МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ЗАТВЕРДЖЕНО
на засіданні кафедри
інформаційних систем.
Протокол № 1 від 22.08.2023 р.

ПОГОДЖЕНО
Проректор з навчально-методичної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ
№02071211

[illegible] Каріна НЕМАШКАЛО

ПРОГРАМУВАННЯ

робоча програма навчальної дисципліни (РПНД)

Галузь знань	**12 "Інформаційні технології"**
Спеціальність	**121 "Інженерія програмного забезпечення"**
Освітній рівень	**перший (бакалаврський)**
Освітня програма	**"Інженерія програмного забезпечення"**

Статус дисципліни	**обов'язкова**
Мова викладання, навчання та оцінювання	**англійська**

Розробник:
к.е.н., доцент підписано КЕП Володимир ФЕДОРЧЕНКО

Завідувач кафедри
інформаційних систем [illegible] Дмитро БОНДАРЕНКО

Гарант програми [illegible] Олег ФРОЛОВ

Харків
2024

[illegible]

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
SIMON KUZNETS KHARKIV NATIONAL UNIVERSITY
OF ECONOMICS

APPROVED
at the meeting of the Information
Systems Departments
Protocol No. 1 of 22.08.2023

AGREED
Vice-rector for educational and methodological
work

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ
№0207[illegible]1

[illegible] Karina NEMASHKALO

PROGRAMMING

Program of the course

Field of knowledge	**12 "Information technologies"**
Specialty	**121 "Software engineering"**
Study cycle	**first (bachelor)**
Study programme	**"Software Engineering"**

Course status	**mandatory**
Language	**English**

Developer: Ph.D., associate professor	digital signature	Volodymyr FEDORCHENKO
Head of Department information systems	[illegible]	Dmytro BONDARENKO
Head of Study Programme	[illegible]	Oleg FROLOV

Kharkiv
2024

INTRODUCTION

Currently, the conditions of industrial production and business conduct require economic management specialists to make comprehensive use of the latest information technologies and modern software (software). The wide possibilities of computer technology in matters of collecting, processing and issuing the necessary information can significantly improve the quality of calculations, make the process of justifying economic decisions more effective.

The course "Programming" is an instrumental basis for the implementation of the analytical part of further special courses, as well as course and diploma projects. It provides the study of the following courses: "Object-oriented programming", "Algorithms and data structures", "Operating systems", "Databases", "Distributed and parallel computing", "WEB technologies and WEB design", " Technologies of software development and testing", "Cross-platform programming" and "Programming for mobile devices".

the purposeteachingof the course "Programming" is: assimilation by students of higher education of the necessary knowledge regarding the basic concepts of algorithmization and the technique of applying basic algorithmic structures (organization of programs) and basic data types in programming, studying the main stages of the software design process and determining the principles of procedural programming in relation to the development of programs in languages C/C++, Python.

The tasks of the course are:

- acquire the ability to analyze the existing state of the subject area, analyze and develop requirements for the software being created;

- definition of concepts and study of basic principles of program organization;

- deepening of personal computer software skills;

- acquire the ability to use modern instrumental software; use previously compiled programs and support programs, make changes to the program, debug programs using built-in tools;

- mastering practical skills to use modern methods and tools for designing software and software product architecture components;

- acquire the ability to use effective algorithms and data structures for the development of software products;

- mastering the skills to use modern information resources and services in the process of solving professional tasks, to be able to design software for solutions in information communications;

- formation of skills in the application of typical approaches to the development and analysis of the most common algorithms for solving economic and mathematical problems;

- acquire the ability to apply and create programs in C/C++, Python programming languages using modern tool environments;
- formation of skills for working with technical documentation for software;
- use their knowledge to build a mathematical model and assess the correctness of problem statements, as well as independently predict the consequences of the obtained results.

The subject of study of the course is the theory and practice of application in programming of basic algorithmic structures and basic data structures based on modern software development technologies.

The object of the course is the software of information systems.

The course "Programming" is studied by students of the "Software Engineering" specialty of all forms of study in the first year during the first and second semesters.

Upon successful completion of the course, the applicant should be able to demonstrate the ability to learn new technologies, methods and techniques for software development, as well as critically analyze them for professional work and further use.

The learning outcomes and competence formed by the course are defined in the table. 1.

Table 1

Learning outcomes and competences formed by the course

Learning outcomes	**Competencies that should acquire a higher education**
LO07	SK7, SK10
LO13	SK3, SK7
LO15	GK2, SK10, SK11, SK13

where, LO07. Know and apply in practice the fundamental concepts, paradigms and basic principles of the functioning of linguistic, instrumental and computing tools of software engineering.

LO13. Know and apply methods of developing algorithms, designing software and data and knowledge structures.

LO15. Motivated to choose programming languages and development technologies to solve the tasks of creating and maintaining software.

GK02. Ability to apply knowledge in practical situations.

GK03. Ability to communicate in the national language both orally and in writing.

SK7. Knowledge of data information models, ability to create software for data storage, extraction and processing.

SK10. The ability to accumulate, process and systematize professional knowledge about creating and maintaining software and recognizing the importance of lifelong learning.

SK11. The ability to implement phases and iterations of the life cycle of software systems and information technologies based on appropriate software development models and approaches.

SK13. The ability to reasonably choose and master software development and maintenance tools.

COURSE CONTENT

Content module 1. Programming in C/C++ languages. Lexical foundations of C/C++ languages

Topic 1. Stages of program development and implementation

Basic information about programming technology: procedural, structural and object-oriented programming. Programming languages: procedural, application, rule systems, object-oriented. Standardization of languages and design environments. Translators. Editors. Compilers. Debuggers. Control structures: operators, expressions and routines. Overview of modern integrated programming systems. The integrated environment of the Visual Studio C++.NET programming system. DOT.NET platform. Stages of program development and implementation. Software code requirements.

The concept of an algorithm. Properties of the algorithm. Typical algorithmic designs. Development of the algorithm by the method of step-by-step refinement.

Topic 2. Computer architecture, principles of John von Neumann

Electronic numerical integrator ENIAC. Logical organization of a universal computing device. Open architecture of system blocks. Principles of John von Neumann.

Topic 3. Positional counting systems

Classification of counting systems. Features of positional counting systems. The basis, the alphabet and the basis of positional number systems. Representation of numbers in positional number systems. Transferring numbers from one number system to another.

Topic 4. Elements of algorithmic languages C/C++: concept of data types, names, values, pointers, variables, constants, operations, expressions

C/C++ programming language standards. The structure of the C++ program.

Lexical elements of the C++ language: alphabet, comments, identifiers, service words, data, expression, operand, variable, operation. Naming conventions.

Concept of data type. Classification and presentation of data. Basic data types: logical, symbolic, integer, substance. Type conversions: implicit conversions, explicit type conversions.

Priorities of operations. Table of priority and associativity of operations.

Operations. Unary operations: unary minus, unary plus, bitwise inversion, logical negation, increment, decrement, sizeof operation. Binary operations: additive, multiplicative, shift operations, bitwise operations, relations operations, logical, assignment, "coma" operation.

Standard mathematical functions.

Constant values: integer, real, enumerable, symbolic (literal), string (strings or literal strings). Rules for defining constants by the compiler. Defining constants using the const keyword.

Topic 5. Structured programming: sequence, branching and loops

General information about the data input-output system.

Expressions, spaces, blocks and complex expressions. Operand, variable. The assignment operator. Declaration and initialization of variables.

Types of operators. The simplest operator, declaration operator, definition operator, expression operator.

Control operators: pass operators; selection operators (single selection – if, double selection – if/else, multiple selection – switch, conditional operation); repetition operators (while operator, do-while operator, for operator). Nested loops. Control statements in loops: break statement, continue statement, goto statement. Recommendations for choosing cycles.

Topic 6. Preprocessing

Preprocessor directives. Basics of the macro apparatus.

#include preprocessor directive and included files. #define preprocessor directive: declaration of constants and macros. Conditional compilation. Using the typedef keyword. The difference between the #define directive and the typedef operator. The typeid operator.

Topic 7. Procedural-oriented programming. Recursion

General information about functions. Function structure.

Return values, parameters, and arguments. Function declaration. Function prototypes. Definition of a function. Execution of a function.

Local and global variables. The rule of variable visibility. Conversion of function argument types. Rules for automatic (implicit) type conversion. Explicit type conversions. Rules for working with functions. Classes of memory.

A list of function parameters. Default settings. Ways of passing parameters. Return value methods.

Substitution functions.

Creating your own header files.

Function overloading. Recursion. Types of recursion.

Functions work. Memory fragmentation. Stack and functions.

Function modifiers.

Topic 8. Dynamic Linkage Libraries (DLL)

Static and dynamic libraries. General information about DLLs. DLL structure. Declaring a function in a DLL.

Ways to load DLLs. Explicit (static) DLL loading. Implicit (dynamic) DLL loading.

Topic 9. Program development methodologies: bottom-up and top-down design, modular programming

Software development methodology: cascade development, interactive development.

Program design, the principle of modularity.

Program design methods: bottom-up and top-down design, kernel extension method.

Content module 2. Programming in C/C++ languages. Basics of programming in C/C++ languages

Topic 10. Arrays

Declaration of arrays. Initialization of arrays. Processing of one-dimensional arrays of economic data.

Array sorting algorithms.

Multidimensional arrays. Initialization of a multidimensional array. Typical examples of matrix processing.

Arrays and graphs. Arrays as function parameters.

Topic 11. Derived data types. Rows in the style of S

Strings as character arrays. String operations. String input-output.

The concept of pointer, link. Pointers and arrays. Address arithmetic. Link. Examples of using pointers and links. Pointers to functions.

Function parameters as references.

Memory organization in modern processors and pointers of the C++ language. Memory models. Static and dynamic variables. Operators new and delete. Dynamic arrays. Dynamic arrays as function parameters.

Topic 12. Structures and associations. Dynamic data structures

Structures, declarations and definitions of structure-type objects. Differences between the structures of the C and C++ languages. Methods in structures. Static structure members. Structures with bit fields. Nested structures. Access to structure elements. Operations with structures. Structures as function parameters. Structures as function return values.

Arrays of structures. Pointers to structures. Transfer by reference of arrays of structures.

Association. Operations with associations. Lists. Variable structures.

Definition, classification of dynamic data structures, methods of declaration, initialization of dynamic structures, methods of accessing data of dynamic structures, placing them in memory, advantages and disadvantages of using dynamic structures in programs.

Linked lists, one-way and two-way lists, cyclic lists. Creation of a singly linked list. Traversal of a singly linked list. Adding a new item to an existing list. Excluding an item from a list.

Creating a bidirectional list. Bidirectional list traversal. Adding a new item to an existing list. Excluding an item from a list.

Topic 13. Introduction to the C/C++ input-output system. File data structures

Basic provisions of the C/C++ input-output system. Streams and buffers. Standard I/O objects. Hierarchy of C++ I/O classes.

Data entry using the global object cin.

Line output. Entering a single character. Using the get() function: without parameters, with parameters. Entering strings from a standard input device. Using the getline() function.

Data output using the global cout object. Clearing the output buffer - flush(). Using the put() and write() functions. Input – Formattable data output. Input-output manipulators. Functions width(), precision(), fill(). User-defined manipulators.

Using files for data input and output. Creating a file. Creating a flow. Opening the stream. "Attach" a file to a stream.

File opening and access modes. Exchanges with a file using a stream. "Disconnecting" the stream from the file. Closing the file. File destruction. Determination of flow state. I/O management functions.

Topic 14. Templates. Standard template library

Basics of the template apparatus. Function templates. Patterns of structures. Function template overloading. Templates of sorting functions.

Standard Template Library (STL). Purpose and composition of STL. Containers. Algorithms. Functional objects. Iterators, thread iterators. Work with vectors, lists, stacks, queues.

Topic 15. Standard string class

Data type string. Communication of the string class with STL. Declaration and definition of objects of type string. Methods and properties of the string class. String operations. Functions for working with strings. String input-output. Convert between string and C-style strings.

Topic 16. Exception handling. Features of standards C17, C++17, C++20

Exception handling in the C language, functions abort(), exit(), atexit().

Principles of exception handling in the C++ language. Exception generation. Exception handling sequence. Features of exception handling during dynamic memory allocation.

Collection cycle. Initialization lists, universal initialization. Lists with strict typing. Additional containers and changes to containers. External templates. Templates with a variable number of arguments. Class std::tuple. Derivation of types. Lamda-functions, alternative function syntax. Regular expressions. Multithreaded programming, Thread library.

Content module 3. Python programming. Lexical foundations of the Python language

Topic 17. Syntax and semantics of the Python language, the concept of data types, names, values, pointers, variables, constants, operations, expressions

Script programming languages. Python standards, application areas. Programming paradigms supported by Python. Overview of modern integrated programming systems for Python. Integrated environment of the PyCharm programming system.

Concept of Python data types. Names, values, pointers, variables, operations, expressions. Comments, Shebang. Priorities of operations. Table of priority and associativity of operations. Operations with deadlines. An if/else expression. Basic functions.

Topic 18. Python language operators

The assignment operator. Declaration and initialization of variables.

Types of operators.

Conditional construction if. Cycles. The word else. Control statements in loops: break statement, continue statement. The range function. Nested loops.

Topic 19. Functions in the Python language

Named functions, instruction def. Function arguments. Functions with an arbitrary number of arguments, functions with positional and named arguments, mandatory and optional. Nested functions. Anonymous functions, lambda instruction. Return operator. Main function. Decorators. Scope of variables, keyword global.

Topic 20. Work with modules

Import and from instructions. Connection of the module from the standard library. Use of pseudonyms. Namespace configuration. Variables __name__ , __main__. Fields of visibility.

Topic 21. Rules for writing and documenting code in the Python language

Appearance of the code. Gaps in expressions and instructions. Comments. Version control. Naming agreements.

Lines of documentation. __doc__ attribute. Single-line and multi-line lines of documentation.

Content module 4. Python programming. Basics of Python programming

Topic 22. Lists, tuples and dictionaries

Indexes and slices. Sorting elements, comparing lists, methods and functions for working with lists. Generators and iterators. Coroutines Adding and removing elements. Check for item availability. Sorting, counting occurrences. Copying lists, joining lists. Lists of lists. Complex tuples. The tuple() function. Convert from a list to a dictionary. Obtaining and changing dictionary elements. The dit() function. The get method. The del operator, the pop() method. Copying and combining dictionaries. Methods items(), keys(), values(). Comprehensive dictionaries.

Set (set, frozenset). Arrays, a NumPy module. Functional programming.

Topic 23. Exception handling in the Python language

Runtime error. The try…except instruction. The finally block and the else instruction. Hierarchy of exclusions. Getting information about an exception, the as operator.

Exception generation, raise operator.

Context managers, construction with ... as.

Topic 24. Working with files

Opening and closing a file, open function, close() method. File opening modes. Text files. CSV files. Binary files, pickle module. Reading from a file. Writing to a file.

Shelve module. The OS module and work with the file system.

Topic 25. Work with lines

String literals. Obtaining a substring, ord and len functions, searching in a string, sorting a string. String functions and methods. String formatting, format method. Byte strings.

Topic 26. Classes and objects

Class definition, methods and attributes, properties, property annotations. Class object reference, constructors, __init() method. Definition of classes in modules and connection. The object class. A temporary representation of the object.

Topic 27. Basic built-in modules

Modules random, math, locale, decimal. Working with dates and times, datetime module. Processes and flows. Graphical interface.

The list of laboratory studies in the course is given in table 2.

Table 2

The list of laboratory studies

Topic name	Content
Topic 1-5.	Familiarity with MS Visual C++.NET integrated program development environment. Preparation and solving of linear and branched problems on a PC in the C/C++ language.
Topic 5.	Preparation and solving problems on a PC using cycles in the C/C++ language.
Topic 6-9.	Preparation and solving problems on a PC using functions and macros in the C/C++ language.
Topic 10.	Preparation and solving of array processing problems in C/C++ on a PC.
Topic 11.	Preparation and solving problems of processing arrays and strings in C style using pointers on a PC.
Topic 12-16.	Preparation and solution of problems of processing arrays of structures in the C/C++ language and dynamic data structures using STL on a PC.
Topic 17, 18.	Familiarity with PyCharm integrated development environment. Preparation and solving of problems on a PC using branches and loops in the Python language.
Topic 19.	Preparation and solving problems on a PC using Python functions.
Topic 20-22.	Preparation and solving of array processing problems in Python on a PC.
Topic 22, 23.	Preparation and solution of collection processing problems on a PC using Python exception handling tools.
Topic 24, 25.	Preparation and solving of string processing problems using Python files on a PC.
Topic 26, 27.	Preparation and solving of structured data processing tasks in Python on a PC.

The list of self-studies by the course is given in the table. 3

Table 3

List of self-studies

Topic name	Content
Topic 1 – 27	Studying lecture material
Topic 1 – 27	Preparation for laboratory classes
Topic 1 – 27	Preparation for the exam

The number of hours of lecture and laboratory studies and hours of self-study is given in the technological card of the course.

TEACHING METHODS

In the process of teaching an course, in order to acquire certain learning outcomes, to activate the educational process, it is envisaged to use such learning methods as:

Verbal (lecture (Topic 1, 4-7, 10-15, 17-20, 22-26), problem lecture (Topic 8, 9, 15), seminar-discussion (Topic 2, 16, 27)).

In person (demonstration (Topic 1, 4-7, 10-15, 17-20, 22-26)).

Laboratory work (Topic 1 - 27), case method (Topic 2, 3, 8, 9, 16, 21, 27).

FORMS AND METHODS ASSESSMENT

The University uses a 100-point cumulative system for assessing the learning outcomes of students.

Current control is carried out during lectures, laboratory classes is aimed at checking the level of readiness of the student to perform a specific job and is evaluated by the amount of points scored:

– for courses with a form of semester control as an exam: maximum amount is 60 points; minimum amount required is 35 points.

The final control includes current control and an exam.

Semester control is carried out in the form of a semester exam.

The final grade in the course is determined:

– for the course's with a form of exam, the final grade is the amount of all points received during the current control and the exam grade.

During the teaching of the course, the following control measures are used:

Current control:

defense of **laboratory work** (60 points);

Semester control: **Exam** (40 points)

An example of an examination ticket

Semyon Kuznets Kharkiv National University of Economics
First (bachelor) level of higher education
Specialty "Software Engineering"
Study programme "Software engineering"
Semester I
course "Programming"

EXAMINATION TICKET No. 1

GENERAL REQUIREMENTS: two tasks in one project; multi-module project; menu availability; processing of exceptional situations; procedural paradigm.

Task 1 (heuristic, 20 points).

Given a rectangular matrix of integers. Determine the number of negative elements in those rows that contain at least one zero element and the sum of positive diagonal elements.
Write the output data to a text file.

Task 2 (heuristic, 20 points).
Enter text from the file. Output the word of maximum length, that is, which has the maximum number of characters. When solving the problem, use a macro with parameters. Use cstring library functions for text processing.

Approved at a meeting of the Department of Information Systems. Protocol No. ____ of "___"__________20___year.

Examiner Ph.D., Assoc. Fedorchenko V.M.

Chief Department of Ph.D., Assoc. Bondarenko D.O.

Evaluation criteria

The examination ticket consists of two heuristic tasks in the form of problems. To solve each problem, you need to develop a program in the algorithmic language specified by the teacher (C(C++) or Python). The final grade for the exam is the sum of the marks for each task. Each task is evaluated from 0 to 20 points in accordance with the following scale (Table 4):

Table 4

Assessment criteria for task

Points	Criterion
20 points	The task is completed in full. The algorithm for solving the problem is optimal. The program works correctly on all tests. The program interface meets the established requirements. The text of the program is accompanied by comments.
18-19 points	The task is completed in full. The algorithm for solving the problem is optimal. The program works correctly. There are small comments on the organization of the user interface. There are no comments in the text of the program.
16-17 points	The task is mostly completed. The algorithm is correct, but not optimal. The program works correctly, but there are small remarks about the organization of the user interface. There are no comments in the text of the program.
14-15 points	The task is mostly completed. The algorithm is correct, but not optimal. The program works correctly, but one of the requirements or capabilities specified in the task is not implemented.
12-13 points	The task is completed, but not in full. The algorithm contains minor errors. The program is running, but the two requirements or capabilities specified in the task are not implemented.
10-11 points	The task is completed, but not in full. The algorithm contains errors. The program is running, but the three requirements or capabilities specified in the task are not implemented.
8-9 points	The algorithm was developed with errors, implemented in the form of a program. The program starts and allows at least one action (possibility) to be performed.
6-7 points	The algorithm was developed with significant errors, implemented in the form of a program. The program starts, but contains gross errors, including when organizing input/output of input/output data. During operation, the program either freezes or crashes.

4-5 points	The algorithm is developed with significant errors, implemented as a program, but the program does not run.
3 points	The algorithm was developed with errors and was not implemented as a program.
2 points	The program does not correspond to the assignment.
1 point	The program does not copy software code developed by the acquirer. The program has clear signs of non-independent development.
0 points	There is no algorithm for solving the problem. The program is missing.

RECOMMENDED LITERATURE

Main

1. Losev M. Yu. Programming in the Python language: a study guide / M. Yu. Losev, V. M. Fedorchenko - Kharkiv, - Lviv: Publishing House of PP "Novyi Svit - 2000", 2023. - 178 p. (Ukrainian language)
2. Lospinoso J. C++ Crash Course: A Fast-Paced Introduction. No Starch Press, 2019, 792 p.
3. Bancila M. Modern C++ Programming Cookbook - Third Edition: Master modern C++ including the latest features of C++23 with 140+ practical recipes: 3rd ed. Edition. Packt Publishing,2024, 816 p.
4. Stroustrup B.Tour of C++, A (C++ In-Depth Series): 3rd Edition.Addison-Wesley Professional, 2022,320p.
5. Ivor Horton, Peter Van Weert. Beginning C++23: From Beginner to Pro, 7th Edition. Apress, 2023, 948 p.

Additional

6. Trofymenko O.G. C++. Algorithmization and programming: a textbook / O.H. Trofymenko, Yu.V. Prokop, N.I. Loginova, O.V. Zadereiko 2nd edition processing and additional Odesa: Phoenix, 2019.–477 p.
7. Bjarne Stroustrup. Tour of C++, A (C++ In-Depth Series), 3rd Edition. Addison-Wesley Professional, 2022. - 320 p.
8. Vasiliev, Oleksiy Mykolayovych. Programming in the Python language / O.M. Vasiliev. Ternopil, "Teaching Book-Bogdan" Publishing House, 2021.–503 p.
9. Vysotska, Victoria Anatolyivna. PYTHON: Algorithmization and programming: study guide / V.A. Vysotska, O.V. Oborsk; Ministry of Education and Science of Ukraine, Lviv Polytechnic National University. Lviv, "New World-2000" Publishing House, 2021.–514 p.
10. C. Matthews, Robert. Coding in Python: A Comprehensive Beginners Guide to Learn the Realms of Coding in Python. 2020.–211 p.
11. Mattes, Eric. Python crash course: a hands-on, project-oriented introduction to programming / Eric Mattes ; translated from English by Olga Belova. Lviv, Stary Lev Publishing House, 2021.–556 p.

Information resources

12. International Standard ISO/IEC 14882:2014(E) – Programming Language C++ [Electronic resource]. - Access mode:https://isocpp.org/std/the-standard.

13. C++ reference [Electronic resource]. - Access mode:https://en.cppreference.com/w/.
14. C++ programming lessons [Electronic resource]. - Access mode:https://acode.com.ua/uroki-po-cpp/
15. C++ language documentation [Electronic resource]. - Access mode:https://learn.microsoft.com/en-gb/cpp/cpp/?view=msvc-170/.
16. Python documentation [Electronic resource]. - Access mode:https://www.python.org/doc/.
17. Learn Python Programming [Electronic resource]. - Access mode:https://www.programiz.com/python-programming.
18. 3Python Tutorial [Electronic resource]. - Access mode: https://w3schoolsua.github.io/python/index.html#gsc.tab=0.