
[13]	ICT students from a secondary school	Flipped Classroom	4	4
[45]	Undergraduate IT students from Durban University of Technology	Hackathon	0	0
[3]	Students enrolled in a computer science and technology course at university	Hybrid Learning (Flipped Classroom based on MOOC)	4	4
[46]	Students enrolled in a MOOC on edX platform	MOOC	0	0
[40]	Pre-university students in Singapore	Blended Learning	4	0
[2]	Students enrolled in a programming course at Qatar University	Flipped Classroom	4	0
[42]	Students enrolled in a programming course at Birla Institute of Technology and Science	MOOC	0	4

Chapter 3.

Selecting a MOOC for Python

Contents

This chapter proposes a step-wise method for selecting a MOOC on Python, including a set of criteria, and its application to reach three final candidates, which are then discussed by an expert panel for reaching a final choice.

[This page has been intentionally left blank]

Chapter 3
Selecting a MOOC for Python

3.1 MOOCs

3.1.1 What are MOOCs?

MOOCs are one of the emerging technologies in the field of education in the 21th century and have gained media attention globally [22][36]. MOOCs have emerged as a key mechanism for millions of learners to access semi-formal learning opportunities and to acquire new knowledge and skills particularly for those whom this has previously been impossible because of constraints of cost or geography [38]. The MOOC acronym is better explained in the following bullet list:

- Massive: the course can enroll an unlimited number of participants;
- Open: the course is open to all Internet users; registering and tracking courses remain free and open; certification fees may apply, but certification must be optional;
- Online: all classes, activities and duties can be taken online;
- Course: the course must meet educational goals, including productions, activities or homework to participants, not just online resources [27].

The European Commission defines a MOOC as *"online courses designed for a large number of participants that can be accessed by anyone anywhere, as long as they have an internet connection. They are open to everyone without entry qualifications, and offer a complete course experience online for free. They are led by subject matter experts from higher education or industry and hosted by learning management systems or dedicated MOOC platforms"* [50]. Also, two forms of MOOCs have emerged. In *cMOOCs*, learners are encouraged (though not required) to contribute actively via blog posts, tweets or other social media posts that are aggregated online by course organisers and shared with all participants via email or newsletters. The "c" stands for "connectivist" and the course approach is typically that learners pursue their own learning outcomes with a focus on community and connections. *xMOOCs*, on the other hand, resemble traditional courses and more traditional higher education teaching methods are used. Pre-recorded video lectures and scalable forms of assessment are provided to learners who can interact in pre-set forums in a single platform rather than creating and/or sharing distributed content on the Web outside the platform [50].

3.2 Comparison of MOOC platforms

There are several MOOC platforms offering courses in many areas. Some of these platforms provide free of charge access, but several others have paid access[1], therefore violating the aforementioned European Commission's definition of a MOOC. In the *BitDegree*[2] website it is possible to find a list with the *"Best Online Learning Platforms of 2020"* which are reviewed and scored by a *BitDegree* experts team. This review process is also explained in detail to provide

[1] see https://www.bitdegree.org/online-learning-platforms
[2] see https://www.bitdegree.org/online-learning-platforms/mooc-review-process

transparency of their choices. To search for the most suitable MOOC to fit this dissertation's purpose, we considered the aforementioned list which details the 10 best MOOC platforms: Coursera, Datacamp, Udacity, Udemy, edX, LinkedIn Learning, SkillShare, BitDegree, Khan Academy and Codecademy.

In those 10 platforms we searched for "Python" and the following filters were applied: course for beginners, most relevance and better classification. This search resulted in 7 courses. In the Udacity platform all the courses found were applied to a specific area (Artificial Intelligence, for example) which does not fit in our purpose of an introductory Python course. The Khan Academy platform was also removed because during the search only isolated videos about Python programming were found. The course found in the edX platform was the same as the one found in Coursera. In these circumstances, it was decided to remove the former and maintain the latter due to the ranking in the list aforementioned. After this, it was considered the classification and number of students enrolled in each course. The only courses with classification were the ones from Coursera, Udemy, BitDegree and Codecademy with 4.8, 4.6 and 4.4 and 4.1, respectively. Thus, the courses without classification were removed, meaning that only 4 courses remained. Since the purpose was to extract 3 courses of the list of 10, the number of students enrolled was the criterion applied to those 4 courses remaining. The 3 courses with more students enrolled were Coursera, Codecademy and Udemy with approximately 2 millions, 4 millions and 0.25 million, respectively (see Table 3.1). Therefore, as a result of this search, the courses from Coursera, Udemy and Codecademy were respectively, "Programming for Everybody[3]", "Learn Python Programming Masterclass[4]" and "Learn Python 2[5]".

In the following subsections the learning process of these three courses will be explained in detail using BPMN modelling as a tool to attain this goal.

Table 3.1: Results of the MOOC search

MOOC Platforms	Course Name	Classification	Number of students enrolled
Coursera	Programming for Everybody	4,8	+-2M
Datacamp	Introduction to Python		+-2M
Udemy	Learn Python Programming Masterclass	4,6	+-0.25M
LinkedinLearning	Python Essential Training		+-0.23M
SkillShare	Python 3: A Beginners Guide to Python Programming		+-0.019M
BitDegree	Master Python Fundamentals	4,4	+-0.015M
Codecademy	Learn Python 2	4,1	+-4M

3.2.1 Codecademy's course

This course is mostly free, since some syllabus topics are only available after the upgrade to the Codecademy's pro version. Regarding the free version, this course has 12 sections and includes

[3] https://www.coursera.org/learn/python?specialization=python
[4] https://www.udemy.com/course/python-the-complete-python-developer-course/
[5] https://www.codecademy.com/learn/learn-python

the main topics of Python such as syntax, conditionals, functions, lists, dictionaries, loops and input and output. The recommended time to finish the course is 25 hours. This course does not focus much on the theory, instead the main focus is coding to learn programming. The Codecademy's interface can be seen in Figure 3.1. On the left side, some key points about the topic and instructions to solve the problem are provided and on the right side an IDE where the learner writes the code to solve the problem. Whenever the learner can not solve the problem, he/she can ask for help and Codecademy provides hints to the problem solution. After running the code successfully by pressing the button "Run", the next level will be available. As mentioned before, this learning process was modelled using BPMN. The macro-process can be seen in Figure 3.2. The process starts with the learner/student pressing the button to login. After that, the learner credentials are verified in Codecademy databases and the session is restored [6]. The sub-process "Do Exercise", where the interaction between the learner and Codecademy's interface while the former is solving a problem is explained in detail in Appendix A.1.

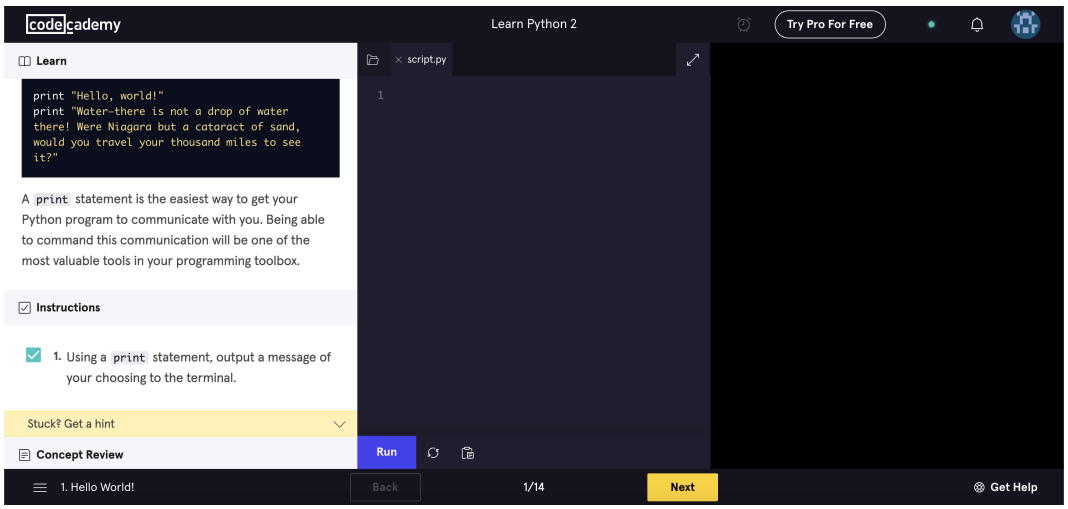

Figure 3.1: Codecademy interface

[6] Wrong credentials were not considered because it is not the objective of this modelling

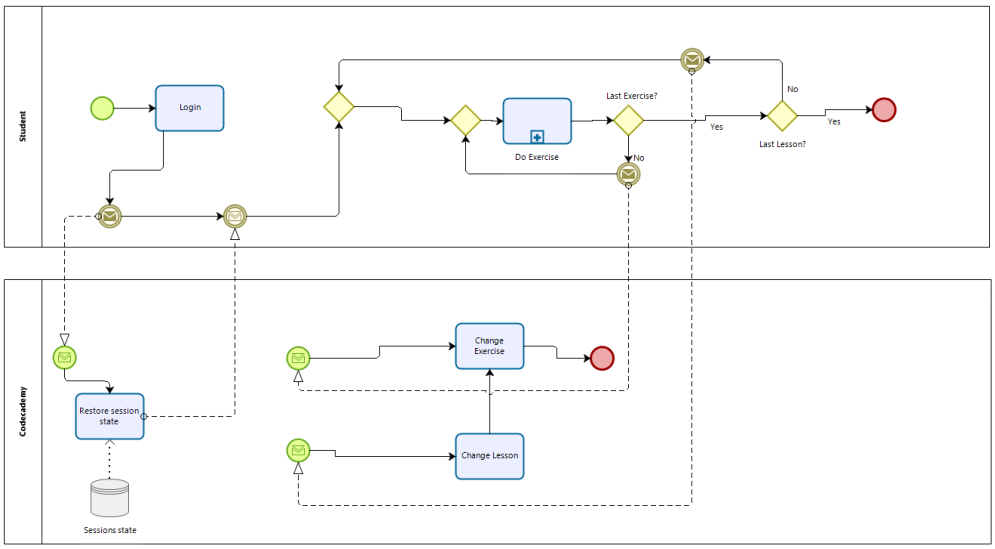

Figure 3.2: Codecademy - macro-process diagram

3.2.2 Udemy's course

This course has 50 hours of video lessons, 14 downloadable resources, 24 coding exercises, all structured into 14 sections. The recommended time to finish this course is approximately 59 hours. The first two sections are dedicated to a course introduction and installation of IntelliJ (IDE). The remaining sections cover all the essential topics of a programming language, such as program flow control, lists, input and output and functions (Figure 3.3). Each section is composed of several short videos about the topic, coding exercises and a final quiz of 10 multiple-choice questions. A particular feature of this quiz is that, in case of failing a question, the student can try again until the correct answer is given. After finishing the quiz, it is shown to the student how many questions did he/she answer correctly on the first attempt and in those that he/she did not. Udemy also reminds which lectures the student should review. Therefore, the student can always go back and watch those video lessons again. Furthermore, the student is free to choose which topics he/she wants to learn first, despite the suggested order given by the course. The macro-process of this course (Figure 3.4) shows the main interactions between the learner and the platform. This process starts with the learner logging in and the platform verifying the credentials. After that, the learner's session state is restored. The sub-process "Do Section" explains in detail the activities when the learner is watching video lessons, doing quizzes or coding exercises. The "Do Section" also details the activities done by the Udemy platform when the learner submits a quiz or assignment. This sub-process can be found in Appendix A.2.

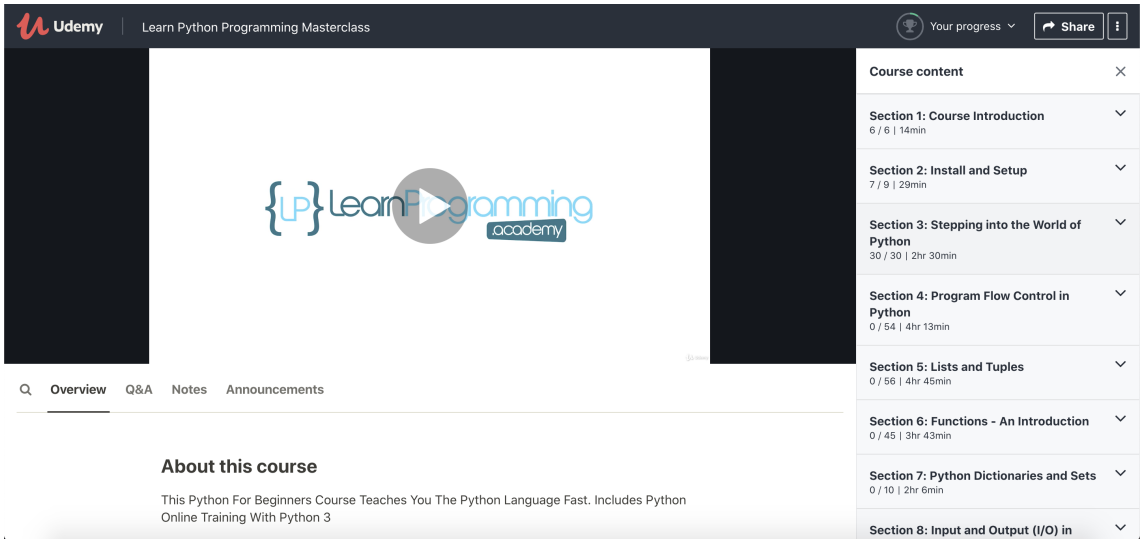

Figure 3.3: Udemy interface

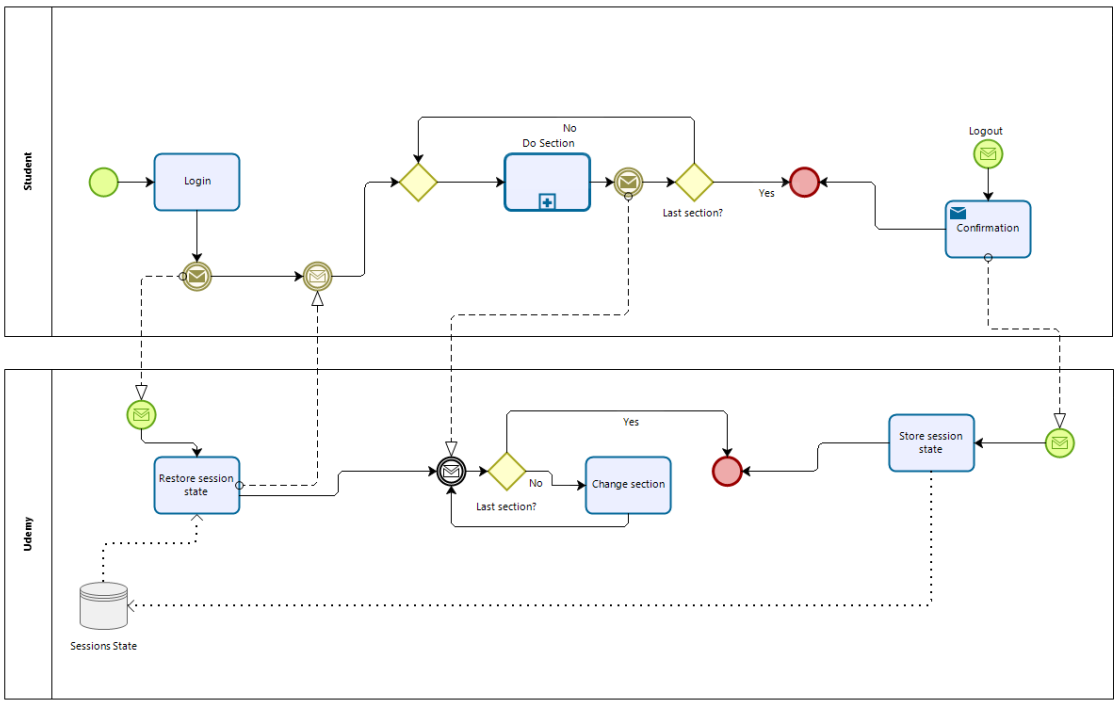

Figure 3.4: Udemy - macro-process diagram

3.2.3 Coursera's course

This course is the first module out of four that belongs to the *Python for Everybody Specialization*. This first module is an introduction to Python programming and is the one that will be analysed from now on. Thus, this module is organized in 7 sections (see Figure 3.5) and the first two sections focus on an introduction to Python programming language and its installation on Windows and Macintosh. In the following sections other topics such as variables and types, expressions, conditional code, functions, loops and iteration are addressed. Differently from the courses analysed before, this course has a component of peer assessment where students

assess each others assignments. Despite the suggested course order given by the course, the student is free to choose his/her own order, meaning that he/she can always turn back to watch videos again, or to repeat the quiz in search for a better grade. In Figure 3.6, a macro-process with the main activities between the learner and the platform is presented. As the other courses analysed before, this course also starts with the learner logging in and his/her credentials being verified by the Coursera's databases. After that, the session is restored and the student is able to watch videos, do quizzes or assignments, which is detailed in the sub-process "Perform weekly workload" (see Appendix A.3). As a conclusion reward, this course, which is provided by the University of Michigan, provides a certificate.

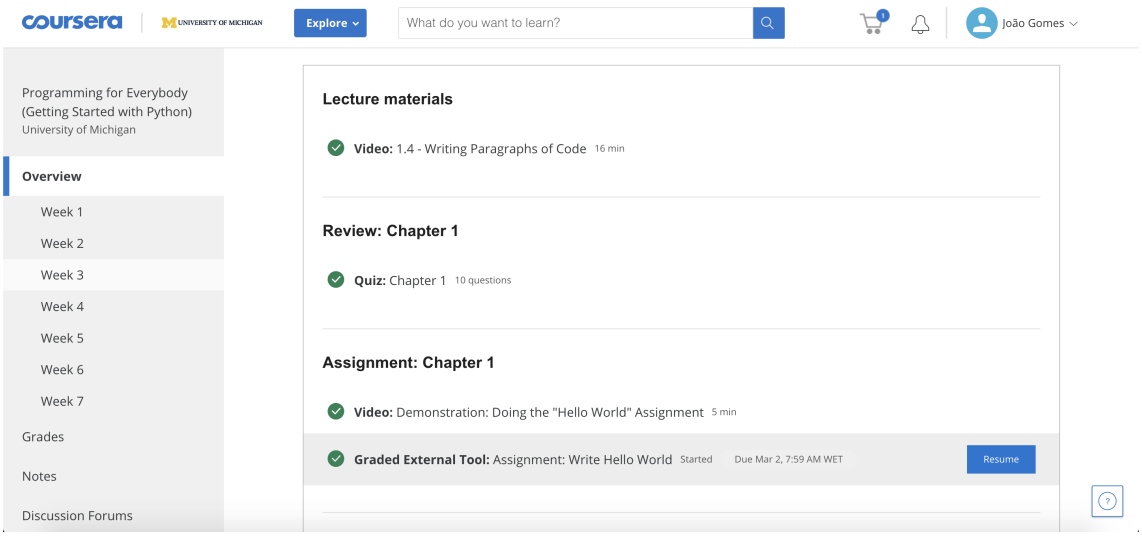

Figure 3.5: Coursera interface

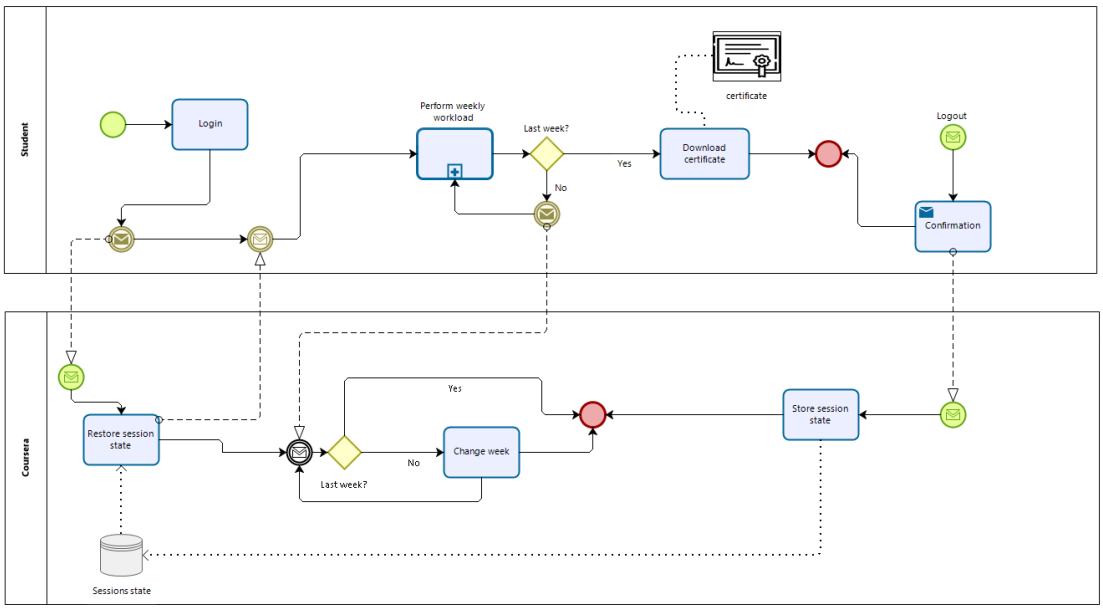

Figure 3.6: Coursera - macro-process diagram

3.2.4 The main features of the three courses

The objective of this subsection was to collect and show the main features of the courses in Coursera, Codecademy and Udemy platforms. In terms of the learning process and evaluation, it is possible to highlight two different approaches in these three courses. Coursera and Udemy use video lessons, coding exercises/assignments and quizzes to teach and evaluate the Python topics. On the other hand, Codecademy gives priority to coding supported by a short text explaining the topics. Coursera's course is the only course that has peer assessment, which allows participants to learn how to evaluate other course participants performance in their assessments. In those three courses, two types of paths were identified and named, as weak order and compulsory order, that students will find when joining the course. In the weak order type, presented in Coursera and Udemy courses, a syllabus order is suggested but students can configure their own path. Meaning that they can skip section 1 and do section 2 and later get back and do section 1. In the compulsory order, presented in Codecademy's course, the syllabus topics must be followed by the proposed order. Another feature that is important to highlight in those three courses are the syllabus topics taught. All three courses teach Python syntax, variables and types, strings, program control flow, functions and input and output. Codecademy, beyond these topics, also teaches dictionaries and sets. Udemy in addition to all the topics mentioned before adds databases, binary number system and object oriented to its syllabus. The possibility of re-attempting an assignment is a feature available only on Udemy and Coursera courses and allows participants to retry and have a better grade or in case of a failure to have the chance to pass in the assignment. Contrarily to Udemy and Codecademy, the Coursera's course is provided by the University of Michigan and gives a conclusion certificate. In order to summarize these features, Table 3.2 was created and helps to understand the next section where these features are assessed by an expert panel.

Table 3.2: Python courses' features

	Codecademy	**Coursera**	**Udemy**
Learning Process and Evaluation	Text to explain the concepts and coding	Video lesson, code exercises and quiz	Video lesson, code exercises and quiz
Peer assessment	No	Yes	No
Course order	Compulsory order	Weak order	Weak order
Possibility of re-attempting an assignment	No	Yes	Yes
Syllabus	Syntax, Variables and Types, Lists, Strings, Program Control Flow, Functions, Dictionaries and Sets, Input and Output	Syntax, Variables and Types, Strings, Program Control Flow, Functions, Input and Output	Syntax, Variables and Types, Lists, Strings, Program Control Flow, Functions, Dictionaries and Sets, Input and Output, Using Databases, Binary Number System, Object Oriented
Provided by a university	No	Yes	No
Certificate	No	Yes	No

3.3 Expert panel

In the previous section, the learning process of three courses from three different platforms was designed using BPMN allowing to collect features and compare those courses (Section 3.2). According to [6], previous studies have used small samples to get expert feedback to evaluate and support model development. Many authors have used expert panels, specially, in software engineering. For example, in [20] 30 experts were interviewed to obtain elements for their instrument to evaluate requirements engineering success. In [19], 11 experts conducted a review process. We gathered an expert panel in order to choose the most suitable MOOC from the previous 3 candidates, following the methodology proposed in [34]. The latter includes eight main steps, which are presented in Figure 3.7.

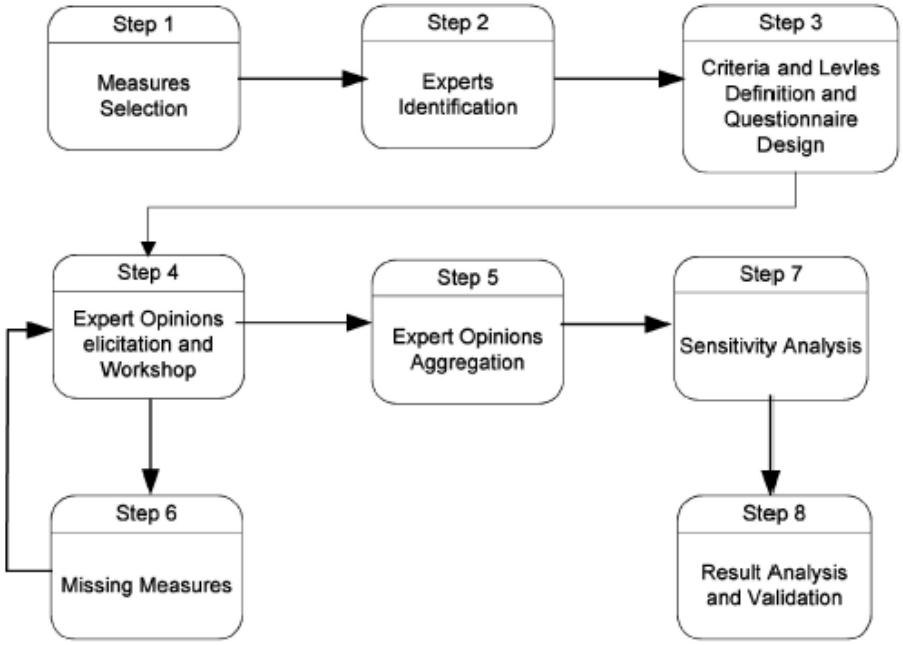

Figure 3.7: Methodology followed from [34]

Based on some of those steps, the organization of this expert panel was divided into five phases:

1. Identify and collect the main features of the three courses:
   - Overall learning process and evaluation;
   - Peer assessment;
   - Course order;
   - Course topics;
   - Course origin;
   - Provision of certificate.
2. Invite four professors with expertise in teaching programming;
3. Build a questionnaire in Qualtrics[7] based on those features found in the first step;
4. Collect expert opinions through an expert panel session;

[7] https://www.qualtrics.com/

5. Outcome analysis of the expert panel session.

The identified Python programming experts were all professors with expertise in teaching programming in the DCTI at our university. In a total of four experts invited, all accepted the invitation to participate in the meeting. To ensure that the discussion among the experts was written in this dissertation as it happened, the experts gave their permission to record the session. In the beginning of the session, a short presentation to describe the background and the reason of the session was made, as well as how it would work. After the presentation it was asked to the experts to fill out the questionnaire.

3.3.1 Questionnaire

The questionnaire had 8 questions and it was built in a way that experts chose individually the features which seemed better in terms of teaching Python. After that, with the features associated with the corresponding course, they had to choose the best combination. This questionnaire use a five-level Likert scale ("Not important at all" to "Very important"), multiple-choice questions with single answer and multiple answer, and a rank order question. The first seven questions were to evaluate the features of the three courses individually and the last question's purpose was to make the experts choose the best course based on its features, not knowing the name of the course or the platform. In the next subsection it will be presented the questions, the respective experts' answers and the discussion between experts. The questionnaire can be seen in Appendix A.4 and A.5

3.3.2 Analysis

The discussion was conducted question by question with experts' answers presented in graphs, and it was asked to the experts to justify their answers. The first question was a multiple-choice question about the learning process and evaluation. The two possibilities of choice, which were the only types of learning process and evaluation presented in the courses, were "video lesson, code exercises and quiz" and "text to explain the concepts and coding". To this question, the answers from all the experts were unanimous since all considered "video lesson, code exercises and quiz" the better approach.

The second question was a Likert scale question referring to the importance of peer assessment in a MOOC. Two experts answered as "Not important at all", one considered peer assessment with "Low importance" and the other has seen it with "Moderate importance" (Figure 3.8). One of the experts who answered "Not important at all" said that peer assessment could be valuable in the teacher perspective, to this point of view the other expert, who answered the same, agreed and added that peer assessment would be important if it was possible that the teacher could see how a student explains the contents to others. The other two experts who answered, "Low importance" and "Moderate importance", ended up agreeing with the colleagues.

In the third question, which the results are presented in Figure 3.9, the goal was to understand which was the best path that students can follow in these three courses. One of the courses has a specific order to follow, which we called "Compulsory order", the other two courses suggest an order, but students can configure their path, "Weak order". The experts

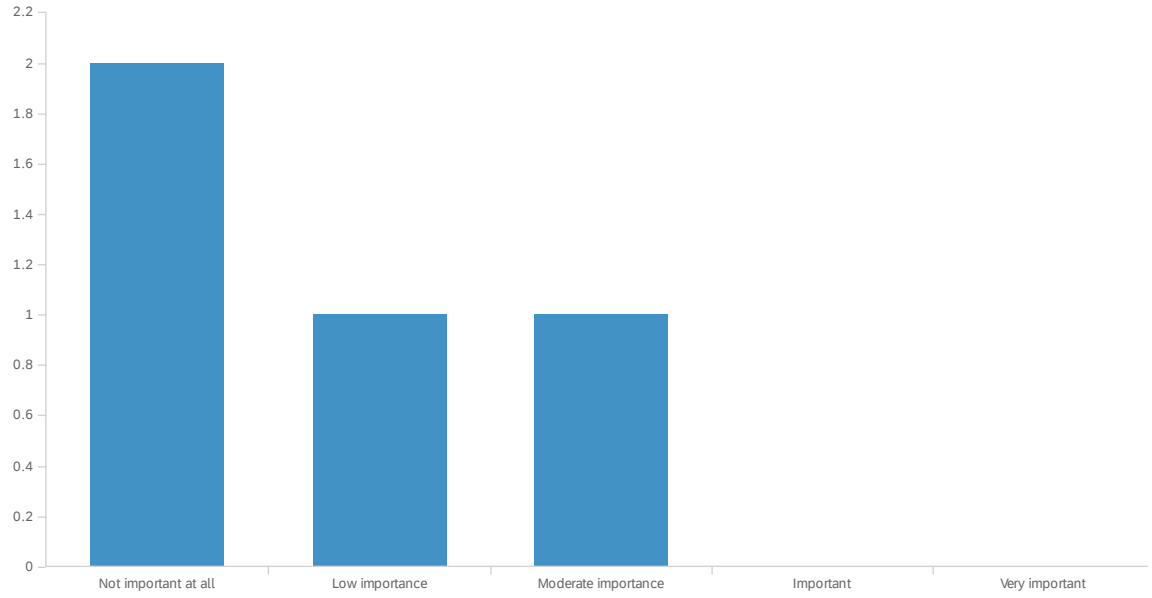

Figure 3.8: Expert panel - peer assessment

had to choose the option that they thought better in these circumstances, and the most chosen approach (chosen by three experts) was the "Compulsory order", while the "Weak order" was chosen just by one expert. One of the experts justified his choice "Compulsory order" saying that, if the course was for students with basic knowledge in programming, the "Weak order" would be more effective. Also, this expert said: *"a long time ago the syllabus changed in a Computer Engineering course that I was enrolled in as a teacher. The topic control flow, which was taught before the topic functions, became to be taught after, and thanks to this order change, the students improved their code, making it less confusing"*. Another expert, who answered "Weak order", replied to the previous expert saying that in case of an online environment and not a traditional one, the students should have a path suggestion, but at the same time they should have the flexibility to watch the video lessons according to their needs. He also added that since the students have basic knowledge in programming, an obligation of a specific order could increase a course dropout caused by students' discouragement. Reacting to this, another expert said that, they had to finish all the topics eventually to have the certificate of completion, and for that reason they should follow the "Compulsory order". To fight the problem of discouragement, this expert added that a component of gamification would be a possible solution.

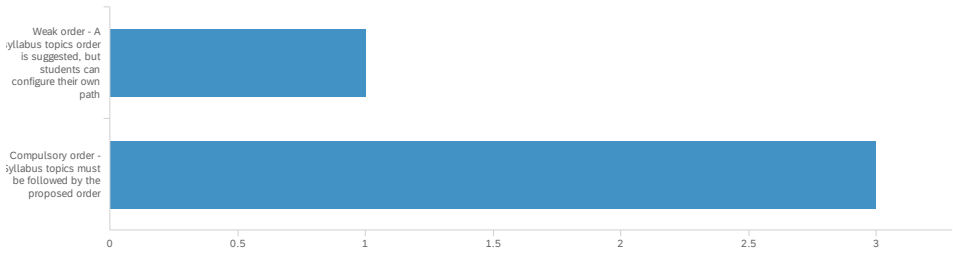

Figure 3.9: Expert panel - course order

The fourth question was about the importance of re-attempting an assignment and one of

the four experts answered as “Not important at all”, two answered as “Important” and one as “Very important”. The expert who answered “Not important at all” was asked to justify his answer and he mentioned that when he was answering he could take “Very important” as well. In his opinion, when a student does an assignment is to evaluate the knowledge learned in the video lessons. For that reason, the assignments only should be repeatable if it is to use as a study component, otherwise, if it is to get a better grade, it does not make sense. The other experts when answering to this question assumed that the goal of re-attempting an assignment was to help students understand that they must review the previous video lessons and do the assignment again to reevaluate their knowledge (Figure 3.10).

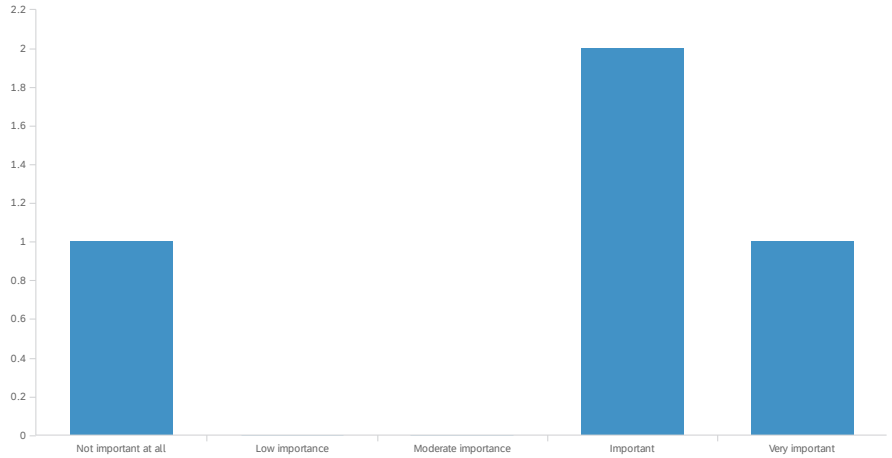

Figure 3.10: Expert panel - re-attempting an assignment

As mentioned before the three courses were for beginners in Python programming who had previous mastering in at least another programming language. Therefore, the fifth question was to understand which topics of the syllabus were more relevant to this type of students. The topics from the three courses were all aggregated in a multiple-choice question.

Given the results presented in Table 3.3, it is possible to see that the topics which were more consensual among the experts to a course for beginners in Python programming were Syntax, Variables and Types, Functions, Dictionaries and Sets and Object-Oriented Paradigm. Strings, Program Control Flow, Lists, Input and Output were considered essential for some experts. Binary Number System was not considered important with zero votes among the experts.

Many of these MOOCs are provided by universities or organizations which provides certificates for students who finish the course. Others are provided by subject experts and usually do not provide a final certificate. Thus, the sixth and seventh question were to understand the importance of these two features in experts’ opinion. In the sixth question, which asked the importance of the course being provided by a university, the experts’ answers were unanimous. It can be seen in Figure 3.11 that they all answered “Moderate Importance”.

The seventh question (Figure 3.12), which was regarding the certificate, had similar responses to the previous question. Although, in this question, one expert answered that certificate is “Important” and the other three answered “Moderate importance”. The expert who classified the certificate as “Important” mentioned that, since the students put the effort in the attempt to finish the course, they must be rewarded somehow if they finished. Also, he added

Table 3.3: Expert panel - syllabus

#	Field	Choice Count	
1	Python Syntax	11.76%	4
2	Variables and Types	11.76%	4
3	Strings	8.82%	3
4	Program Control Flow (If, for, ...)	8.82%	3
5	Lists	8.82%	3
6	Functions	11.76%	4
7	Dictionaries and Sets	11.76%	4
8	Input and Output	8.82%	3
9	Using Databases	5.88%	2
10	Binary Number System	0.00%	0
11	Object Oriented Paradigm	11.76%	4

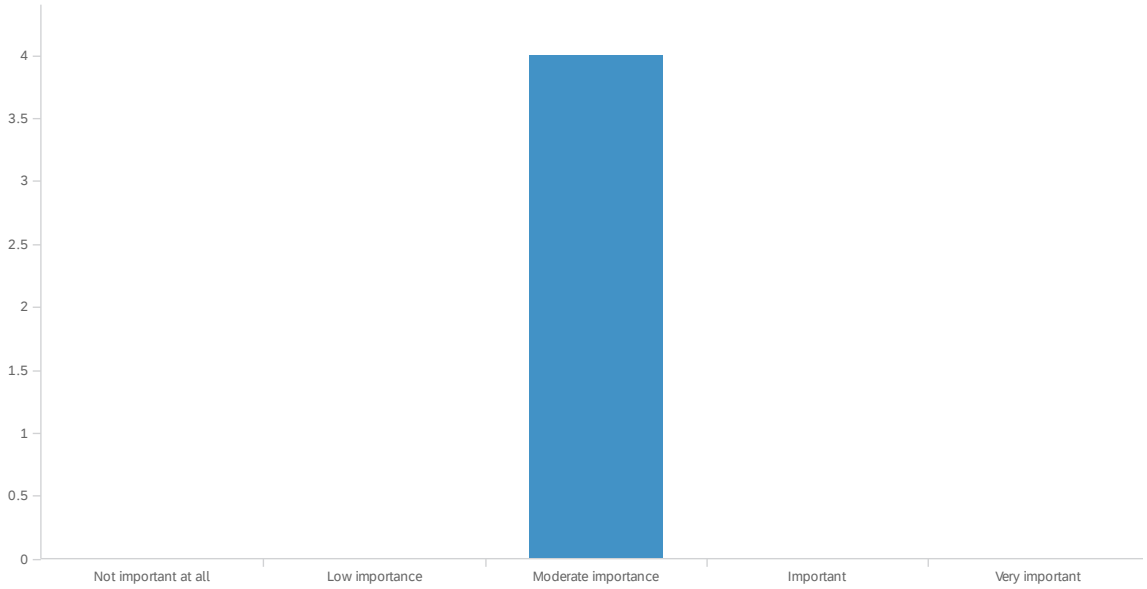

Figure 3.11: Expert panel - provided by a university

that in the perspective of the student, this certificate is important because they can use it as a component of their curriculum vitae. All the other experts agree with each other that the certificate has some importance but is not the most important feature that an online course should have. Instead, they considered more important other features, such as the course being well structured and the syllabus is appropriate to beginners in Python programming, that they consider more important.

The last question of this questionnaire was a ranking order question. As it is possible to see in Table 3.4, the features, which were assessed individually before, were now linked to the corresponding course. The courses were named X,Y and Z in order to avoid experts

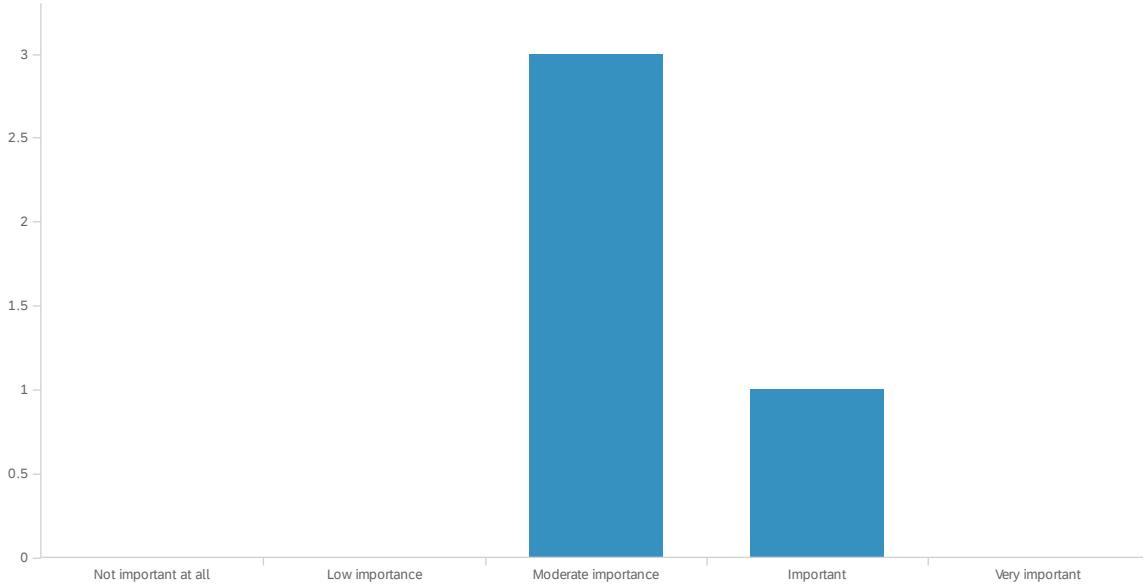

Figure 3.12: Expert panel - certificate

from being influenced by any experience that they could have with the platforms/courses. The main objective was to make them choose based on the features of each course without knowing its name. Taking these images into account, the experts had to rank them according to their preference and knowing its purpose, which was to choose the best course for students who had knowledge of Java but want to learn Python. The results of this question are presented in Table 3.5.

Table 3.4: Course X,Y and Z

	Course X
Learning Process and Evaluation	Text to explain the concepts and coding
Peer assessment	No
Course order	Compulsory order - Syllabus topics must be followed by the proposed order
Possibility of re-attempting an assignment	No
Syllabus	Python Syntax, Variables and Types, Lists, Strings, Program Control Flow, Functions, Dictionaries and Sets, Input and Output
Provided by a University	No
Certificate	No

	Course Y
Learning Process and Evaluation	Video lesson, code exercises and quiz
Peer assessment	Yes
Course order	Weak order - A syllabus topics order is suggested, but students can configure their own path
Possibility of re-attempt an assignment	Yes
Syllabus	Python Syntax, Variables and Types, Strings, Program Control Flow, Functions, Input and Output
Provided by a University	Yes
Certificate	Yes

	Course Z
Learning Process and Evaluation	Video Lesson, code exercises and quiz
Peer assessment	No
Course order	Weak order - A syllabus topics order is suggested, but students can configure their own path
Possibility of re-attempt an assignment	Yes
Syllabus	Python Syntax, Variables and Types, Lists, Strings, Program Control Flow, Functions, Dictionaries and Sets, Input and Output, Using Databases, Binary number system, Object Oriented
Provided by a University	No
Certificate	No

Table 3.5: Expert panel - courses' rank order

Field	1	2	3	Total
Course X	25.00% 1	0.00% 0	75.00% 3	4
Course Y	50.00% 2	25.00% 1	25.00% 1	4
Course Z	25.00% 1	75.00% 3	0.00% 0	4

Course X had one vote for the best course and had three votes for the third best course. Course Y was considered by two experts the best course and the other two experts voted as the second and the third best course. Course Z earned one vote for the best course and three for the second best course. Course X was the least preferred in the experts' opinion. However a good point of view was given by the expert who chose course X, he mentioned that video lessons can be uninteresting to students who know already how to program and if they had instead text to explain the syllabus topics it would be more interesting and easier to them. The race for the

best course was between courses Y and Z. Three experts were unanimous that video lessons was the best approach to take in this type of course. The main reason that made them choose between Y or Z was the syllabus topics. One of the experts mentioned that taking into account the fact that course Y was provided by a university could mean that the course has more quality than the other. With this in mind, he added that he prefers to have less syllabus topics but more quality in the course. The two remaining experts, one agreed with the previous said. The other mentioned that the personal reason to choose course Z over course Y was the importance of all the syllabus topics for beginners presented in course Z.

3.3.3 Final result

Course X, Y and Z are Codecademy, Coursera and Udemy courses, respectively. The discussion between the Coursera's course and the Udemy's course was tight. Coursera's course had two votes for the first place and one vote for the second place. On the other hand, the Udemy's course had one vote for the first place and three for the second place. Despite the tight discussion between these two courses, the choice was for Coursera's course. As mentioned before, the purpose of this expert panel was to choose one of these three MOOCs to use the best and most suitable in a pedagogical event named Pythacon that will be explained in detail in the next chapter.

[This page has been intentionally left blank]

Chapter 4

In-class MOOC approach trial: Pythacon event

Contents

This chapter describes the plan and execution of a controlled experiment, named Pythacon, that allowed to validate the proposed learning approach, *In-class MOOC*, using the previously selected MOOC. Finally, the results of the experiment are analysed.

[This page has been intentionally left blank]

Chapter 4
In-class MOOC approach trial: Pythacon event

4.1 Introduction

Pythacon was a pedagogical event organized at Iscte and its duration was one week divided into two phases from September 14th to 18th, 2020. The main goal was to understand the effectiveness of learning Python using the proposed learning approach (*In-class MOOC*) and compare it with the traditional learning approach.

The planning step in the organization of Pythacon included the development of its website[1] in order to disseminate all the information about the event and attract students to register and participate in it. Direct emails were sent to all students, providing the link to the inscription page in Pythacon's website.

The first phase of this event, which lasted four days, was dedicated to the *In-class MOOC* learning using the Coursera's Python course and was targeted to students from the three following undergraduate degrees: Computer Science and Business Management[2], Computer Engineering and Telecommunications[3] and Computer Engineering[4]. The participants from these three undergraduate degrees formed the experimental group. They had previous programming knowledge, mostly in Java, acquired in traditional learning, but Python was not part of their degrees' syllabus. The main objective of this phase was to provide these participants the opportunity to learn Python with a MOOC in class. Therefore, learning Python through a MOOC in class was the treatment applied to our subjects. Furthermore, in order to measure participants' workload in our proposed learning approach, the NASA TLX was used.

The second phase, inspired by the International Collegiate Programming Contest (ICPC)[5] event, was the contest day where participants put to test their knowledge on Python programming. In order to get to this phase, participants from the experimental group had to complete at least the first module of the selected MOOC. As for the control group, it was composed of Data Science[6] undergraduate students, that learned Python through the traditional in-class learning approach. Therefore, the latter had no precondition for joining the second phase.

4.2 Planning

4.2.1 Participants description

As mentioned in the previous section, students had to register in Pythacon website. This registration page was a questionnaire developed in the Qualtrics platform[7] containing questions

[1] https://sites.google.com/iscte-iul.pt/pythacon/
[2] https://www.iscte-iul.pt/course/6/bachelor-bsc-in-computer-science-and-business-management
[3] https://www.iscte-iul.pt/course/2/bachelor-bsc-in-telecommunications-and-computer-engineering
[4] https://www.iscte-iul.pt/course/3/bachelor-bsc-in-computer-engineering
[5] https://icpc2019.up.pt/
[6] https://www.iscte-iul.pt/course/291/bachelor-bsc-in-data-science
[7] https://www.qualtrics.com/

about students' identification such as student number, email, undergraduate degree and corresponding year of inscription. The questionnaire also had other questions about programming skills, programming courses completed and respective grades, previous mastering in other programming languages and familiarity with MOOCs.

In total, 102 registrations were received. As represented in Figure 4.1, the undergraduate degree with more registrations was Computer Engineering, with a total of 36 registrations, followed by Computer Science and Business Management with 33. Telecommunications and Computer Engineering had 19 registrations and Data Science was the undergraduate degree with less registrations, only 14. The most common year of the students was the second year with a total of 68 students. The first and third year were represented by 4 and 23 students, respectively. From the fourth year, which is only applicable in the Computer Science and Business Management degree there were 7 students.

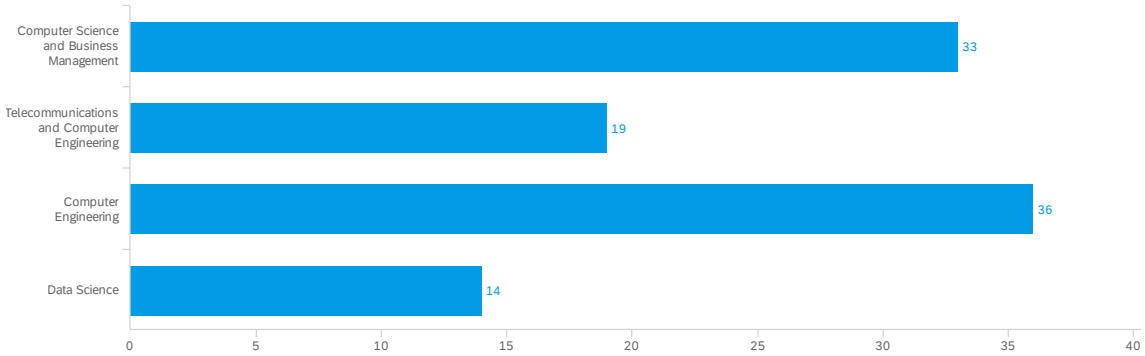

Figure 4.1: Pythacon registrations by undergraduate degree

Most students only obtained their programming skills at the undergraduate degree (4.1), although some of them mentioned other situations when they gained or improved their programming skills, which are: high school, MOOCs, books, videos online and internet.

Table 4.1: Pythacon participants former programming skills

#	Field	Percentage
1	High School	8.82%
2	Undergraduate degree	100.00%
3	MOOCs (Massive Open Online Courses)	5.88%
4	Books	3.92%
5	Other	5.88%

The programming courses offered on each undergraduate degree and the number of students that completed them are presented on Table 4.2, thereby confirming that the vast majority of students in the experimental group had taken successfully at least 3 programming courses in Java and the vast majority of students in the control group had taken successfully 2 courses on Python.

Table 4.2: Programming courses formerly concluded by Pythacon participants

	Computer Engineering	Computer Science and Business Management	Telecommunications and Computer Engineering	Data Science	No. Students and %
Introduction to Programming	1st year	1st year	1st year	-	88 (100%)
Algorithms and Data Structures	1st year	1st year	1st year	-	79 (89.7%)
Objected Oriented Programming	1st year	1st year	1st year	-	75 (84.3%)
Concurrent and Parallel Programming	2nd year	2nd year	2nd year	-	29 (33%)
Programming	-	-	-	1st year	13 (92.9%)
Data Structures and Algorithms	-	-	-	1st year	12 (85.7%)

In Figure 4.2 and Figure 4.3, the grades' distribution across the different programming courses is presented . Regarding the experimental group, it can be observed the median grade of the four programming courses, presented here in the same order as in Figure 4.2: *(Mdn = 14)*, *(Mdn = 15)*, *(Mdn = 12)* and *(Mdn = 14)*. Additionally, by observing the interquartile range (Q3-Q1) in the Introduction to Programming course, it is possible to affirm that about 50% of the students had their grades between 11 and 16.4, meaning that this programming course was the one with the largest grade distribution. The programming course with the shortest distribution was the Objected Oriented Programming course where about 50% of the students had their grades between 13 and 17. In Figure 4.3, the median grade of the Programming course and Data Structures and Algorithms course was 13 and 15, respectively. In the Data Structures and Algorithms course it is possible to affirm that about 50% of the students had their grades between 13 and 17. Furthermore, in the Programming course it is noticeable the presence of outliers, meaning that there were students with grades that stretched an abnormal distance from the remaining distribution.

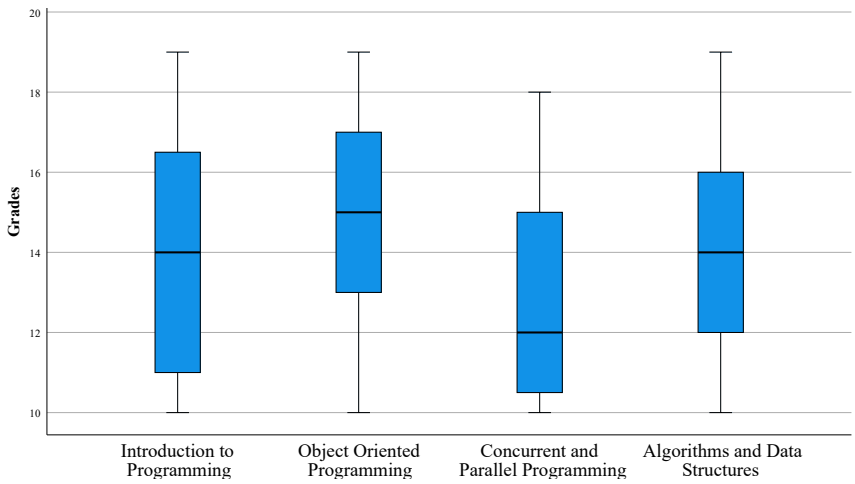

Figure 4.2: Pythacon - programming courses grades' distribution in the experimental group

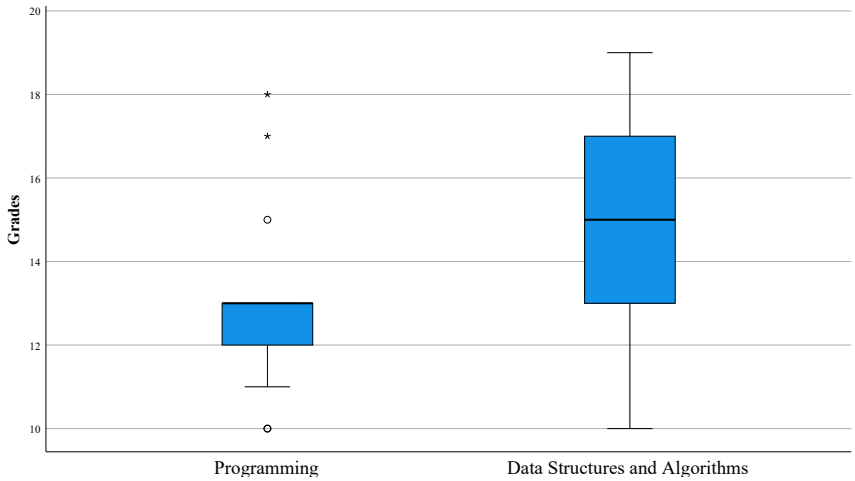

Figure 4.3: Pythacon - programming courses grades' distribution in the control group

Towards understanding of the familiarity with MOOCs, students were questioned regarding which platforms of online courses they used before. Observing Figure 4.4 one can see that only 9 participants had contact with MOOCs, which corresponds to 8.82% of the total. Regarding which platform they used, 4 answered Coursera, 2 the Udemy platform, 1 Codecademy and the other 2 mentioned Code.org in the "other" option.

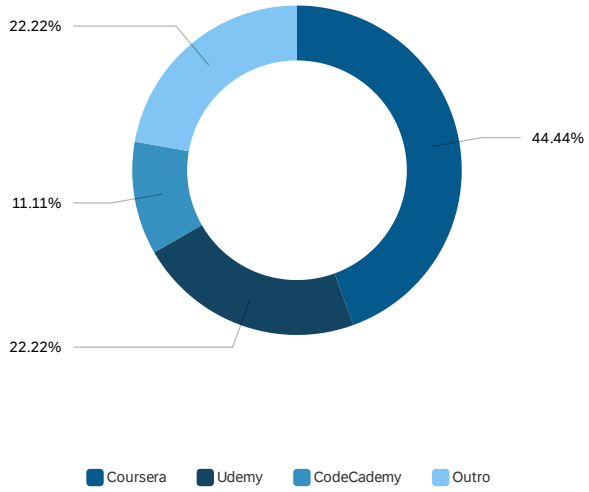

Figure 4.4: Pythacon - MOOC platforms used

In Table 4.3 are presented the programming languages that students had contact with. In this question, students were asked to classify in a scale of 0 to 10 their knowledge in the selected programming languages. With no surprise Java was the most recurrently selected programming language and with the higher mean score (6.23). The least chosen programming language was PHP which also had the lowest mean score (1.16). Python was chosen by 63 students and had a mean score of 3.37, meaning that despite being the second most chosen programming language these students do not have much knowledge in Python programming. This particular aspect about the Python programming language confirms the "Python gap" mentioned in Section 1.1.

Table 4.3: Pythacon - Mastering of other programming languages

#	Field	Minimum	Maximum	Mean	Count
1	Java	0.00	9.00	6.23	91
2	C, C++, C#	0.00	8.00	3.49	49
3	Python	0.00	8.00	3.37	63
4	SQL	0.00	8.00	3.76	45
5	JavaScript	0.00	8.00	2.26	39
6	PHP	0.00	7.00	1.16	25
7	HTML	0.00	10.00	2.98	52
8	Visual Basic	0.00	9.00	2.00	29
9	Outra	0.00	9.00	2.18	17

4.3 Execution

4.3.1 First phase description

Based on the inscriptions, we were expecting 88 participants in the first phase. The additional 14 participants inscribed were from the Data Science degree and were automatically admitted in the second phase.

Given that Pythacon took place during the COVID-19 pandemic, albeit in a period of slowing down between two peaks, participants were organized into two 5-hour shifts each, to maximize the physical distance between each other's tables. One shift worked from 8:00 am to 1:00 pm and another from 1:30 pm to 6:30 pm. Therefore, since this first phase spanned for 4 days, each student had 20 hours scheduled for *In Class MOOC* learning. This duration corresponds roughly to the recommended 19 hours for the first module of the selected MOOC course, the "Olympic minimums" to be fulfilled by these first phase participants to be accepted in the second phase.

At the room doors, participants used the provided hand disinfectant gel and, in the interval between the two shifts, the room was disinfected by the cleaning staff at Iscte. The chosen room had excellent natural lighting and good air circulation. Each participant had their own table with an electrical connection for their laptop and coffee breaks were offered approximately halfway through each shift. WiFi traffic workload was monitored continuously by the IT department to guarantee that it did not cause any disruptions in participants' work, especially because it encompassed the visualization of many videos that require a good bandwidth. All of this was important to guarantee good working conditions since the course was intensive.

In the first day we had 76 participants, which means a dropout of only 13.6%. One participant concluded the first module in just 1 hour and 23 minutes and following this participant other 34 finished the first module in the first day, i.e. within the first 5 hours, that corresponds to less than 25% of the aforementioned recommended duration for the first module. This confirmed the importance of choosing a MOOC that allowed "fast-forward" for participants with

previous knowledge on other programming language(s), as mentioned in section 3.2. To maximize the learning outcome, many participants progressed to the following MOOC modules in Coursera which were *Python Data Structures*, *Using Python to Access Web Data* and *Using Databases with Python*.

In Figure 4.5 it is possible to observe the mortality rate in Pythacon participation. The increase of participants from the Computer Engineering degree and Telecommunications and Computer Engineering degree from day 3 to day 4 was probably due to the fact that we announced that by the end of each shift in day 4 we would provide additional information and instructions regarding the contest to be hold on day 5.

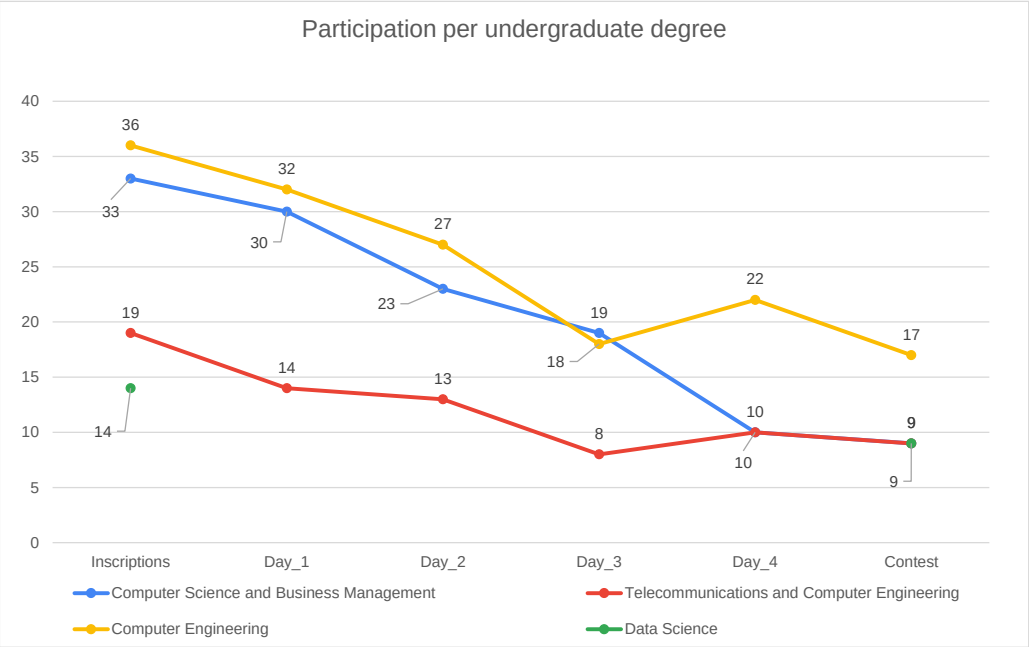

Figure 4.5: Participation in Pythacon

The number of MOOC modules finished by participants on the first phase is presented in Figure 4.6. The undergraduate degree with better performance was the Computer Engineering degree with 32 participants finishing the first MOOC module, 19 who finished the second module and 5 finishing the third module. It is important to refer that the first module was finished by all the 76 participants with different paces, which will be discussed later in this section. Observing Figure 4.6, it is possible to note that just 1 participant, who was from the Computer Science and Business Management degree, finished the fourth module. The average of modules finished by student was 1.47.

In this first phase the students could have used the 20 hours (1200 minutes) scheduled to finish the maximum modules as possible. Based on that, the average time that students spent in the four days of the first phase was 808 minutes. All the modules had a suggested time given by the Coursera platform which was 19 hours (1140 minutes) for the first, second and third modules and 14 hours (840 minutes) for the fourth. In Figure 4.7, it is presented the distribution of the elapsed time in each MOOC module. The median of the elapsed time for modules 1, 2 and 3 was respectively, $Mdn = 217.5$, $Mdn = 255$ and $Mdn = 262.5$. The largest distribution of the elapsed time in all the three modules was in the second module,

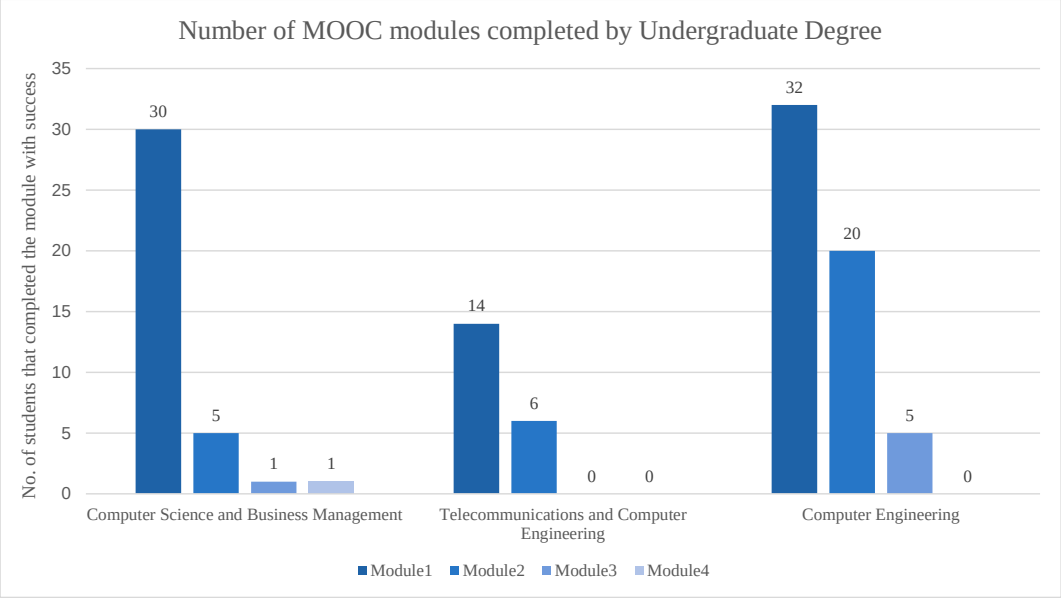

Figure 4.6: Number of MOOC modules completed with success by undergraduate degree

where about 50% of the students concluded it between approximately 190 and 300 minutes. The third module had the shortest distribution where about 50% of the students finished it between approximately 250 and 260 minutes. This may be due to the small number of students (probably the more proficient ones) who finished module 3 when compared to the other two modules. Still regarding the third module, it is possible to observe an outlier, meaning that one student's elapsed time stretched an abnormal distance from the remaining distribution.

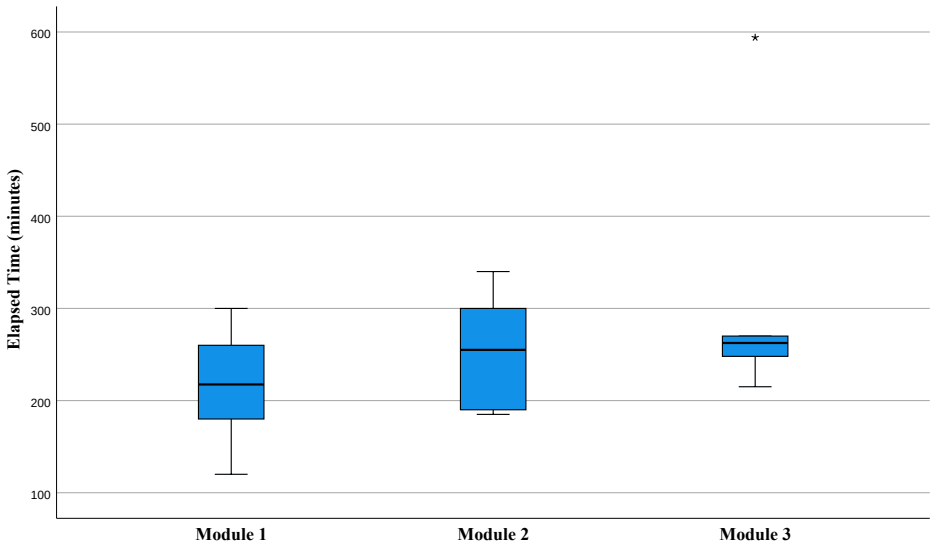

Figure 4.7: Distribution of the elapsed time in each MOOC module

4.3.2 Second phase description

In this second phase of the Pythacon, the main goal was to test the knowledge acquired in Python programming either by the experimental group and the control group. All the 76 participants from the experimental group finished the first MOOC module in the first phase that guaranteed the possibility for being in the second phase. However, only 35 out of 76 were present in the contest day. Additionally to these 35 participants from the experimental group, only 9 out of 14 students from the control group went to the competition. Thus, in the contest day a total of 44 participants were present. A group of professors, who were involved in the organization of this event, created a set of six exercises which were divided in three categories of two exercises each. Hence, there was two easy, two medium and two complex exercises. These exercises, or later called problems along with input and correspondent output data, were inserted in the Mooshak automatic judge. Mooshak's interface allows students to the problem statement and submit solutions. The submissions can be classified as observed in Table 4.4, re-adapted from a tutorial[8] at Mooshak's website. This automatic judge ran in a virtual machine with the following specifications: SO – Debian 9; CPU – Intel(R) Xeon(R) CPU E5-2650 2.20GHz; RAM – 4GB.

Table 4.4: Submissions' classification on Mooshak

Severity	Classification	Meaning
9	Requires Reevaluation	For some reason the program has to be re-evaluated
8	Runtime Error	The program crashed, i.e. it exited prematurely due to a run-time error
7	Invalid Exit Value	The program terminated with an invalid code
6	Invalid Function	The program or evaluator has called an invalid function and/or an internal error occurred
5	Time Limit Exceeded	The program did not finish within the allocated amount of time
4	Memory Limit Exceeded	The program exceeded the allocated amount of memory
3	Output Limit Exceeded	The program generated an output too long for this problem
2	Wrong Answer	The program runs through one or more test cases without a run-time error but the output did not match the expected output
1	Presentation Error	The output seems to be correct but it is not presented in the required format
0	Accepted	The program passed all tests and is accepted as correct

The contest duration was 5 hours and during this period it was received 754 submissions. There were 5 types of submissions registered: "Accepted", "Wrong Answer", "Time Limit Exceeded", "Invalid Submission" and "Runtime Error", which are presented in detail in Figure 4.8. The "Wrong Answer" submissions had the highest percentage of the total submissions. However, in this dissertation it will only be discussed the "Accepted" submissions by problem and undergraduate degree in order to find out if the undergraduate degree had influence on the learning performance in the contest.

[8] https://mooshak2.dcc.fc.up.pt/mooshak/automated-judge-newcomers

In Figure 4.9 allows observing differences on accepted submissions among undergraduate degrees. Later in this dissertation we will check if those differences are statistically significant.

In Figure 4.10 it is presented the average time of the six problems. The first problem had the lower average time of all the problems *(M = 00:57:43)*. In the second problem there was an increase in the average time that students took to solve this problem *(M = 02:47:31)*. In the two following problems, the third and fourth, there was a slight dropout in the average time of the third problem *(M = 02:44:24)* and in the fourth problem the dropout was much more noticeable *(M = 02:15:35)*. In the fifth problem, the solving average time increased *(M = 02:52:46)*. The sixth problem did not have any accepted submission, which means that none of the participants managed to solve this last and most complex problem. For each problem it was given a corresponding value in points. P1 and P2 were worth 1 point each, P3 and P4 were worth 2 points each and P5 and P6 were worth 3 points each. At the end of the 5 hours, the points were calculated towards a final contest ranking. Several students had equivalent scores, therefore these points were adjusted considering the time that each student took to solve each problem.

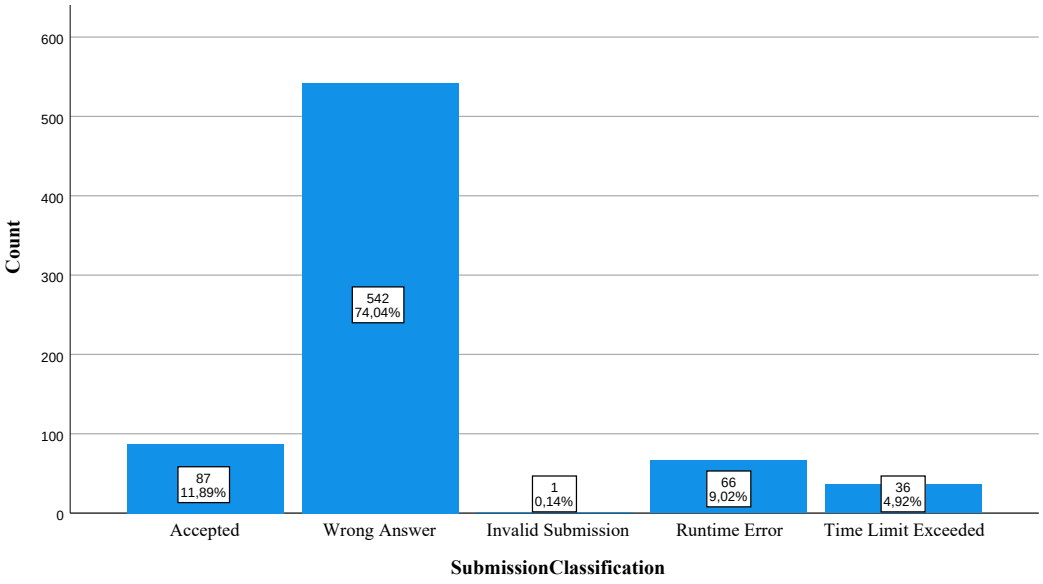

Figure 4.8: Pythacon submissions

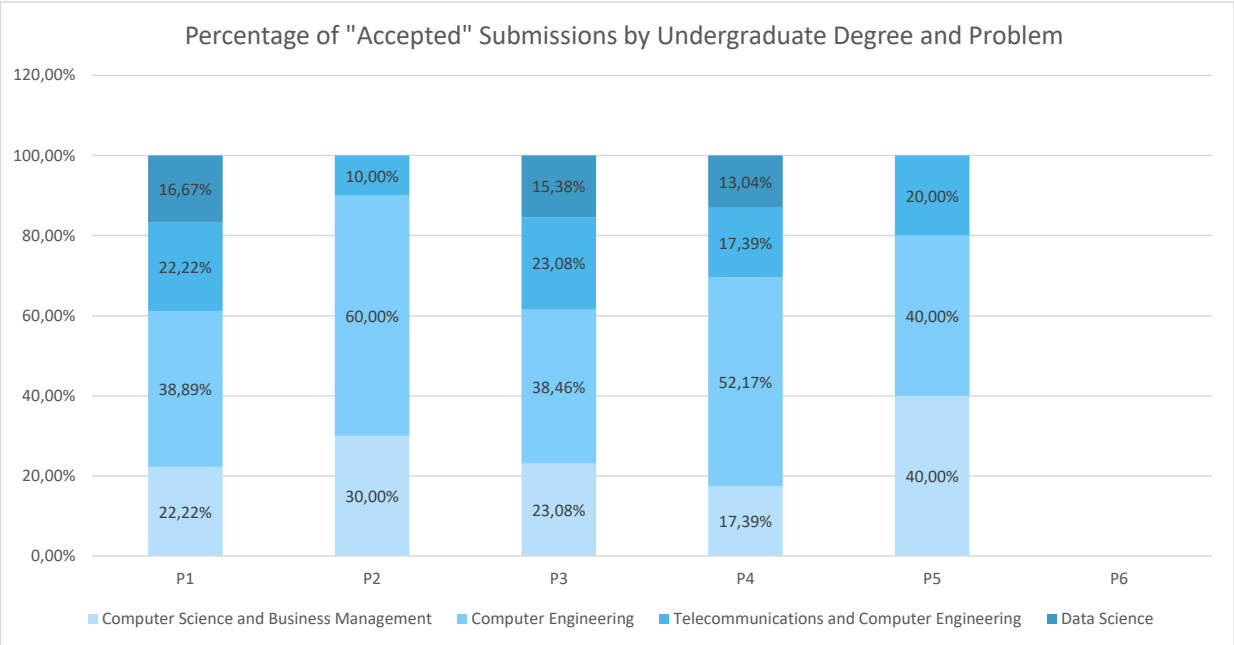

Figure 4.9: Percentage of accepted submissions by undergraduate degree and problem

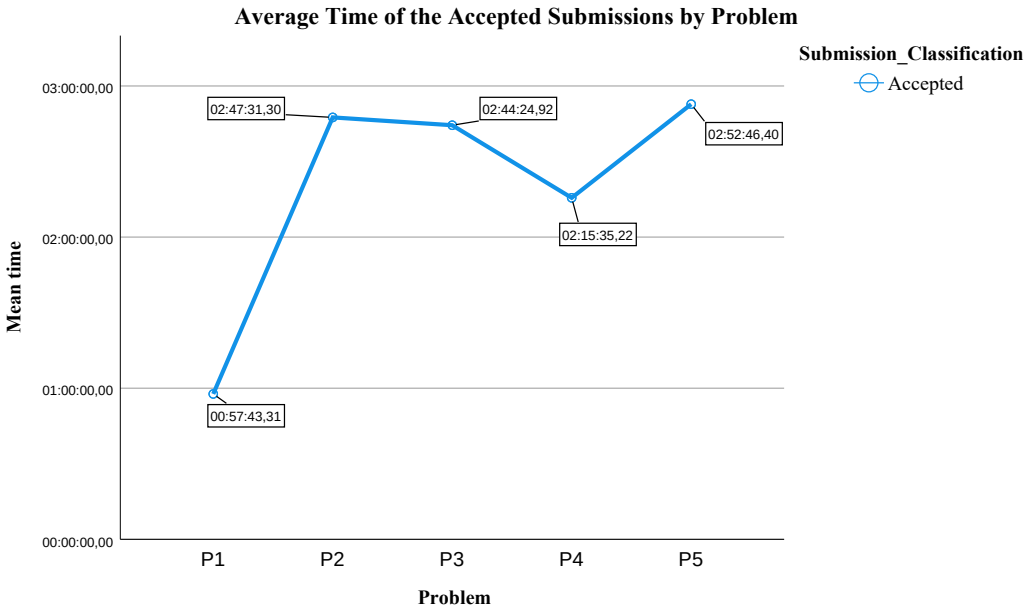

Figure 4.10: Average time of accepted submissions by problem

4.4 Analysis and results

4.4.1 NASA Task Load Index

NASA TLX was developed more than 30 years ago to measure workload in aviation [26] [25]. Since then this questionnaire was used in several areas and estimated that it was used in more

than 300 studies [25]. It is claimed in [49] that this questionnaire became a dominant scale in workload measurement with an explosive use between 1985-2012. According to [49] it can be stated that workload has become synonymous with TLX. Despite the fact that NASA TLX being mostly used in aviation, in [30] the authors apply it in health care, that showed that NASA TLX is a reliable and valid tool to measure workload among ICU nurses. Also, the NASA TLX is easy to administer and allows the researcher to measure different dimensions of workload and overall workload [30].

NASA TLX consists in six sub scales that represent somewhat independent clusters of variables: Mental, Physical, and Temporal Demands, Frustration, Effort, and Performance [25]. The assumption is that some combinations of these dimensions are likely to represent the workload experienced by most people performing most tasks [26] [25]. In [23] there is a figure adapted from [26] which contains the description of the six sub scales. This figure is reproduced in this dissertation for a better understanding of each dimension (4.5).

Table 4.5: NASA TLX rating-scale descriptions from [23]

Title	Endpoints	Descriptions
Mental Demand	Low, High	How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy or demanding, simple or complex, exacting or forgiving?
Physical Demand	Low, High	How much physical activity was required (e.g., pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
Temporal Demand	Low, High	How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
Performance	Good, Poor	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
Effort	Low, High	How hard did you have to work (mentally and physically) to accomplish your level of performance?
Frustration Level	Low, High	How insecure, discouraged, irritated, stressed, and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task? (NASA Task Load Index, p. 13)

In this dissertation we applied NASA TLX to measure participants workload during the first phase. The participants were asked to fill the questionnaire (see Annex I.1) at the end of the first and fourth day. In the first day, 76 participants filled this questionnaire, while in the last day of the first phase only 48 filled the TLX. The data were analysed using SPSS software[9] and it will be applied the American Psychological Association (APA) format[10] to report the results. As mentioned before there are six dimensions in NASA TLX: Mental Demand (MD), Physical Demand (PD), Temporal Demand (TD), Performance (P), Effort (E) and Frustration (F). To differentiate both time instances where the questionnaire was filled by the participants, the variables in the first time instance were named MD1, PD1, TD1, P1, E1 and F1, and in the second time instance were named MD2, PD2, TD2, P2, E2 and F2. To test whether these variables are normally distributed, it was used the Kolmogorov-Smirnov test. The result of this

[9] https://www.ibm.com/analytics/spss-statistics-software
[10] https://apastyle.apa.org/

test indicated that the following dimensions followed a normal distribution, with a confidence level of 95%:

- MD2 *(D(48)=0.119, p=0.085)*;
- TD2 *(D(48)=0.124, p=0.063)*;
- E2 *(D(48)=0.077, p=0.2)*.

The Kolmogorov-Smirnov test indicated that the remaining dimensions do not follow a normal distribution:

- MD1 *(D(76)=0.125, p=0.005)*;
- TD1 *(D(76)=0.171, p=0.0001)*;
- PD1 *(D(76)=0.221, p=0.0001)*;
- P1 *(D(76)=0.244, p=0.0001)*;
- E1 *(D(76)=0.120, p=0.009)*;
- F1 *(D(76)=0.237, p=0.0001)*;
- PD2 *(D(48)=0.229, p=0.0001)*;
- P2 *(D(48)=0.165, p=0.002)*;
- F2 *(D(48)=0.195, p=0.0001)*.

Considering the fact that some of these dimensions are not normally distributed, a non-parametric test was undertaken. The Wilcoxon test compared the two paired groups of the different time instances. Therefore, it was compared Mental, Physical and Temporal Demand, Performance, Effort and Frustration at the beginning and at the end of the first phase.

On average, the students performed better[11] at the beginning *(Mdn =3.00)* than at the end of the first phase *(Mdn = 4.50)* but a Wilcoxon signed-rank test indicated that this difference was not statistically significant, $T = 633$, $Z = -1.946$, $p = 0.052$. Regarding effort, on average the students showed more effort at the end *(Mdn =10.0)* than at the beginning of the first phase *(Mdn =7.0)*. The Wilcoxon signed-rank test indicated that this difference was statistically significant, $T = 650.5$, $Z = -3.241$, $p = 0.001$. Concerning the mental demand, on average the students were more mental actives at the end *(Mdn = 10.0)* than at the beginning of the first phase *(Mdn = 7.0)* and the Wilcoxon signed rank-test indicated that this difference was statistically significant, $T = 679.5$, $Z = -4.051$, $p < 0.001$. On average, the students showed a higher physical demand at the end *(Mdn =6.0)* than at the beginning of the first phase *(Mdn 5.0)*. The Wilcoxon signed-rank test indicated that this difference was not statistically significant, $T = 203.5$, $Z = -0.352$, $p = 0.725$. In terms of temporal demand, on average the students felt more pressure at the end *(Mdn = 8.0)* than at the beginning of the first phase *(Mdn = 6.0)* and a Wilcoxon signed-rank test indicated that this difference was not statistically significant, $T = 518.0$, $Z = -1.793$, $p = 0.073$. As regards the frustration, on average the students showed more frustration at the end *(Mdn = 4.0)* than at the beginning of the first phase *(Mdn = 2.0)*. The Wilcoxon signed-rank test indicated that this difference was statistically significant, $T = 639.5$, $Z = -3.492$, $p < 0.001$.

[11] Note: See Table 4.5 that shows that the performance scale is inverted.

4.4.2 First and second phase

In order to understand whether or not there were factors influencing the learning resilience in the *in-class MOOC* learning approach, the following null hypothesis with the respective factors were tested. "learning resilience of a new programming language does not depend on:"

- the previous mastering of other programming language(s);
- the undergraduate degree followed;
- the learning pace.

The first step was to test whether or not the dependent variable (learning resilience) followed a normal distribution. In Table 4.6 it is possible to observe that this assumption is not satisfied, meaning that the data do not follow a normal distribution $(W(76) = 0.931, p < 0.001)$ and the use of non-parametric tests is recommended.

Table 4.6: Learning resilience - normality test

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Learning_Resilience	,129	76	,003	,931	76	,000

a. Lilliefors Significance Correction

The Spearman's rank-order correlation is the non-parametric version of the Pearson product-moment correlation. Spearman's correlation coefficient (ρS) measures the strength and direction of association between two ranked variables. It is usually adopted when the assumption of the normal distribution is not tenable [5]. Also, in [16] the authors mentioned that the most popular non-parametric correlation measure is Spearman's rank-order correlation. In order to determine if there was any relationship between the learning resilience, which is the total dedication time learning through MOOCs in the first phase, and the previous mastering of other programming language(s) and the learning pace, a Spearman rank-order correlation test was conducted. The results of the Spearman correlation indicated that there was a very weak and positive correlation between learning resilience and the previous mastering of other programming language(s) ($\rho S(76) = 0.027, p=0.814$) and a weak and positive correlation between learning resilience and the learning pace ($\rho S(76) = 0.361, p=0.001$).

The Kruskal-Wallis test is the non-parametric analogue of a one-way anova, which does not make assumptions about normality. In [28], the author ended concluding that: *"For non-symmetrical distributions the non-parametrical Kruskal-Wallis test results in a higher power compared to the classical one-way anova. The results of the simulations show that an analysis of the data is needed before a test on differences in central tendencies is conducted. Although the literature and textbooks state that the F-test is robust under the violations of assumptions, these results show that the power suffers a significant decrease"*. Therefore, in order to compare the effect of the undergraduate degree followed on the learning resilience, a Kruskal-Wallis test was undertaken. The results of this test showed that there were no differences found among the three undergraduate

degrees *($\chi 2(2) = 0.561$, $p = 0.756$)*.
Despite the very weak and weak correlations, a Kruskal-Wallis test was undertaken as well to explore the effect of the previous mastering of other programming language(s) on the learning resilience and the effect of the learning pace on the learning resilience. Regarding the effect of the previous mastering of other programming language(s) on the learning resilience, the Kruskal-Wallis test showed no differences *($\chi 2(2) = 0.141$, $p = 0.932$)* among the three categories of this variable. Differently, the Kruskal-Wallis test showed that there was a statistically significant difference *($\chi 2(3) = 8.192$, $p = 0.042$)* in learning resilience between the four different groups of learning pace (fast, moderate, slow and very slow). *Post-hoc* Mann-Whitney tests using a Bonferroni-adjusted alpha level of .008[12] were used to compare all pairs groups. As it is possible to see in Table 4.7, none of the comparisons among the different groups were significant after the Bonferroni adjustment.

Table 4.7: *Post-hoc* Mann-Whitney test using Bonferroni adjustment

Pairwise Comparisons of LearningPace_Categories

Sample 1-Sample 2	Test Statistic	Std. Error	Std. Test Statistic	Sig.	Adj. Sig.[a]
Fast-Moderate	-7,966	6,604	-1,206	,228	1,000
Fast-Slow	-17,406	7,804	-2,231	,026	,154
Fast-Very Slow	-23,964	10,002	-2,396	,017	,099
Moderate-Slow	-9,440	6,604	-1,429	,153	,917
Moderate-Very Slow	-15,998	9,098	-1,759	,079	,472
Slow-Very Slow	-6,558	10,002	-,656	,512	1,000

To the extent of understanding which factors influenced or not the learning effectiveness, the contest points were analysed in SPSS as well as the following factors: learning mode, overall proficiency level and MOOC modules completed. Therefore, the null hypothesis to test was "learning effectiveness does not depend on":

- the learning mode (traditional vs. *In-class MOOC*;
- the overall proficiency level (average grades in degrees' programming courses);
- the number of MOOC modules completed.

A test of normality to the dependent variable was undertaken in order to understand whether or not the data followed a normal distribution. The Kolmogorov-Smirnov and Shapiro-Wilk test (Table 4.8) showed that these data do not follow a normal distribution (*$W(44) = 0.89$, $p = 0.001$*). With this in mind, it is recommended the appliance of non-parametric tests.

[12] To minimise the problem of obtaining an increase in false positive results, the 0.05 alpha threshold is divided by the number of tests intended to run

Table 4.8: Points - normality test

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Points_Adjusted	,141	44	,028	,890	44	,001

a. Lilliefors Significance Correction

Therefore, a Spearman rank-order correlation test was conducted in order to determine if there was any relationship between the learning effectiveness, measured by the points earned in the contest, and the number of MOOC modules completed in the first phase of the event. The results of the Spearman correlation indicated that there was a strong and positive correlation between points and modules completed ($\rho S(35) = 0.675$, $p<0.001$). The same test was used to assess the relationship between the learning effectiveness and the students' average grades in the programming courses taken in their undergraduate degrees. The results of the Spearman correlation indicated that there was moderate and positive correlation between the learning effectiveness and the average grades ($\rho S(44) = 0.438$, $p=0.003$).

Regarding the learning effectiveness among the two learning modes, it was compared the mean points of the two conditions (traditional and *In-class MOOC*) and the differences between them. The results presented in Table 4.9 and the Kruskal-Wallis test showed that there was no significant differences ($\chi 2(1) = 2.208$, $p = 0.137$) among the traditional ($M = 2.14$) and *In-class MOOC* ($M = 3.65$) conditions.

Table 4.9: Mean points of each learning mode

Points_Adjusted

Learning_Mode	Mean	N	Std. Deviation	Minimum	Maximum
In-Class MOOC	3,6509	35	2,95608	,00	9,80
Traditional	2,1462	9	2,15823	,00	5,75
Total	3,3431	44	2,85534	,00	9,80

Complementing the Spearman test results regarding the existing relationship between the contest points and the number of MOOC modules and the contest points and the average grades, it was undertaken a Kruskal-Wallis test to examine the differences between them. Regarding the contest points and the average grades, the Kruskal-Wallis test showed no differences ($\chi 2(2) = 5.154$, $p = 0.076$) among the three different categories of the average grades. Contrarily, the Kruskal-Wallis test showed that there was a statistically significant difference in the contest points between the different number of MOOC modules taken ($\chi 2(3) = 15.867$, $p = 0.001$). *Post-hoc* Mann-Whitney tests using a Bonferroni-adjusted alpha level of .008 were used to compare all pairs groups. The comparison between the groups who completed 1 module and 2 modules

was significant after Bonferroni adjustment *($p = 0.0029$)*. None of the other comparisons were significant after the Bonferroni adjustment, see Table 4.10.

Table 4.10: *Post-hoc* Mann-Whitney test using Bonferroni adjustment

Pairwise Comparisons of Modules_Completed

Sample 1-Sample 2	Test Statistic	Std. Error	Std. Test Statistic	Sig.	Adj. Sig.[a]
1-2	-13,178	3,686	-3,576	,000	,002
1-3	-16,731	7,778	-2,151	,031	,189
1-4	-21,731	10,626	-2,045	,041	,245
2-3	-3,553	7,612	-,467	,641	1,000
2-4	-8,553	10,506	-,814	,416	1,000
3-4	-5,000	12,541	-,399	,690	1,000

Chapter 5

Conclusion

Contents

This chapter summarizes the main contributions and presents the conclusions of this dissertation. Finally, presents the threats to the validity of the experiment and possible future works.

[This page has been intentionally left blank]

Chapter 5
Conclusion

5.1 Conclusions

This dissertation intended to mitigate the "Python gap" problem faced by the majority of the IT undergraduate students at Iscte. As explained in detail in Section 1.1, this problem arises with the increasing popularity of the programming language Python and its high demand on the job market. In Section 1.2, a learning approach is proposed to solve the problem above mentioned. This learning approach that we named *In-Class MOOC* was then used as a learning method in the event "Pythacon", where the students had the possibility to learn Python.

In Chapter 3, a step-wise method for selecting a MOOC course on Python was proposed to answer the first research question (see Section 1.3). This method included a set of criteria, which was applied to reach three final candidates: "Learn Python", "Programming for Everybody (Getting Started with Python)" and "Learn Python Programming Masterclass" from the Codecademy, Coursera and Udemy platforms, respectively. These three courses were graphically represented using BPMN and then discussed by an expert panel for reaching a consensus on the final choice. Therefore, those were the steps taken in this dissertation to answer the first research question (see Section 1.3).

The second research question of this dissertation aimed to understand whether or not there were factors influencing the learning resilience in the *In-Class MOOC* approach. Based on the results presented in Section 4.4, we can conclude that the previous mastering of other programming language(s) and the learning pace, despite weakly, influenced the learning resilience positively but that influence is not statistically significant. Furthermore, it is also possible to conclude that the undergraduate degree followed by the participants did not affect the learning resilience.

Regarding the third research question "Which factors influence learning effectiveness¿', it is possible to conclude that the number of MOOC modules done by the students directly influenced the number of points they had in the contest. Additionally, the students' grades were also related to the number of points they got in the contest. Observing Figure 5.1 and taking into account the results in Section 4.4, we conclude that there is a significant difference between finishing only the first module and finishing the first and the second module. Students who finished only the first module did not have good marks in the contest and the students who had good marks finished at least two modules. In Figure 5.2, we can affirm that to get a high number of points, the average grades had to be high too. However, the opposite to this is not necessarily true. In other words a good background was necessary but not sufficient condition for maximizing learning effectiveness, it is also required persistence in following more MOOC modules.

The fourth and last research question of this dissertation aimed to find out whether the *In-class MOOC* learning approach is as effective as the traditional learning approach. In Section 4.4, the results showed that the learning mode (traditional learning and *In-class MOOC*) did not affect the points obtained in the contest, meaning that regardless the learning mode, students

were on equal terms as regards the know-how in Python programming. With that being said, we can conclude that the proposed learning approach is as effective as the traditional learning approach.

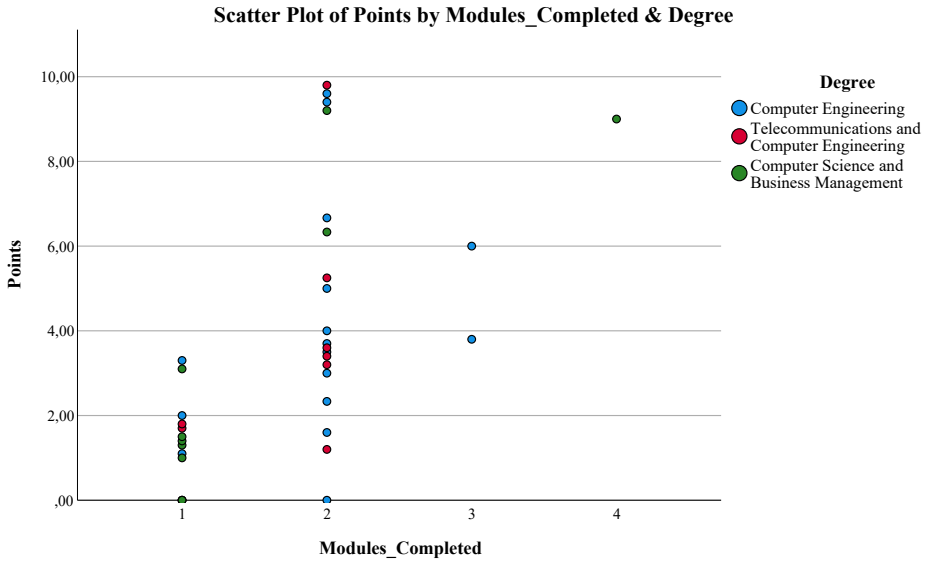

Figure 5.1: Scatter plot of points by MOOC modules completed and undergraduate degree

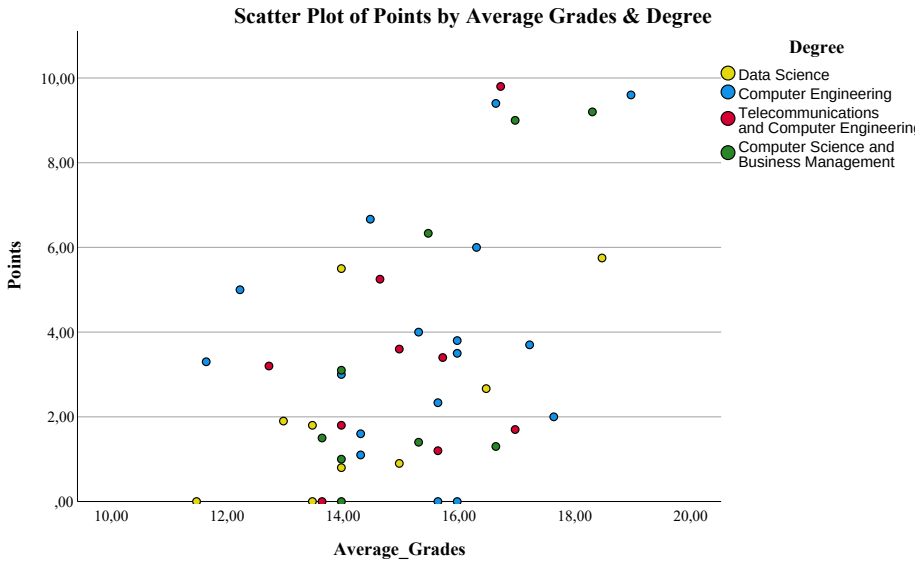

Figure 5.2: Scatter plot of points by average grades and undergraduate degree

5.2 Validity threats

Based on [47], this section dedicates to the identification and discussion of the validity threats found in the experiment conducted in this dissertation.

5.2.1 Internal threats

- **Social:** The students from the control group did not have the same treatment as the experimental group. As treatment, we mean the *In-Class MOOC* where the students from the experimental group had the opportunity to learn Python programming prior to the day of the competition. Since the students from the control group were in disadvantage, the compensation was the automatic qualification for the second phase (compensatory equalization of treatments). Additionally, the experimental group faced a pre-selection because they had to complete the first MOOC module to qualify for the contest[1], while the control group did not have to face any pre-selection because all the students from this group were automatically qualified for the contest.
- **Instrumentation:** In order to collect the data regarding the students' presence, dedication time to the MOOC and breaks during the 5 hour period of each day, the main objective was to use their students' card and the card reader available at the room doors. However this was not possible, so this information was collected manually on a worksheet (excel), meaning that it may have deviations from reality. Since, for instance, the time spent on coffee breaks was not registered.
- **Mortality:** In this experiment, the dropout in both groups was noticeable (see Figure 4.5), meaning that different types of students did not come to the contest day and this may have influenced the results obtained by each group.

5.2.2 Construct validity

- **Social:** The set of prizes offered to the best students in the contest may have influenced their behaviour (e.g. already had programming Python skills when registered in the event or carried an in-depth study of the Python programming language outside each 5 hour period in the first phase).

5.2.3 External threats

- **Selection:** The small number of subjects in both groups, as well as the discrepancy between them, prevented us from generalizing the results of the experiment to the population, which is the students at DCTI.
- **Evaluate Apprehension:** Some of the students could have not been comfortable with being assessed, which may have influenced their behaviour in the experiment (known as the Hawthorne effect [37]).

[1] All the students from the experimental group finished the first module, and as a consequence, qualified for the contest

5.3 Future work

The objectives defined for this dissertation were accomplished. However, there are some aspects which could be addressed in the future to extend this study:

- Undertake an experiment as the one presented in this dissertation with more subjects in both groups, as well as equivalence between them towards an understanding of whether or not the results obtained can be extrapolated to the population.
- The *In-Class MOOC's* integration in a programming course of the undergraduate degrees at DCTI. The main objective of this integration would be learning the programming concepts through the MOOC and then be assessed with a project development and a final test. After the integration and based on the results obtained, compare this learning approach with the previous traditional approach. Additionally, undertake an interview/questionnaire to obtain the students' feedback regarding their experience, the negative and positive points and suggestions to improve in the future.
- Conduct an experiment with a small number of subjects in order to measure their cognitive effort in MOOCs using an EEG[2] headset and an eye-tracking device.

[2] Electroencephalography

Bibliography

[1] K. Abuhlfaia and E. d. Quincey. "The usability of E-learning platforms in higher education: a systematic mapping study." In: *Proceedings of the 32nd International BCS Human Computer Interaction Conference 32*. 2018, pp. 1–13. DOI: 10.14236/ewic/HCI2018.7.

[2] S. Alhazbi. "Using flipped classroom approach to teach computer programming." In: *Proceedings of 2016 IEEE International Conference on Teaching, Assessment and Learning for Engineering, TALE 2016*. Institute of Electrical and Electronics Engineers Inc., Feb. 2017, pp. 441–444. ISBN: 9781509055982. DOI: 10.1109/tale.2016.7851837.

[3] S. An, W. Li, J. Hu, L. Ma, and J. Xu. "Research on the reform of flipped classroom in computer science of university based on SPOC." In: *ICCSE 2017 - 12th International Conference on Computer Science and Education*. Institute of Electrical and Electronics Engineers Inc., Oct. 2017, pp. 621–625. ISBN: 9781509025084. DOI: 10.1109/iccse.2017.8085567.

[4] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. "Engaging with massive online courses." In: *WWW 2014 - Proceedings of the 23rd International Conference on World Wide Web*. New York, New York, USA: Association for Computing Machinery, Inc, Apr. 2014, pp. 687–697. ISBN: 9781450327442. DOI: 10.1145/2566486.2568042. arXiv: 1403.3100.

[5] R. Artusi, P. Verderio, and E. Marubini. "Bravais-Pearson and Spearman Correlation Coefficients: Meaning, Test of Hypothesis and Confidence Interval." In: *The International Journal of Biological Markers* 17.2 (Apr. 2002), pp. 148–151. ISSN: 1724-6008. DOI: 10.1177/172460080201700213.

[6] S. Beecham, T. Hall, C. Britton, M. Cottee, and A. Rainer. "Using an expert panel to validate a requirements process improvement model." In: *Journal of Systems and Software* 76.3 (June 2005), pp. 251–275. ISSN: 01641212. DOI: 10.1016/j.jss.2004.06.004.

[7] A. Bralić and B. Divjak. "Use of moocs in traditional classroom: blended learning approach." In: *Forging new pathways of research and innovation in open and distance learning* 34 (2016).

[8] B. Cartaxo, G. Pinto, and S. Soares. "The Role of Rapid Reviews in Supporting Decision-Making in Software Engineering Practice." In: *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*. Ease'18. Christchurch, New Zealand: Association for Computing Machinery, 2018, 24–34. ISBN: 9781450364034. DOI: 10.1145/3210459.3210462.

[9] B. Cartaxo, G. Pinto, and S. Soares. "Rapid Reviews in Software Engineering." In: *Contemporary Empirical Methods in Software Engineering*. Ed. by M. Felderer and G. H. Travassos. Cham: Springer International Publishing, 2020, pp. 357–384. ISBN: 978-3-030-32489-6. DOI: 10.1007/978-3-030-32489-6_13.

[10] J. M. Case and H. Heydenrych. "Trade-offs in curriculum design: Implementation of an integrated curriculum in chemical engineering." In: *2018 IEEE Frontiers in Education Conference (FIE)*. Ieee. 2018, pp. 1–9. DOI: 10.1109/fie.2018.8658519.

[11] L. Chamberlin and T. Parish. "MOOCs: Massive open online courses or massive and often obtuse courses?" In: *ELearn* 2011.8 (2011). DOI: 10.1145/2016016.2016017.

[12] A. Chauhan. "Massive open online courses (MOOCS): Emerging trends in assessment and accreditation." In: *Digital Education Review* 25 (2014), pp. 7–17. DOI: 10.1344/der.2014.25.7-17.

[13] G. Cheng and W. S. Ng. "Secondary students' views on using flipped classroom to learn computer programming: Lessons learned in a mixed methods study." In: *Communications in Computer and Information Science*. Vol. 1048. Springer Verlag, Mar. 2019, pp. 27–36. ISBN: 9789811398940. DOI: 10.1007/978-981-13-9895-7_3.

[14] D. Coetzee, S. Lim, A. Fox, B. Hartmann, and M. A. Hearst. "Structuring interactions for large-scale synchronous peer learning." In: *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. 2015, pp. 1139–1152. DOI: 10.1145/2675133.2675251.

[15] N. Croft, A. Dalton, and M. Grant. "Overcoming isolation in distance learning: Building a learning community through time and space." In: *Journal for Education in the Built Environment* 5.1 (2010), pp. 27–64. DOI: 10.11120/jebe.2010.05010027.

[16] C. Croux and C. Dehon. "Influence functions of the Spearman and Kendall correlation measures." In: *Statistical Methods and Applications* 19.4 (2010), pp. 497–515. ISSN: 16182510. DOI: 10.1007/s10260-010-0142-z.

[17] J. Daniel. "Making sense of MOOCs: Musings in a maze of myth, paradox and possibility." In: *Journal of interactive Media in education* 2012.3 (2012). DOI: 10.5334/2012-18.

[18] D. Davis, G. Chen, T. Van der Zee, C. Hauff, and G.-J. Houben. "Retrieval practice and study planning in MOOCs: Exploring classroom-based self-regulated learning strategies at scale." In: *European conference on technology enhanced learning*. Springer. 2016, pp. 57–71. DOI: 10.1007/978-3-319-45153-4_5.

[19] T. Dybå. "Instrument for measuring the key factors of success in software process improvement." In: *Empirical Software Engineering* 5.4 (Dec. 2000), pp. 357–390. ISSN: 13823256. DOI: 10.1023/a:1009800404137.

[20] K. El Emam and N. H. Madhavji. "An instrument for measuring the success of the requirements engineering process in information systems development." In: *Empirical Software Engineering* 1.3 (1996), pp. 201–240. ISSN: 13823256. DOI: 10.1007/bf00127446.

[21] T. Eriksson, T. Adawi, and C. Stöhr. "Time is the bottleneck: a qualitative study exploring why learners drop out of MOOCs." In: *Journal of Computing in Higher Education* 29.1 (2017), pp. 133–146. DOI: 10.1007/s12528-016-9127-8.

[22] D. Gašević, V. Kovanović, S. Joksimović, and G. Siemens. "Where is research on massive open online courses headed? A data analysis of the MOOC research initiative." In: *International Review of Research in Open and Distance Learning* 15.5 (2014), pp. 134–176. ISSN: 14923831. DOI: 10.19173/irrodl.v15i5.1954.

[23] V. J. Gawron. *Human performance measures handbook*. Lawrence Erlbaum Associates Publishers, 2000.

[24] B. E. Guajardo Leal, C. Navarro-Corona, and J. R. Valenzuela González. "Systematic mapping study of academic engagement in MOOC." In: *International Review of Research in Open and Distributed Learning* 20.2 (2019).

[25] S. G. Hart. "Nasa-Task Load Index (NASA-TLX); 20 Years Later." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50.9 (Oct. 2006), pp. 904–908. ISSN: 1541-9312. DOI: 10.1177/154193120605000909.

[26] S. G. Hart and L. E. Staveland. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research." In: *Advances in Psychology* 52.C (Jan. 1988), pp. 139–183. ISSN: 01664115. DOI: 10.1016/s0166-4115(08)62386-9.

[27] A. Hassani and S. A. Ghanouchi. "Modeling of a collaborative learning process in the context of MOOCs." In: *Proceedings - 2016 3rd International Conference on Systems of Collaboration, SysCo 2016*. Institute of Electrical and Electronics Engineers Inc., Jan. 2017. ISBN: 9781509049264. DOI: 10.1109/sysco.2016.7831336.

[28] T. V. Hecke. "Power study of anova versus Kruskal-Wallis test." In: *Journal of Statistics and Management Systems* 15.2-3 (May 2012), pp. 241–247. ISSN: 0972-0510. DOI: 10.1080/09720510.2012.10701623.

[29] Y. Hirata and Y. Hirata. "Flipped Classroom Approaches in Computer Programming Courses in Japan." In: *Proceedings - 2020 International Symposium on Educational Technology, ISET 2020*. Institute of Electrical and Electronics Engineers Inc., Aug. 2020, pp. 109–113. ISBN: 9781728171883. DOI: 10.1109/iset49818.2020.00032.

[30] P. Hoonakker, P. Carayon, A. P. Gurses, R. Brown, A. Khunlertkit, K. McGuire, and J. M. Walker. "Measuring workload of ICU nurses with a questionnaire survey: the NASA Task Load Index (TLX)." In: *IIE Transactions on Healthcare Systems Engineering* 1.2 (Apr. 2011), pp. 131–143. ISSN: 1948-8300. DOI: 10.1080/19488300.2011.609524.

[31] V. Johri and S. Bansal. "Identifying Trends in Technologies and Programming Languages Using Topic Modeling." In: *Proceedings - 12th IEEE International Conference on Semantic Computing, ICSC 2018*. Vol. 2018-Janua. Institute of Electrical and Electronics Engineers Inc., Apr. 2018, pp. 391–396. ISBN: 9781538644072. DOI: 10.1109/icsc.2018.00078.

[32] S. Khangura, K. Konnyu, R. Cushman, J. Grimshaw, and D. Moher. "Evidence summaries: The evolution of a rapid review approach." In: *Systematic Reviews* 1.1 (Feb. 2012), pp. 1–9. ISSN: 20464053. DOI: 10.1186/2046-4053-1-10.

[33] J. Leinonen, P. Ihantola, A. Leinonen, H. Nygren, J. Kurhila, M. Luukkainen, and A. Hellas. "Admitting students through an open online course in programming: A Multi-year Analysis of Study Success." In: *Proceedings of the 2019 ACM Conference on International Computing Education Research (ICER 2019)*. New York, NY, USA: Association for Computing Machinery, Inc, July 2019, pp. 279–287. ISBN: 9781450361859. DOI: 10.1145/3291279.3339417.

[34] M. Li and C. S. Smidts. "A ranking of software engineering measures based on expert opinion." In: *IEEE Transactions on Software Engineering* 29.9 (Sept. 2003), pp. 811–824. ISSN: 00985589. DOI: 10.1109/tse.2003.1232286.

[35] M. Liu, L. Wang, Q. Chi, L. Wang, and Q. Liu. "Research on Flipped Classroom about Java Programming Course Based on MOOC." In: *Proceedings of the 3rd International Conference on Education & Education Research (EDUER 2018)*. Uk: Francis Academic Press, 2018, pp. 296–300. DOI: 10.25236/eduer.18.065.

[36] R. Mcgreal, W. Kinuthia, and S. Marshall. *Open Educational Resources: Innovation, Research and Practice*. Vancouver, Canada: Commonwealth of Learning and Athabasca University, 2013. ISBN: 9788188770267.

[37] F. Merrett. "Reflections on the Hawthorne effect." In: *Educational Psychology* 26.1 (2006), pp. 143–146. DOI: 10.1080/01443410500341080.

[38] C. Milligan and A. Littlejohn. "Why study on a MOOC? The motives of students and professionals." In: *International Review of Research in Open and Distance Learning* 18.2 (2017), pp. 92–102. ISSN: 14923831. DOI: 10.19173/irrodl.v18i2.3033.

[39] H. Mok. "Teaching tip: The flipped classroom." In: *Journal of Information Systems Education* 25.1 (2014), pp. 7–11. ISSN: 1055-3096.

[40] H. N. Mok and V. R. Rao. "Introducing basic programming to pre-university students: A successful initiative in Singapore." In: *2018 17th International Conference on Information Technology Based Higher Education and Training, ITHET 2018*. Institute of Electrical and Electronics Engineers Inc., Aug. 2018. ISBN: 9781538646236. DOI: 10.1109/ithet.2018.8424783.

[41] I. Nawrot and A. Doucet. "Building engagement for MOOC students: introducing support for time management on online learning platforms." In: *Proceedings of the 23rd International Conference on world wide web*. 2014, pp. 1077–1082. DOI: 10.1145/2567948.2580054.

[42] T. Ray, A. Malapati, and N. L. Murthy. "Teaching Computer Programming Using MOOCs in Multiple Campuses: Challenges and Solutions." In: *Proceedings - IEEE 8th International Conference on Technology for Education, T4E 2016*. Institute of Electrical and Electronics Engineers Inc., Jan. 2017, pp. 160–163. ISBN: 9781509061150. DOI: 10.1109/t4e.2016.041.

[43] B. Riyami and A. Bouaine. "MOOC's Integration approach: Assessment and comparative studies of all moroccan universities." In: *ACM International Conference Proceeding Series*. Association for Computing Machinery, July 2020, pp. 11–15. ISBN: 9781450375757. DOI: 10.1145/3411681.3411693.

[44] M. Sade and R. Suviste. "Using a programming MOOC as an admission mechanism for CS." In: *Proceedings - IEEE 20th International Conference on Advanced Learning Technologies, ICALT 2020*. Institute of Electrical and Electronics Engineers Inc., July 2020, pp. 42–44. ISBN: 9781728160900. DOI: 10.1109/icalt49669.2020.00019.

[45] M. D. Sakhumuzi and O. K. Emmanuel. "Student perception of the contribution of Hackathon and collaborative learning approach on computer programming pass rate." In: *2017 Conference on Information Communication Technology and Society, ICTAS 2017 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., May 2017. ISBN: 9781509059959. DOI: 10.1109/ictas.2017.7920524.

[46] J. Samuelsen and M. Khalil. "Study Effort and Student Success: A MOOC Case Study." In: *Advances in Intelligent Systems and Computing*. Vol. 916. Springer Verlag, 2020, pp. 215–226. ISBN: 9783030119317. DOI: 10.1007/978-3-030-11932-4_22.

[47] M. K. Slack and J. L. R. Draugalis. "Establishing the internal and external validity of experimental studies." In: *American Journal of Health-System Pharmacy* 58.22 (Nov. 2001), pp. 2173–2184. ISSN: 10792082. DOI: 10.1093/ajhp/58.22.2173.

[48] M. Stigmar. "Peer-to-peer teaching in higher education: A critical literature review." In: *Mentoring & Tutoring: partnership in learning* 24.2 (2016), pp. 124–136. DOI: 10.1080/13611267.2016.1178963.

[49] J. C. de Winter. "Controversy in human factors constructs and the explosive use of the NASA-TLX: A measurement perspective." In: *Cognition, Technology and Work* 16.3 (May 2014), pp. 289–297. ISSN: 14355566. DOI: 10.1007/s10111-014-0275-1.

[50] G. Witthaus, A. Inamorato dos Santos, M. Childs, A. Tannhauser, G. Conole, B. Nkuyubwatsi, and Y. Punie. "Validation of non-formal MOOC-based learning: An analysis of assessment and recognition practices in Europe (OpenCred)." In: *JRC Science for Policy Report* (2016). ISSN: 1831-9424. DOI: 10.2791/809371.

[51] M. Yağcı. "Web-Mediated Problem-Based Learning and Computer Programming: Effects of Study Approach on Academic Achievement and Attitude." In: *Journal of Educational Computing Research* 56.2 (Apr. 2018), pp. 272–292. ISSN: 15414140. DOI: 10.1177/0735633117706908.

[This page has been intentionally left blank]

APPENDIX A.

SELECTING A MOOC FOR PYTHON

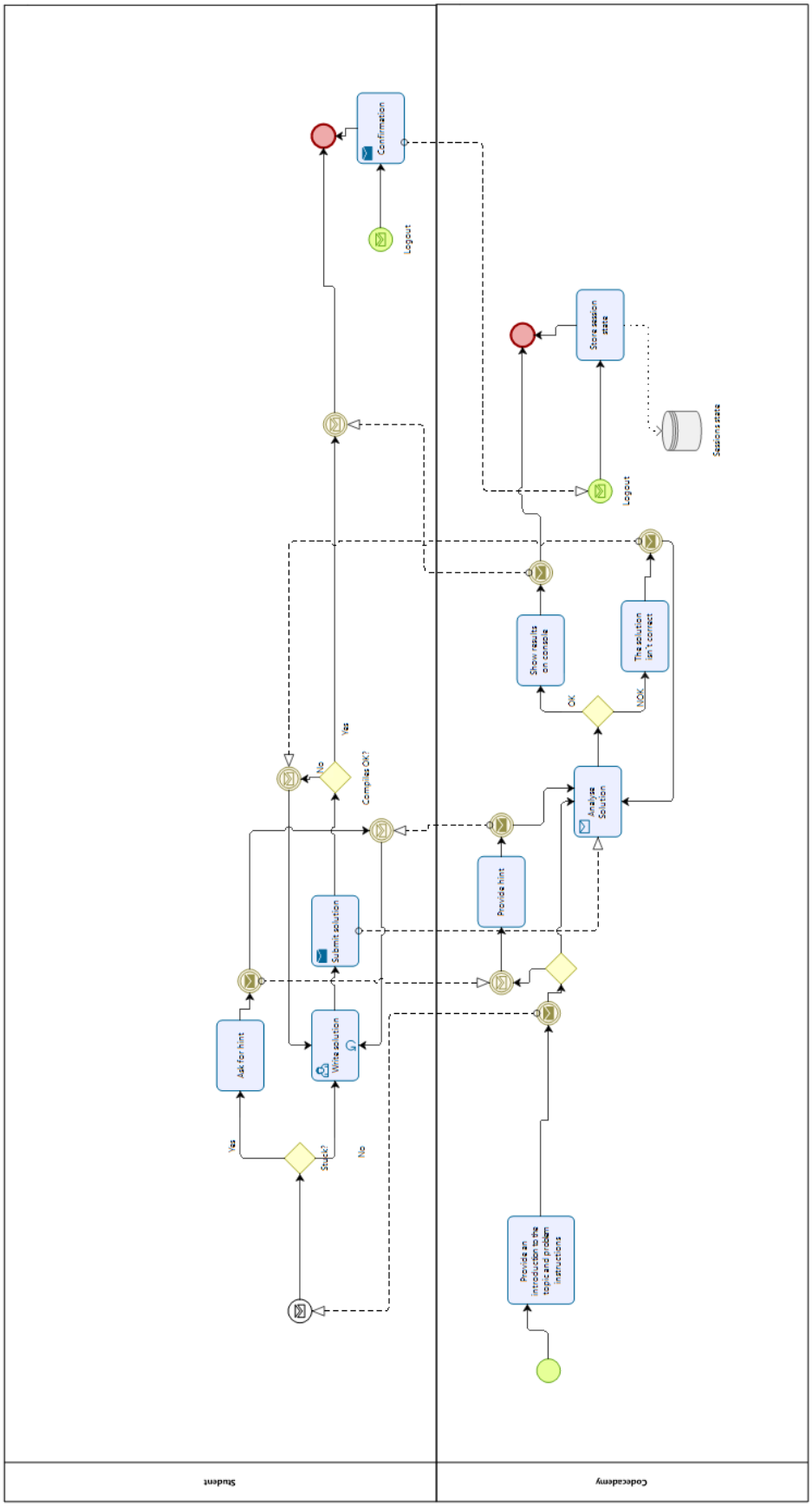

Figure A.1: Codecademy - "Do Section" process diagram

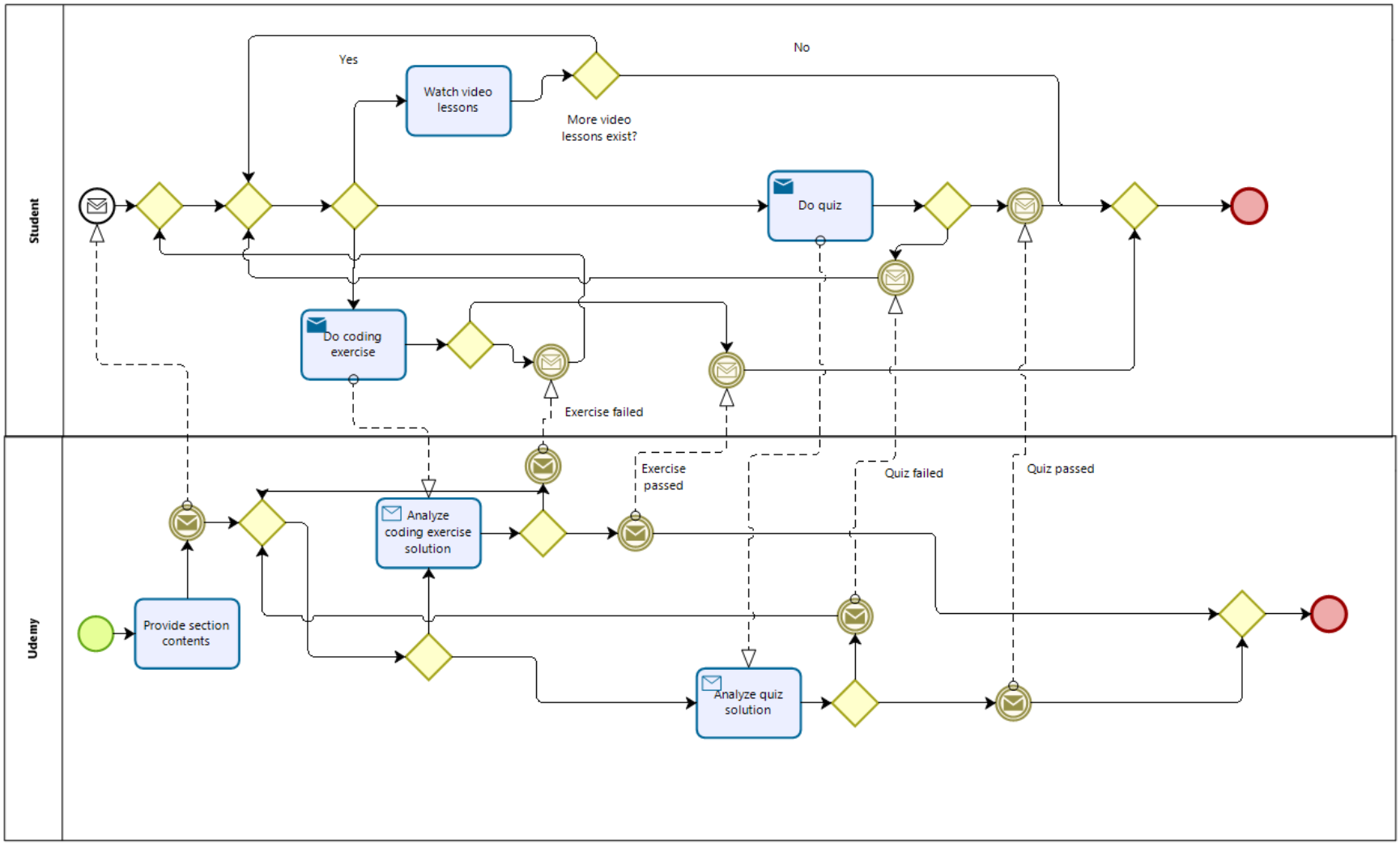

Figure A.2: Udemy - “Do Section“ process diagram

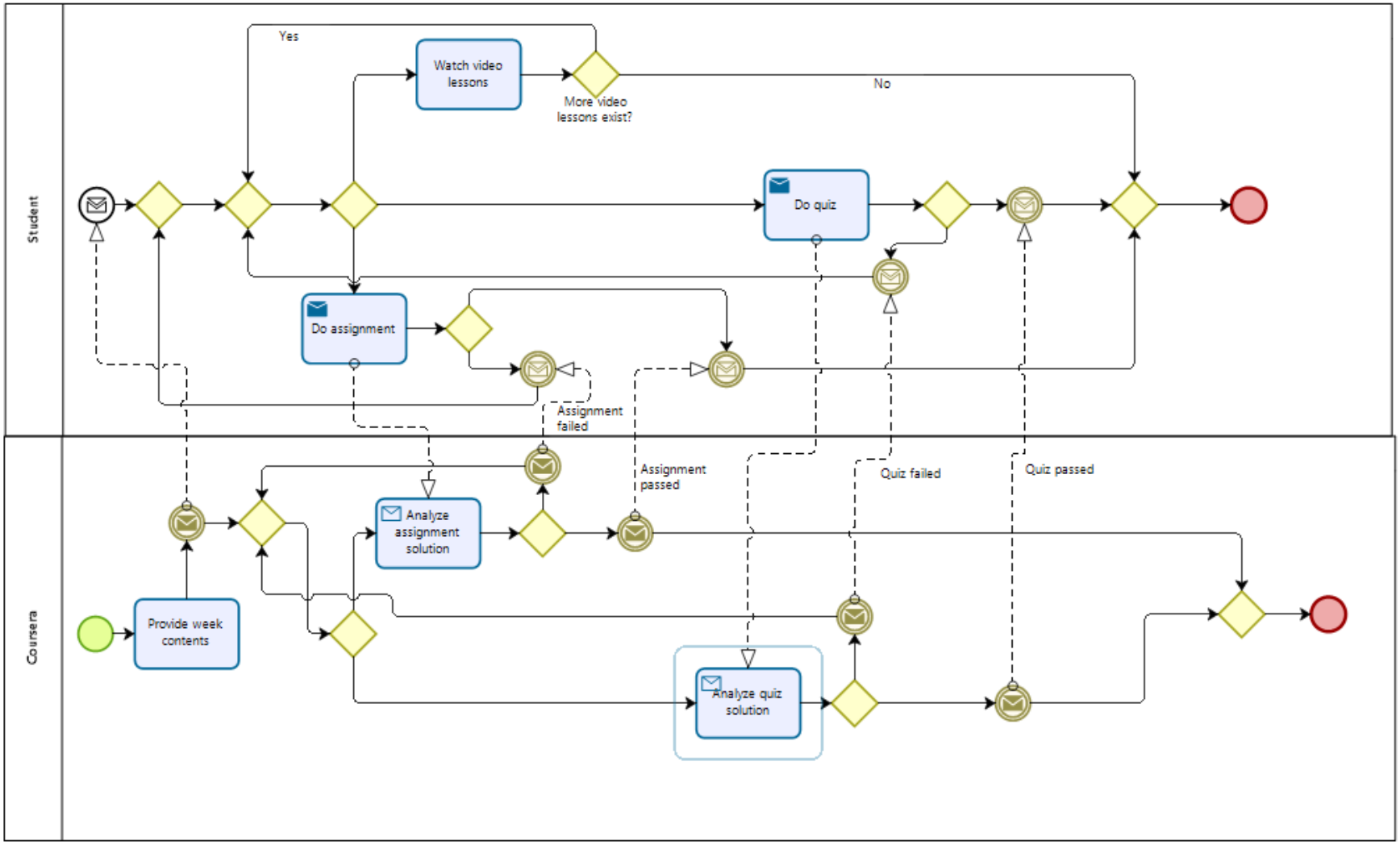

Figure A.3: Codecademy - process diagram

Default Question Block

Learning Process and Evaluation:

- ◯ Video Lesson, code exercises and quiz
- ◯ Text to explain the concepts and coding

Peer assessment (assignments are assessed by other course participants):

	Not important at all	Low importance	Moderate importance	Important	Very important
	◯	◯	◯	◯	◯

Course order:

- ◯ Weak order - A syllabus topics order is suggested, but students can configure their own path
- ◯ Compulsory order - Syllabus topics must be followed by the proposed order

Syllabus:

- ☐ Python Syntax
- ☐ Variables and Types
- ☐ Strings
- ☐ Program Control Flow (If, for, ...)
- ☐ Lists
- ☐ Functions
- ☐ Dictionaries and Sets
- ☐ Input and Output
- ☐ Using Databases
- ☐ Binary Number System
- ☐ Object Oriented Paradigm

Possibility of re-attempting an assignment:

	Not important at all	Low importance	Moderate importance	Important	Very important
	◯	◯	◯	◯	◯

Provided by a University:

	Not important at all	Low importance	Moderate importance	Important	Very important
	◯	◯	◯	◯	◯

Certificate:

	Not important at all	Low importance	Moderate importance	Important	Very important
	◯	◯	◯	◯	◯

Figure A.4: Expert panel questionnaire - 1

Based on your opinion, rank these three courses by preference order:

- Course X
- Course Y
- Course Z

Figure A.5: Expert panel questionnaire - 2

ANNEX I.

NASA Task Load Index

NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Name	Task	Date

Mental Demand — How mentally demanding was the task?

Very Low — Very High

Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?

Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?

Very Low — Very High

Figure I.1: NASA Task Load Index

iscte UNIVERSITY INSTITUTE OF LISBON

Learning to code in class with MOOCs: process, factors and outcomes

João Gomes