Syllabus

Web Development with Python

Course duration: 184 classes (1 class = 80 minutes)

Schedule: 2 classes twice a week and at least 6 hours of homework.

The teacher may choose either of the two ways of teaching this subject:

- successively: HTML/CSS, JavaScript, Python;
- HTML/CSS first; then, after the HTML/CSS course is over, begin the Python course: twice per week; starting with the 4th module of the Python course, teach JS one time per week and Python one time per week; after JS is over, continue Python twice per week.

Terms of admission:

- Age from 15 to 55;
- Strong PC skills.

Topic Plan

Creating Web Pages with HTML5 and CSS3

Version 3.0.1

Course duration: 14 classes (1 class = 80 minutes)

Course objective

To teach the students to develop and create layout of static web pages using the HTML5 and CSS3 technologies. To present a full picture about the technological chain of creating websites and to form understanding of trends in the development of web-based technologies. To teach the students to pick the most suitable way to develop web pages. To teach them to test and check the code of web pages.

Upon completion of the course, the students will be able to:

- know and apply HTML basics: tags, attributes, and way of structuring contents of web pages in order to create formatted documents;
- know and apply CSS basics: values, lists, colors, fonts, and other formatting metrics;
- have skills of checking and debugging the code of web documents;
- have skills of formatting contents of web documents for various screens: from standard browsers to mobile devices;
- have skills of fast and quality formatting of complex web documents;
- know HTML5 and CSS3 basics.

Any free editor can be used to create a website: Notepad++, Microsoft Visual Studio Community.

Upon completion of this course, the student submits all practical tasks of the course and takes a test of knowledge on the course materials. In order to be admitted to the test, all homework assignments and practical tasks must be submitted.
The student is graded based on all submitted tasks.

The student should create a web site and put it on the Internet. Basic requirements: box layout, valid code.

Topic Plan

Module 1

Introduction to Web Technologies. Structure of HTML. Text Formatting with HTML

1. **Introduction to the subject.**
2. **Introduction to markup languages. Hypertext Markup Language (HTML).**
   - Internet.
   - HTTP.
   - Evolution of HTML, versions. HTML5.
   - Cross browser compatibility issues. Browser war.
   - W3C.
3. **Tags as a basic element of the HTML structure. The rules for writing tags and their attributes in the HTML5 standard. Syntax differences of HTML4, XHTML, and HTML5.**
4. **Common mistakes when writing tags.**
   - <!DOCTYPE HTML> specifications.
   - Document validation with FireFox: HTML Validator add-on.
   - The concept of well-formed.
   - HTML5 ancestors: SGML and XML.
5. **Structure of the HTML5 document.**
   - Main elements and their purpose.
   - New tags used to set structure: <header>, <nav>, <section>, <article>, <aside>, <footer>. Availability of new tags in modern browsers. How new tags are displayed in obsolete browsers.
6. **Character encodings of a page and <meta> tags.**
   - Use of the <meta> tag: set information about the page (expires, refresh, author, copyright, keywords, description).
   - Set page encoding using the <meta> tag.
   - Character mapping and encodings.
7. **Classification of tags: inline and block.**
   - Inline.
   - Block.
8. **Text formatting model: headings and paragraphs. The <p>, <h1>...<h6> elements. Text alignment within block elements: the align attribute.**

9. **Classification of tags: logical and physical tags.**
    - Physical tags.
    - Logical tags.
    - Brief overview of the main logical tags: <abbr>, <acronym>, <cite>, <code>, <del>, <dfn>, <ins>.
10. **Practice: creating a basic web page.**

Module 2

Formatting with CSS. Lists. CSS Margins and Paddings

1. **Cascading Style Sheets (CSS).**
    - Introduction. Overview of versions. Purpose: HTML sets structure, CSS formats the document.
    - Adding CSS to HTML using the style attribute. Rules for writing CSS properties.
2. **Tags containing no formatting: <div> – block, <span> – inline.**
3. **Analogy of HTML and CSS by the example of inline and block tags.**
4. **Additional properties of CSS for text formatting: letter-spacing, line-height, text-indent, text-transform, white-space, word-spacing.**
5. **The use of class and id attributes to set styles.**
    - Create styles for tags, classes, identifiers inside the <style> tag. The concept of selectors. Rules for writing selectors: tag selector, class selector, identifier selector, universal selector *.
    - Priority of using styles (tag / class / id / style). Using !important rule to take precedence.
    - Style inheritance. Standard property values.
    - Track styles using the firebug environment (Firefox add-on).
6. **Usage of external CSS files.**
    - Add CSS files using the <link> tag and the @import instruction.
    - CSS files and browser cache.
7. **Practice: text formatting by means of CSS.**
8. **Creating lists.**
    - Unordered lists: <ul>, <li> elements.
    - Ordered lists: <ol>, <li> elements.
    - The type, value, start attributes.

9. **Creating nested lists.**
10. **Formatting lists using CSS.**
    - The list-style-type, list-style-image, list-style-position properties.
    - Shorthand notation for the list-style property.
    - Design of multilevel lists. Nested selectors.
11. **Description lists: <dl>, <dd>, <dt> elements.**
12. **Managing margins and paddings.**
    - The margin property and its children: margin-left, margin-top, margin-right, margin-bottom.
    - The padding property and its children: padding-left, padding-top, padding-right, padding-bottom.
    - Difference between padding and margin and their purposes.
    - Cancel default margins for some tags: <body>, <h1>...<h6>, <p>.
13. **Practice: create lists.**

Module 3

Graphics in Web Design. Optimizing Web Graphics. Hyperlinks. Website Navigation Principles

1. **Graphics file formats in web.**
2. **The <img /> tag and its attributes (src, alt, width, height, border).**
    - The border property as an analog for the border attribute.
    - Setting margin, padding, border properties for an image.
    - Aligning images on a page using the align attribute. Analogue for the align attribute is the float property.
3. **Background of a page: the background property.**
    - Setting background as a color: background-color. Required background setting for the <body> element.
    - Setting background as an image: background-image, background-repeat, background-position, background-attachment.
    - Images and browser cache.
4. **General information about hyperlinks.**
    - The <a> tag and its attributes (href, target).
    - Ergonomics, user-friendly navigation.

5. **Absolute and relative addressing.**
    - Organization of external links.
    - Organization of internal links using the <a> element. The id and name attributes.
    - Organization of mixed links (to the specified element in the external HTML document).
    - Image links. Cancel link borders.
6. **Creating a menu with the list structure (<ul>, <li>), its formatting. The display property. Converting a link into a block element.**
7. **Pseudo classes.**
    - Link pseudo-classes: active, hover, link, visited.
    - Pseudo-classes for regular elements: first-child, first-line, first-letter.
8. **CSS cursor property.**
9. **Practice: development of an image gallery.**
10. **Properties from CSS3.**
    - Working with background: creating gradients, changing background size: the background and background-size properties.
    - Working with borders: blocks' rounded borders – the border-radius property.
    - Setting transparency to page elements: the opacity property.
    - Full support of the CSS 2.1 selectros.
11. **Working with multimedia.**
    - Add video to a page using the <video> tag.
    - Add audio to a page using the <audio> tag.
    - Create images and animation using the <canvas> tag.
    - Using the SVG format.

Module 4

Tables

1. **Creating a basic table. The <table>, <tr>, and <td> tags.**
    - The border, cellspacing, cellpadding attributes. Their possible CSS analogues: border, padding.
    - Specify cell width and height: the widths, height attributes. Rules for setting width and height. CSS analogues: width, height properties.
    - Data alignment in a table: align and valign attributes. CSS analogues: text-align, vertical-align properties.

- Management of background color and border color of a table (of an individual row, cell).
- Using images as table background (of an individual row, cell).

2. **Spanning cells: colspan, rowspan attributes.**
3. **Logical tags for tables: <thead>, <tbody>, <tfoot>. Logical tags for grouping columns: <colgroup>, <col>.**
4. **Managing table frames: frame, rules attributes.**
5. **Practice: create complex tables.**
6. **Basics of table layout. Example of table layout: its disadvantages.**

Module 5

Positioning. Box Layout

1. **The position property.**
   - Overview of positioning: relative and absolute.
   - The top, left, bottom, right properties.
2. **The visibility, overflow properties.**
3. **Practice.**
4. **Basics of box layout. Layout rules.**
   - Nested boxes.
   - Setting the width and height to boxes using the width and height properties.
   - Wrapping around blocks. Cancel wrapping around blocks. The float and clear properties.
   - Rules for setting margins and paddings.
   - Setting minimum height and width of a box: the min-height, min-width properties. Setting these properties in the IE6 browser.
   - Alignment inside boxes (margin, text-align, line-height, position). Cross-browser alignment.
5. **Overview of the simplest structures of pages.**
   - Overview of the simplest structures of pages.
6. **Responsive structure. Boxes with negative margin.**

Module 6

Forms. Frames

1. **Introduction to forms.**
2. **Form controls.**
   - Buttons (Send, Cancel, etc.)
   - Checkboxes.
   - Radio buttons.
   - Drop-down lists.
   - Text input.
   - File selection.
   - Hidden controls.
3. **Creating forms using HTML.**
   - The <form> element.
   - The <input> element.
   - The <button> element.
   - The <select>, <optgroup>, and <option> elements.
   - The <textarea> element.
   - The <label> tag.
   - Form structure: <fieldset> and <legend>.
4. **HTML5 form elements.**
5. **Form validation using HTML5.**
6. **Formatting form elements using CSS.**
7. **Frames and their structure (theory).**
   - The <iframe> tag.
   - Using frames to add external resources (YouTube, Google Maps etc.).

Developing Client-Side Scripts with JavaScript

Version 2.0.0

Course duration: 30 classes (1 class = 80 minutes)

Course objective

To teach the students to develop client-side scripts using JavaScript. To teach them to pick appropriate mechanisms and structures to solve a particular problem.

Upon completion of the course, the students will:

- master basic structures of JavaScript, such as variables, conditionals, loops, strings, arrays, functions, etc.;
- get acquainted with OOP and its basic terms;
- be able to handle errors;
- be knowledgeable in the concepts of event, event handler;
- create handler functions of various events;
- understand differences between BOM and DOM;
- be able to interact with objects from BOM and DOM;
- be knowledgeable in subtle details of implementing client-side scripts for various browsers;
- master the principles of creating forms and analyzing user data with regular expressions;
- be able to store user data using cookie mechanism;
- understand subtle details of applying HTML5 in relation to JavaScript;
- be able to serialize and parse data using JSON;
- master principles of creating asynchronous requests using AJAX.

Upon completion of this course, the students submit all practical tasks of the course.
The students get a grade in this subject based on all submitted tasks.

Topic Plan

Module 1

Introduction to JavaScript

1. **Client-side scripts.**
2. **What is JavaScript?**
3. **Origin of JavaScript.**
4. **Differences between JavaScript and Java, JScript, ECMAScript.**
5. **JavaScript versions.**
6. **Document Object Model.**
7. **Browser Object Model.**
8. **Implementation into HTML documents. JavaScript code editors.**
9. **The <noscript> tag.**
10. **Basics of syntax.**
    - Case sensitivity.
    - Comments.
    - Keywords and reserved words.
11. **Variables. Variable naming rules.**
12. **Data types.**
13. **Operators.**
    - Arithmetic operators.
    - Relational operators.
    - Logical operators.
    - Assignment operator.
    - Bitwise operators.
    - Operator precedence.
    - The typeof operator.
14. **Data input/output. Dialog boxes.**
15. **Conditions.**
    - What is a condition?
    - if.
    - if else.
    - Ternary operator ?:
    - switch.

16. **Loops.**
   - What are loops?
   - while.
   - do while.
   - for.
   - break.
   - continue.
   - The concept of a label.
17. **What is a function?**
   - Syntax of function declaration.
   - Function parameters.
   - Returned function value. The return keyword.
18. **The arguments object.**
   - Aims and objectives of an object.
   - The length property.
19. **Variable scope. The this keyword.**
20. **Recursion.**

Module 2

Object. Arrays. Array Object. Strings. String Object. Date Object. Math Object. Introduction to OOP

1. **Objects.**
   - What is an object?
   - Introduction to object data type.
   - The Object object.
   - The new keyword.
   - The concept of a property.
   - Adding properties. Syntax for adding properties.
   - Syntax for accessing properties.
2. **Arrays.**
   - What is an array?
   - Array object.
   - Creating an array.

- Accessing array elements.
- Properties and methods of Array.

3. **Strings.**
   - String object.
   - Properties and methods of String.
4. **Delays and intervals. Periodic function call.**
5. **Date object. Processing date and time.**
6. **Math object. Properties and methods. Random numbers.**
7. **What is OOP?**
8. **Three fundamental principles of OOP.**
   - Encapsulation.
   - Inheritance.
   - Polymorphism.
9. **The concept of class and object in terms of JavaScript.**
10. **Properties.**
11. **Methods.**
12. **Property accessors.**
    - Getters.
    - Setters.
13. **Constructor.**
14. **The concept of prototype.**
    - What is a prototype?
    - Aims and objectives of prototype.
15. **Inheritance.**

Module 3

Event Handling

1. **What is an event?**
2. **What is an event handler?**
3. **Event handling in scripts.**
4. **Controlling styles of web page elements.**

5. **The event object and its properties.**
6. **Default event handlers (standard handlers), restriction on calling a standard handler.**
7. **The Image object. Managing the drawings and rollovers.**

Module 4

Browser Object Model. Document Object Model

1. **What is Browser Object Model?**
2. **Browser Object Model objects.**
   - Window object. Open, move, and resize windows.
   - Navigator object. Browser management.
   - Screen object. Screen properties.
   - Location and History objects. Navigating through pages.
   - Frames collection. Frame management.
3. **What is Document Object Model?**
4. **Differences between DOM and BOM.**
5. **Representing HTML documents as a tree.**
6. **DOM objects. Hierarchy of nodes.**
7. **Properties and methods of DOM. The DOM event model.**
8. **Changing the DOM tree.**
9. **Introduction to the Document and Link objects.**
10. **Managing the selection and text range: Selection and TextRange objects.**
11. **Features of DOM in HTML5.**

Module 5

Forms

1. **Application of forms. Arrangement of form elements in HTML.**
2. **The Forms collection. Creating and programming elements of form.**
   - Buttons: Button, Submit, Reset elements.

- Text boxes: Text, Password, File Upload, Textarea elements.
- Hidden form field: general concept of the Hidden element.
- Checkbox: Checkbox element.
- Radio button: Radio element.
- List: Select, Option elements.

Module 6

Form Validation. Using Cookie

1. **The RegExp object. The rules for writing regular expressions.**
2. **Methods of the String and RegExp objects for handling regular expressions.**
3. **Validation of the form data.**
4. **What is cookie?**
5. **Advantages and disadvantages of cookie.**
6. **Creation, use, and removal of cookie.**

Module 7

JSON, AJAX

1. **What is JSON?**
2. **Aims and objectives of JSON.**
3. **JSON syntax.**
   - Variables.
   - Objects.
   - Arrays.
4. **JSON object.**
   - What is serialization?
   - What is parsing?
   - The stringify and parse methods.
5. **Configuring the user serialization in JSON.
   The toJSON method.**
6. **Synchronous and asynchronous requests.**
7. **What is AJAX?**

8. **XMLHttpRequest object.**
    - Creation through the ActiveX object.
    - Creation through the XMLHttpRequest object.
9. **Methods and properties of XMLHttpRequest.**
10. **The concept of the HTTP header.**
11. **The use of the GET method. URL encoding.**
12. **The use of the POST method.**

Module 8

Introduction to jQuery

1. **What is jQuery?**
2. **Aims and objectives of jQuery.**
3. **Origin of jQuery.**
4. **Versions of jQuery.**
5. **Adding jQuery.**
6. **Access to page elements using the $ function.**
7. **The concept of a selector.**
8. **Types of selectors.**
    - CSS selectors.
    - jQuery selectors.
9. **Interaction with DOM.**
    - Creating new DOM elements.
    - Adding DOM elements.
    - Moving DOM elements.
    - Copying DOM elements.
    - Interaction with attributes.
    - Traversing. Methods to traverse DOM. Methods filter, next, nextAll, prev, prevAll, siblings, etc.
10. **Events and jQuery.**
    - Creation of event handlers using jQuery.
    - Deletion of event handlers.
    - Application examples.

11. Animation and jQuery.

12. AJAX and jQuery.

- JSON.
- AJAX mechanisms inside a jQuery library.
- The use of the GET method.
- The use of the POST method.
- Events and AJAX within jQuery.
- Error handling.

13. Usage of jQuery plugins.

- The concept of the plugin.
- Adding a plugin.
- Examples of plugins.

Web Development with Python

Version 1.0.4

Course duration: 140 classes (1 class = 80 minutes)

Course objective

To teach the students create web apps using Python.

Upon completion of the course, the students will:

- understand subtle details of building web apps using Python;
- understand features of OOP implementation in Python;
- process and analyze form data;
- use standard features in Python;
- apply regular expressions;
- understand principles of functional programming;
- save user data in the cookies files;
- work with sessions;
- understand principles of network interaction;
- interact with data sources;
- add AJAX to web apps;
- know how to use version control system;
- understand basics of teamwork;
- apply design patterns;
- use unit testing;
- understand subtle details of the MVC pattern;
- create web projects using Python and MVC pattern;
- use Flask/Bottle;
- develop web apps using Django framework.

Upon completion of this course, the student submits practical task and takes test of knowledge on course materials. In order to be admitted to the test, all homework assignments and practical tasks must be submitted. Practical task must cover a maximum of the material from various sections of the course.

Students should be provided with access to Microsoft Imagine Academy: "Introduction to Python" before the subject starts. This course prepares them for the MTA 98-381 exam: "Introduction to Programming Using Python."

Topic Plan

Module 1.	Introduction to Web Programming with Python	**4 classes**
Module 2.	Branching Operators, Loops, Exceptions	**12 classes**
Module 3.	Strings, Lists	**8 classes**
Module 4.	Functions and Modules	**8 classes**
Module 5.	Sort, Search	**6 classes**
Module 6.	Tuples, Sets, Dictionaries	**5 classes**
Module 7.	Files	**5 classes**
Module 8.	Version Control Systems	**4 classes**
Module 9.	OOP	**12 classes**
Module 10.	Data Structures	**5 classes**
Module 11.	Data Packing	**4 classes**
Module 12.	Design Patterns	**4 classes**
Module 13.	MVC Pattern	**2 classes**
Module 14.	Principles of Designing SOLID Classes	**2 classes**
Module 15.	Unit Testing	**2 classes**
Module 16.	Parallel, Multithreaded, and Network Programming	**5 classes**
Module 17.	Introduction to Databases	**10 classes**
Module 18.	Use of Databases in Python	**6 classes**
Module 19.	Teamwork, Software Project Management	**4 classes**
Module 20.	Frameworks	**24 classes**
Module 21.	Creating Chatbots with Python	**6 classes**
Module 22.	Exam	**2 classes**

Module 1

Introduction to Web Programming with Python

1. **Introduction to web programming and operation principles of web apps.**
   - Basics of the client-server technology.
   - Overview of HTTP, getting acquainted with the Request and Response structure.
   - Reasons and origin of web programming. Differences between server and client web programming. Aims, objectives, trends, brief history.
   - Principles and stages of web page loading.
2. **Overview of programming languages.**
   - Getting acquainted with the main paradigms of programming.
   - Overview of the modern programming languages.
   - The concept of algorithm.
   - Getting acquainted with Python, application areas.
3. **Introduction to Python. Python interpreter and its environment.**
   - Introduction to Python.
   - The concept of interpreter and installation order.
   - Getting acquainted with programming environments:
     - Standard programming package: IDLE and Python Shell;
     - IDE PyCharm, Spyder, Visual Studio;
     - Atom.
4. **Data types, variables, and syntax constructs.**
   - Type and values.
   - Variables as an object in Python.
   - Variable names and reserved words.
   - Statements.
   - Operators and operands.
   - Operator precedence.
   - Variable operations.
   - Order of program execution.
   - Input/output.
   - Type conversion.
   - Syntax and logical errors, how to work with them.

Module 2

Branching Operators, Loops, Exceptions

1. **Conditional statements and their syntax.**
    - The concept of the execution block.
    - Logical expressions and operators.
    - Branching statements if...else.
    - Nested statements.
2. **The concept of exceptions.**
    - Types of exceptions.
    - Catching an exception.
    - Features of working with try...except.
3. **Loops.**
    - The concept of iteration.
    - The while loop.
    - Infinite loops.
    - The continue, break, and else control statements.
    - The for loop.
    - Local and global variables.

Module 3

Strings, Lists

1. **Strings.**
    - The ASCII, Unicode, UTF-8, Byte-code encoding.
    - String – unchangeable sequence of characters.
    - String methods.
    - Features of working with strings.
    - Slicing strings.
    - Escape sequences.
    - "Raw" strings.
    - Formatted output.
    - The string module.
    - Bytes and encodings.
    - Regular expressions, the re module.

2. **Lists.**
    - The concept of classic array.
    - The concept of object collection.
    - Reference data type list.
    - Creating lists.
    - List generators.
    - Working with lists.
    - List methods.
    - The in membership operator.
    - List features, references and cloning.
    - Search for element.
    - Matrices.

Module 4

Functions and Modules

1. **Functions and modules.**
    - What is a function?
    - Aims and objectives of a function.
    - Built-in functions.
    - Math functions and random numbers.
    - Syntax of function declaration.
    - Arguments and returned values.
2. **Advanced methods of working with functions.**
    - Packing and unpacking arguments.
    - Default arguments, keyword arguments.
    - Scope, LEGB-rule.
    - Local and global variables in functions.
    - Functions as first-class objects.
    - Recursion.
3. **Functional programming.**
    - What is functional programming?
    - Anonymous lambda functions.
    - The functools module.
    - The map(), reduce(), filter(), zip() functions.

- Higher order functions.
- Closure.

4. **Closure.**
5. **Currying.**
6. **Decorators.**

Module 5

Sort, Search

1. **Sorting.**
   - Optimality.
   - Bubble sort.
   - Merge sort.
   - Shell sort.
   - Heap sort.
   - Quick sort.
2. **Search.**
   - Linear search.
   - Binary search.

Module 6

Tuples, Sets, Dictionaries

1. **Tuples.**
   - Collection of immutable objects.
   - Application and features of a tuple.
2. **Sets.**
   - The mathematical concept of sets.
   - The set(), frozenset() data type.
   - Set operations.
   - Application of sets.
3. **Dictionaries.**
   - Associative arrays.
   - Hash tables.

- Creating a dictionary.
- Dictionary methods.
- The concept of sparse matrix.

4. **Application examples.**

Module 7

Files

1. **Files.**
    - File system, features of format implementation.
    - Working with files:
        - Open;
        - Close;
        - Read;
        - Write.
2. **Context manager.**
3. **File types, text and binary.**
4. **Application examples.**

Module 8

Version Control Systems

1. **What is version control?**
2. **Why do we need version control?**
3. **Overview of version control systems.**
    - CVS.
    - SVN.
    - Git.
    - Other version control systems.
4. **Git.**
    - What is Git?
    - Aims and objectives of Git.
    - Basic terms:
        - repository;

    - commit;
    - branch;
    - working directory.
- Git operations:
    - installation;
    - creating a repository;
    - adding a file to the repository;
    - recording changes to the repository;
    - getting the current state of the working directory;
    - display branches;
    - accumulation buffer operations;
    - git remote;
    - git push;
    - git pull;
    - other operations;
    - use of external services (github).

Module 9

OOP

1. **Introduction to OOP.**
    - The concept of OOP.
    - Encapsulation.
    - Inheritance.
    - Polymorphism.
    - Features of OOP implementation in Python, duck typing.
2. **User defined data types.**
    - Class instance.
    - Classes and objects.
    - Attributes, fields (properties), class methods.
    - Method overload.
    - Magic methods, constructors.
    - Static methods and class methods.
3. **Inheritance and encapsulation.**
    - Public, protected, and private method.
    - Multiple inheritance and method resolution order (MRO).

4. **Polymorphism.**
    - Operator overload.
    - Implementation of magic methods.
5. **Creation and behavior management of class instance.**
    - Functors.
    - Decorators.
    - Managed attributes.
    - Properties.
    - Descriptors.
6. **Meta classes.**
    - Model of meta classes.
    - The __new__() method.
    - The class protocol.

Module 10

Data Structures

1. **Linked lists.**
    - What is a list?
    - Single linked and double linked list.
    - Application examples.
2. **Stack.**
    - What is a stack?
    - The LIFO principle.
    - Application examples.
3. **Queue.**
    - What is a queue?
    - Types of queues.
    - Application examples.
4. **Trees.**
    - What is a tree?
    - Types of trees.
    - Application examples.

Module 11

Data Packing

1. **Serialization and deserialization.**
   - What is serialization?
   - What is deserialization?
   - Aims and objectives of serialization and deserialization.
   - Application examples.
2. **The pickle module.**
3. **The json module.**
4. **Third-party serialization modules.**

Module 12

Design Patterns

1. **What are design patterns?**
2. **Reasons for occurrence of design patterns.**
3. **The concept of a design pattern.**
4. **Principles of applying design patterns.**
5. **Principles of selecting design patterns.**
6. **Principles of separating patterns into categories.**
7. **Introduction to UML.**
   - Class diagram.
   - Object diagram.
   - Interaction diagram.
8. **Usage of UML when analyzing design patterns.**
   - Class diagram.
   - Object diagram.
   - Interaction diagram.
9. **Creational patterns.**
   - What is a creational pattern?
   - Aims and objectives of creational patterns.
   - Overview of creational patterns.

- Analysis of creational patterns:
  - Abstract Factory:
    - pattern objective;
    - reasons for pattern occurrence;
    - pattern structure;
    - results of pattern use;
    - application example of using pattern.
  - Builder:
    - pattern objective;
    - reasons for pattern occurrence;
    - pattern structure;
    - results of pattern use;
    - application example of using pattern.
  - Factory Method:
    - pattern objective;
    - reasons for pattern occurrence;
    - pattern structure;
    - results of pattern use;
    - application example of using pattern.
  - Prototype:
    - pattern objective;
    - reasons for pattern occurrence;
    - pattern structure;
    - results of pattern use;
    - application example of using pattern.
  - Singleton:
    - pattern objective;
    - reasons for pattern occurrence;
    - pattern structure;
    - results of pattern use;
    - application example of using pattern.

10. Structural patterns.

- What is a structural pattern?
- Aims and objectives of structural patterns.
- Overview of structural patterns.

- Analysis of structural patterns:
    - Adapter;
    - Composite;
    - Facade;
    - Proxy;
    - other structural patterns.

11. **Behavioral patterns.**
    - What are behavioral patterns?
    - Aims and objectives of behavioral patterns.
    - Overview of behavioral patterns.
    - Analysis of behavioral patterns:
        - Command;
        - Iterator;
        - Observer;
        - Strategy;
        - other behavioral patterns.

Module 13

MVC Pattern

1. **What is an MVC pattern?**
2. **Aims and objectives of the Model-View-Controller pattern.**
3. **Model.**
    - What is Model?
    - Aims and objectives of Model.
4. **View.**
    - What is View?
    - Aims and objectives of View.
5. **Controller.**
    - What is Controller?
    - Aims and objectives of Controller.
6. **Application examples of an MVC pattern.**

Module 14

Principles of Designing SOLID Classes

1. **Review of issues occurring when designing and developing classes.**
2. **Design principles of SOLID classes.**
   - The Single Responsibility Principle.
   - The Open Closed Principle.
   - The Liskov Substitution Principle.
   - The Interface Segregation Principle.
   - The Dependency Inversion Principle.
3. **Example use of SOLID principles.**

Module 15

Unit Testing

1. **What is unit testing?**
2. **Aims and objectives of unit testing.**
3. **The need for unit testing.**
4. **Overview of tools used for unit testing.**
5. **Unit testing tool for Python apps.**

Module 16

Parallel, Multithreaded, and Network Programming

1. **Parallel and multithreaded programming.**
   - Creating threads.
   - Synchronizing threads.
   - Task queues.
   - GIL and features of multithreading implementation in Python.
   - Processes and sharing data between processes.
2. **Network programming.**
   - HTTP/HTTPS.
   - OSI model, TCP/UDP.

- Client-server model.
- Implementation of a simple, multithreaded, and asynchronous echo server.
- Apache and Ngnix.

Module 17

Introduction to Databases

1. **Introduction to database theory.**
    - History and development stages.
    - The concept of databases and database management system.
    - Comparison of the existing database models:
        - file model;
        - network model;
        - hierarchical model;
        - relational model;
        - object-oriented model.
    - The concept of the relational database model.
    - Codd’s 12 rules.
    - MySQL DBMS:
        - what is MySQL?
        - MySQL background;
        - MySQL versions;
        - MySQL installation.
    - Tables:
        - primary key;
        - default value;
        - uniqueness.
    - Data types.
    - Indexes.
    - Queries:
        - introduction to structure query language (SQL);
        - SQL. SQL standards;
        - DDL, DML, DCL.
2. **SELECT, INSERT, UPDATE, DELETE requests.**
    - SELECT statement:
        - SELECT clause;

    - FROM clause;
    - WHERE clause;
    - ORDER BY clause.
  - IN, BETWEEN, LIKE keywords.
  - INSERT operator.
  - UPDATE operator.
  - DELETE operator.

3. **Multi-table databases.**
   - Anomalies of interaction with a single-table database:
     - update anomalies;
     - insert anomalies;
     - update anomalies;
     - delete anomalies.
   - Principles of creating multi-table databases:
     - reasons to create multi-table databases;
     - foreign key;
     - links. Link types;
     - data integrity.
4. **Normalization.**
   - The need for normalization.
   - The concept of normal forms.
   - First normal form.
   - Second normal form.
   - Third normal form.
   - Boyce-Codd normal form.
5. **Multi-table queries.**
   - Principle of creating multiple table queries.
   - Cartesian product.
6. **Aggregation functions.**
   - COUNT function.
   - AVG function.
   - SUM function.
   - MIN function.
   - MAX function.
7. **The concept of grouping. GROUP BY keyword.**

8. **HAVING keyword. Comparative analysis of HAVING and WHERE.**
9. **Subqueries.**
   - The need to create and use subqueries.
   - Comparison between subqueries and multiple table queries.
   - Basic idea of subqueries.
10. **Operators used in subqueries, unions.**
    - Operators used in subqueries:
      - EXISTS operator;
      - ANY/SOME operators;
      - ALL operator.
    - Uniting query results:
      - union principles;
      - UNION keyword;
      - UNION ALL keyword.
    - JOINs:
      - the concept of inner join;
      - the concept of left join;
      - the concept of right join;
      - the concept of full join.
11. **Execution plan of query.**
12. **Query optimization.**
13. **The concept of transaction. Use of transaction.**
14. **Views.**
    - Create views;
    - Modify views;
    - Delete views.
    - Change data via views.
15. **Stored procedures.**
16. **Triggers.**

Module 18

Use of Databases in Python

1. **Use of databses.**

2. **ORM systems.**
    - Theory and practice of use.
    - SQLAlchemy.
    - PonyORM and others.
3. **NoSQL databases.**
    - Basics of NoSQL, CAP theorem.
    - Main types of NoSQL databases.
    - Redis DBMS.
    - MongoDB DBMS.
4. **Working with databases, data serialization.**
    - Read and write in the XML DOM format.
    - StAX and SAX parsers.

Module 19

Teamwork, Software Project Management

1. **What is software project management?**
2. **Causes for occurrence of the software project management discipline.**
3. **Gantt charts.**
4. **Important issues in software project management.**
    - What is a project and a software project?
    - What is a life cycle of the software development process?
    - What is project management?
    - What is solo development?
    - What is team development?
    - Analysis of problems of solo and team software development.
5. **Analysis of terms of the topical area.**
    - Process.
    - Project.
    - Staff.
    - Product.
    - Quality.
6. **Project characteristics.**
    - Project type.

- Project objective.
- Quality requirements.
- Budget requirements.
- Deadline.

7. **Costs associated with the project.**
   - Direct.
   - Indirect.

8. **General review of models and methods of development process.**
   - Process stages:
     - requirements identification;
     - design;
     - implementation (coding);
     - integration;
     - testing and debugging (verification);
     - installation;
     - support.
   - Waterfall model.
   - Spiral model.
   - Iterative model:
     - Agile;
     - Scrum;
     - XP.
   - RUP.
   - MSF.
   - Analysis of the existing models and methods.

9. **About Scrum in detail.**
   - What is Scrum?
   - Reasons for Scrum occurence.
   - Roles in Scrum:
     - product owner;
     - team;
     - scrum master.
   - Product backlog:
     - what is product backlog?
     - how to create backlog?
     - how to estimate tasks in backlog?

    - what is scrum board?
    - examples of backlog creation.
  - Sprint:
    - what is sprint?
    - sprint planning;
    - daily scrum;
    - overview of a sprint;
    - retrospective meeting.

Practice
Simulate teamwork according to the Scrum method.
For example, the so-called Lego Scrum.
More details: Scrum Simulation with LEGO Bricks.

Module 20

Frameworks

1. **Classification of web frameworks.**
   - Flask web framework.
   - Template mechanism and Jinja2 template engine.
   - Sessions and forms.
   - Bottle web framework.
2. **Asynchronous web apps.**
   - Tornado framework.
   - Twisted library.
3. **Django – a framework to create web apps.**
   - Django installation.
   - Creating a Django project.
   - Structure of a Django project (url-view-model-template).
   - MVC/MVT pattern.
4. **Models and ORM.**
   - Models and fields.
   - ORM relations between tables, design and DB implementation via the ORM mechanism.
   - Migration.
   - Model managers.
   - Administrative part.

5. **Working with the administrative part, display settings.**
   - Configuration of the administrator interface.
   - Sort, filters, field editing.
   - Data input.
6. **Template language and web form creation.**
   - Templates.
   - Routing, views.
   - Syntax, logical constructs.
   - Context processor.
7. **Creating forms and pages.**
   - Adding pages.
   - GET/POST requests.
   - Static and dynamic content.
   - Authorization.
   - Access right limitations.
   - Validation.
   - AJAX.
8. **Standard tasks.**
   - Built-in class-based views.
   - Using forms with Django CBV.
9. **Standard tasks.**
   - Authentication.
   - Pagination.
10. **Jumping into Django.**
    - Middleware.
    - Signals.
    - Messages.
    - Sessions.
11. **Continue.**
    - Tree like data structures in Django.
    - Mixins.
    - Django debugger.
    - Logging, sending mail.

Module 21

Creating Chatbots with Python

1. **What is a chatbot?**
2. **Aims and objectives of chatbots.**
3. **Chatbot architecture.**
4. **Application example of creating a chatbot.**

Module 22

Exam