Jupyter Notebook, JupyterLab – Integrated Environment for STEM Education

2022 VI International Conference on Information Technologies in Engineering Education (Inforino) | 978-1-6654-0577-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/Inforino53888.2022.9782924

Valery F. Ochkov
National Research University "MPEI"
Moscow, Russia
OchkovVF@mpei.ru

Alan Stevens
Moscow, Russia
alanstevens5@virginmedia.com

Anton I. Tikhonov
National Research University "MPEI"
Moscow, Russia
TikhonovAI@mpei.ru

***Abstract*— The article deals with the application of integrated environments Jupyter Notebook and JupyterLab in the learning process together with STEAM (Science, Technology, Engineering, Art, Mathematics) technology. It is shown that these environments support basic learning activities including formulating problem statements with formulas and images, performing scientific and engineering calculations, computational experiments, creating graphical user interfaces, publishing interactive e-textbooks, and interactive web-applications. The article describes how to conduct classes using Jupyter Notebook and JupyterLab.**

Keywords*— *STEM, STEAM, Jupyter project, JupyterLab, Jupyter Notebook, computational document, integrated environment, Python, Markdown, Latex

I. INTRODUCTION

Nowadays, STEM (Science, Technology, Engineering, Mathematics) technologies are increasingly used in teaching science and engineering disciplines [1, 2]. STEM makes teaching visual, reveals interdisciplinary connections, and introduces elements of creativity into the learning process: adding A (Art) symbol to the STEAM acronym [3].

STEM application implies solving a large number of tasks [4-8] and intensive interaction of students and teachers in the learning process.

At the same time, STEM imposes additional requirements to the environment in which classes are conducted. Indeed, such an environment should provide all the following kinds of activities in the classroom:

- Description of problem statement (formatted text, formulas, pictures, videos).
- High-level tools for solving scientific and technical problems.
- Visualization tools for presentation of results.
- Tools for computational experiments, including the solution of both direct and inverse problems. In the latter case, input values must be determined, given restrictions on the values of output parameters.
- Publication of solved problems in the form of class notes with active content. Depending on the task, the source code of the solution can be available to students or it must be hidden. The latter is necessary when the student is asked to solve an inverse problem, which is possible in a reasonable time only if the student knows about the relationships between the input and output parameters of the system, at least at a qualitative level.
- Additional opportunities to work with the environment, for example, the possibility of transforming the solution of the problem into a presentation for use in a discussion of the solution, or simple tools for generating tasks for a group of students.
- The possibility of using the environment to conduct classes not only in the computer classrooms of the university, but also in distance learning, which has become relevant in the conditions of the COVID-19 pandemic.

STEM and STEAM classes typically use one or more general-purpose mathematical systems as an integrated environment, such as the proprietary Mathcad, Matlab, Maple, Mathematica, and the freely available Octave, Scilab, and SMath. Common to the above systems is an integrated environment for solving scientific and technical problems, based on the concept of the *computational document*, a set of libraries, built-in programming tools. Basically, the systems support numerical methods of solving problems, but some of them, such as Mathematica, Maple, Mathcad allow one to apply analytical methods. The system to be used is related to the set methodological tasks, the traditions established in a particular university and the availability of licenses (enough to carry out the educational process in full-time and distance learning).

This article analyzes the use of the Jupyter Notebook (JN) [9] and JupyterLab (JL) [10] as an integrated environment for STEM classes. Currently, the primary integrated environment is considered to be JL, which has significantly more features compared to JN. Last year we also started to use the VS Code editor [11], which supports JN; moreover, VS Code also allows us to work conveniently with other language environments, including html and Javascript.

At a glance, let us note that JN, JL, VS Code are distributed freely, and whose environments fully meet the requirements listed above.

II. METHODOLOGY OF CLASSES

The authors give courses on scientific, technical and engineering calculations [11, 12]. During the classes, problems are solved and the main tools used are Mathcad and the Python ecosystem. Mathcad as a working tool has been used for more than 25 years. The Python ecosystem has

been used in the educational process since 2015. It was chosen for the following reasons: it has an extremely wide set of libraries for scientific, technical and engineering calculations, a low entry threshold, and is free of charge.

A number of requirements are imposed on the tasks to be solved in the classroom:

- They should be interesting for students, otherwise students stop solving problems. This is extremely important in distance learning.
- Problem definition should not take too much time; in this connection JN, JL notebooks with problem definitions are sent to students by e-mail together with an invitation to attend classes via Webex or Zoom. The flipped class approach gives the best results, but it can be applied only if there is a motivated audience. Before the class, students should become familiar with the problem statement and the numerical and analytical methods used to solve it. During the obligatory assignments students themselves prepare formulations of problem conditions with necessary formulas, inserting images in JB, JL.
- The problems considered in the class should be chosen so that their solution does not exceed 30-45 minutes. This implies the use of high-level library procedures.

To reduce the time spent on solving problems in class, it is often enough to use pre-designed code in class mainly for visualization. The students use this code when solving homework, but sometimes they offer their own solutions to the problems discussed in class.

Experience shows that the processes of problem decomposition into subtasks, selection of library procedures to solve subtasks, coordination of input and output data of subtasks ("gluing") are the most difficult ones for modern students. It is the process of decomposition – selecting tools to solve subtasks – ensuring data transfer between subtasks – to which students have to repeatedly pay attention in class. The second difficulty is the process of verification of the problem solution, the need to check on test data the correctness of problem solution. To overcome it, you have to intentionally leave hidden errors in the process of implementing the problem solution, which are then localized and eliminated by using test data, naturally attracting students' attention to this.

Another skill that students should acquire while solving problems is to analyze the sensitivity of solution to errors in the initial data.

For this purpose, specially selected problems are solved, which lead to the loss of stability either due to physical processes or numerical methods applied. In the latter case, we implement, for example, an animation of the loss of stability when using an explicit difference scheme to solve the thermal conductivity equation.

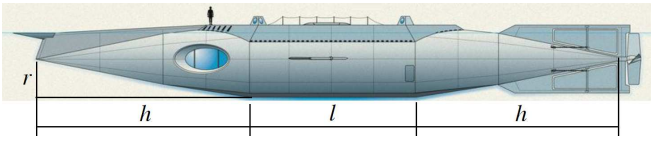

Fig. 1. Schematic representation of the Nautilus submarine.

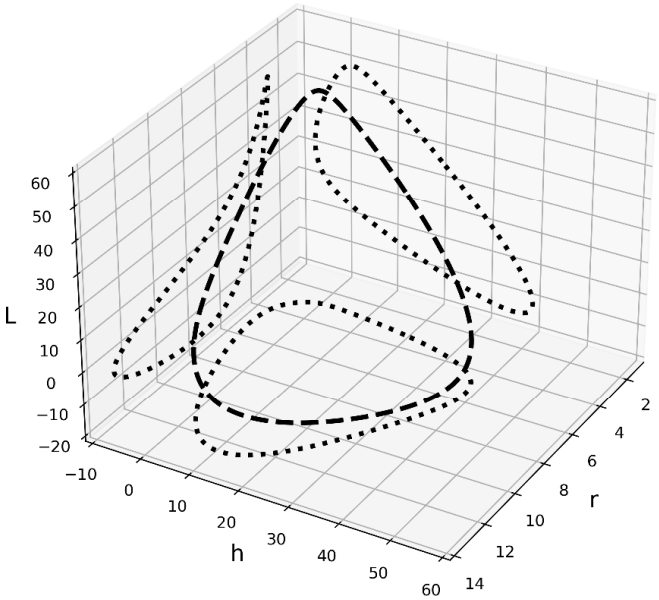

Fig. 2. The spatial curve of the exact solution of the problem and its projections onto coordinate planes.

Let's take a simple example of the problem discussed in the class. In the famous novel 20,000 Leagues Under the Sea by Jules Verne, when asked by Professor Aronax about the dimensions of the submarine Nautilus, Captain Nemo answers that the submarine has the shape of a geometric body composed of two identical straight circular cones (the bow and stern of the boat) and a straight circular cylinder (Fig. 1). The radii of bases of the two cones and the cylinder are equal. We know the volume of the boat (displacement) V=1500 m^3 and the area of its outer surface S=1010 m^2. It is necessary to determine its geometrical dimensions – radius of bases of two cones and one cylinder r, height of two cones (length of bow and stern) h and height of cylinder (hull length) l. This simple problem leads to an underdetermined system of algebraic equations and, depending on the composition of the audience, allows us to consider numerical and symbolic methods for solving algebraic equations, setting initial approximations, limitations related to physical feasibility, and dependence of problem solutions on parameters.

Fig. 2 shows how the solution to the problem depends on the ratio of r, L, and h.

III. Jupyter Project

JN, JL is an evolution of the computational document concept, integrating static formatted text, formulas, multimedia, program code, problem solving results and their visualization in one document – a notebook. Initially it was only possible to work with this environment in Python, but since 2014 the number of supported language processors has expanded, now it is possible to work in JN, JL with more than 30 programming languages and mathematical systems, including Julia, Matlab, Octave, SQL, Python, Javascript and others. Because of this, the IPython (Interactive Python) project, initiated in 2011, was renamed Jupyter in 2014.

Jupyter notebooks are text files in JSON format consisting of at least two types of cells: document cells and computational cells. Document cells can contain static text and video, images. Markdown markup languages [12], html are supported. LaTeX [12, 14] is used for formulas.

Computational cells contain pieces of program code. A set of computational cells in a notebook forms a single namespace.

Notebook cells can be in two states: edit and execute. To run a cell, just click the Run button on the toolbar of JN, JL. To put a cell into edit mode, just click on it with the mouse. Cells can be executed in any order; any cell can be executed an arbitrary number of times.

The programming language of the computational document (notebook) is determined at its creation, for example, selecting one of the versions of Python installed on the user's computer. At the same time, fragments of code in other languages supported with JN, JL can be embedded in the notebook. For example, below is a cell with JavaScript code.

```
%%javascript
alert('Javascript alert function')
```

The first line is a "magic" command indicating that the given cell contains Javascript code, and the result of the second line is the opening of a dialog box with the message text.

In addition, "magic" commands allow one to control the embedding visualizations in JN, JL notebooks, for example,

```
%matplotlib inline
```

implies displaying graphs embedded in Notebook as images.

"Magic" commands allow one to access the operating system and install Python ecosystem libraries without leaving JN, JL.

IV. JN, JL Operating Environment

JN, JL can be used in a variety of ways in the classroom. Most often they are installed together with the Anaconda [15] distribution for Windows, Linux and macOS operating systems or WinPython [16] for Windows only. This approach provides a completely ready working environment for scientific and technical calculations. In case additional packages need to be installed, one can use the package manager conda in the first case and pip in the second. As of version 3.0 or higher, JL provides interactive debugging, but one must install the xeus-python kernel for that. Installing the Anaconda and WinPython distributions is easy enough and can be done without difficulty by students on their home computers.

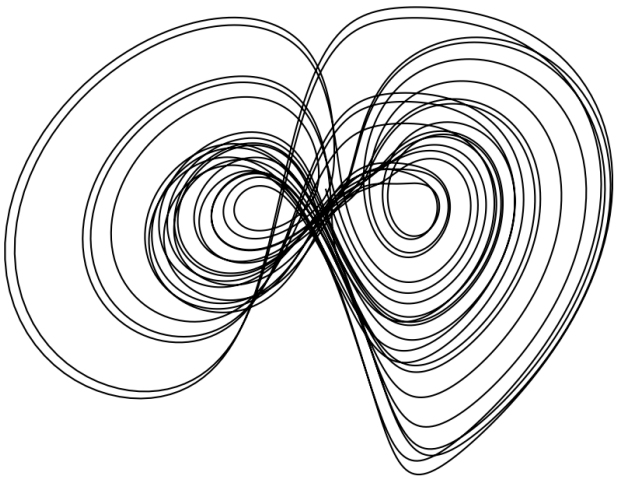

Fig. 3. Phase portrait of the Sakarya Attractor.

The installation of the desktop environment is greatly simplified when using the JupyterLab Desktop App (JLDA) [17], which is a desktop application based on Electron. JLDA integrates with the Anaconda distribution.

JupyterLite [18] is a JL environment fully functioning in a browser with most of the libraries needed for the learning process. The advantage of JupyterLite is that the distribution can be saved on a web server, and users need to specify the url of the distribution file to start working, once downloaded, JupyterLite starts working immediately. The disadvantage of JupyterLite is the difficulty of installing additional libraries, as this requires compilation to WebAssembly.

An integrated environment for working with JN, JL for groups of users not exceeding 30 people can be created with The Littlest JupyterHub (TLJH) [19]. TLJH is a server on which notebooks run, the users' computing devices serve only to work with the user interface. Thus, when using TLJH, only a browser is required, nothing needs to be installed on the user side, and it is possible to use not only personal computers but also tablets, and some students even try to work from smartphones. In addition, TLJH can be used to run Voila [20] and Streamlit [21] web applications. The same server is also useful for publishing teaching materials as well as student works.

Almost all problems are solved with the help of the Python ecosystem; the basic libraries are NumPy (Numerical Python) and SciPy (Scientific Python), as well as libraries for solving problems in specific subject areas. For visualization the matplotlib and seaborn libraries are used, and plotly used for interactive visualization. Animation is widely used. For example, when solving problems containing systems of differential equations, students build animations of phase portraits for the systems of equations (Fig. 3), with rotation of the animation around one of the coordinate axes.

To perform computational experiments, students build graphical user interfaces (GUI) for the tasks they solve using JN, JL widgets (ipywidgets library). For the tasks, solved in class, the automated means of GUI construction are mainly used – decorators interact and interact_manual. For home tasks students build more complex GUI, including adaptive ones, adjusted to the size of the browser window, in which they are executed, thanks to all means available in the library ipywidgets. Fig. 4 shows the user interface animating the construction of so-called "magic" curves on the complex plane. The curves are constructed according to the formula:

$$w(t)=\Sigma_k\, a_k\, exp(ib_k t)$$

where a_k are complex numbers b_k are integers, t is a parameter taking values from 0 to 2π, and i is the unit imaginary number. The geometrical placement of points describing "magic" curves is a sequence of circles, the center of the next circle moves along the previous one, as shown in Fig. 4. The application was written with student participation and is given as an illustration of STEAM approach – mathematics generates attractive artifacts.

In our opinion, it is important in the application of STEM and STEAM technologies to solve inverse problems, when the student is given a digital twin – a realized mathematical model from the subject area and he/she needs to select the values of input parameters so that the values of output parameters lie within the specified limits. In fact, this is an engineering design problem. Even with three or four input

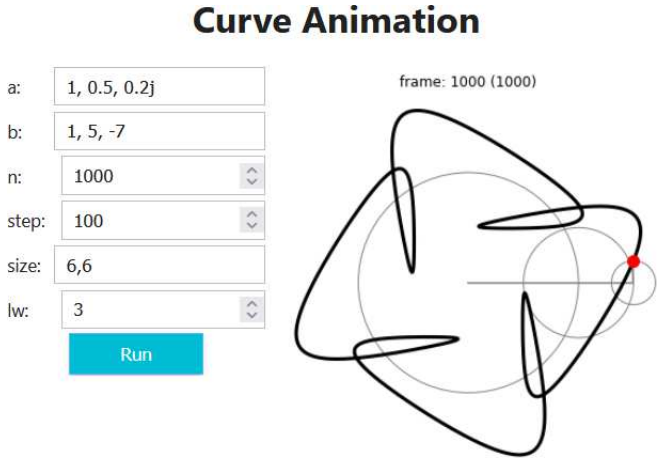

Fig. 4. "Magic" curve animation application.

parameters it is impossible to solve the inverse problem by brute force without qualitative and quantitative knowledge of relations among parameters of the designed system.

This raises the question of how to provide students with the ability to interact with digital twins without access to the source code of their implementation. Voila [20], PyViz Panel [22], Plotly Dash [23] and Streamlit [21] technologies give an answer to this question. When applying JN, JL in the learning process, Voila is the most attractive because it allows one to run JN, JL notebook applications without additional modification by only pressing the button on the toolbar. At the same time, only document cells and interactive applications with GUI built with ipywidgets are available to the user.

Panel requires the use of its own set of widgets and installation of an additional server, the use of Panel is reasonable only when working intensively with geographical data using the GeoViews library built into the Panel. At present, the Panel is not used in the educational process.

The application area of Dash is the creation of virtual labs. You can work with Dash applications in JN, JL, but it is the only technology listed above that allows integration into traditional Flask-based web applications. This technology is studied as an elective and is used in homework.

Streamlit – a new technology for creating interactive applications with graphical user interface, incompatible with JN, JL, but simple and intensively developed.

Naturally, most of the interactive applications used in solving inverse problems are developed by teachers, but in the last two years we are beginning to use in the educational process interactive applications created by students when doing homework and graduate qualification works.

When solving inverse problems, sensitivity analysis plays an important role. Indeed, if small variations in input parameters values lead to a significant change in output parameters, then such a system becomes unstable. This is why we include statistical modeling in our disciplines. Due to its simplicity and clarity, statistical modeling is well received by students and allows simulation of "technological processes" of production of digital twins developed by students when studying other sections of the disciplines. Special attention is paid to the presentation and visualization of the results of statistical modeling.

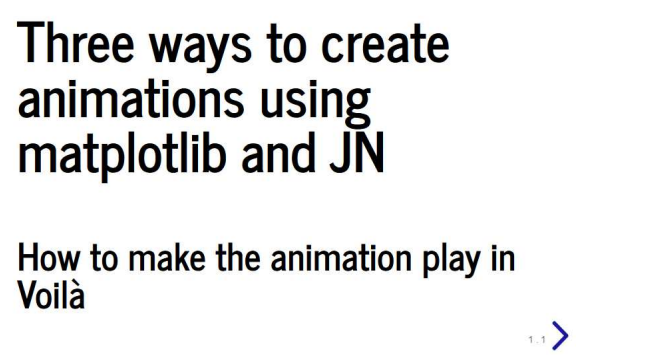

Fig. 5. Slideshow in JN.

An important advantage of JN, JL is the ability to use extensions built into notebooks. The most common extension is Rise, which allows one to turn a JN, JL notebook into an interactive slideshow; it is widely used in conducting classes by teachers, as well as in the presentation of completed homework by students. To do this you need to mark up the cells of the notebook in order to create a navigational structure of the slideshow. Cell markup is quite simple and can be mastered in 5-10 minutes. Switching to the demonstration mode is done by pressing the button on the toolbar. Fig.5 shows an image of the presentation cell used in the class, at the bottom right there is a button to move to the next slide of the presentation.

Another very attractive extension is Mito, which allows you to embed full-featured spreadsheets in JL notebooks. Mito is quite tightly integrated with the pandas library, providing data input and output in the formats supported by the library. The formulas of spreadsheet cells can use Python code.

nbgrader [25] is usually installed on the TLJH server and performs LMS functions, supporting course structures, users account management, test assignments. It should be noted, however, that the documentation for nbgrader is written so that mastering it requires considerable effort. Individual assignments for students are generated by filling in templates using utilities written in Python and Jinja2 [26], and sent to students in the form of JN, JL notebooks. This is done quite simply because JN, JL notebooks are text documents.

Finally, the ready JN, JL notebooks are published. Notebooks intended for general use can be published on GitHub [27].

To publish tutorials, we use our own static site generator that assembles electronic tutorials from heterogeneous material, including html and pdf files, videos, images, The structure of tutorials is described in the form of Excel spreadsheet. The result of compilation is a zip archive, which is deployed both on the discipline's web server and on students' local computers. The static site with tutorial integrates with the TLJH server, allowing web applications prepared with Voila and Dash technologies to run in the context of the e-textbook. Conversion of JN, JL notebooks into html and pdf is done using the nbconvert utility and JN, JL built-in tools.

V. Conclusion

JN, JL is a convenient universal, integrated environment that supports almost all kinds of activities when using STEM and STEAM technologies in the learning process, from

describing problem conditions, developing interactive electronic textbooks, through calculations, visualizations, animation and computational experiments to their publication in the form of web applications embedded in electronic textbooks. JN, JL notebooks are turned into interactive slideshows that can be used for classes with a minimum of effort. For JN, JL the developers and the community of users developed a large number of extensions that increase the functionality and convenience of working with notebooks.

However, it should be noted that JN, JL has a number of drawbacks, the main one of which is that notebook cells form a global namespace and allow non-linear interactive execution. This leads to a number of undesirable effects. For example, if you assign values to variables in a cell and then delete that cell, the variables will remain in the application context and affect the execution of other cells. This can be avoided by running the Kernel | Restart & Run All command after repeated or non-linear execution of notebook cells.

Another major drawback is that the lack of an interactive debugging facility has now been eliminated in JL in versions starting with 3.0.

Finally, JN, JL is not suited for writing large programs. This is not too relevant to the learning process and can be overcome by developing separate modules in the form of JN, JL notebooks, saving them as Python modules and then importing them into other notebooks.

REFERENCES

[1] N. A. Wahab, O.Talib, F. Razali, N. Kamarudin, "The Big Why of Implementing Computational Thinking In STEM Education. A Systematic Literature Review". Malaysian Journal of Social Sciences and Humanities (MJSSH), vol. 6, # 3, 2021, pp 272-289.

[2] T. R. Kelley, J. G. Knowles, "A conceptual framework for integrated STEM education". International Journal of STEM Education vol 3, # 1, 2016, pp 1-11.

[3] C. Liao, "From Interdisciplinary to Transdisciplinary: An Arts-Integrated Approach to STEAM Education," Art Education, vol. 69, $. 6, 2016, pp. 44-49.

[4] N. Atqiya , L. Yuliati , M. Diantoro, "Argument-driven inquiry for STEM education in physics: Changes in students' scientific reasoning patterns". 2021 THE 4TH INTERNATIONAL CONFERENCE ON MATHEMATICS AND SCIENCE EDUCATION (ICoMSE) 2020. Innovative Research in Science and Mathematics Education in The Disruptive Era, vol. 2330, # 1, pp 50022

[5] J. W. Bequette, M. B. Bequette. "A Place for Art and Design Education in the STEM Conversation. Art Education". 2012, vol. 65, # 2, 2012, pp 40-47.

[6] P. Sengupta, A. Dickes, A.Farris, Toward a Phenomenology of Computational Thinking in STEM Education". arXiv: Physics Education, 2017, pp 49-72

[7] L. Thibaut, S. Ceuppens, H. De Loof, J. De Meester, L. Goovaerts, "Integrated STEM Education: A Systematic Review of Instructional Practices in Secondary Education" European Journal of STEM Education, vol. 3, # 1, 2018, pp 1-12.

[8] C. Garibay, R. M. Teasdale. "Equity and Evaluation in Informal STEM Education". 2019 New Directions for Evaluation, vol. 2019, # 161, pp 87-106

[9] D. Toomey, Jupyter Cookbook. Packt, Birmingham-Mumbai, 2018.

[10] "Jupyter. JupyterLab: Jupyter's Next-Generation Notebook Interface". Jupyter, available at: https://jupyter.org/, [accessed: 20 November 2020].

[11] M.A.K. Ovais, H. Khusro, Developing Multi-Platform Aps with Visual Studio Code. 1st edn., Packt, Birmingham-Mumbai, 2020.

[12] A.I. Tikhonov, "Scientific and technical calculations in Python", 1st edn, MPEI Publishing House, Moscow, 2020 (in Russian).

[13] Engineering calculations. Biligual Book. Lan'.Sanct-Petersburg, 2021 (In Rusian and English)

[14] M. Goossens, F. Mittenbach. S. Rathz, The LaTeX Grapic Companion. Second Edition. Addisson-Wesley, Boston, 2008.

[15] Anaconda Individual Edition. The World's Most Popular Python/R Data Science Platform. URL: https://www.anaconda.com/distribution/ (accessed on 20 November 2021)

[16] WinPython. The easiest way to run Python, Spyder with SciPy and friends out of the box on any Windows PC, without installing anything! URL: https://winpython.github.io/ (accessed on 20 November 2021).

[17] JupyterLab Desktop. URL: https://github.com/jupyterlab/jupyterlab-desktop (accessed on 22 November 2021).

[18] JupyterLite. URL: https://github.com/jupyterlite/jupyterlite URL: https://github.com/jupyterlite/jupyterlite (accessed on 22 November 2021).

[19] The Littlest JupyterHub. URL: https://tljh.jupyter.org/en/latest/ (accessed on 22 November 2021).

[20] Voila-dashbords. URL: https://github.com/voila-dashboards/voila (accessed on 22 November 2021).

[21] Awesome Streamlit. URL: https://awesome-streamlit.readthedocs.io/ (accessed on 22 November 2021).

[22] Panel. A high-level app and dashboarding solution for Python, available at: https://panel.pyviz.org/ (accessed: 22 November 2021).

[23] Dash User Guide, available at: https://dash.plotly.com/ (accessed: 22 November 2021)..

[24] K. Gupta, "Exploring Mito: Automatic Python Code for SpreadSheet Operation". URL: https://www.analyticsvidhya.com/blog/2021/06/exploring-mito-automatic-python-code-for-spreadsheet-operations/ (accessed: 22 November 2021).

[25] H. Liaqat. "5 Steps to Auto-grade your Jupyter Notebooks-nbgrader simplified". URL: https://medium.com/analytics-vidhya/5-steps-to-auto-grade-your-jupyter-notebooks-nbgrader-simplified-4cbebf8943ef (accessed: 22 November 2021).

[26] Welcome to Jinja2. URL: https://jinja2docs.readthedocs.io/en/stable/ (accessed: 22 November 2021).

[27] Where the world builds software. URL: https://github.com/ (accessed: 22 November 2021).