Brazilian Journal of Experimental Design, Data Analysis and Inferential Statistics

Federal University of Rio de Janeiro

BJEDIS, Rio de Janeiro, Special Edition, v. 01, 2021, Page:88

DOI: https://doi.org/10.55747/bjedis.v1i1.48409

E-ISSN: 2763-6925

Special Edition **Received 01/28/2021. Revised 02/18/2021. Accepted 03/12/2021. Published 05/13/2021.**

Preliminary Study on the Application of the Python Language as a Tool for the Randomization of Laboratory Experiments: a Short Course at ConBraPA 2020

Igor Leão dos Santos[*1], Emanuele Nunes de Lima Figueiredo Jorge[2], Sérgio Thode Filho[2] e Fernando Gomes de Souza Júnior[3]

[1]*Programa de Pós-graduação em Engenharia de Produção e Sistemas (PPPRO), Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ), Rio de Janeiro, Brasil;*

[2]*Instituto Federal de Educação, Ciência e Tecnologia do Rio de Janeiro, IFRJ, Rio de Janeiro, Brasil.*

[3]*Universidade Federal do Rio de Janeiro, UFRJ, Rio de Janeiro, Brasil.*

Abstract: Digitization, network operation and an emphasis on data become the main attributes of smarter production, the core of a phenomenon declared as Industry 4.0, or the fourth industrial revolution, or even the fourth technological era. Amid the revolution of the fourth technological age that occurs in our society, the demand for professionals with programming skills stands out. A first challenge in this context concerns how to train professional programmers, in a current, effective, and versatile programming language. A second challenge concerns how to provide this professional with a useful programming base for experimental planning and data analysis, already integrating his programming learning with experimental planning theory. In line with these challenges, a Python language computer programming course was developed with application in the context of the Completely Randomized Design (CRD). The objective of this work is to present and discuss the short course, its construction methodology, and the results of this short course, which took place at the 1st Brazilian Conference on Experimental Planning and Data Analysis (ConBraPa 2020). The main results were: (i) participants learnt about basic programming logic, (ii) participants learnt about using basic instructions in the Python programming language, (iii) participants learnt how to create a randomized experimental sketch in the context of CRD using Python, (iv) elaboration of a specific short course construction methodology. It is noteworthy that, through evaluation with a questionnaire, it was possible to conclude that 100% of the participants in the short course who answered the questionnaire evaluated the short course as beneficial.

Keywords: Python, Programming, Completely Randomized Design, Application, Education, Short Course.

Adherence to the BJEDIS' scope: This work is closely related to the scope of BJEDIS as it presents the result of a short course carried out at ConBraPA2020 that allows the use of the python language in several experimental tests.

*Address correspondence to this author at the Department of Graduate Program in Production and Systems Engineering (PPPRO) at the Federal Center for Technological Education Celso Suckow da Fonseca (CEFET / RJ), CEP: 20271-110, Rio de Janeiro, Brazil; Tel / Fax: +55 (21) 2566-3022 / -3111 / -3120 / -3015; E-mail: igor.santos@cefet-rj.br.

1. INTRODUCTION

The global shift towards a completely connected society is underway, around the entire planet (1). The digital transformation influences business and private life in a radical and sustainable way. Important topics in this context are the physical-cyber systems (CPS) (2), the Internet of Things (IoT) (3), and the Cloud of Things (CoT) (4), in addition to virtualization [5] in general, which permeates all these technologies. The economic potential is evident, with an emphasis on these network and Internet technologies, which mark the great digital businesses of the future. Thus, the world becomes increasingly digital. Digitization, network operation and an emphasis on data become the main attributes of smarter production, the core of a phenomenon declared as Industry 4.0, or the fourth industrial revolution, or the fourth technological era (1, 6).

Amid the revolution of the fourth technological age that occurs in our society, the demand for professionals with programming skills stands out. Several authors point out the need to develop software solutions to enable the benefits of advanced manufacturing (7, 8). These same authors emphasize that, in addition to the capabilities of human beings to interact with machines, there is a great need for the development of analytical capacity and problem solving. For this case, a professional profile of experimental planning and data analysis is very welcome to the professional of the fourth technological age (9). Thus, two challenges arise. The first concerns how to train professional programmers in a current, effective and versatile programming language. The second challenge concerns how to provide this professional with a useful programming base for experimental planning and data analysis, already integrating his programming learning with experimental planning theory.

In the field of experimental planning, computer programming has many applications. Especially, among the countless programming languages available today, the Python language [10] is highlighted. Python is one of the most widely used languages in the world [11], due to its versatility, its simplicity of learning and its power. The authors in [11] highlight the following characteristics of this language that are ideal for novice programmers: (i) ease of learning, (ii) existence of an interactive shell that allows the programmer to run the program in an easy and fast way, (iii) existence of several code libraries that facilitate the learning process, (iv) portability and (v) installation flexibility, being able to be installed on any operating system. Therefore, to solve the first challenge, solutions using the Python programming language are very welcome, as they facilitate learning.

As for the second challenge, it is important to provide the professionals of the fourth technological age with notions of experimental planning and data analysis, together with their computer programming knowledge. In this sense, a simple application that can be developed with basic programming knowledge is the one that generates randomized experimental sketches, based on the Completely Randomized Design (CRD) [12]. The CRD is a methodology used to ensure that the data is obtained to provide a correct analysis, leading to valid conclusions regarding the problem under study. Thus, it aims to minimize the set of random effects in an experiment and seeks to minimize the random variation. The CRD is the simplest of the experimental designs and involves two basic principles of experimentation: repetition and randomization. Thus, it is assumed that local conditions are homogeneous and have no significant effect on treatments, therefore, local control is not necessary. It is widely used in laboratory tests (test benches), tests in plant nurseries and in tests with vases, carried out in greenhouses, in which the experimental conditions can be perfectly controlled. The elaboration of an experimental sketch using a Python program is a safe and effective way to guarantee the randomization of a study.

In line with the previously mentioned challenges, the objective of this work is to present and discuss a Python language computer programming short course with application in the context of the CRD, and its construction methodology. In addition, the results of this short course, which took place at the 1st Brazilian Conference on Experimental Planning and Data Analysis (ConBraPa 2020), are discussed. This work is organized in four other sections, in addition to this introduction. Section 2 presents the methodology for the development of this work and construction of the short course. Section 3 presents the implementation of the short course. Section 4 presents an assessment of the results of the short course. Section 5 concludes this work.

2. Methodology

This section presents the methodology used in the development of this research. From the point of view of the classification of the research, the taxonomy presented by Vergara [13] was taken as a basis, which qualifies it in relation to two aspects: as to the ends and as to the means. As for the ends, this research can be considered exploratory. Exploratory, as there is a lack of studies on the use of the Python language applied to the randomization of experimental designs. As for the means, the research can be classified as field research, as it took place through field research with the participants of the short course to collect primary data.

For the construction of the short course, a series of steps was carried out in sequence, based on the authors' experience with teaching in computer programming using the Python language. These steps are presented below.

Step 1: Establishment of the requirements of the short course. At this stage it was established that the duration of the short course is two hours, limited by the conference ConBraPa 2020. The display of the short course has been defined as asynchronous online mode. The student's profile was defined as a lay person in computer programming and in Python language, with no prerequisites necessary to take the course. The learning objective of the course is stated as “being able to write the final program (FP)”. Where FP is a Python computer program capable of making randomized experimental sketches. We considered Python version 3.7.8 in this short course.
Step 2: Survey of the requirements of the final program (FP). In this stage, the requirements to which the final program must meet were raised. What are its functionalities and attributions?
Step 3: Implementation of the final program (FP) in Python, meeting the requirements of the previous step, in the simplest possible way, to facilitate learning.
Step 4: Survey of the basic programming concepts in Python involved in each part of FP.
Step 5: Elaboration and implementation of small Python code examples about each basic concept raised in the previous step.
Step 6: Analysis of the code examples from the previous step to identify the minimum concepts necessary for their understanding and listing these concepts in a screenplay.
Step 7: Preparation of the screenplay for the short course and the slides.
Step 8: Recording the short course.
Step 9: Editing the short course.
Step 10: Preparation of a questionnaire to assess the learning and quality of the short course.
Step 11: Display of the short course and collection of responses to the questionnaires.
Stage 12: Writing of the present work and consolidation and presentation of results.

3. Implementation of the short course

This section presents the short course, its code examples and discusses its realization, all according to the methodology presented in Section 2.

3.1. Initial considerations

No prior computer programming knowledge is required to complete this short course. Specifically, in the first stage of the course, the student learns the basics about computer programming using Python, which involves the following topics: Variables, Data Types, Print/Input, Logical and Conditional Structures, Repeating Structures, Tuple and Lists Manipulation, Functions, Modules and finally Reading and Writing to files. All these topics are presented with examples in the short course and support the second stage of the course. This second step aims to demonstrate the applicability of the Python language in experimental planning, specifically in the CRD.

3.2. Presenting the final program

Following step 2 of the methodology, the following requirements for the final program (FP) were raised:

Requirement 1: Receive, as input data entered by the user, (i) the number of treatments and (ii) the number of replicates.
Requirement 2: Build a randomized experimental sketch, with a control group.
Requirement 3: Write the suggested sketch in a text file.
Following step 3 of the methodology, the final program implemented in Python is shown in Figure 1.

```python
import random

def createsketch():
    ntreatments = int(input("Type the number of treatments: "))
    nreplicates = int(input("Type the number of replicates: "))

    lista = []
    for i in range(0, ntreatments):
        for j in range(0, nreplicates):
            lista = lista + ["T%d.%d" % (i, j)]

    for i in range(0, nreplicates):
        lista = lista + ["TC.%d" % (i)]

    random.shuffle(lista)

    str_to_write = ""

    for i in range(0, nreplicates):
        for j in range(0, ntreatments):
            str_to_write = str_to_write + lista.pop() + " "
        str_to_write = str_to_write + "\n"

    arq = open("sketch.txt", "w")
    arq.write(str_to_write)
    arq.close()

createsketch()
```

Figure 1. Final Program in Python

Following step 4 of the methodology, the following topics were raised as basic concepts necessary to understand the main program: (i) First steps and Print; (ii) Variables and Data Types; (iii) Input; (iv) Functions; (v) Modules; (vi) Logical and Conditional Structures; (vii) Repetition Structures; (viii) Manipulation of Tuples and Lists; (ix) Reading and Writing to files.

3.3. Python code examples

Following a step 5 of the methodology, Python examples were implemented for each of the basic concepts mentioned in the previous section. The relevant ones related to step 6 of the methodology are identified for each figure.

```python
print("Hello world")
print("Igor")
print("2020")
print("Jose")
```

Figure 2. Example - (i) First steps and Print

In Figure 2, it is necessary to discuss the use (call) of a function (without the need for a declaration of functions yet), to understand how the program works. It is also necessary to explain the concept of character string, stating that character strings (text information) are written in quotation marks. A brief notion of the sequential flow of program execution is also needed.

```
x = (1+1) * 2
y = (10%3)
print(x)
print(y)
```

Figure 3. Example - (ii) Variables and Data Types

In Figure 3, it is necessary to discuss variables and the assignment of values to variables. The basic operators (sum, multiplication, division and integer division, modulus and exponentiation should be discussed. Operations between constants and variables should also be discussed. Precedences between operations using parentheses should also be discussed. The main idea in this example is to teach how to use Python as a simple calculator. It is important to mention that this code example is very changeable, and that during the display in the realization of the short course, several changes must be made in the code, showing the results of each change on screen to the students.

```
v = int(input("Insert a value: "))
x = int(input("Insert another value: "))

print(v + x)
print(v - x)
print(v * x)
print(v / x)
```

Figure 4. Example - (iii) Input

In Figure 4, it is necessary to discuss the use of the input function, as well as the conversion between the text data type to the integer data type. Concepts from the previous example are taken up, with respect to operations between variables and assignments. It is important to note that values are not reported directly in the Python console in the examples of this short course, in order to eliminate the need for explanations to understand the use of the console. All results displayed on the screen are performed using the print function, from the module being built in each example. For each example, a new empty Python module is initially considered.

```
def sum(x, y):
    return x + y

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    return a / b

def circular_crown_area(r1, r2):
    return circ_area(r1) - circ_area(r2)

def circ_area(r):
    return (3.14*(exponentiation(r, 2)))

def exponentiation(x, y):
    return x**y

print(circular_crown_area(2, 1))
```

Figure 5. Example - (iv) Functions

In Figure 5, the need for explanations regarding the use of the keyword “def” to define a new function is evident. The possibility of a function calling another function must also be raised and explained. The program flow, regarding the definitions of the functions and later their calls, must also be explained. It is necessary to mention that the functions are composed of a header / signature, parameters, and body of the function. The keyword “return” must also be explained, and the concept of the function's return value. A brief notion of scope of a function should also be given to students.

```
import math

def sqroot(x):
    return x**(1/2)

def sin_angle_deg(angle):
    return math.sin(math.radians(angle))

print(sqroot(4))
print(math.sqrt(4))
print(sin_angle_deg(30))
```

Figure 6. Example - (v) Modules

In Figure 6, it is necessary to explain the use of the “import” statement, giving as an example the “math” module. To mention that there are other modules available to be imported, and that the student can implement his modules and import them. To demonstrate that calls to the imported module’s functions must be made by mentioning the module name with a period and then the function name. Show that the import can search for specific functions within the imported module or even consider all functions, using “from math import *”.

```
def eval_number(x):
    if (x > 0):
        return "positive"
    else:
        if (x < 0):
            return "negative"
        else:
            return "zero"

print(eval_number(0))
```

Figure 7. Example - (vi) Logical and Conditional Structures

In Figure 7, it is necessary to first present and discuss the Boolean data type, Boolean, Comparison and Logical operators, and Boolean expressions that return true or false values. It is necessary to display on screen the result of several Boolean operations so that the student understands the value being evaluated by the conditional structure that will be presented in Figure 8. It is then necessary to also present the keywords “if” and “else”. In addition, it is necessary to mention that, in a function, multiple “return” statements may exist, but only one is executed, returning the result of the function immediately when it is executed. Finally, it is important to mention the possibility of nesting several conditional blocks.

```
import random

def accum_random():
    x = 0
    accumulator = 0
    while x < 10:
        k = random.randint(1,5)
        accumulator = accumulator + k
        x = x + 1
    return accumulator

def accum_even_numbers():
    accumulator = 0
    for i in range(2, 10 + 1, 2):
        accumulator = accumulator + i
    return accumulator

print(accum_random())
print(accum_even_numbers())
```

Figure 8. Example - (vii) Repetition Structures

In Figure 8, it is necessary to inform about the use of the “random” library, and the “randint” function. It is also necessary to explain the use of the keywords "while" and "for". Mention the concept of a list in Python. Explain the use of the “range” function. Mention the concept of control loop and accumulation variables.

```
T = (1, "Igor", 3, True, 5)
L = [1, "Igor", 3, 4.0, 5]
L2 = [10, 20, "10"]
L3 = L + L2

print(L3)

def returnList(a, b):
    if a > b:
        return list(range(b, a))
    else:
        return list(range(a, b))

print(returnList(4, 5))
```

Figure 9. Example - (viii) Manipulation of Tuples and Lists

In Figure 9, it is important to understand that tuple is an immutable type, while the list type is mutable. Both tuple and list data types are used to store a set of values that can be of varying types, including the list or tuple type itself. Simple operations with lists such as concatenation (sum) and scalar multiplication must be introduced.

```
def create_file():
    arq = open("text.txt", "w")
    arq.write("Igor")
    arq.close()

def read_file():
    arq = open("text.txt", "r")
    content = arq.readlines()
    arq.close()
    return content

print(create_file())
print(read_file())
```

Figure 10. Example - (ix) Reading and Writing to files

In Figure 10, it is necessary to clarify the procedure for creating a text file. Explain the function of the “open” function, which creates a text file in the directory of the module being executed, if the file does not exist, and opens the file in the indicated mode (the main modes are reading “r” or writing “w”), if the file already exists. It is necessary to explain the functioning of the “readlines” function, which obtains the complete contents of the file opened in a list within the program, allowing its processing. Emphasize the importance of closing the file after opening and carrying out the necessary operations with it.

3.4. Discussion

Following step 7 of the methodology, a short course screenplay was developed to be followed during the recordings, with the observations mentioned above about each code example. A 21-slide presentation was prepared in Microsoft Power Point to guide the execution of the short course. Approximately two slides per topic were used, constructed in a lean manner, to present the topic and the statement of the respective example. After the presentation of the topic and the statement of the example, the presenter switches to the IDLE screen, the basic Python IDE, where the presenter solves the code example. Subsequently, the presentation is resumed, returning to the main flow of the short course. This procedure is repeated for each of the 10 topics (9 introductory examples, plus the implementation related to the CRD).

Following step 8 of the methodology, the short course was recorded using the OBS Studio tool. Short clips were recorded, one for each topic, relating to the execution of the examples and the final program, all of such clips displaying IDLE on the screen. In addition, a larger and continuous clip was recorded with the slide show moments of the short course, displaying the slides on the screen.

Following step 9 of the methodology, the short course was edited using the Kdenlive tool. In the edition, we inserted the recordings of the clips of the topics within the larger clip of the slide show. The short course was also edited to remove some errors during the implementation of the examples and final program, however, some errors were left in the final recording, as they were considered didactic.

4. Results

This section presents the elaboration of the evaluation of the short course, and the consolidation of the results of the present work.

4.1. Development and application of the questionnaire

Following step 10 of the methodology, a questionnaire was prepared to assess the learning and quality of the short course. The questionnaire was composed of the following questions: (i) Did you know the Python programming language? (ii) Do you use randomness in your experiments? (iii) What tools do you use to produce

randomness? (iv) Within your working reality, would you use the tool for your experiments? (v) Was the course, in general, beneficial for you? Where questions (i), (ii), (iv) and (v) have a closed answer (yes or no) and question (iii) is open.

Following step 11 of the methodology, the short course was displayed to the students, and the subsequent application and collection of responses to the questionnaires happened. The short course took place during the 1st Brazilian Conference on Experimental Planning and Data Analysis (ConBraPA), held on November 24, 2020. The short course was displayed to students from 9:00 am to 11:10 am, in virtual mode. There was a presentation of the short course (approximately 2 hours) with the rest of the time dedicated to a live meeting with the participants (students and professor) to help students in their difficulties in learning. The questionnaire was applied during the meeting, therefore immediately after the end of the short course.

Following step 12 of the methodology, finally the consolidation and presentation of results and the writing of the present work occurred.

4.2. Analysis of results

The short course class was composed of 43 participants, however only 21 answered the questionnaire, totaling approximately 49% of the population. The topic was approached in a simple and direct way, stimulating the debate, to favor an exchange of information among participants. During the post-short course meeting, questions were raised by students regarding how to perform mathematical operations using Python. Another question raised by the students was "Does the function circ_area being defined after circular_crown_area make a difference?" (referring to the example in Figure 5). Another question was "for the list, is it necessary to write ‘10 + 1’ or could you simply write ‘11’?" (referring to the example in Figure 8). A third question was “What are the advantages of Python over R in statistical applications?”. Finally, a question about IDEs for Python development was made: "Is there any difference/advantage in using Python or Spyder?", In this case, by Python, IDLE was understood, since this was the Python development environment used in the short course.

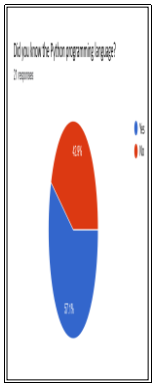

Figure 11. Result of question (i)

According to Figure 11, the survey showed that approximately 57% of the students had previous knowledge of the Python language.

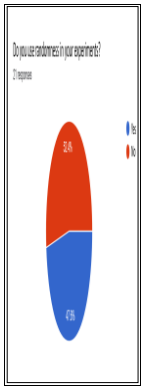

Figure 12. Result of question (ii)

In addition, when asked about the use of randomness, approximately 52% of respondents said they use this criterion in their work, according to Figure 12.

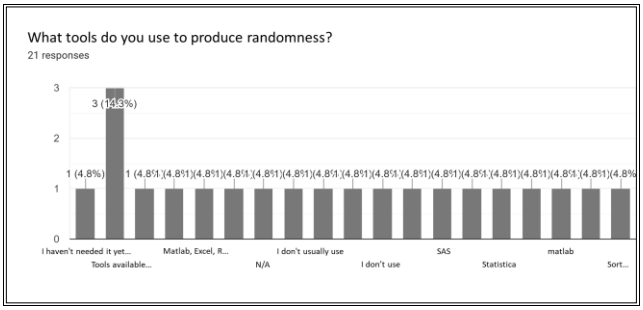

Figure 13. Result of question (iii)

In relation to the tools used to generate randomness in experiments, some software such as Matlab, Microsoft Excel, SAS, and STATISTICA have been reported, as shown in Figure 13. In Figure 13, several software had a single occurrence by the participants. The highest occurrence (3 occurrences) was for “tools available in Excel”.

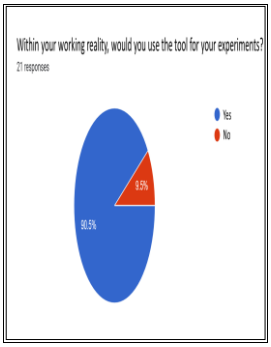

Figure 14. Result of question (iv)

Regarding the use of the Python tool in personal experiments, approximately 91% of respondents said they would use the tool, as shown in Figure 14.

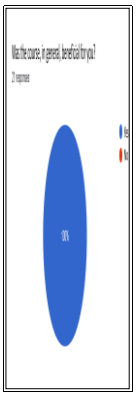

Figure 15. Result of question (v)

And finally, Figure 15 shows that, for all participants (100%), the short course was beneficial.

5. CONCLUSION

In line with the challenges previously mentioned in this work, a short course on computer programming in Python was developed with application in the context of the CRD, which took place at the 1st Brazilian Conference on Experimental Planning and Data Analysis (ConBraPa 2020). The short course, its construction methodology and the results achieved were discussed. The main results were: (i) participants learnt about basic programming logic, (ii) participants learnt about basic instructions in the Python programming language, (iii) participants learnt how to create a randomized experimental sketch in the context of CRD using Python, (iv) elaboration of a specific methodology for the construction of the short course. It is noteworthy that, through evaluation with a questionnaire, it was possible to conclude that 100% of the participants in the short course who answered the questionnaire evaluated the short course as beneficial. It is also noteworthy that approximately 91% of respondents said they would use the Python tool in their personal experiments. These numbers demonstrate an excellent use of the short course by the participants. As future work, the following three directions are suggested: (i) substantiate the methodology in the literature; (ii) re-plan the short course, based on the public's feedback, and the substantiation of the methodology; and (iii) apply the methodology in the elaboration of other short courses.

Acknowledgments

The authors would like to thank the ConBraPA 2020 team for the invitation and opportunity to hold the short course. Author Emanuele Nunes de Lima Figueiredo Jorge thanks Proppi IFRJ.

Conflict of interest: None

References

1. Bauer W, Hämmerle M, Schlund S, Vocke C. Transforming to a Hyper-connected Society and Economy – **Towards an "Industry 4.0**." Procedia Manuf. 2015;3(Ahfe):417-424. doi:10.1016/j.promfg.2015.07.200.

2. Gorecky D, Schmitt M, Loskyll M, Zühlke D. Human-machine-interaction in the industry 4.0 era. Proc - 2014 **12th IEEE Int Conf Ind Informatics,** INDIN 2014. Published online 2014:289-294. doi:10.1109/INDIN.2014.6945523.

3. Colistra G, Pilloni V, Atzori L. The problem of task allocation in the Internet of Things and the consensus-based approach. **Comput Networks**. 2014;73:98-111. doi:10.1016/j.comnet.2014.07.011.

4. 1. Sajid A, Abbas H, Saleem K. Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges. **IEEE Access**. 2016;4:1375-1384. doi:10.1109/ACCESS.2016.2549047.

5. Alves MP, Delicato FC, Santos IL, Pires PF. Architecture for Virtualization and Collaboration in Edge Computing: an implementation based on FIWARE building blocks. In: **ANAIS DO XXXVIII SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS**. Sociedade Brasileira de Computação; 2020:686-699.

6. Fatorachian H, Kazemi H. A critical investigation of Industry 4.0 in manufacturing: theoretical operationalisation framework. **Prod Plan Control.** 2018;29(8):633-644. doi:10.1080/09537287.2018.1424960.

7. Nelles J, Kuz S, Mertens A, Schlick CM. Human-centered design of assistance systems for production planning and control: The role of the human in Industry 4.0. Proc **IEEE Int Conf Ind Technol.** 2016;2016-May:2099-2104. doi:10.1109/ICIT.2016.7475093.

8. Gorecky D, Schmitt M, Loskyll M, Zühlke D. Human-machine-interaction in the industry 4.0 era. Proc - 2014 12th I**EEE Int Conf Ind Informatics**, INDIN 2014. Published online 2014:289-294. doi:10.1109/INDIN.2014.6945523.

9. Schuh G, Gartzen T, Rodenhauser T, Marks A. Promoting work-based learning through industry 4.0. **Procedia CIRP**. 2015;32(Clf):82-87. doi:10.1016/j.procir.2015.02.213.

10. van Rossum G, Drake FL. The Python Language Reference Manual. **Network Theory Ltd**.; 2011.

11. Orfanakis V, Papadakis S. Teaching basic programming concepts to novice programmers in Secondary Education using Twitter, Python, Arduino and a coffee machine. **Hell Conf Innov STEM Educ** [HiSTEM2016]. Published online 2016.

12. Ferreira PV. Estatística experimental aplicada às ciências agrárias. Maceió: **Edufal,** 1996.

13. Vergara, SC. Projetos e relatórios de pesquisa em administração. São Paulo: **Atlas,** 2010.