1. **Course description**

   In the 21st century programming has become an essential skill for anyone working in the IT-field, as it provides significant insight into how numerous technical devices and systems surrounding us operate. Programming languages and tools have evolved significantly over the years, which now enables people without a solid technical background to successfully master related courses. The present course is mandatory for students of the 38.03.05 “Business informatics” bachelor’s program at the Higher School of Economics.

   The course covers four modules and is split into two distinct parts:

   Part 1: Introduction to Programming (304 hours including 40 hours of lectures, 68 hours of practice sessions and 196 hours for self-study) taught in the first year (modules 3-4) and based on the Python programming language.

   The first part gives students an in-depth understanding of the main principles used in any high-level programming language, the basic data structures and algorithms on them.

   Part 2: Object-oriented Programming (152 hours including 34 hours of lectures, 36 hours of practice sessions and 82 hours for self-study) taught in the second year (modules 1-2) and based on the C# programming language.

   The main goal for the second part is to provide students with knowledge of the main techniques of modern software development using object-oriented approach and the corresponding practical skills. Although the course part is based on a specific programming language – C#, it includes many general techniques that can be applied to any other modern object-oriented language.

   The course program is developed in accordance with:

   - Educational standards of the National Research University – Higher School of Economics
   - Educational program for bachelor’s degree 38.03.05 “Business Informatics”

**Tutors (2018-2019)**

Lecturer: Sergey Efremov

Class instructors: Sergey Efremov, Sergey Petropavlovsky, Kirill Mayboroda, Sergey Kalashnikov, Alexander Gorbunov, Tatiana Repetskaya, Leonid Smelov

**Prerequisites**

- High school mathematics

2. **Learning outcomes**

   On successful completion of the course students are expected

   **To know:**

   - Computational approaches to a wide range of problems
• Fundamental elements of any high-level programming language: type system, variables, flow control, functions
• Main principles of object-oriented software development.
• Mechanisms of building event-driven programs.
• Capabilities provided by the Python language as well as the C# programming language and .NET Framework.
• Modern paradigms of software engineering and testing.

To have practical skills of:
• Using Python with its standard libraries for solving data processing tasks
• Formalizing a practical problem using object-oriented approach.
• Using C# and .NET Framework for creating graphical applications with adaptive UI.
• Using modern IDEs (PyCharm, Visual Studio Code, Visual Studio) with major extensions for application development.
• Developing applications with relational databases as the storage mechanism.
• Building applications that interconnect with Restful web services.
• Testing and debugging programs using modern approaches and toolkits.

The discipline forms a basis for several 38.03.05 courses to follow:
- Data management (rus. Управление данными)
- Modelling of processes and systems (rus. Моделирование процессов и систем)
  It also gives students practical skills that can be used in their course and diploma projects.

3. Topic-wise course plan

Part 1. First year. Modules 3 – 4

<table>
<thead>
<tr>
<th>#</th>
<th>Topic</th>
<th>Total hours</th>
<th>In-class</th>
<th>Self-study</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Lectures</td>
<td>Practice</td>
</tr>
<tr>
<td>----</td>
<td>-------------------------------------------------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Overview of modern computer architecture</td>
<td>12</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Introduction to programming and the Python language</td>
<td>18</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Boolean expressions. Conditional execution</td>
<td>20</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Loops</td>
<td>24</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Control work I: program flow</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Bitwise operations. Error handling techniques.</td>
<td>14</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Functions and modules. Name scopes. Recursive algorithms.</td>
<td>24</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>#</td>
<td>Topic</td>
<td>Total hours</td>
<td>In-class</td>
<td>Self-study</td>
</tr>
<tr>
<td>----</td>
<td>------------------------------------------------------------</td>
<td>-------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lectures</td>
<td>Practice</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Introduction and brief revision of 1st year’s topics. Collections and generics</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>Object-oriented programming in detail</td>
<td>20</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>Programming with abstractions</td>
<td>24</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>Libraries and packages. Collaborative software development.</td>
<td>20</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td>UI Design with XAML (ONLINE)</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>LINQ</td>
<td>16</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Part 2. Second year. Modules 1 – 2

<table>
<thead>
<tr>
<th>7</th>
<th>Python data model. Data structures I. Introduction to algorithmic complexity.</th>
<th>36</th>
<th>4</th>
<th>6</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Control work II: Functions and standard data structures</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Module 4

<table>
<thead>
<tr>
<th>8</th>
<th>Data structures II</th>
<th>42</th>
<th>6</th>
<th>12</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>File input-output</td>
<td>28</td>
<td>4</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Control work III: File input-output</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Processing text. Regular expressions</td>
<td>16</td>
<td>2</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Control work IV: Theoretical test</td>
<td>12</td>
<td>2</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>Specialized Python packages. Python for data analysis</td>
<td>18</td>
<td>2</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>Introduction to object-oriented programming</td>
<td>26</td>
<td>4</td>
<td>6</td>
<td>16</td>
</tr>
</tbody>
</table>

Total: 304 40 68 196
4. Requirements and grading

Part 1:

<table>
<thead>
<tr>
<th>Type of control</th>
<th>Form of control</th>
<th>Modules</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current control</td>
<td>Classroom assessment</td>
<td>2</td>
<td>The first three are coding exercises, each to be completed during a 2-hour practice session, the last is a 45-minute test on theoretical aspects with multiple choice and open response questions carried out during the lecture</td>
</tr>
<tr>
<td>Homework assignments</td>
<td>1</td>
<td>Coding problems focused on groups of topics</td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>1</td>
<td>Test on theoretical aspects of the course with multiple choice and open response questions</td>
<td></td>
</tr>
<tr>
<td>Final control</td>
<td>Exam</td>
<td>One or two coding problems</td>
<td></td>
</tr>
</tbody>
</table>

Part 2:

<table>
<thead>
<tr>
<th>Type of control</th>
<th>Form of control</th>
<th>Modules</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current control</td>
<td>Homework assignment</td>
<td>2</td>
<td>Coding problems focused on groups of topics</td>
</tr>
<tr>
<td>Test</td>
<td>1</td>
<td>Test on theoretical aspects of the course with multiple choice and open response questions</td>
<td></td>
</tr>
<tr>
<td>Team project</td>
<td>1</td>
<td>Development of a complete software solution in teams</td>
<td></td>
</tr>
<tr>
<td>Final control</td>
<td>Exam</td>
<td>One or two coding problems</td>
<td></td>
</tr>
</tbody>
</table>

Current control includes regular classroom assessment, homework assignments, a team project and a theoretical test.

Each of the homework assignments focuses on a particular topic of the course, which allows students to gain practical skills related to it.

The aim of the team project is to learn main techniques of collaborative development of software projects as well as testing and debugging principles using tools and methods of team development (version control system, bug tracking, pair programming and others). To complete the team project, students form groups of 2-3 members. The project is graded as a whole from 1 to 10, each team member gets the overall mark.

Late policy
Students are required to follow deadlines of the assignments. Failure to comply with deadlines reduces the grade of the respective assignment by:

- 20% - delay less than 8 days
- 30% - delay from 8 to 14 days
- 40% - delay from 15 to 21 days

Assignments submitted with a delay of more than 21 days are not graded.

Final control is carried out in the form of a written exam. At the exam students are given one or two short practice tasks on the key topics of the course.

**Plagiarism penalties**

Plagiarism is considered a severe violation of the principles of academic integrity at HSE. Apart from the team project in module 4, all assignments are individual, and students are expected to submit their own solutions. If a group of students is caught cheating, all of them receive a zero grade for the respective assignment. Repeated cases of plagiarism may result in suspension from the University.

See official policy for detail: https://www.hse.ru/org/hse/antiplagiat_info/plagiat

**Grading**

**Part 1**

The aggregated grade (AG₁) is determined by a weighted average of the scores on classroom assessment (CA₁) and homework assignments (HW₁). The percentage breakdown is the following:

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classroom assessment (CA₁)</td>
<td>50</td>
</tr>
<tr>
<td>Homework assignment (HW₁)</td>
<td>50</td>
</tr>
</tbody>
</table>

\[ AG₁ = 0.5 \cdot CA₁ + 0.5 \cdot HW₁ \]

The final grade (FG₁) is determined by a weighted average of the aggregated grade and the exam score with the following percentages:

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregated grade (AG₁)</td>
<td>60</td>
</tr>
<tr>
<td>Exam (EX₁)</td>
<td>40</td>
</tr>
</tbody>
</table>

\[ FG₁ = 0.6 \cdot AG₁ + 0.4 \cdot EX₁ \]

**Part 2**

The final grade (FG₂) for the second part of the course is determined by a weighted average of the scores on classroom assessment, which includes a theoretical test (CA₂), average for homework assignments including the final team project (HW₂) and the final exam (EX₂).

\[ FG₂ = (0.3 + w_α) \cdot HW₂ + (0.3 - w_α) \cdot CA₂ + 0.4 \cdot EX₂ \]

The weight of the final exam is constant and equal to 40%.

The remaining 60% is split between in-class and homework assignments with a weight adjustment \( w_α \) that each student is free to set individually within the following limits:

\[ |w_α| \leq 0.1 \cdot (LQ + Act) \]

where

- \( LQ \) – average score of lecture quizzes normalized to [0,1]
- \( Act \) – additional bonuses for activity at practice sessions awarded by
seminar teachers, $Act \leq 0.3$, $LQ + Act \leq 1$
Determination of $w_a$ should be completed before the final exam, otherwise the default value $w_a = 0$ will be taken into calculation.
If a student is caught cheating at any assessment during the course, the right to determine the weight adjustment $w_a$ is taken away from him/her.
Rounding is not applied at any stage except for the final grade.
None of the grade components are blocking.
The final exam is obligatory for all students.

**Overall course grade**
The overall grade (OG) for the two-year programming course is determined by a weighted average of the final grades for the two parts. The percentage breakdown is the following:

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>First year grade (FG$_1$)</td>
<td>50</td>
</tr>
<tr>
<td>Second year grade (FG$_2$)</td>
<td>50</td>
</tr>
</tbody>
</table>

$$OG_1 = 0.5 \times FG_1 + 0.5 \times FG_2$$

5. **Course content**

**Introduction.** Course overview.

**Topic 1. Overview of modern computer architecture.**

**Topic 2. Introduction to programming and the Python language.**

**Topic 3. Boolean expressions. Conditional execution.**

**Topic 4. Loops.**
Pre-condition and post-condition loops. Algorithms on loops: number multiplication, greatest common divisor, sequence processing, nested loops.

**Topic 5. Bitwise operations.**

**Topic 6. Functions and modules. Name scopes. Recursive algorithms**

**Topic 7. Python data model. Data structures I. Introduction to algorithmic complexity**
How are values stored? The concept of a memory reference. Reference assignment.

**Topic 8. Data structures II.**

**Topic 9. File input-output.**

**Topic 10. Processing text. Regular expressions.**
Specialized methods for string manipulation and processing. Regular expression language: main capabilities and samples. Extracting data to groups.

**Topic 11. Specialized Python packages. Python for data analysis.**
Overview of available packages and their features. Integration with Excel.

**Topic 12. Introduction to object-oriented programming.**
Classes and objects. Instance and class members. Basics of OOP. Python syntax related to OOP.

**Topic 13. Review: basics of programming, C# language and .NET Framework.**

**Topic 14. Object-oriented programming in detail.**

**Topic 15. Abstract programming.**
The principle of loose coupling and its importance. Abstract programming using delegates: function call mechanism, reference to function, delegate, delegate instances and events, anonymous delegates, generic delegate types in C#, lambda expressions. Abstract programming using interfaces and inheritance: “Object” type and its methods, polymorphism, interfaces and abstract classes, multiple inheritance, inversion of control principle.

**Topic 16. Libraries and packages. Collaborative software development**

**Topic 17 (ONLINE). UI design with XAML**

Resources:
Microsoft Virtual Academy – XAML for Windows 10: Layout

**Topic 18. LINQ.**

**Topic 19. Interfacing databases. Object-to-relational mapping.**

**Topic 20. Basics of parallel programming.**

**Topic 21. Integration with the Web.**
6. Teaching methods and recommendations

6.1 Recommendations for instructors

Lectures are primarily targeted at presenting new material to students in a sequential systematic way. They form a theoretical basis for practice sessions, homework assignments and the final programming project. Focus of the lecture material should be placed on explaining why a certain technology or mechanism is used in programming rather than on exploring it to the smallest details.

For better understanding of the lecture material it is highly advisable to prepare lecture handouts and send them to students via e-mail or through the course website two or three days before the lecture.

Presentation at the lecture can be done in a few different ways: slides, ready code demonstration, on-the-fly code development, short questions quiz. Source code templates should be published on the course website before the lecture.

Practice sessions are carried out in computer-equipped classes (see requirements, sections 8-9). They are designed to reinforce the concepts presented in the lectures. If necessary groups are divided into two or more subgroups so that each student can work on the problems individually on a separate computer.

During self-study activities students gain additional knowledge by reading recommended resources and completing assignments. Typically, each week or two a new homework assignment is given. This does not apply to periods of working on the team project and preparing for the theoretical test. A homework assignment should be focused on a narrow topic and can be of two types:

- “Do from scratch”. In this case an instructor gives a description of the task in written (electronic) form. Students build an entire solution themselves.
- “Complete a template/Correct mistakes”. In this case a partly-functional project is given to students along with a description of tasks to be done for the project to be completed.

6.2 Recommendations for students

The course is practice-oriented and requires active student’s involvement in its activities.

Lectures organize course material in a logical systematic way and allow students to understand the ideas behind key technologies and aspects used in modern programming. Lectures may involve real-time code demonstrations. Students can bring their laptops and follow these demonstrations.

Practice sessions are intended to show implementation details of a specific programming mechanism and to help students complete a related homework assignment.

Homework assignments form the key part of the course as they give students the hands-on experience in developing applications. An assignment typically takes from 2 to 4 hours (including required readings) to be fully solved. Late policy applies to assignments sent after the deadline (see section 4).

A group project allows students to understand basic principles of developing software products in a small team. They will learn a number of collaborative tools, which have become a standard in modern software engineering.

Collaboration with classmates

Discussions on common concerns are allowed between classmates, however these discussions must be kept on a general level without exposing details of code solutions. Students may:

- Ask for / provide clarification of an error message / exception, appearing during compilation or program execution
- Help each other with the IDE functionality
- Help to find the cause of an error without providing a step-by-step explanation of how to fix the error
- Discuss lecture videos, handouts, textbooks, online resources or assignment specifications

The following activities are strictly prohibited:
- Posting assignment solutions online or sending them via e-mail, social networks, messaging services or any other communication channels
- Writing, typing, dictating code to classmates
- Providing step-by-step instructions on how to complete an assignment
- Using other students’ code in your solutions

**Plagiarism**
All submitted solutions are checked by sophisticated automated tools to detect plagiarism. Each solution is compared against a code database, which includes all submissions from current and previous years of the course. See the grading section for specification of penalties that apply in case plagiarism is detected.

**Using code fragments from online resources**
Students can use code fragments from publicly available online resources (resources that don’t require registration, e.g. [stackoverflow](https://stackoverflow.com)) in their programs, however citing is required (ask course instructors for details of how citing should be specified). It should be noted that overusing other people’s code can lower the grade of the respective assignment.

### 7. Assignment topics

#### 7.1 Topics for homework assignments
- Delegates, events and lambda expressions
- Abstract programming with interfaces and inheritance
- LINQ and Entity Framework
- Parallel programming

#### 7.2 Examples of test questions
- Give examples of low-level and high-level programming languages.
- What are the main features of C# as a programming language.
- What is the difference between a compiler and a Just-In-Time Compiler.
- What are the main tasks solved by CLR
- Give examples of value and reference types
- What is the difference between a value type and a reference type
- What is a “null” value and what is it used for
- Explain boxing and unboxing mechanisms
- What is the difference of class- and struct-based objects
- What are class constructors used for and how they differ from ordinary class methods
- What is the purpose of the “Singleton” template
- In what order are constructors called in case of an inheritance hierarchy
- How can a child class gain access to fields of its base class
- What is the difference in the call of a virtual and non-virtual method
- When is an object deleted from memory in managed execution frameworks
- What are the differences between an interface and an abstract class
- What is the main purpose of a delegate as a data type
- How does an event differ from a delegate instance
7.3. Topics for team project

The team project should use the following mechanisms covered in the course:

- Abstract programming using interfaces, delegates and possibly inheritance chains
- Database storage
- Web requests
- LINQ
- Version control system

Topics for the team project can vary, below are just a few examples:

- Store inventory
- HR management
- Transport scheduling
- University rating system
• Internet-store
• Forum/blog
• Hotel booking system
• Processing sport event results

8. Course resources

8.1. Textbooks
Interactive version with program execution in the browser available at: http://interactivepython.org/runestone/static/thinkcs/tp/index.html
• Троелсен, Э. Язык программирования C# 6.0 и платформа .NET 4.6 / Э. Троелсен, Ф. Дженикс; Пер. с англ. под ред. Ю. Н. Артеменко. – 7-е изд. – М.; СПб.; Киев: Вильямс, 2017. – 1438 с.
• C# 5.0. Справочник: полное описание языка / Дж. Албахари, Б. Албахари; Пер. с англ. под ред. Ю. Н. Артеменко. – 5-е изд. – М.; СПб.; Киев: Вильямс, 2014. – 1007 с.

8.2. Primary online resources
• Coursera online course: Programming for Everybody (Getting started with Python): https://www.coursera.org/learn/python
• Microsoft Virtual Academy. URL: https://mva.microsoft.com/

8.3. Complementary resources
• 101 LINQ Samples [Electronic resource]. URL: http://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b (access date 09.2018).

8.4. Software
• Python 3
• PyCharm Community Edition or Visual Studio Code
• Microsoft Visual Studio 2017 Community Edition (default installation with SQL Server Express / LocalDb or a separate SQL Server)

9. Technical equipment

Lectures require an internet-connected PC and a projector.
Practice sessions are carried out in computer-equipped classes. Each student should be provided with an individual PC with the necessary software installed (see previous
section).