doi: 10.22363/2521-442X-2020-4–4-23-38
rudn.tlcjournal.org

# Original Research

## Programming for specific purposes in linguistics: A new challenge for the humanitarian curricula

by Alexey I. Gorozhanov and Innara A. Guseynova

**Alexey I. Gorozhanov** Moscow State Linguistic University a_gorozhanov@mail.ru
**Innara A. Guseynova** Moscow State Linguistic University ginnap@mail.ru
**Received** 24.07.2020 | **Revised** 4.11.2020 | **Accepted** 10.12.2020
**Recommended citation format:** Gorozhanov, A. I., & Guseynova, I. A. (2020). Programming for specific purposes in linguistics: A new challenge for the humanitarian curricula. *Training, Language and Culture, 4*(4), 23-38. Doi: 10.22363/2521-442X-2020-4-4-23-38

*The article describes the ways of teaching the linguistic students programming for specific purposes and deals with the global problem of training future teachers of a foreign language and culture in the context of digitalisation. The issues related to the linguistic component in teaching programming are also revealed in the framework of the paper. An innovative tutorial in programming on Python is described as an example, which is built on the classical principles of comparison, contrast and 'from simple to complex', and considers the following topics: basic operations on Python (Part 1), basic knowledge about the PyQt5 graphics library (Part 2) and introduction to developing web applications on the web2py framework (Part 3). The tutorial is provided with examples of programming code and screenshots of the application interfaces. Part 1 of the tutorial is mandatory for learning, as it is intended for developing basic skills of programming (active learning) that can help the students to understand the structure of the algorithms analysed in the 2nd and the 3rd parts (passive learning). The first two parts of the tutorial were tested by the students of the German Faculty at Moscow State Linguistic University (about 150 people in total in the period from 2018 to 2020). Most of the students successfully coped with the Part 1 tasks, which were actually the goal of the work. Though a deep understanding of the materials of Part 2 was achieved by 5% of the students, this could be considered a success, especially since some of them used programming methods for their research. The conclusion draws upon the results of the testing of the tutorial. The proposed approach to teaching the linguistic students skills in programming for specific purposes seems to be promising and effective.*

**KEYWORDS:** *applied linguistics, programming for specific purposes, Python programming language, PyQt5 graphics library, web2py framework*



### 1. INTRODUCTION

The humanities scientists inevitably face the need to process large amounts of data in the course of their work in the context of the global pivot to *digital reality*. Modern linguistics successfully applies methods of computer analysis and uses ready-made databases (text corpora). Nevertheless, ready-made solutions, firstly, can be ex-

doi: 10.22363/2521-442X-2020-4-4-23-38
rudn.tlcjournal.org

pensive, and secondly, they are not universal, so there remains a need for the development of new software products, in which the active participation of a linguist is strictly necessary.

Among the analysed studies on the topic, those that prioritise the technological component dominate, without taking into account the role and importance of the professionally oriented (in our case linguistic) component (Ardimento et al., 2020; Egan & McDonald, 2020; Perez et al., 2020; Rashkovits & Lavy, 2020; Papadakis, 2018; Orfanakis & Papadakis, 2016). Modern research should, in our opinion, have a real interdisciplinary basis, and involve the search for intersections of the practical interests of various sciences. We believe that information technology and linguistics have the potential to form the integrative foundations of a new model of academic education. The educational model, which has an integrative basis, acquires stability, contributing to the preservation and development of scientific and educational schools, maintains the continuity of scientific and educational processes, preserves the individual learning path of the students, realises the immersive principle, which can be achieved only in multimodal or interdisciplinary educational programmes, has the parameter of consistency and requires constant updating of both the content and the software (Amelkin & Guseynova, 2020). As a result of technical and humanitarian interaction, intellectual software products are created to meet the challenges of our time. Obviously, these products are untenable without high-quality content; for this reason, the role of a linguist who has knowledge of the world, language and language use is a priority in the process of interacting with a technical specialist. The linguist sets the vector for the development of the information environment.

In this regard, it seems important to pay attention to a special area of knowledge – *professionally oriented programming* or *programming for specific purposes*. The goal of teaching programming for specific purposes, on the one hand, is to form a general understanding of the main tasks that a modern linguist faces and which require the creation of software solutions (general developmental goal), and on the other hand, to relate knowledge about specific techniques that can already be introduced into practice (practice-oriented goal). Taken together, these goals should ensure an understanding of the process of developing specialised software, which will allow the future specialist to act, if not in the role of a programmer, then in the role of a project manager or/and of a developer of technical specifications.

*'Taken together, these goals should ensure an understanding of the process of developing specialised software, which will allow the future specialist to act, if not in the role of a programmer, then in the role of a project manager or/and of a developer of technical specifications'*

Analysis of subject-specific literature, as well as Internet resources showed that the terms *professionally oriented programming* or *programming for specific purposes* are new ones. And this can be explained at least by the fact that these terms are relative to a certain extent, since any specific programming code is focused on solving a specific problem. However, we consistently use these terms to determine a specific area of knowledge and to separate it from programming in general as a broader area. Actually, the terms were invented by the authors by analogy with such widespread terms as *English for specific purposes* or *professionally oriented translation*.

In this case we are talking about programming not for the applied linguists, for whom such disciplines are traditionally included in the curriculum, but primarily about programming for the future teachers of a foreign language and linguistic researchers (germanists, discourse analysts, etc.), who, in principle, do not study it as major or minor subjects at the Bachelor and Master levels.

## 2. THEORETICAL BACKGROUND

As an example of software products created by the linguists, who are teachers of foreign languages, we can cite over 30 online courses devel-

oped in the Laboratory of Fundamental and Applied Issues of Virtual Education at Moscow State Linguistic University. The component of the programming for specific purposes lies here in the fact that when creating the mentioned online courses, not only ready-made LMS Moodle templates were used, but modifications were made to the LMS programming core in order to adapt this universal system for the needs of teaching foreign languages (Guseynova et al., 2019).

Based on this vision, let us outline the range of tasks in the field of programming for specific purposes now. In our opinion, this includes automatic and automated text processing, as well as automatic and automated creation of digital educational materials in foreign languages. The former can serve the purpose of formatting, tokenisation, text analysis, including the interpretation of the works of art, as well as the primary preparation of text arrays for solving the second circle problems – optimising the process of making electronic textbooks and tutorials, as well as online courses in foreign languages.

Moreover, we have to decide which programming language (and maybe several languages) one needs to learn in the framework of programming for specific purposes. In our opinion, the designated general developmental and practice-oriented goals narrow the range of studies to one universal, popular, multi-paradigm and comprehensible programming language, which should be not only a teaching tool, but also a tool used in professional practice. Python was chosen as such a programming language, which quite accurately meets all these requirements. In addition, Python is able to create both stand-alone programmes and graphical applications (GUI programmes), and web applications, although the latter two varieties require studying additional libraries and frameworks.

The Python programming language is a very flexible tool that enables developing functional applications by a single programmer or small teams (Morawiec et al., 2020). A large amount of data display and analysis software tools are currently being developed in Python (Romero et al., 2020; Medina-Ortiz et al., 2020; Sun & Kennett, 2020; Cid-Fuentes et al., 2020; Garcia-Fossa et al., 2020). According to the TIOBE ranking in 2020, Python is the third most popular programming language in the world, traditionally behind C/C++ and Java (TIOBE, 2020). Considering that Python was developed in 1991 and C and Java in 1972 and 2005 (C++ in 1983), respectively, the third place is an outstanding result. Further, Python is widely used for natural language processing (Qi et al., 2020; Burns, 2020; Japhne & Murugeswari, 2020; Senekal & Kotzé, 2019). Thus, we can conclude that Python is a relevant tool both for universal use and for the field of linguistics.

## 3. MATERIAL AND METHODS

Another problem is the creation of standard demonstration programmes, which, after the didacticisation, will become learning materials for teaching programming for specific purposes. To solve this problem, we developed and tested the tutorial *Introduction to Programming. Python*, which is intended for the acquisition of basic knowledge in the field of programming. The tutorial includes three parts: Basic Python (Sections 1-6), Introduction to the PyQt5 Graphics Library (Sections 7-10), and Introduction to the web2py Framework (Sections 12-17).

The parts devoted to the PyQt5 graphics library and web2py framework are more complex than the first part and are intended for those who would like to get acquainted with the use of Python in the field of GUI applications and web applications. Such an increase in the level of complexity is justified, in our opinion, since the tutorial is not intended to be used for self-study work outside the classroom.

According to the author's intention, Part 1 should be mandatory for studying at the undergraduate level, and parts 2 and 3 should be additional ones, designed to merely demonstrate the capabilities of Python at the undergraduate level or for detailed study at the graduate level, as well as for use in writing the graduation papers and theses by the students at both undergraduate and graduate (or postgraduate) levels who are interested in programming for specific purposes.

doi: 10.22363/2521-442X-2020-4-4-23-38
Volume 4 Issue 4, 2020, pp. 23-38
rudn.tlcjournal.org

*'As an example of product development on web2py, we can cite a wiki web application, which serves to teach linguistic students to work together on composing a text with the given parameters, as well as to teach them the ability to assess the texts written by their colleagues'*

The choice in favour of PyQt5 and web2py was not made by chance. PyQt5 is a popular graphics library which is the Python version of the Qt graphics library for the C programming language (Python itself is also written in C). With the help of PyQt5 the graphical user interfaces for software development environments are designed, as well as big data analysis, visualisation of photometry data, spectroscopy, etc. (Combrisson et al., 2019; Reddy et al., 2017; Titze et al., 2018; Liu et al., 2011; Chabaud et al., 2018).

The web2py framework is interesting, firstly, because it works with the Python programming language. And if Python completely suits as a tool for teaching programming, then web2py is an intuitively transparent tool for creating web applications, built on the concept of *rapid development* (Di Pierro, 2013, p. 21). The concept foresees practically endless modifications of ready-made template solutions instead of programming a web application from the very beginning. In this regard, the developer saves a lot of time focusing not on printing the code, but on solving the problem in the most efficient way; in other words, he or she applies first of all the professional competences, in our case – the linguistic one.

Secondly, the web2py installation package comes by default with a local web server, which makes it possible to make and to test products offline. In fact, this means that the full cycle of software development and deployment can be performed without using the Internet. At the same time, this does not exclude the use of the created products through the local network of the university. Digital learning materials created on this framework can be used in conditions of limited access to high-speed Internet, which seems to be relevant for the linguistic students who use, in our observation, mainly a mobile connection outside the university. This service helps motivate the students to learn foreign languages and cultures, because there is an additional incentive that forms a conscious attitude to the need to regularly develop their linguistic skills *always and everywhere*.

Finally, despite the fact that the framework is inferior in use to such products as Django or Wordpress, it is used in the academic environment, including for creating applications for data processing and data visualisation (Miles, 2016; Di Pierro, 2011; Alberti & Zuccon, 2011; Burger et al., 2013). As an example of product development on web2py, we can cite a wiki web application, which serves to teach linguistic students to work together on composing a text with the given parameters, as well as to teach them the ability to assess the texts written by their colleagues (Gorozhanov, 2018a, p. 356-361). Along with the linguistic competence, the social competence plays a significant role, which is aimed at learning to work in teams. This is facilitated by the joint work associated with the generation of a text, taking into account its genre specificity, which is, first of all, in global text structures. So, the students can train the creation of persuasive texts (advertising, scientific, journalistic, etc.) or descriptive texts with a narrative structure (biography, composition, etc.). Collaboration leads to intense creativity that fosters interest in learning foreign languages and cultures.

Using web2py allows one to study the Model-View-Controller (MVC) web applications and the principles of working with databases, in order to ensure move on transition, if necessary, to the projects in other frameworks or in the PHP programming language.

In this paper, the content of the first and second parts of the tutorial is considered, since they have already passed the testing stage. The materials devoted to learning how to use the web2py framework require, in our opinion, a separate detailed study, which is our further goal.

In general, the described tutorial uses the following techniques (methods) for teaching the linguistic students programming for specific purposes.

1. The similarity of the studied programming languages and the natural languages is constantly emphasised, the ability to read the programming code as a coherent natural text is instilled.

2. When working on difficult tasks, a relatively simple template is taken as a basis and then is modified step by step, gradually becoming more complex and approaching the solution of the linguistic task. In this regard the students are taught to see the solution to the problem as a sequence of many tiny steps.

3. After getting acquainted with the basic operations in the programming language, the students immediately proceed to solving professionally oriented (linguistic) tasks.

4. In case the solution is technically difficult, the students are invited to act as project managers (linguistic advisors) and draw up a technical task for the programmer, using their knowledge of programming to understand what can be done and what cannot be done by the programmer.

## 4. TRAINING CONTENT

### 4.1. The first part of the tutorial

The tutorial *Introduction to Programming. Python* is built on the classical principle *from simple to complex*. It includes three parts, each of which is divided into sections. The learning materials are supplied with numerous illustrations (about a hundred figures) and examples of the programming code (listings).

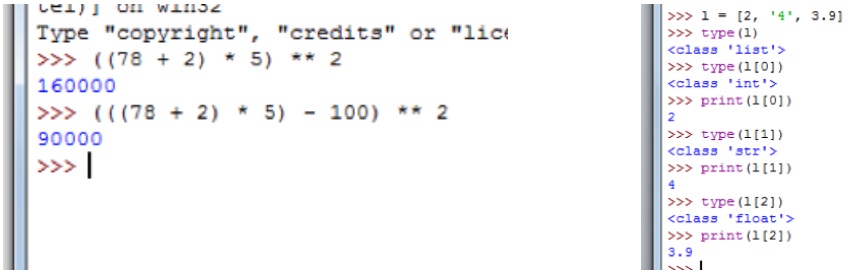
The first part, which deals with the basic operations on Python, begins with the instructions on how to install the Python IDLE development environment on the user's computer. The students are introduced to the interactive Python Shell, as well as the instruction on how to work with Python from the command line interface (CLI) using the *print()* command.

At the end of the section, there are some questions for self-control: (1) How can Python be installed? (2) What is IDLE? (3) What is Python Shell? (4) What are the principles of programming in the command line interface? (5) How can you verify that Python is installed correctly? (Gorozhanov, 2018b, p. 11).

The second section of the first part of the tutorial covers basic arithmetic operators and introduces the concept of a variable. Among the arithmetic operations addition ('+'), subtraction ('-'), division ('/') and multiplication ('*') are distinguished, then exponent ('**'), floor division ('//'), modulus ('%') and grouping using brackets (Gorozhanov, 2018b, p. 12-13). Since all these operations are written in Python intuitively, their understanding is not difficult for the linguistic students.

Working with the materials of the tutorial, the students perform all these actions by themselves and check with the results shown in the illustrations. The concept of a variable is introduced and the Python Reserved Keywords that cannot be used as such are listed (*and*, *as*, *assert*, *break*, *class*, *continue*, etc.). The symbol '=' is introduced, which is the assignment operator in Python. The *type()* method is described for defining the type of the variable (Gorozhanov, 2018b, p. 13-15). The types of variables *integer*, *float*, *string* are given. It is explained that Python reassigns variable types automatically. A sequence data type *list* is introduced: *l = ['Ivan', 'Maxim', 'Yaroslav']*, and it is emphasised that the elements of the list are numbered from zero, not from one (Gorozhanov, 2018b, p. 16).

Thus, in the second section of the first part of the tutorial, very important fundamental materials are given that form the basis for further training in programming, including professionally oriented ones. Intuitive phenomena are described according to the principle of comparison (the operators of addition, subtraction, multiplication and division are similar to those used on modern calculators), differing phenomena are described according to the principle of contrast (the symbol '=' is the assignment operator, not the equality sign). Increased attention is paid to phenomena that differ from the usual ones in the school course of mathematics. The screenshots illustrating the learning materials allow the students to compare their result with a sample (Figures 1-2).




*Figures 1-2*. Using the brackets and Demonstration of a list containing three different data types (Gorozhanov, 2018b, p. 13-17)

Just like after the first section, there are questions for self-control, in particular: (1) What operators are used to perform the simplest arithmetic operations in Python? (2) What is modulus? (3) What is floor division? (4) What are data types in Python? (5) What are the Reserved Keywords? (6) What is a Python list? (7) Can data of different types fit in one list? (8) What does *l[0]* mean for the list *l*? (9) How can one determine the type of a variable? (Gorozhanov, 2018b, p. 17-18).

Section 3 of the tutorial focuses on creating the first Python programmes in separate files. In Python IDLE a single-line file *print('Hello, world!')* is created. The programme is saved and executed. The interactive Python Shell outputs the result (Gorozhanov, 2018b, p. 19-20).

Then, the Python syntax is described. Here, the students are told that high-level programming languages have a syntax like natural languages (taking into account the specifics of the audience). The concept of a comment is introduced: *'A line comment is a single line, at the beginning of which one or more '#' characters are put. A block comment can span multiple lines, which are placed between triple single straight quotes (''') or triple double straight quotes (""")*. *Comments are not taken into account during programme execution – they are meant only for the human'* (Gorozhanov, 2018b, p. 21).

In the course of studying the syntactic structure of the programming language, the ability to work with error alerts is developed (Figure 3).

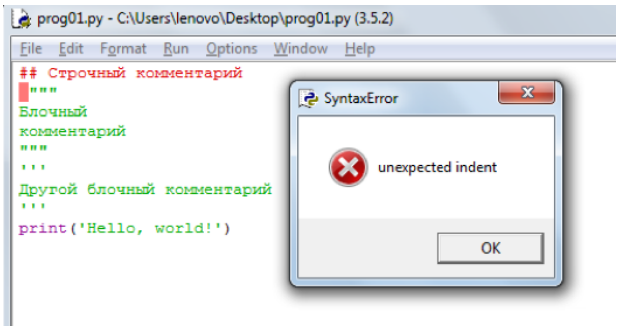



*Figure 3*. Syntax error alert – an unexpected indent (Gorozhanov, 2018b, p. 22)

The next step is to write an interactive programme that would receive data from the user, process it and produce the output: *'Let's create a programme that would receive an integer from the user and determine if the entered number is even, odd, or it is no integer at all'* (Gorozhanov, 2018b, p. 22). To solve this problem, the conditional expression *if...else* is introduced. Since the meaning of the conditional expression is intuitive (*if*, *then*), we can completely focus on solving the problem. The latter is solved by demonstrating to the students the process of analysing the data received

from the interpreter: it is shown to students that one step in solving the problem, after which an error occurs. The error is analysed, corrected, and then the next step of the solution is revealed, which also leads to an error, etc.

The input is organised using the *input('Enter an integer')* method. The entered data is validated for even/odd parity. But the user enters characters other than numbers. This results in an error (Figure 4). A detailed explanation on how to read the error alert correctly is given to the students. The way of exception handling with *try...except* is demonstrated (Figure 5). Then the task becomes more complex: *'For example, we might be interested in whether the user input is a number less than 18, a number over 60, or some other number. Such a check may be related to the person's age'* (Gorozhanov, 2018b, p. 25). This allows the use of *if... elif...else* conditional statements. The user enters a number less than zero, which results in an error. Additional validation is added to the code: *'Inside each if…elif…else statement a so-called 'nested' statement can be placed. For example, if the entered age is less than 18, but at the same time it is less than zero, then the entry may be considered invalid'* (Gorozhanov, 2018b, p. 25) (Figure 6).

```
Введите целое число: ффф
Traceback (most recent call last):
  File "C:\Users\lenovo\Desktop\prog01.py", line 2, in <module>
    inp = int(inp)
ValueError: invalid literal for int() with base 10: 'ффф'
>>>
```

*Figure 4*. Incorrect input error alert (Gorozhanov, 2018b, p. 24)

```
prog01.py - C:\Users\lenovo\Desktop\prog01.py (3.5.2)
File  Edit  Format  Run  Options  Window  Help
inp = input('Введите целое число: ')
try:
    inp = int(inp)
    if inp % 2 == 0:
        print('Ваше число четное')
    else:
        print('Ваше число нечетное')
except (ValueError):
    print('Вы ввели не целое число')
```

*Figure 5*. Handling exceptions (Gorozhanov, 2018b, p. 24)

```
prog01.py - C:\Users\lenovo\Desktop\prog01.py (3.5.2)
File  Edit  Format  Run  Options  Window  Help
inp = input('Введите целое число: ')
try:
    inp = int(inp)
    if inp < 18:
        if inp < 1:
            print('Внимание! Число не может быть меньше единицы')
        else:
            print('Введено число меньше 18')
    elif inp > 60:
        print('Введено число больше 60')
    else:
        print('Введено число в промежутке от 18 до 60 включительно')
except (ValueError):
    print('Вы ввели не целое число')
```

*Figure 6*. Nested *if* statements (Gorozhanov, 2018b, p. 26)

The following questions are given for self-control: (1) How can one run a programme in IDLE? (2) What are the specifics of the syntax in Python? (3) How is a comment formatted and what is its function? (4) How can one use conditions in Python? (5) How does the *exception handling* work? (6) What is the *if...elif...else* model? (Gorozhanov, 2018b, p. 26).

Section 4 deals with the *for* and *while* loops. The *for* loop is illustrated by displaying the contents of the list *lst = [3, 6, 1, 0, 13, 70]*. At the same time, it is demonstrated by means of natural language, as the interpreter thinks: *'While there are elements in the list 'lst', I take the zero element and output it. I check if there are more elements; it is true, so I take the first element and output it, etc.'* (Gorozhanov, 2018b, p. 27). It shows how the list items are sorted and introduces a professional component: basic string processing routines. By analogy with a list of integers, the *for* loop displays the *Hello!* string, letter by letter. It is emphasised that in Python all strings are lists by default.

Further, we enhance the professional component using step-by-step problem solving. The string *Hello!* is sorted using the *sorted()* method, but the output is unusual: *!Hello*. The students are encouraged to comment on why the sorting is done this way. The *for* loop and *char()* method print the first 300 elements of the character table. In the table, the symbol '!' has position 33, and only then the uppercase Latin letters and lowercase Latin letters follow. The sorting logic becomes clear. In addition, the *for* loop is used with the numeric range: *for char in range (0, 300):* (Gorozhanov, 2018b, p. 29).

The functioning of the *while* loop is revealed by the example of solving the following problem: *'The programme randomly 'guesses' a number from 1 to 9 inclusively and prompts the user to guess it. The game ends only after entering the guessed number'* (Gorozhanov, 2018b, p. 30). When presenting this problem, the method of analysis is used, i.e. the students are invited to study in detail the ready-made solution (Figure 7).

Listing 1

```
1.  import random
2.  guess = random.randint(1,9)
3.  print("Загадано число от 1 до 9 включительно.") ## A number from 1 to 9 is guessed
4.  iterVar = True
5.  while iterVar:
6.      name = input('Пожалуйста, введите это число: ') ## Please enter this number
7.      if name == str(guess):
8.          print('Правильно!') ## Correct
9.          iterVar = False
10.     else:
11.         print('Попробуйте снова...') ## Try again
12. print('Игра окончена') ## Game over
```

*Note: Here and further the translation of the text from Russian into English is given as a comment.*

*Figure 7*. Code of the programme 'Guess Number' (Gorozhanov, 2018b, p. 30)

Besides demonstrating the functioning of the *while* loop, this example introduces the concept of importing a module (*import random*) and a Boolean variable. Here is the complete comment on the code from listing 1.

*'Line 1 imports the 'random' module, which is responsible for generating random numbers (various Python modules must be included manually to save memory). Line 2 assigns to the 'guess' variable an integer value between 1 and 9 inclusively.*

*Line 4 declares the boolean variable 'interVar' with the value 'True'. Line 5 initiates the while loop, whose code is in lines 6-11. The loop is running as long as 'interVar' is 'True'. Each time it passes through the loop, the programme asks for an input from the user and compares it to the value of the 'guess' variable, transforming it to 'string'. If the values match, the message is displayed: 'Correct!' The 'interVar' variable becomes 'False', and the loop is broken. The programme goes to line 12 and displays the end of the game. If the user fails to guess the number, the loop repeats'* (Gorozhanov, 2018b, p. 31). Similar to all the previous sections the questions for self-control follow, but due to the limited scope of the paper we will not give them here and further.

Section 5 is devoted to working with functions. Following the principle *from simple to complex*, the first function contains just one line of code: *def greeting(): print('hello')* (Gorozhanov, 2018b, p. 31). It shows how a function can be called within a programme. The function receives one argument (Figure 8). The number of arguments increases to three (Figure 9).

Listing 2

```
1.  def greeting(name):
2.      print('Привет, %s!' % name) ## Hello
3.  greeting('Петя') ## Peter
```

*Figure 8*. A simplest function with one argument (Gorozhanov, 2018b, p. 32)

Listing 3

```
1.  def exampleCount3(num1, num2, num3):
2.      return num1 * num2 * num3
3.  print(exampleCount3(1, 2, 3))
```

*Figure 9*. An example of a function with three arguments (Gorozhanov, 2018b, p. 33)

To demonstrate the solution of a real task the following problematic situation is suggested to the students: *'Let's solve a problem from the field of cryptology, associated with the so-called 'Caesar cipher'. The essence of this cipher is that each letter of the source text is 'shifted' by a specified number of positions to the right or left alphabetically. For example, if a shift to the right by five is selected, then the Russian 'А' will be replaced with 'А(0), Б(1), В(2), Г(3), Д(4), Е(5)'. If the letters end with the shift, then the alphabet starts anew, for example, 'Ю' is replaced by 'Ю(0), Я(1), А(2), Б(3), В(4), Г(5)'* (Gorozhanov, 2018b, p. 33).

Five programmes are given step by step, the last of which is the complete solution. First, the function *def alphOut():* is built, which displays the Russian alphabet in capital letters (except for *Ё*): *АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ* (Gorozhanov, 2018b, p. 33). This function is modified and displays the original alphabet and the alphabet with a shift to the right by three positions – a kind of encryption key. Moreover, in the shifted alphabet, the letter *А* stands in front of the letter *Я* (left), and after the letter *Я* stands the letter *А* (on the right) (Gorozhanov, 2018b, p. 34). In the following programme, the user can set the shift (including the negative value), thanks to the previously learned *input()* method.

The fourth programme receives from the user a text that must be encrypted. In this case, the default shift value is three. The text is capitalised. All letters of the received text are enumerated and checked for code compliance. If these are capital letters of the Russian alphabet (except for *Ё*), then

they are shifted by three positions. The rest of the characters do not change (Gorozhanov, 2018b, p. 35-36). The final programme contains two functions that encode and decode the text with a default shift of three (Figure 10).

Thus, the result of studying Section 5 is the creation of a fully functional application that solves a linguistic problem. In keeping with the principle of repetition, the programme uses conditional expressions and loops, and the methods learned before.

The last section of the first part of the tutorial covers the class phenomenon and the basic principles of working with files. The classes are seen as a tool for organising a number of functions into a single block of code for subsequent reuse. First of all, the programme from the previous section is formed into a class. As a result of using a previously prepared class by importing it, the programme becomes compact. All functions become class methods (Figure 11).

The work with files is revealed by using the same encoding and decoding programme. The materials to be encoded are read from a text file, and the result is also placed in a newly created text file. Here only the methods for working with files *open (), read (), write (), close ()* are new.

As a result, such complex materials as functions and classes are considered practically on the same example, with the gradual introduction of some new code fragments.

Listing 4

```
def textToCode(shift):
    codeText = ''
    text = input('Введите исходный текст: ').upper() # Input the text to encode
    for letter in text:
        if ((ord(letter)) > 1039) and ((ord(letter)) < 1072):
            if (ord(letter) + shift) < 1040:
                codeText += chr(ord(letter) + 32 + shift)
            elif (ord(letter) + shift) > 1071:
                codeText += chr(ord(letter) – 32 + shift)
            else:
                codeText += chr(ord(letter) + shift)
        else:
            codeText += letter
    print(text)
    print('=>')
    print(codeText)
def codeToText(shift):
    text = ''
    shift = shift * –1
    codeText = input('Введите закодированный текст: ') # Input the text to decode
    for letter in codeText:
        if ((ord(letter)) > 1039) and ((ord(letter)) < 1072):
            if (ord(letter) + shift) < 1040:
                text += chr(ord(letter) + 32 + shift)
            elif (ord(letter) + shift) > 1071:
                text += chr(ord(letter) – 32 + shift)
            else:
                text += chr(ord(letter) + shift)
        else:
            text += letter
    print(codeText)
    print('=>')
    print(text)
textToCode(3)
codeToText(3)
```

*Figure 10*. Text encoding and decoding programming code (Gorozhanov, 2018b, p. 36-37)

Listing 5

```
1.  import caesar
2.  a = caesar.Caesar()
3.  a.textToCode(3)
4.  a.codeToText(3)
```

*Figure 11*. Class import example (Gorozhanov, 2018b, p. 39)

**4.2. The second part of the tutorial**

Section 7 of the tutorial, which opens the second part, explains the procedure for installing the PyQt5 graphics library (Gorozhanov, 2018b, p. 43-44). The material of Section 8 is devoted to the first window application written in Python and PyQt5, which must get a certain string from the user and display its length in characters. The interface is created in Qt Designer (it is included in the PyQt5 installation package by default) and consists of three widgets: input field (*QLineEdit*), button (*QPushButton*) and label (*QLabel*) (Figure 12).

In the command line, which was covered in the first part of the tutorial, the interface file with the *ui* extension is formatted to the *py* file. The latter opens in Python IDLE and runs. The resulting programme consists of 52 lines. The remaining pages of the section are devoted to the analysis of the resulting programming code, as well as to the analysis of the ways to modify it (Gorozhanov, 2018b, p. 53-56).

In Section 9, functions for handling the user input are added to the programme file. Note that the functions are added directly to the interface file, which is a good (but not the best) solution within the framework of creating the very first window application, when, as below, a more efficient way will be considered – to create a separate data processing file. So, the function *myFunction()* must be activated by clicking a button and perform the following three actions: *'to take the text entered by the user from the input field, calculate its length, and display the result in the label'* (Gorozhanov, 2018b, p. 57) (Figure 13).

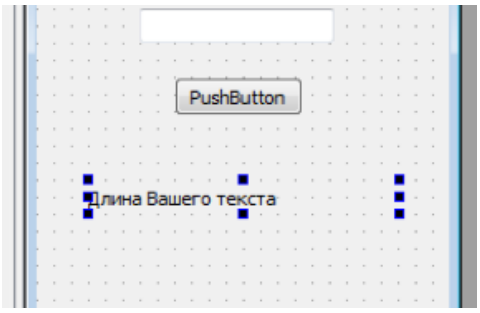



*Figure 12*. The interface of the application made in Qt Designer (Gorozhanov, 2018b, p. 48)

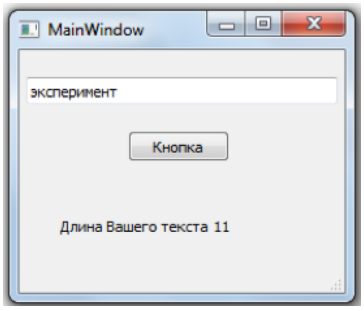



*Figure 13*. A functioning window application (Gorozhanov, 2018b, p. 58)

Using the resulting programme as an example, the *architecture* of a PyQt5 window application is demonstrated: creating an interface, creating a function for processing signals received from this interface, binding functions to signals from widgets (in our case, the *clicked* signal is considered) (Gorozhanov, 2018b, p. 58). As an additional task, the students are encouraged to install PyQt5, follow the steps outlined above, and compare their result with the output in the illustrations to the tutorial.

The next section demonstrates how to separate the data processing file from the GUI file so as to easily make adjustments to the interface file using Qt Designer. In the command line interface, the *ui* file is converted to a *py* file with the *-o-* parameter: *pyuic5 myinterface.ui -o myinterface.py* (Gorozhanov, 2018b, p. 61). After that, it will no longer run on its own, i.e. it will not receive a graphical loop, so for achieving this one needs to create a main executable file in which an eternal graphical loop (lines 9-13) is created (Figure 14). For the programme to work properly, one needs to put the function code in it and bind it to the signal.

This code is inserted between lines 8 and 9: *self.ui.push Button.clicked.connect (self.myFunction)* and *def myFunction (self): self.ui.label.setText ("Your text length% d "% len (self.ui.lineEdit.-text ()))*.

Listing 6

```
1.  import sys
2.  from myinterface2 import *
3.  from PyQt5 import QtCore, QtGui, QtWidgets
4.  class MyWin(QtWidgets.QMainWindow):
5.      def __init__(self, parent=None):
6.          QtWidgets.QWidget.__init__(self, parent)
7.          self.ui = Ui_MainWindow()
8.          self.ui.setupUi(self)
9.  if __name__ == "__main__":
10.     app = QtWidgets.QApplication(sys.argv)
11.     myapp = MyWin()
12.     yapp.show()
13.     sys.exit(app.exec_())
```

*Figure 14*. The code of the programme 'myintmain.py' (Gorozhanov, 2018b, p. 61)

Now one can reopen a separate GUI file in Qt Designer and resize the widgets, change the labels, colours, etc. The modified *ui* file must be converted to a *py* file again. The changes will take effect immediately when the main programme starts running.

In the sections described above, step by step, with a gradual increase in the level of complexity, the same window application is considered, starting with the creation of an interface and ending with a functioning software package of two files. As an additional task, students are encouraged to change the graphical interface file in Qt Designer without changing the main executable file.

The second part of the tutorial concludes with a section on how to create a *Guess Word* game. At the beginning, a technical task is formulated. The programme must: have a graphical user interface (GUI); receive from the user a certain number of Russian letters; display in a special window the Russian words of 4 or 5 characters long, which can be composed of the entered letters; use each of the entered letters in one word as many times as it was entered by the user; allow the user to choose a parameter of length 4 or 5 (it determines the words of which length to search for); use a list of the Russian words (dictionary) from a separate file for searching; display the number of tested combinations; display the execution time in seconds; have a progress bar (Gorozhanov, 2018b, p. 64).

Based on the difficulty of the task, we assign this section a strictly demonstrative function, which, however, does not exclude a thorough line-by-line analysis of the materials. Like in the previous case, the development starts with creating a graphical user interface in Qt Designer. The pro-

gramme uses the following widgets in the grid layout (Figure 15). The *for* loop is used to find and check the matches with the words in the dictionary. The final point of the technical task is especially difficult, but PyQt5 has a built-in *QProgressDialog* widget that allows one to track long processes, in particular loops, as in our case. We will bind the indicator to the external *for* loop and make it so that whenever the loop variable is incremented, the progress bar is updated (Gorozhanov, 2018b, p. 71). As a result, the code of the executable file eventually takes up about 150 lines (Gorozhanov, 2018b, p. 72-76). Its work is illustrated in Figure 16.

Thus, the second part of the tutorial introduces the students to the basics of developing professionally oriented Python window applications using the PyQt5 graphics library.

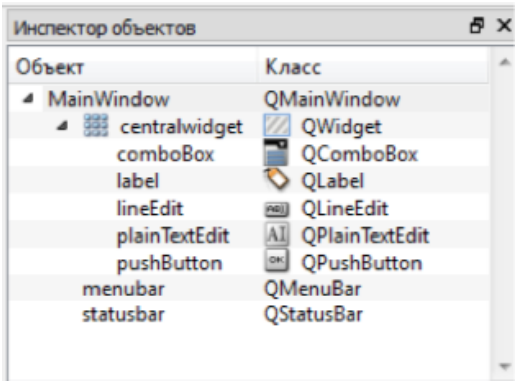



*Figure 15*. A widget tree for the interface of the programme *Guess Word* (Gorozhanov, 2018b, p. 65)

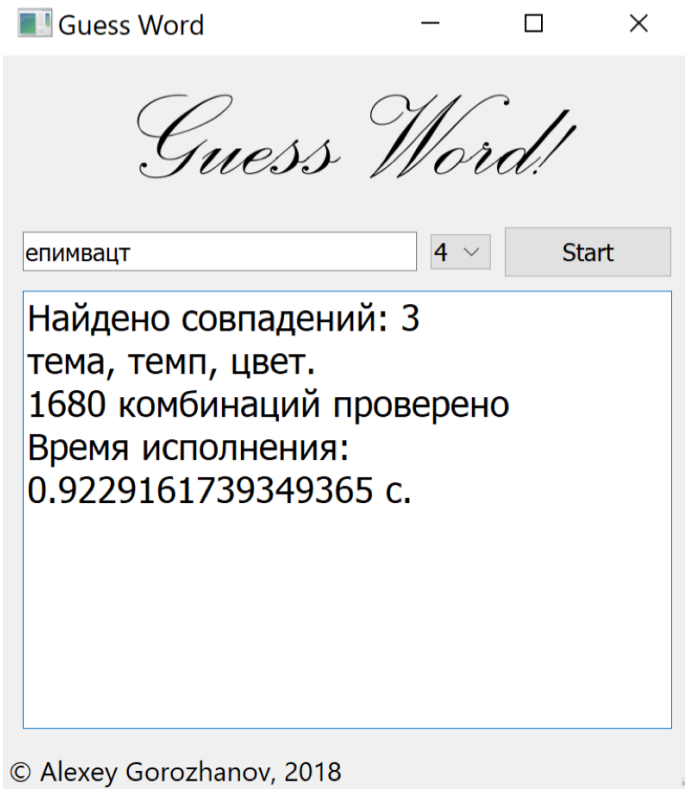



*Figure 16*. The output after entering *епимвацт* with a word length of four (Gorozhanov, 2018b, p. 77)

doi: 10.22363/2521-442X-2020-4-4-23-38
Volume 4 Issue 4, 2020, pp. 23-38
rudn.tlcjournal.org

## 5. CONCLUSION

Let us note that our article is one of the stages in the development of the issue of teaching the linguists programming languages taking into account a professionally oriented component, and it does not pretend to exhaustively describe all aspects of this subject area. However, our research is intended to be a contribution to the theory of teaching the linguists in our digital world.

The value of the study is that its results can be immediately implemented into the educational process, within the framework of the concept of lifelong learning (teaching programming for specific purposes not only the students, but also the acting teachers of foreign languages). It is important that the toolkit we offer does not need specialised materials and technical support, since free software is used.

The described materials of the tutorial were successfully tested at Moscow State Linguistic University in the framework of the disciplines *Application of modern information and communication technologies in teaching* and *Modern information technologies in linguistics* with undergraduate students majoring in *Linguistics: Theory and methods of teaching foreign languages and cultures* (45.03.02). The testing was attended by the students of the first and fourth year of full-time study, as well as the fifth year of part-time study (about 150 people in total in the period between 2018 and 2020).

Training in programming for specific purposes was only one of the modules of these disciplines (other modules were mainly devoted to the special theory of teaching foreign languages online and studying the capabilities of modern learning management systems) and was carried out within the framework of the seminars on the first part and the lectures on the second part of the tutorial (we underline that the third part of the tutorial has not been tested within the disciplines mentioned above).

Practice has shown that the materials can be most successfully arranged as follows. The first section of the first part of the tutorial is given in lectures. This is followed by a series of seminars, during which the material of the Part 1 of the tutorial is studied to the end. Then the material of Part 2 is introduced. Thus, working with the material on the PyQt5 graphics library, the students are already able to read the algorithms and understand their structure. The questions for self-control are used to form exam questions.

*'Despite some fears at the beginning of the testing period, most of the students successfully coped with the task. About 30% of students studied programming at school at a fairly high level, mainly in Pascal and Basic, and were familiar with such concepts as algorithm, variable, conditional expressions and loop. The fact that several first-year students later used the knowledge gained when working on their research can be considered success'*

Despite some fears at the beginning of the testing period, most of the students successfully coped with the task. About 30% of students studied programming at school at a fairly high level, mainly in Pascal and Basic, and were familiar with such concepts as *algorithm*, *variable*, *conditional expressions* and *loop*. The explanation and demonstration of the materials took place in the classroom. The homework accounted for the revision and better understanding of the materials and the solution to additional problems of increased complexity that were not included in the materials of the tutorial, with which, for one reason or another, no more than 5% of the students could cope. About 22% of them could offer a general solution scheme, but found it difficult to write the programming code. The fact that several first-year students later used the knowledge gained when working on their research can be considered success.

We conclude that the results of the testing showed the effectiveness of the proposed approach to teaching the linguistic students skills in programming for specific purposes.

## References

Alberti, G., & Zuccon, P. (2011). AMI: AMS monitoring interface. *Journal of Physics: Conference Series, 331*(8), art. 082008. Doi: 10.1088/1742-6596/331/8/082008

Amelkin, S. A., & Guseynova, I. A. (2020, October 28-30). Integrativnaya model' akademicheskogo obrazovaniya [Integrative model of academic education]. In *Proceedings of the International Conference Science Without Borders: Synergy of Theories, Methods and Practices* (pp. 497-500). Moscow, Russia: Moscow State Linguistic University.

Ardimento, P., Bernardi, M. L., Cimitile, M., & De Ruvo, G. (2020). Reusing bugged source code to support novice programmers in debugging tasks. *ACM Transactions on Computing Education, 20*(1), art. 2. Doi: 10.1145/3355616

Burger, D., Stassun, K. G., Pepper, J., Siverd, R. J., Paegert, M., De Lee, N. M., & Robinson, W. H. (2013). Filtergraph: An interactive web application for visualization of astronomy datasets. *Astronomy and Computing, 2*, 40-45. Doi: 10.1016/j.ascom.2013.06.002

Burns, P. J. (2020). Ensemble lemmatization with the classical language toolkit. *Studi e Saggi Linguistici, 58*(1), 157-176. Doi: 10.4454/ssl.v58i1.273

Chabaud, P. Y, Arnouts, S., Borges, R., Fauchier, F., Le Brun, V., Le Fèvre. O., … Vidal, C. (2018, May 28 – June 1). ARIADNE: A system for evaluation of AMAZED's spectroscopic redshift estimation efficiency. In *Proceedings of the 2018 SpaceOps Conference* (pp. 2378-2381). Marseille, France: AIAA. Doi: 10.2514/6.2018-2378

Cid-Fuentes, J. A., Àlvarez, P., Amela, R., Ishii, K., Morizawa, R., & Badia, R. (2020). Efficient development of high performance data analytics in Python. *Future Generation Computer Systems*, *111*, 570-581. Doi: 10.1016/j.future.2019.09.051

Combrisson, E., Vallat, R., O'Reilly, C., Jas, M., Pascarella, A., Saive, A., … Jerbi, K. (2019). Visbrain: A multi-murpose GPU-accelerated open-source suite for multimodal brain data visualization. *Frontiers in Neuroinformatics*, *13*, art. 14. Doi: 10.3389/fninf.2019.00014

Di Pierro, M. (2011). web2py for scientific applications. *Computing in Science & Engineering*, *13*(2), 64-69. Doi: 10.1109/MCSE.2010.97

Di Pierro, M. (2013). *web2py complete reference manual* (5th ed.). Chicago, IL: Experts4Solutions.

Egan, M., H., & McDonald, C. (2020). An evaluation of SeeC: A tool designed to assist novice C programmers with program understanding and debugging. *Computer Science Education*. Doi: 10.1080/08993408.2020.1777034

Garcia-Fossa, F., Gaal, V., & De Jesus, M. B. (2020). PyScratch: An ease of use tool for analysis of scratch assays. *Computer Methods and Programs in Biomedicine*, *193*, art. 105476. Doi: 10.1016/j.cmpb.2020.105476

Gorozhanov, A. I. (2018a). *Formirovanie obuchayushchei virtual'noi sredy v kontekste novykh informatsionnykh tekhnologii* [Developing a learning virtual environment in the context of new information technologies] (Doctoral dissertation). Moscow State Linguistic University, Russia.

Gorozhanov, A. I. (2018b). *Vvedenie v programmirovanie. Python. Tsifrovoi uchebnik* [Introduction to programming. Python. Digital tutorial]. Moscow, Russia: Moscow State Linguistic University.

Guseynova, I. A., Gorozhanov, A. I., & Kosichenko, E. F. (2019, October 25). Development of linguistic institutional educational virtual environment at Moscow State Linguistic University (2016-2018). In *Proceedings of the International Scientific and Practical Conference Current Issues of Linguistics and Didactics: The Interdisciplinary Approach in Humanities and Social Sciences CILDIAH-2019* (art. 00045). Volgograd, Russia: EDP Sciences. Doi: 10.1051/shsconf/20196900045

Japhne, A., & Murugeswari, R. (2020, February 26-28). Opinion Mining based complex polarity shift pattern handling for improved sentiment classification. In *Proceedings of the 5th International Conference on Inventive Computation Technologies* (pp. 323-329). Coimbatore, India: IEEE. Doi: 10.1109/ICICT48043.2020.9112565

Liu, Y., Gao, J., & Liu, D. (2011, May). Design of software system of mobile robot with mixed programming based on Eclipse + pydev. In *Proceedings of the International Conference on Theoretical and Mathematical Foundations of Computer Science ICTMF 2011* (pp. 231-238). Berlin, Heidelberg: Springer. Doi: 10.1007/978-3-642-24999-0_33

Medina-Ortiz, D., Contreras, S., Quiroz, C., Asenjo, J. A., & Olivera-Nappa, Á. (2020). DMAKit: A user-friendly web platform for bringing state-of-the-art data analysis techniques to non-specific users. *Information Systems*, *93*, art. 101557. Doi: 10.1016/j.is.2020.101557

Miles, M. (2016). Using web2py Python framework for creating data-driven web applications in the academic library. *Library Hi Tech*, *34*(1), 164-171. Doi: 10.1108/LHT-08-2015-0082

Morawiec, K., Zajkowska, W., Dłużewski, P., Shiojiri, M., & Kusińskic, J. (2020). PyHoLo software, a new tool for electron hologram analysis and magnetic investigation. *Computer Physics Communications*, *256*, art. 107471. Doi: 10.1016/j.cpc.2020.107471

Orfanakis, V., & Papadakis, S. (2016). Teaching basic programming concepts to novice programmers in secondary education using Twitter, Python, Ardruino and a coffee machine. In *Proceedings of the Hellenic Conference on Innovating STEM Education HISTEM* (pp. 16-18). Athens, Greece: University of Athens.

Papadakis, S. (2018). Is pair programming more effective than solo programming for secondary education novice programmers? A case study. *International Journal of Web-Based Learning and Teaching Technologies (IJWLTT)*, *13*(1), 1-16. Doi: 10.4018/IJWLTT.2018010101

Perez, J. E., Dinawanao, D. D., & Tabanao, E. S. (2020). JEPPY: An interactive pedagogical agent to aid novice programmers in correcting syntax errors. *International Journal of Advanced Computer Science and Applications, 11*(2), 48-53. Doi: 10.14569/IJACSA.2020.0110207

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. D. (2020, July). Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 101-108). Stroudsburg, PA: Association for Computational Linguistics. Doi: 10.18653/v1/2020.acl-demos.14

Rashkovits, R., & Lavy, I. (2020). Novice programmers' coping with multi-threaded software design. *Journal of Information Technology Education: Innovations in Practice*, *19*, 75-89. Doi: 10.28945/4609

Reddy, T. D., Chugh, K. L., & Karthik, R. (2017, December 7-8). Course outcome assessment tool for objective questions. In *Proceedings of the International Conference on Intelligent Sustainable Systems ICISS 2017* (pp. 1116-1119). Palladam, India: IEEE. Doi: 10.1109/ISS1.2017.8389355

Romero, J., Bisson, M., Fatica, M., & Bernaschi, M. (2020). High performance implementations of the 2D Ising model on GPUs. *Computer Physics Communications*, *256*, art. 107473. Doi: 10.1016/j.cpc.2020.107473

Senekal, B., & Kotzé, E. (2019). Open source intelligence (OSINT) for conflict monitoring in contemporary South Africa: Challenges and opportunities in a big data context. *African Security Review, 28*(1), 19-37. Doi: 10.1080/10246029.2019.1644357

Sun, W., & Kennett, B. L. N. (2020). Common-reflection-point-based prestack depth migration for imaging lithosphere in Python: Application to the Dense Warramunga array in Northern Australia. *Seismological Research Letters*, *91*(5), 2890-2899. Doi: 10.1785/0220200078

TIOBE. (2020). *TIOBE index for October 2020*. Retrieved from https://tiobe.com/tiobe-index

Titze, B., Genoud, C., & Friedrich, R. W. (2018). SBEMimage: Versatile acquisition control software for serial block-face electron microscopy. *Frontiers in Neural Circuits*, *12*, art. 54. Doi: 10.3389/fncir.2018.00054