Journal of Education and Development; Vol. 7, No. 1; February, 2023
ISSN 2529-7996 E-ISSN 2591-7250
Published by July Press

Expert Validation of a Python Test, Reliability, Difficulty and Discrimination Indices

Héctor Manuel Belmar Garrido[1,2]

[1] School of Computer Engineering, Ñuñoa Campus, INACAP, Santiago, Chile

[2] Dr. in Education, Universidad Metropolitana de Ciencias de la Educación, Santiago, Chile

Correspondence: Héctor Manuel Belmar Garrido, School of Computer Engineering, Ñuñoa Campus, INACAP, Santiago, Chile. E-mail: hector.belmar@hotmail.com

Received: January 7, 2023 Accepted: February 4, 2023 Online Published: February 17, 2023

doi:10.20849/jed.v7i1.1320 URL: https://doi.org/10.20849/jed.v7i1.1320

Abstract

In recent years, different countries have implemented the teaching of computer programming from the first grade in schools, with the aim of incorporating computational thinking as a new way of thinking that is a necessary skill for scientific and technological development, a fundamental axis for development in the 21st century. Recent reviews of the state of the art have shown that this task is only being carried out by countries that have given space to technology and science, incorporating it at the elementary school level. However, developing countries do not yet consider the significance of the issue, so to date they have not taken the necessary steps in this direction. In addition, learning computer programming is fundamental for countries to join technological development, so that they can be creators of technology and not just users of it. The problem is that there is no didactic development for the teaching of programming, nor validated evaluation instruments to quantify the learning of computer programming. The objective of this research is to validate a programming test that evaluates technical skills in the Python programming language. The instrument proposed to be validated is a 90-item test, which after validation by experts was reduced to 70 items, and after the psychometric analysis that considered the calculation of reliability, difficulty and discrimination indexes, resulted in the proposed 45-item instrument, as a standard instrument for the evaluation of learning in the Python programming language. It should be noted that the validation methodology was carried out using the classical theory of psychometric analysis.

Keywords: computer programming, algorithm, programming didactics, evaluation instrument

1. Introduction

Computing since its beginnings in the 20th century with the theoretical conceptualization by Alan Turing of a machine capable of making calculations and with memory to store data, has led us in a spiral of technological advances, which has evolved exponentially. The Englishman Alan Turing was the founder of Computer Science, he was a mathematician, philosopher and cryptographer, a visionary of his time. Turing lived from [1912 to 1954]. It should be noted that, without that initial idea and the construction of the first computers as electromechanical machines, computing would not be what we know today (Kulkarni, 2015).

In this context, computer science has evolved during the second half of the twentieth century, starting with computer programming in assembler through procedural languages to focus at the beginning of the twenty-first century on Object Oriented programming and code generators. Currently there are several initiatives in the world to incorporate programming in schools, one of them is the Bebras challenge, which is aimed at motivating, practicing and evaluating the skill levels of students at an extracurricular level. Along with the Bebras challenge is the initiative called Code.org, which provides free material to anyone who requests it, in order to facilitate access to computer programming teaching material. But let's see what each of these initiatives is.

The Bebras challenge is a community computer education network that was born in Lithuania in 2005 and has been consolidated in more than 40 countries to discuss computer science concepts for school computer education. The Bebras Algorithm Development Workshops, which have been organized annually since its inception, bring together representatives from all these countries for rigorous work and decision making on good tasks to promote computer science education in primary and secondary schools. On the one hand, the Bebras challenge is an international assembly driven to respond to the needs of computer education worldwide, and on the other hand, almost all activities are nationally based, organized by communities in the participating countries. Bebras is an attractive way to promote computer science learning worldwide. It has a community-based, distributed organization, which makes it a democratic organization, as everything comes from the bottom up. The workshops function as extracurricular activities so participants do so voluntarily and based on their motivation to learn (Dagiene & Stupuriene, 2016).

In addition, there are several quality aspects in the Bebras challenge: first, tasks (development of criteria for good tasks, visualization, correctness, etc.), second, online quiz management systems (effective, easy to use, flexible), third, importance of learning (computer science concepts), among others. The classification of tasks is what is proposed in the research. The proposed topics started in 2006 with certain basic categories and in 2008 Bebras' task categories were revised and a modified classification was published, which included the categories; understanding information, algorithmic thinking, computer system structures, patterns and arrangements, puzzles, social, ethical, cultural, international and legal issues, whose categories are maintained to date of publication (Dagiene & Stupuriene, 2016).

Code.org is a non-profit organization dedicated to expanding access to computer programming in schools and increasing the participation of all who need and want to learn computer programming. Code.org's vision is that every student in every school has the opportunity to learn to program as part of their elementary and secondary education. Code.org, the leading provider of school computer science curriculum to the largest school districts in the United States, also created the annual "Hour of Code" campaign, which has engaged more than 15% of all students worldwide. This curriculum at Code.org makes connections between computer science learning and traditional subjects such as mathematics, language, science, and social science (Kale & Yuan, 2021).

At present, there are several problems in implementing the teaching of computer programming in schools, which begin with the political discussion of the countries and the allocation of sufficient budget to equip the educational system with the appropriate number of computers in laboratories suitable for teaching. After overcoming the issue of computer infrastructure, an additional problem arises, which is the availability of a sufficient number of teachers properly trained to teach programming. In this sense, the political discussion will begin when a diagnosis is made about the implications of the advance of technology and how it could displace a significant percentage of workers (due to the automation of work) and with it an increase in unemployment that will imply socio-political instability, which will shake the economy and governments that are not properly prepared for these changes. Once the first barrier is overcome, the implementation will come, which would imply an operational diagnosis, how many trained teachers exist, how they are distributed in the country, also regarding the digital infrastructure, laboratories, software, etc.

In addition to the above, there is the lack of a didactics of programming to guide teachers on what to teach and at what level to teach it, since there are 12 levels in which the teaching of programming should be distributed, and there is also the lack of tests and ways to evaluate knowledge in programming. The issue of didactics for the teaching of programming is not a minor issue, since so far one of the big problems of teaching programming in the first course of computer programming at the entrance to higher education is that those who teach it do it as they have learned or as they were trained, Therefore, novices must overcome the difficulty of learning a new subject with the lack of an andragogical character of teaching, together with the inexistence of a didactic of programming, as it exists in mathematics, language or science, and the inexistence of properly validated evaluation instruments. In this sense, the present publication proposes a 45-question test validated as an effective evaluation tool to measure the knowledge achieved by students in the Python language.

2. Theoretical Framework

2.1 Computer Programming and Computational Thinking

While it is true that this research aims to perform a psychometric analysis of a computer programming evaluation test in Python, it is also true that the concept of "Computational Thinking" has taken some prominence whenever the topic of computer programming is addressed. So, what is the link between both concepts? What is happening is that the world is incorporating the teaching of computer programming for elementary and high school students, in order to develop transversal skills, which go far beyond learning to program and what it may mean for the professional training of each subject in the future. Thus the teaching of computational programming serves as a scaffold for the development of computational thinking skills, which are transferable to other areas of knowledge, not only remain in the scholars of computer science, but also radiate to the whole area of science and technology (STEM), also includes art (STEAM) and beyond (Rojas & Garc á, 2020).

The contributions of computational thinking in education are very broad. Thus universities around the world are revising their undergraduate computer science curricula, as a result of which they are changing their first course in computer science to cover fundamental concepts, not just programming. In addition, interest around computational thinking has grown beyond undergraduate education, with many focusing on incorporating computational thinking into K-12 education. Computer scientists know the value of thinking abstractly, thinking at multiple levels of abstraction, abstracting ideas to manage complexity, abstracting to scale, iteration, debugging, and software testing, among others. Our immediate task, says Wing, is to better explain to non-computational scientists what we mean by computational thinking and the benefits associated with this thinking skill (Wing, 2006, 2008, 2011).

For Garc á (2018), information technologies are the beginning in the construction of the digital infrastructure that will move the world of the 21st century. In this sense, education is impacted by digital technology. Given this situation, from which we cannot abstract, schools must take measures with our young people to operate in a virtual world, for which they must be prepared in the language of this century, without which they will become digitally illiterate. Therefore, the school should train the youth with the skills of computational thinking, since what the present century demands is to acquire the skills of computer science, to live a new way of thinking and solving problems, so the current challenge is to prepare our young people to succeed in the digital world, that is, instead of teaching students to be users of a changing technology, they should be trained and trained in the new paradigm of computational thinking, so as to be creators of new technologies (Garcia, 2018).

Complementing the above, researchers Jacob and Warschauer from the University of California, published a work with which they seek to influence the definitions and scope of computational thinking, considering that people who enter the workforce today, do so in a world that works with computing and computerization of productive, educational and service processes, so that to succeed in the changes of the productive apparatus, students must learn to think algorithmically and computationally, to solve problems with different levels of abstraction, skills that have been integrated into the social fabric and our lives. However, computational thinking has not been taught in high schools. Efforts to create computer science standards and frameworks have yet to make their way as a requirement in mandatory courses in schools (Jacob & Warschauer, 2018).

Furthermore, Jacob and Warschauer (2018), note that computational thinking is the critical skill of the 21st century. While in education policy initiatives have focused on assessing literacy by measuring discrete reading and writing skills, economics continues to value computational skills that solve problems across a wide variety of disciplines. Although computer science initiatives take time to impact policy, integrating computational thinking with current literacy practices leverages students' existing literacy skills to improve computational outcomes and, conversely, fosters students' literacy development through computational practice. Teaching computational thinking beginning in the elementary grades will enable students to become developers and creators of new technologies in a natural way (Jacob and Warschauer, 2018).

From the reviews and articles read, the contribution made by Belmar (2022) stands out in his work entitled "Review on the teaching of programming and computational thinking in the world", in whose work he incorporates the concept of modeling and simulation as foundational skills of computational thinking, so that it includes formulating problems, organizing and analyzing data logically, representing data through models and automating solutions. Thus, looking towards school training, these skills improve attitudes such as: confidence in dealing with complexity, persistence in working with difficult problems, tolerance for ambiguity, the ability to deal with open-ended problems, and the ability to work with others to achieve a common goal and communicate it. Which is complemented by Grgurina (2021) who describes computational thinking in terms of its main concepts, such as: data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, modeling, and simulation Belmar (2022).

When comparing the progress of countries in the teaching of computational thinking in schools, says Belmar(2022), it is necessary to see the progress of states in public policies that establish this subject in the mandatory curriculum, as is the case of England in 2013 and the countries of Europe in general since 2016, or other Asian countries such as Japan, South Korea and China where computational thinking is defined as the skills of the 21st century and will be the engine of technological and economic development. However, in developing countries, such as Latin America and Africa, although there are several initiatives of universities in doing research on the teaching of computational thinking, the states have other priorities, so the issue is not in the discussion of public policies, but rather the development of the teaching of computational thinking is done by particular initiatives of universities and some other foundation, which leaves them behind in the race to create and develop new technologies to deliver better employment alternatives and thus greater welfare to the population Belmar (2022).

3. Materials and Methodology

The materials for this work include a personal computer with Internet connection, which is at least equipped with Office software and SPSS (Statistical Package for the Social Sciences) software.

3.1 Methodology

The methodology of the study corresponds to a quasi-experimental design and is based on psychometric analysis based on classical test theory. The quasi-experiment consists of validating a test, which comprises; the elaboration, application and validation of a multiple-choice test that covers all the relevant dimensions to measure the knowledge of the Python language, the opinion of 3 experts on the test, to validate the questions and elaborate the pilot test that was taken to computer science students in the second semester of the year 2021 in a Chilean University, with whose results the definitive test proposed was elaborated.

For the expert validation, the Hernández-Nieto theory is used (Hernández-Nieto, 2002 in Pedraza et al., 2013). For the calculation of reliability (Cronbach's alpha) the SPSS software version 22 of IBM was used and to calculate the difficulty and discrimination indexes we worked with some of the authors of the classical test theory (Pedraza et al., 2013).

Design guidelines and principles

- Objective: The test aims to measure the knowledge of Python programming in a student of professional technical education who have done their 1st semester.

- Construct definition: Python programming, implies the ability to solve problems based on the concepts of computing and using the logical syntax of the language: basic sequences, loops, iteration, conditionals, functions, variables and data structures such as arrays, tupas and dictionaries.

- Population: The test is designed for students who have taken the introductory programming course in vocational technical education.

- Type of test: multiple choice test with 4 answer options (only one correct).

- Length: 45 items.

- Estimated completion time: 80 minutes.

3.2 Population and Sample of the Pilot Study

The sampling procedure will be non-probabilistic; specifically, convenience or opinion sampling was used (Hernández, Fernández and Baptista, 2010, p. 396). The population of the pilot study corresponds to the students of the computer science careers of the indicated branches of a Chilean university, which are the following: see Table 1.

Table 1. 1st year computer science students

INACAP Headquarters	Number of students in 1st year 2021
Iquique	127
La Serena	189
Maipú	212
Santiago-Centro	326
Ñuñoa	95
Concepción -Talcahuano	104
Puerto Montt	65
TOTAL	**1093**

Source: Own elaboration.

It should be noted that the sample was defined by the national direction of the Informatics Area of the institution that arranged the collaboration with the present research, according to convenience or opinion sampling (Hernández, Fernández and Baptista, 2010, p. 396).

3.3 Elaboration of the Pilot Study Instrument

The pilot test was built by the author, which is based on his professional training in the area of Computer Science and Informatics and his experience as a teacher in the subjects of Introduction to Programming, C Programming, Data Structure and Software Engineering and the contributions of training tests for Python certification, which can be found on the web (https://www.netacad.com/courses/programming/pcap-programming-essentials-python). Once the test was constructed, it was submitted to expert judgment, with the purpose of ratifying or discarding some test questions.

The test was submitted to the rating of 3 experts, who evaluated on the scale of 1 to 5, where 1 contributes less and 5 contributes completely to the fulfillment of the validation criterion. The experts are Civil Engineers in Computer Science, with a Master's degree in various specializations within the area of Computer Science, such as Software Development, Programming Languages, Operating Systems and Artificial Intelligence.

The instrument was developed for the Python programming language, a language that was incorporated as a teaching-learning tool in a Chilean University since March 2021. This language was added as a teaching-learning tool, due to its benefits as a computational language, compared to other professional languages such as Java or C++, which pose greater challenges to students, due to their complexity and rigor in the definitions of data and data structures, in addition to the development of the code that manifests itself with greater complexity in its structure and commands. However, Python is more flexible and user-friendly, from its interface to code creation.

Based on the specific needs of evaluating a computer programming course of computer science students at a Chilean University, I have focused on an instrument that measures students' skills in the application of control structures, since these form part of the very core of programming, as demonstrated, for example, in the curriculum of the extensive and successful British school computing initiatives, where control structures are mentioned as a prominent part of the intended learning. Depending on the school level, students should be able to: Design and write programs that include sequencing: doing one step after another. Selection (if-then-if-if-not): doing one thing or another. Repetition (iterative loops or recursion), conditional repetition (while - condition - do) and multiple branching (case - condition - then do 1, then do 2, ... then do n, - otherwise - do) (Mühling et al., 2015).

It should be noted that the initial version of the pilot instrument has a set of 90 multiple-choice questions, with 4 alternatives, which are categorized into 2 dimensions: Algorithms using simple data (Logical, numeric, character and alphanumeric) in Python, and algorithms in Python using structured data (Array, Dictionaries and Tuples), which will be the input for expert validation.

4. Results

4.1 Validation of Experts

Regarding the validation of experts, 5 experts in the computer area were summoned, of which 3 of them responded, which is accepted according to Hernandez-Nieto's theory (Hernandez-Nieto, 2002 in Pedraza et al., 2013). The procedure carried out was to make available to the experts the 90-item test that evaluates Python programming learning, and an Excel file with the validation criteria.

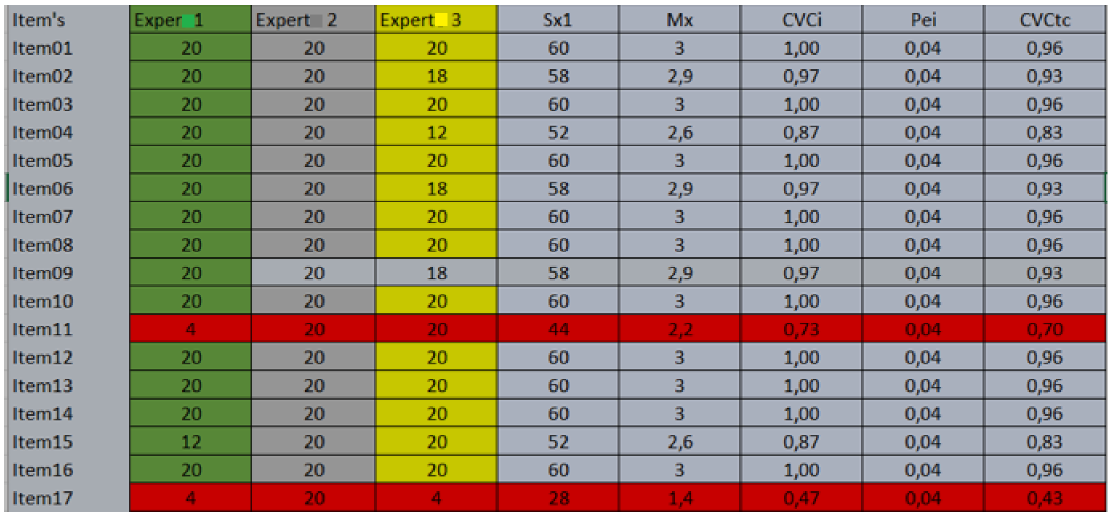
Once the three forms were received from the three experts who responded, a consolidated form was constructed, as shown in the image below for the first 8 items. It should be noted that the value 20 is the sum of the validation of the 4 validity criteria (Coherence, Clarity, Scale, and Relevance), in which each one takes the value from 1 to 5 according to the degree of validity considered by the expert with respect to each criterion for each item, so the maximum score per item is 20 points.

However, before calculating the content validity coefficient, a brief description of what the author proposes will be made. The Content Validity Coefficient (CVC) of Hernández-Nieto, like other classic coefficients such as the method based on the factor analysis of Tucker (1961), the content validity index of Lawshe (1975), the item-objective congruence index of Rovinelli and Hambleton (1977), among others, Hernández-Nieto (2002) makes it possible to assess the degree of agreement of the experts, who can be between three and five experts, with respect to each of the different items. For this purpose, after applying a Likert-type scale of five alternatives, the mean obtained in each of the items is calculated and based on this, the CVC is calculated for each item (Pedrosa et al., 2013).

Thus for an item j, $CVC_j = \frac{M_x}{V_{máx}}$, where M_x represents the mean of the item in the score given by the experts and

V_{max} represents the maximum score that the item could reach. On the other hand, the error assigned to each item (Pe_j) must be calculated, thus reducing the possible bias introduced by any of the judges, which is obtained by the formula $Pe_i = (\frac{1}{j})^j$ where j is the number of experts who responded. Finally, the CVC is calculated by applying CVC = CVC_i - Pe_j. Regarding its interpretation, Hernández-Nieto (2002) recommends keeping only those items with a CVC higher than 0.80 (Pedrosa et al., 2013).

Applying the formula proposed by Hernandez-Nieto (2002) resulted in the following table, which shows the first 17 items, of which the questions that do not meet the acceptance criteria are shown with a red background: see Figure 1.



Item's	Exper 1	Expert 2	Expert 3	Sx1	Mx	CVCi	Pei	CVCtc
Item01	20	20	20	60	3	1,00	0,04	0,96
Item02	20	20	18	58	2,9	0,97	0,04	0,93
Item03	20	20	20	60	3	1,00	0,04	0,96
Item04	20	20	12	52	2,6	0,87	0,04	0,83
Item05	20	20	20	60	3	1,00	0,04	0,96
Item06	20	20	18	58	2,9	0,97	0,04	0,93
Item07	20	20	20	60	3	1,00	0,04	0,96
Item08	20	20	20	60	3	1,00	0,04	0,96
Item09	20	20	18	58	2,9	0,97	0,04	0,93
Item10	20	20	20	60	3	1,00	0,04	0,96
Item11	4	20	20	44	2,2	0,73	0,04	0,70
Item12	20	20	20	60	3	1,00	0,04	0,96
Item13	20	20	20	60	3	1,00	0,04	0,96
Item14	20	20	20	60	3	1,00	0,04	0,96
Item15	12	20	20	52	2,6	0,87	0,04	0,83
Item16	20	20	20	60	3	1,00	0,04	0,96
Item17	4	20	4	28	1,4	0,47	0,04	0,43

Figure 1. Content validity index by items

Source: Own elaboration.

With the information obtained, all those items whose CVC was less than 0.80 according to the theory of Hernández-Nieto (2002) were removed from the test, which resulted in a pilot test of 70 items. See complete test in appendix.

4.2 Application of the Pilot Instrument

The application of the pilot test was carried out in the second semester of the year 2021 during the month of December by disposition of the National Direction of the Computer Science Area, to a sample of computer science students detailed in the following Table 2:

Table 2. Computer Science students participating in the pilot test by site

INACAP Headquarters	**Students 1st year 2021**	**1st year students answering the test**	**% of participation by site**
Iquique	127	28	22%
La Serena	189	21	11,1%
Maipú	212	9	4,2%
Santiago Centro	326	96	29,4%
Ñuñoa	95	77	81%

Concepción-Talcahuano	104	15	14,4%
Puerto Montt	65	43	66,2%
TOTAL	**1093**	**289**	**26,4%**

Source: Own elaboration.

For the pilot test, 1,093 students from 7 sites were invited to participate, of which 289 male and female students responded, representing a 26.4% participation rate, as detailed in Table 2 above, corresponding to first year students in the computer science careers of a Chilean University from the sites selected by the Computer Science Area.

Regarding those who answered the test, the participation by gender per site is shown in the following Table 3:

Table 3. Women and men who answered the test

INACAP Headquarters	participants	**man**	**female**
IQUIQUE	**28**	23	5
LA SERENA	**21**	21	0
MAIPU	**9**	9	0
SANTIAGO CENTRO	**96**	86	10
ÑUÑOA	**77**	68	9
CONCE-TALCAHUANO	**15**	13	2
PTO. MONTT	**43**	42	1
TOTAL	**289**	**262**	**27**

Source: Own elaboration.

Regarding the age of the students, 96% are between 18 and 24 years old, with the majority of them between 18 and 21 years old (80%). Regarding gender, 90.4% are men and only 9.6% are women, which explains why only 27 women responded out of the 289 who answered the test in total. Regarding the type of school in which they attended high school, 41.3% attended a municipal school, 45.7% attended a subsidized private school, 10.9% attended a private school and 2.1% took free exams or another type of school. The average NEM of the participants in the test was 5.6. Finally, with regard to the computer career they are studying, 53.5% are Programmer Analysts and 46.5% are studying Computer Engineering.

The test was carried out by means of an online google form during free time, which was distributed by the IT Department to the IT career directors of the participating sites, who shared it with the professors of each participating site. As the activity was voluntary, weekly progress reports were made, which were informed to the career directors of the participating sites with a copy to the national direction of the IT area of the University.

4.3 Análisis Psicométricos Del Test

El principal objetivo del estudio piloto es calcular la validez y confiabilidad de las preguntas del test. Los análisis de datos del estudio piloto pueden dividirse en dos fases: (1) *Estudio de confiabilidad y análisis de las preguntas*; (2) *Estudio de validez*. A continuación, se describen los análisis psicométricos y estadísticos efectuados en cada fase.

4.3.1 Confiabilidad

El análisis de confiabilidad incluyó la estimación de la consistencia interna para cada cuestionario mediante el cálculo del *coeficiente alfa de Cronbach*. Para el análisis de las preguntas se calculó el *coeficiente alfa si se elimina el ítem* (índice de confiabilidad). Para el cálculo del coeficiente Alfa de Cronbach se utilizó el software SPSS (*Statistical Package for the Social Sciences*) versión 22 de IBM.

The data output was the value of Cronbach's alpha, which for the test performed resulted in 0.917, but which can go up by one to two thousandths if some questions are eliminated. For example, if questions 33, 41, 55 or 70 are removed separately, Cronbach's alpha increases to 0.918, but if question 40 is removed, Cronbach's alpha increases to 0.919. Advancing the analysis a little, it turns out that, when calculating the difficulty and discrimination indexes, precisely some of the items that are eliminated are the questions indicated, so that the Cronbach's alpha of the corrected test is 0.919, a value that is quite good.

4.3.2 Difficulty and Discrimination

The classical theory has been widely studied, of which authors such as Muñiz (2010), García-Cueto (2005) and Master (1988) stand out with a finished study of the calculation of the discrimination index. The calculation for both indexes considers dividing the sample between the subjects who obtained higher performance, called the upper group, and the group of subjects with lower performance, called the lower group. For this purpose, the total score per individual is obtained and then the database is ordered from highest to lowest, according to the level of achievement. The criterion for separating the upper group from the lower group is to take the 30% with the best results and assign them as the upper group, and the 30% with the lowest results will be assigned as the lower group, leaving an intermediate 40% that is not used in the calculation of the indexes.

According to Masters (1988) the calculation is made using the following formulas:

p = $\frac{U_p+L_p}{U+L}$ for the index of difficulty

D = $\frac{U_P-L_P}{U}$ for the discrimination index.

Where:

U: Number of subjects in the upper group,

L: Number of subjects in the lower group,

Up: Number of subjects in the upper group who answered correctly, Lp: Number of subjects in the lower group who answered correctly.

Lp: Number of subjects in the lower group who answered correctly.

For the case under study U=L= 90 subjects.

To perform the analysis of difficulty and discrimination it is necessary to know the ranges recommended by the authors: see Table 4.

Table 4. Range of difficulty and discrimination indexes

Difficulty	**Interpretation**		**Discrimination**	**Interpretation**
> 0,9	Very easy		>0,4	Discriminates very well
0,61 a 0,89	Easy		0,3 a 0,39	Discriminates well
0,4 a 0,6	Moderate (complies)		0,2 a 0,29	Discriminates poorly
< 0,4	Difficult		0 a 0,19	Does not discriminate

Suorce: Masters (1988).

In the following table, the difficulty and discrimination index was calculated for each item. After painting all the items that meet the acceptance criteria, the items that do not meet the criteria were numbered and then eliminated from the test, leaving only the items that meet both criteria, the difficulty index and the discrimination index, in order to construct the final test. See Table 5.

Table 5. Difficulty Index and Discrimination Index

Item	**U_p**	**L_p**	**Difficulty**	**Discrimination**	**Decision**
Q1	87	69	0,9	0,2	Rejected
Q2	78	54	0,7	0,3	Rejected
Q3	56	21	0,4	0,4	Accepted
Q4	79	56	0,8	0,3	Rejected
Q5	69	50	0,7	0,2	Rejected

Q6	79	62	0,8	0,2	Rejected
Q7	85	58	0,8	0,3	Rejected
Q8	73	44	0,7	0,3	Rejected
Q9	80	33	0,6	0,5	Accepted
Q10	49	21	0,4	0,3	Accepted
Q11	84	22	0,6	0,7	Accepted
Q12	77	34	0,6	0,5	Accepted
Q13	83	30	0,6	0,6	Accepted
Q14	20	13	0,2	0,1	Rejected
Q15	89	45	0,7	0,5	Rejected
Q16	85	30	0,6	0,6	Accepted
Q17	87	63	0,8	0,3	Rejected
Q18	65	24	0,5	0,5	Accepted
Q19	58	19	0,4	0,4	Accepted
Q20	64	48	0,6	0,2	Rejected
Q21	84	37	0,7	0,5	Rejected
Q22	77	43	0,7	0,4	Rejected
Q23	66	26	0,5	0,4	Accepted
Q24	79	34	0,6	0,5	Accepted
Q25	70	19	0,5	0,6	Accepted
Q26	57	11	0,4	0,5	Accepted
Q27	50	23	0,4	0,3	Accepted
Q28	83	31	0,6	0,6	Accepted
Q29	59	31	0,5	0,3	Accepted
Q30	44	10	0,3	0,4	Rejected
Q31	66	12	0,4	0,6	Accepted
Q32	56	19	0,4	0,4	Accepted
Q33	19	23	0,2	0,0	Rejected
Q34	74	25	0,6	0,5	Accepted
Q35	66	13	0,4	0,6	Accepted
Q36	84	28	0,6	0,6	Accepted
Q37	30	7	0,2	0,3	Rejected
Q38	75	32	0,6	0,5	Accepted
Q39	83	32	0,6	0,6	Accepted
Q40	28	40	0,4	0,0	Rejected
Q41	13	17	0,2	0,0	Rejected
Q42	86	17	0,6	0,8	Accepted
Q43	84	33	0,7	0,6	Rejected
Q44	85	29	0,6	0,6	Accepted
Q45	89	39	0,7	0,6	Rejected
Q46	56	22	0,4	0,4	Accepted
Q47	89	38	0,7	0,6	Rejected
Q48	85	28	0,6	0,6	Accepted
Q49	75	26	0,6	0,5	Accepted
Q50	78	16	0,5	0,7	Accepted

Q52	54	17	0,4	0,4	Accepted
Q53	68	34	0,6	0,4	Accepted
Q54	53	20	0,4	0,4	Accepted
Q55	40	38	0,4	0,0	Rejected
Q56	46	16	0,3	0,3	Rejected
Q57	72	20	0,5	0,6	Accepted
Q58	77	27	0,6	0,6	Accepted
Q59	66	20	0,5	0,5	Accepted
Q60	60	19	0,4	0,5	Accepted
Q61	67	8	0,4	0,7	Accepted
Q62	86	36	0,7	0,6	Rejected
Q63	72	31	0,6	0,5	Accepted
Q64	57	10	0,4	0,5	Accepted
Q65	65	16	0,5	0,5	Accepted
Q66	61	24	0,5	0,4	Accepted
Q67	69	20	0,5	0,5	Accepted
Q68	78	31	0,6	0,5	Accepted
Q69	68	25	0,5	0,5	Accepted
Q70	57	56	0,6	0,0	Rejected

SOURCE: Own elaboration.

See complete test in Annex.

5. Discussion and Conclusions

5.1 World Context of Computer Programming Education

The teaching of computer programming involves resources ranging from the implementation of digital infrastructure (computers and software), the availability of sufficient human resources to cover the entire primary and secondary education, to the non-existence of a didactic that guides the delivery of computer programming knowledge. In this context, the questions arise: what to teach and at what level? which programming languages to teach; Python, Scratch, Alice or Java? in which subjects to incorporate programming? only to teach in the STEM area or to extend to the humanistic area? In addition, there is the aspect of how to measure learning, how to know that what is being done pays off in new learning for students and that this learning truly constitutes the acquisition of computational thinking skills.

At the 21st edition of the International Conference on Interactive Collaborative Learning and the 47th edition of the International Conference on Engineering-Pedagogy held in 2018 at the Aristotle University of Thessaloniki of Greece, they published a text with the papers presented entitled "The Challenges of the Digital Transformation in Education", in which among the many papers presented, a section on research conducted on preschool, primary and secondary education stands out. Some of the titles observed are: "Cyber and Internet Module Using Python in Junior-High School", "Children's Reflection-in-Action During Collaborative Design-Based Learning", "Internet Addiction and Anxiety Among Greek Adolescents: An Online Survey", "Intelligent Robotics in High School: An Educational Paradigm for the Industry 4.0 Era", "Design and Use of Digitally Controlled Electric Motors for Purpose of Engineering Education", among others. It should be noted that, from the observed titles, didactic teaching strategies do not appear and neither are observed validations of tests to measure learning (Auer & Tsiatsos, 2019).

For its part, UNESCO proposes a master plan for the development of digital skills, in which in one of the sections it highlights some important points in which it points out that the plan should have described aspects on the technological infrastructure in the school, as a necessary prerequisite, teachers trained in the area of digital technologies, and the integration of digital technology within the curriculum, not only in specific courses but within the goals or objectives (Fau & Moreau, 2018). These aspects must be supported by public policies that guide in teaching methodologies and didactics, beyond providing digital infrastructure and human resources. It should be noted that implementing the teaching of computational thinking is an enormous task, which should start in universities by preparing teachers for such a gigantic task, who after the curricular policies have been dictated by governments and the economic resources have been allocated, will be able to implement the new teaching in schools (Law et al., 2018).

Complementing the above, the European Commission published in 2016 a paper entitled "Developing Computational Thinking in Compulsory Education", in which it makes various diagnoses and the state of progress on the subject in the member countries. The research points out, among other things, that the challenge for governments is to prepare future generations for this digitalized world that is coming, however, countries that have implemented educational reforms have found a dramatic shortage of teachers trained in the areas of Computer Science, and where they have them, although few, there is the problem that there are no pedagogical and didactic models for teaching the new skills. The report analyzes the most significant of the development of computational thinking for compulsory education in Europe and in synthesis provides evidence including practical and policy implications (Bocconi et al., 2016).

5.2 Two Cases of Test Validations in Primary Education

Although both UNESCO and the European Commission - Joint Research Centre (JRC) have made significant progress in the right direction, these advances mainly benefit developed countries, particularly those countries that have implemented the teaching of computational thinking or have decided to do so. However, there is still currently a lack of standardized tests to measure computational thinking skills and the level of achievement in learning computational programming languages, such as Python, Scratch or Alice. This occurs at the three educational levels, primary, secondary and tertiary, although there are some publications regarding primary education, which have been developed in those countries that have incorporated computational thinking as part of the school system, this marks the first signs that the subject is beginning to develop research gradually. Thus, for example, there is the case of a publication carried out in Spain in 2019 entitled "Computational thinking test: design guidelines and content validation", research conducted with students in the first basic cycle (González, 2015).

In another research conducted by Mühling and his team, the evaluation of competences in computational programming is directly addressed, in which a test is validated based on the algorithmization of procedures, where decision procedures are highlighted with the use of the if statement, conditional repetitive cycle processes with the use of the while statement, and the unconditional cycle for, from which algorithms can be built that together with basic assignment, reading and writing statements of the console, allow to make an application more complex until it becomes a solution to a real problem. It should be noted that computational sentences by themselves do not allow making an application, but this will require emulating human performance, performing nesting in the cycles and elementary sentences (Mühling et al., 2015).

5.3 Findings and Comments

In the present test validation study, perhaps one of the interpretations with certain bias and degrees of limitation is the idea that not enough is being done in the world to implement the teaching of computer programming in order to achieve computational thinking skills, a key skill for the technological development of the 21st century. In fact, there are studies that indicate that international standard tests such as PISA and TIMSS are gradually incorporating elements of computational thinking and programming, which in the medium and long term will increase the gap between developed countries and countries that are not developed, or rather a new gap will open up between countries that have implemented the teaching of computational thinking with respect to countries that have not. Now, as there is an overlap between developed countries and countries that are implementing the teaching of computational thinking in school, the gap will increase and will be determinant in future international test results, opening an even bigger gap between developed and non-developed countries (Alyahya & Alotaibi, 2019).

In the research, it is observed that the area of teaching computer programming and the evaluation of student achievement, come together and give way to the need to generate both teaching materials for programming and tests that measure such learning, where both topics converse in a coherent manner, that is, what is evaluated is consistent with what is taught and vice versa, what is taught is consistent with what is evaluated. In the institution where the research was conducted, it is observed that the teaching strategy that is mostly practiced is that they are given one or more titles with two or three associated contents and the student must learn alone, i.e., there is no gradual delivery of contents with clear objectives class by class. Teachers in the area of computer science, who do not have pedagogical training, deliver the contents as they learned them in their time as students and do not break down the contents into smaller units, so as to teach with due gradualness for students to build their knowledge in a pleasant and effective way.

In the case of the application of the test, it was observed that in the performance of the students, none obtained 100% of the score, which is explained by the excessive difficulty of some items, however, this contrasts with the opinion of two of the experts, that more than 50% of the students who graduate from the institution do not know how to program computers in any of the 5 languages they study during the 2.5 years of the Programmer Analyst career. This shows that late training in programming is ineffective if done in short periods of time, and reinforces the idea of establishing the teaching of programming from elementary school, which would allow learning in computer programming to be integrated into the student's knowledge, such as mathematics, science or history, so that when they reach university or professional technical education, it would allow teaching to focus only on the professional training of the career they are studying.

5.4 Projections

In the future there should be many investigations that test different didactic strategies for teaching programming, and that promote the creation of validated tests to measure the contributions of the didactic strategies that are implemented with the students. It should be noted that the didactics for primary and secondary students should be different from the one applied in tertiary education, since the latter works with adult students who have a specific purpose for which they carry out the study and not general as when they are in primary and secondary education. Thus emerges the concept of andragogy, which is the analog of pedagogy, but for adult education. Moreover, in the future it will not only be necessary to evaluate algorithms with conceptual or quantitative results, but it will be necessary to evaluate actions such as those performed by a robot or an automated machine (Carrillo, 2018).

The educational systems of the future, should incorporate not only the teaching of computational thinking, but also elements of neuroscience, so that once the basis of knowledge in computer programming, educational robotics and gamification is laid, it can go further, making the connection between computational processes and brain processes, in order to understand and create applications that are able to take advantage of brain waves in the activation of electronic devices that perform actions as an extension of the body, as there are already developments in the war industry, but with peaceful motivations and focused on the physical disabilities of human beings and also on brain disabilities with the aim of correcting diseases such as deafness, blindness or go beyond, correcting Alzheimer's disease.

The teaching of computational thinking should flood all knowledge formation in the educational system, integrating STEM and non-STEM areas or better known as STEAM. Currently there are lost subjects such as technology education when students are made to do crafts related to handicrafts or issues that contribute very little to the formation of knowledge. The subject of technology education should change its content from craft to technology, and should be composed of electronics, integrating electronic devices with the management of programs that allow students to be true creators of technology and not just users of this or observers of technology through the use of social networks that contribute little or nothing to build knowledge.

The formation of knowledge in schools should cut across all areas of knowledge, starting with mathematics and natural sciences, where one could experiment with the creation of virtual reality and/or augmented reality applications for chemistry and biology, passing through history and geography where one could teach through the creation of games and stories located in certain territories and eras, such as the Age of Empires game where different versions of the game show the ancient civilizations of Europe until about 1700, and could create applications for schoolchildren on other continents, such as Africa, Latin America, Asia and Oceania. In language, tales and stories could be recreated to give life to the letters, and in art it could be integrated with developments with virtual reality, so that students navigate within the technology building the various knowledge of education, all of which would be done in a progressive and interactive way training students in programming and gradually incorporating the skills of abstraction, decomposition of problems, algorithmization, debugging and problem solving.

Some of the topics in computer science that could be studied and evaluated in the future would be computer security, computer programming in education, disconnected activities in learning computational thinking, didactic strategies in the teaching and learning process of programming, internet of things, data science and big data, artificial intelligence, students with special needs, and studies on psychological aspects of technology and humans, data science and big data, artificial intelligence, students with special needs and studies on psychological aspects of technology and humans, in addition to deepening on gamification and educational robotics in primary school and robotics and industrial automation in secondary school, so that project-based learning, learning by doing, is practiced. It should be noted that all this should be duly distributed throughout the 12 years of primary and secondary education.

In addition to the above, there are emerging technologies such as nanotechnology and quantum computing, topics that should be part of the educational system in research phases and emerging technology topics. It should be noted that companies such as IBM already have prototypes of quantum computers that are fully operational. It is important to keep in mind that a quantum computer can decode all existing computer security systems in the world in just minutes, which will leave governments and organizations around the world unprotected. This happens because of the processing speed of this type of computers, since they are based on the parallelism of the binary system. While in the current electronic system, the bits (1 and 0) manifest themselves sequentially, in the quantum system they do so in parallel, which means that an algorithm that takes x units of time to execute in a current computer, in a quantum computer will take at most $\log_2(x)$, that is, if x takes the value in hours x = 8,760 hours (1 year), log2(8,760) corresponds to 13 hours and 6 minutes, a little more than half a day.

Finally, I would like to make a reflection on the difference in priorities between countries, which occurs on multiple levels; cultural, economic, technological, political and social, which leads to think, where should we start from, should we promote the teaching of computational thinking as a way to transfer knowledge and thus in the medium and long term the countries solve their social problems, or should we categorize where to start from, besides the fact that each country is independent. For example, in Haiti in Central America, with more than 200 years of independence, they still have not managed to have a stable political system that allows them to overcome extreme poverty and hunger. In this situation there are several countries in Africa and Latin America, while the world, represented by the developed countries, is faced with the dilemma of climate change, which is just around the corner and will affect us all. I firmly believe that, in order of priority, the teaching of computational thinking is second only to climate change.

References

Alyahya, D., & Alotaibi, A. (2019). Computational thinking skills and its impact on TIMSS achievement: An Instructional Design Approach. *Issues and Trends in Learning Technologies, 7*(1).

Auer, M. E., & Tsiatsos, T. (Eds.). (2019). *The Challenges of the Digital Transformation in Education: Proceedings of the 21st International Conference on Interactive Collaborative Learning (ICL2018)*-Volume 1 (Vol. 916). Springer.

Belmar, H. M. (2022), Review on the teaching of programming and computational thinking in the world. *Front. Comput. Sci.,* 4:997222. https://doi.org/10.3389/fcomp.2022.997222

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). Developing computational thinking in compulsory education – Implications for policy and practice; EUR 28295 en. https://doi.org/10.2791/792158

Carrillo, F. (2018). *Formación de Competencias para el Trabajo en Chile.* Tech. rep., Comisión Nacional de Productividad.

Dagiene, V., & Stupuriene, G. (2016). Bebras--A Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in education, 15*(1), 25-44.

Fau, S., & Moreau, Y. (2018). Managing tomorrow's digital skills-what conclusions can we draw from international comparative indicators?. *Education 2030 – UNESCO.*

García, F. (2018). Editorial Computational Thinking. *IEEE Ibero-American Journal of Learning Technologies, 13*(1), 17-19.

González, M. R. (2015). Computational thinking test: Design guidelines and content validation. In *Proceedings of EDULEARN15 conference* (pp. 2436-2444).

Grgurina, N. (2021). Getting the Picture: Modeling and Simulation in Secondary Computer Science Education.

Hernández, R., Fernández, C., & Baptista, P. (2010). *Research Methodology (5 ed.).* Mexico, D.F.: McGraw-Hill Interamericana.

Hernández-Nieto, R. A. (2002). Contributions to statistical analysis. *Mérida: Universidad de Los Andes,* 193.

Jacob, S., & Warschauer, M. (2018). Computational thinking and literacy. *Journal of Computer Science Integration, 1*(1).

Kale, U., & Yuan, J. (2021). Still a new kid on the block? Computational thinking as problem solving in Code. org. *Journal of Educational Computing Research, 59*(4), 620-644.

Kulkarni, V. (2015). Looking Back: Alan Turing-The Father of Computer Science.

Law, N., Woo, D., de la Torre, J., & Wong, G. (2018). A global framework of reference on digital literacy skills for indicator 4.4. 2. *UNESCO - Institute for Statistics*.

Lawshe, C. H. (1975). A quantitative approach to content validity. *Personnel Psychology, 28*, 563-575.

Masters, G. N. (1988). Item discrimination: When more is worse. *Journal of Educational Measurement, 25*(1), 15-29.

Mühling, A., Ruf, A., & Hubwieser, P. (2015). Design and first results of a psychometric test for measuring basic programming abilities. In *Proceedings of the workshop in primary and secondary computing education* (pp. 2-10).

Pedrosa, I., Suárez-Álvarez, J., & García-Cueto, E. (2013). Evidence on content validity: theoretical advances and methods for its estimation. *Psychological Action, 10*(2), 3-18.

Rojas, A., & Garcia, F. J. (2020). Evaluation of computational thinking for learning computer programming in higher education. *Distance Education Magazine (RED), 20*(63).

Rovinelli, R. J., & Hambleton, R. K. (1977). On the use of content specialists in the assessment of criterion-referenced test item validity. *Dutch Journal of Educational Research, 2*, 49-60.

Tucker, L. R. (1961). Factor Analysis of Relevance Judgments: An Approach to Content Validity. In A. Anastasi (Ed.), *Testing Problems in Perspective* (pp. 577-586). Washington, DC.: American Council on Education

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences, 366*(1881), 3717-3725.

Wing, J. (2011). Research notebook: Computational thinking—What and why. *The Link Magazine,* 6.

Annex Validated Python test.

1.- How do you open a file to read it?

a) f = open("archive.txt", "read")

b) f = open("read", "archive.txt")

c) f = open("file.txt", "w")

d) f = open("archive.txt", "r")

2.- What is the correct way to write a while loop?

a) while a<5

b) while a foreach[0..4]

c) while a in range(0..4)

d) while (a=5)

3.- To show the value of the position 2 of an array called mycollection we use

a) print(mycollection[1])

b) print(mycollection[2])

c) pp(mycollection[2])

d) puts(mycollection[2])

4.- To add an alternative condition to a conditional if statement we use

a) elseif

b) else if

c) elsif

d) elif

5.- Which of the following is an object of type dictionary?

a) dictionary = {'Number': 1, 'Name': 'Miguel'}

b) dictionary = {'Number' -1, 'Name' -> 'Miguel'}

c) dictionary = {'Number' => 1, 'Name' => 'Miguel'}

d) dictionary = ('Number': 1, 'Name': 'Miguel')

6.- What is the correct way to write a for loop?

a) for a in range(0..3)

b) for (a in range [0..3])

c) for(a=0; a<3; a++)

d) for a in range(0, 3):

7.- How is a variable defined by assigning it a value?

a) v = 0

b) int v = 0

c) var v = 0

d) number v = 0

8.- Which of these data types is mutable?

a) bool (Boolean)

b) decimal

c) float (Floating point number)

d) None of the above

9.- Which of these data types is immutable?

a) dictionaries

b) bytearrays

c) sets

d) None of the above

10.- Which of the options allows us to obtain a list with the following values: [9, 16, 25], given that; lst_num = [3, 4, 5]?

a) output = [n*2 for n in lst_num].

b) output = [n**2 for n in lst_num]

c) output = [n**2 for lst_num in n]

d) output = [n^2 for n in lst_num]

11.- Fill in the missing line of code to obtain the uppercase texts from the following list: lst_lp=['Python','c','java','php']]

a) output = [lp.upper() for lp in lst_lp]

b) output = [lp.capitalize() for lp in lst_lp]

c) output = [lp.lower() for lp in lst_lp]

d) output = [lp.uppercase() for lp in lst_lp]

12.- What elements will the list have when executing the following code?

num = [1, 2, 3, 4] num = [1, 2, 3, 4]

out = [n-1 for n in num if n<=3]

print(out)

a) [1, 2, 3]

b) [1, 2, 3, 4]

c) [0, 1, 2]

d) [0, 1, 2, 3]

13.- Which of the following options allows us to obtain a list with the following values [2, 3, 3, 4], given that lst_num = [1, 2, 3, 4]?

a) output = [n + 1 if n<4 else n for n in lst_num].

b) output = [n + 1 if n<2 else n for n in lst_num].

c) output = [n + 1 if n<=3 else n for n for n in lst_num]

d) output = [n + 1 if n<=2 else n for n in lst_num] 14.

14.- Which of the options allows us to obtain a list with the following elements: ['P', 'P', 'H'], given that:

letters_1 = ['P', 'Y', 'T', 'H', 'O', 'N']

letters_2 = ['P', 'H', 'P']?

a) output = [a for a in letters_1 for b in letters_2 if a=b]

b) output = [a if a == b for a in letters_1 for b in letters_2]]

c) output = [a for a for a in letters_1 for b in letters_2 if a==b]

d) output = [a for a for a in letras_2 for b in letras_1 if a=b]

15.- What elements will the list have after the execution of the following code?

lst_test_a = [9, 34.90, 'Python']

lst_test_a.append('Flask')

a) ['Flask', 34.9, 'Python']

b) [9, 34.9, 'Python', 'Flask']

c) ['Flask', 9, 34.9, 'Python']

d) [9, 34.9, 'Flask'] 16.

16.- Which instruction is necessary to obtain the following elements: [1, 5, 10, 15] in the list "lst_test_b", by defining; lst_test_b = [1, 10, 5].

a) lst_test_b.insert(1,5)

b) lst_test_b.extend(5,1)

c) lst_test_b.append(2,5)

d) lst_test_b.insert(5,1)

17.- What is the output in the following instruction block?

lst_test_d = ['PHP', 'Python', 'Go', 'Java']

language = lst_test_d.pop()

print (language)

a) PHP

b) Java

c) Python

d) ['PHP', 'Python', 'Go', 'Java'] ['PHP', 'Python', 'Go', 'Java']

18.- How do I get the element whose value is 'Django' from the lst_test_e list?

```
lst_test_e = ['Ruby on Rails', 'Laravel', 'Django', 'Zend Framework']
```

a) lst_test_e.pop(2)

b) lst_test_e.pop(2,1)

c) lst_test_e.get(2)

d) lst_test_e.get(2, 1)

19.- What numbers does the following Python code print?

```
x = range(7, 15, 2)
for n in x:
    print (n)
```

a) 7, 9, 11, 13 and 15

b) 9, 11, 13 and 15

c) 7, 9, 11 and 13

d) 9, 11 and 13

20.- What is the output produced by this Python code block?

```
x = 0
while x < 3 :
        x= x + 1
else:
        for n in range (x, x+3):
                print(n)
```

a) 0, 1 and 2

b) 1, 2 and 3

c) 2, 3 and 4

d) 3, 4 and 5

21.- How do I get the element whose value is 'Spain' from the list of countries (lst_countries)?

```
lst_paises = ['Germany', 'Belgium', 'Denmark', 'Spain', 'France']]
```

a) lst_paises[-1]

b) lst_paises[2]

c) lst_countries[3]

d) lst_countries[4]

22.- What value does the execution of the following code return?

```
lst_6 = ['Python', 'PHP', 'Java', 'Javascrip']
index = lst_6.index('Abap')
```

a) -1

b) false

c) True

d) Error

23.- If I use the * operator as shown in the following code, what is the result obtained?

In [1]: 'two' * 3

a) seid

d) 6

c) twotwotwo

d) two3

24.- What is the result obtained when executing this Python instruction?

>>> print(17 % 3)

a) 5

b) 3

c) 2

d) 1

25.- The output of the following Python instruction is:

>>> print(2**3)

a) 6

b) 5

c) 8

d) 9

26.- For the following instruction in Python

if('1'='1'): print('1')

What is missing for it to work?

a) =

b) Nothing

c) Badly identified

d) No error

27.- What would be your output of the following Python code segment?

def value(x): return x+x

print(value(1))

a) 2

b) 4

c) x

d) 1

28.- An operator capable of verifying if two values are not equal is:

a) =/=

b) <>

c) !=

d) not ==

29.- The following code fragment:

def func1(a):

return None

```
                    def func2(a):
return func1(a) * func1(a)
                    print(func2(2))
```

a) will output 2

b) will output 16

c) will output 4

d) will result in a runtime error

30.- What value will be assigned to the variable x?

```
z = 0
y = 10
x = y < z and z > y or y > z and z < y
```

a) False

b) 1

c) 0

d) True

31.- What is the result of the following code fragment?

```
list = [x * x for x in range(5)].
def fun(lst):
       del lst[lst[lst[2]]
                            return lst
       print(fun(list))
```

a) [1, 4, 9, 16]

b) [0, 1, 4, 16]

c) [0, 1, 4, 9]

d) [0, 1, 9, 16]

32.- What is the result of the following code?

```
x = 1
y = 2
x, y, z = x, x, y
z, y, z = x, y, z
print(x, y, z)
```

a) 1 1 2

b) 2 1 2

c) 1 2 1

d) 1 2 2

33.- What will be the output of the following code fragment?

```
a = 1
b = 0
a = a ^ b
b = a ^ b
a = a ^ b
print(a, b)
```

a) 0 1

b) 0 0

c) 1 0

d) 1 1

34.- What is the result of the following code?

```
print("a", "b", "c", sep="sep")
```

a) a b c

b) asepbsepc

c) abc

d) asepbsepcsep

35.- What is the result of the following code?

```
x = 1 // 5 + 1 / 5
print(x)
```

a) 0.4

b) 0.0

c) 0

d) 0.2

36.- What is the result of the following code fragment?

```
dct = { 'one':'two', 'three':'one', 'two':'three' }
v = dct['three']
for k in range(len(dct)):
        v = dct[v]
print(v)
```

a) ('one', 'two', 'three')

b) two

c) one

d) three

37. How many elements does the list lst contain?

```
lst = [i for i in range(-1, -2)].
```

a) two

b) zero

c) three

d) one

38.- What is the result of the following code fragment?

```
def fun(x, y):
        if x == y:
        return x
        else:
        return fun(x, y-1)
print(fun(0, 3))
```

a) 2

b) 1

c) the code fragment will result in a runtime error

d) 0

39.- What is the result of the following code fragment?

```
dd = { "1": "0", "0": "1" }
for x in dd.vals():
    print(x, end=" ")
```

a) 0 0

b) 0 1

c) the code is wrong (object dd does not contain the vals() method)

d) 1 0

40.- What is the result of the following code fragment?

```
dct = {}
dct['1'] = (1, 2)
dct['2'] = (2, 1)
for x in dct.keys():
    print(dct[x][1],end="")
```

a) (1,2)

b) (2,1)

c) 21

d) 12

41.- What is the result of the following code fragment?

```
def fun(inp=2, out=3):
    return inp * out
print(fun(out=2))
```

a) the code fragment is wrong

b) 6

c) 2

d) 4

42.- How many (#) will the following code fragment print on the console?

```
lst = [[x for x in range(3)] for y in range(3)]: [[x for x in range(3)]: [[x for x in range(3)].
for r in range(3):
    for c in range(3):
        if lst[r][c] % 2 != 0:
            print("#")
```

a) three

b) nine

c) six

d) zero

43.- What will be the output of the following code?

```
x = 16
while x > 0:
    print('*', end='')
    x //= 2
    break
```

a) *

b) *****

c) the code will enter an infinite loop

d) ***

44.- What will be the result of the following fragment?

```
d = { 'one':1, 'three':3, 'two':2 }
for k in sorted(d.values()):
    print(k, end=' ')
```

a) 3 2 1

b) 2 3 1

c) 1 2 3

d) 3 1 2

45.- What will be the result of the following fragment?

```
d = {}
d['2'] = [1, 2]
d['1'] = [3, 4]
for x in d.keys():
    print(d[x][1], end="")
```

a) 24

b) 13

c) 42

d) 31