The OSCAR PUBLISHING Services
American Journal of Applied Science and Technology

VOLUME Vol.05 Issue 10 2025
PAGE NO. 212-217
DOI 10.37547/ajast/Volume05Issue10-37

Writing Chemistry-Related Programs In The Python Programming Language

Atavulloyev Hafiz Hayotovich
Assistant of Bukhara State Medical Institute, Uzbekistan

Received: 30 August 2025; **Accepted:** 24 September 2025; **Published:** 28 October 2025

Abstract: In recent days, programming has become the most popular and interesting subject among our youth. Programming languages that are relatively easy to learn for chemists were reviewed, and the relationship between programming and chemistry was studied. For this reason, in order to create a connection between disciplines, we created a GUI program based on chemical calculations in the Python programming language and provided detailed information about its capabilities. When writing this program, a programming environment that is considered somewhat convenient for writing codes was chosen, and the codes were written in this programming environment. The results obtained were compared with previous results. Through this article, we have shown that we need to be able to find solutions to many problems in chemistry through programming and that we need to study all sciences in more depth. We have shown that the created program can be used not only in educational processes and in laboratory enterprises, but also in a number of its achievements in the article.

Keywords: Python, Pycharm, PyQt5, GUI window, chloride, hardness, crucible, Mercurometric, titration, EDTA.

INTRODUCTION:

Nowadays, programming languages are among the most widely studied fields in our country. Programs created using programming languages have also found broad application in the fields of chemistry and scientific analysis. Programming languages are being continuously improved and updated year by year. Among the most widely used of these languages is Python.

Python is a powerful general-purpose programming language that is widely used in web applications, software development, data science, machine learning, and the natural sciences. It was developed in 1989 by Guido van Rossum. Python is classified as an interpreted programming language that automates many low-level operations (such as memory management) that are normally executed at the processor level ("machine code"). Thanks to its expressive syntax (which is, in many cases, close to natural language) and the rich variety of built-in data structures—such as lists, tuples, sets, and dictionaries—Python is considered a high-level language compared to, for example, C++. [1]

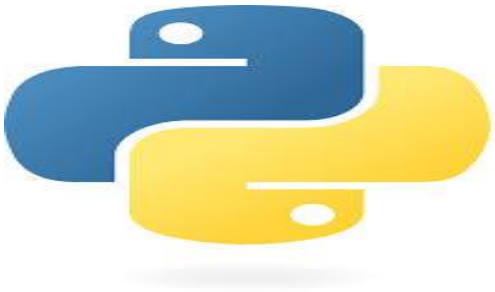

Figure 1. Logo of the Python program.

An essential skill for all chemists is the ability to process, analyze, and visualize data. A key tool that aids in this is the computer, and the ability to use it effectively for such tasks is a crucial skill for both current and future chemists. While electronic spreadsheets can help perform some of these tasks, there are many challenges—such as analyzing very large datasets, conducting multidimensional analyses, and analyzing images—that require more advanced tools and knowledge to solve [2]. Unfortunately, most chemists today prefer to work with ready-made software rather than learning programming languages and how to use them. However, most of these ready-made programs require paid subscriptions or offer limited versions

with restricted functionality. This, in turn, hinders the processing of certain types of data.

The Python programming language, on the other hand, is completely free and available for all operating systems. Many open-source [3] scientific computing libraries also provide Python interfaces. Users can install Python and its various extension libraries on any computer. Due to its simplicity, clarity, and extensibility, more and more research institutions are using Python for scientific computing, and an increasing number of universities are adopting Python for teaching computer programming.

In addition, many scientific computing libraries specific to the Python programming language—such as NumPy, SciPy, and Matplotlib—have been developed. These libraries provide Python with capabilities for fast array processing, numerical operations, and scientific plotting.

This programming language can also access geospatial data processing libraries, such as those provided by the Open Source Geospatial Foundation, which enable the processing of raster and vector data as well as the reprojection of spatial information.

At present, scientists around the world are widely using Python in inorganic chemistry to understand vibrational modes, predict infrared (IR) and Raman spectra, and accurately assign and interpret observed spectra by applying group theory and symmetry principles [4]. It is also extensively applied in structural chemistry, quantum chemistry, thermodynamics, kinetics [5], molecular visualization of proteins and DNA [6], and electronic structure calculations [7].

Basics of Coding:

First of all, many project libraries and auxiliary tools are based on this programming language. It is known that the Python programming language is slower than C++ and similar languages.

On the other hand, most libraries such as NumPy [8] are written in C++, highly optimized, and provide an interface that allows for much easier and faster development in Python. Python code is also more readable and concise, allowing us to present more examples in a simpler form.

To create programs with a graphical user interface (GUI), Python provides libraries such as Tkinter [9] and PyQt5 [10].

Nowadays, most available software is paid or operates online through the Internet. For this reason, we developed a Python-based program with an interface designed to calculate the concentration of ions in water samples. This newly developed program is relatively easy to use, does not require an Internet connection, and most importantly, is completely free.

METHODOLOGY

In this study, the PyCharm programming environment was chosen for software development. For the chemical analysis of ions in water, relevant methodologies and GOST standards were taken as the basis.

Before writing the code for the program we intended to create, it was necessary to select an appropriate programming environment. There are several available programming environments, such as PyCharm, Visual Studio, and Atom.

We chose PyCharm because it is free and easy to use, and then installed it on our computer.

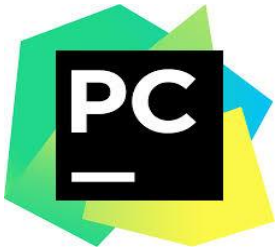

Figure 2. PyCharm environment logo

We download the required libraries through the Terminal section and then begin entering the necessary data.

```
class WaterQualityCalculation (QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setWindowTitle("Water Quality Calculation")
        self.setGeometry(100, 100, 700, 500)
```

```
        palette = self.palette()
        palette.setColor(QPalette.Window, QColor("lightblue"))  #
Background color for the main interface
        self.setPalette(palette)

        main_layout = QHBoxLayout()
        form_layout = QGridLayout()
        result_layout = QVBoxLayout()
```

Figure 3. The process of writing the program

After all the necessary libraries have been installed, we begin coding based on the previously prepared information. The input data are as follows:

1. Water volume – determined according to the density of the water. For example, the higher the density, the larger the volume of the sample to be analyzed should be.
2. Density – the density of the water mixture being analyzed. This is usually measured using an areometer.
3. Chloride – the volume of the solution consumed during the titrimetric (mercurimetric) determination.
4. Excess chlorine – the volume of the mercury-containing solution determined by back titration.
5. Hardness – the volume of EDTA (Trilon B, disodium salt) solution consumed during the titrimetric determination of carbonate hardness.
6. Calcium – the volume of EDTA (Trilon B, disodium salt) used in the determination of calcium content.
7. Bicarbonate, carbonate, and total carbonate content – the volume of HCl solution consumed during conductometric titration.
8. Crucible 1 – the mass of $BaSO_4$ measured gravimetrically.
9. Crucible 2 – the mass of the dry crucible measured gravimetrically.

After encoding the input data, we proceed to code the formulas used in the results section.

```
self.result_text.setText(
        f" Chlorine: {xl:.2f} mg/l\n"
        f" Hardness: {bb:.2f}\n"
        f" Calcium: {ca:.2f} mg/l\n"
        f" Magnesium: {mn:.2f} mg/l\n"
        f" Bicarbonate: {gd:.2f} mg/l\n"
        f" Carbonate: {ka:.2f} mg/l\n"
        f" Sulfate: {o4:.2f} mg/l\n"
        f" Sodium: {na:.2f}\n"
        f"CO2: {co:.2f} mg/l\n"
        f" Total Mineralization: {mi:.2f} mg/l"
    )
except ValueError:
    QMessageBox.critical(self, " Error", "Numbers entered are in the
wrong format!")
```

Figure 4. The code used to transfer data to the Graphical User Interface (GUI).

We repeatedly test the accuracy of the code using PyCharm and then proceed to the results section.

RESULTS AND DISCUSSION

After testing the developed program, we analyze its output results. First of all, we create the main GUI window and the program components.

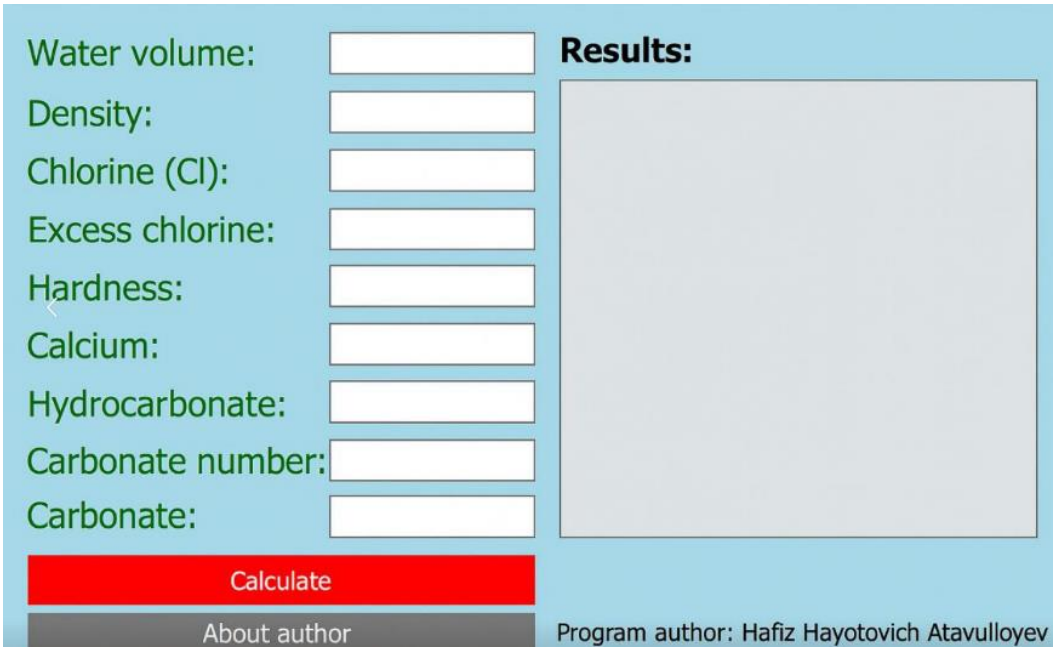

Figure 5. The initial version of the program with a Graphical User Interface (GUI).

The program window displayed exactly as we had written in the code. Our next test involves leaving one of the input fields empty when entering the data.

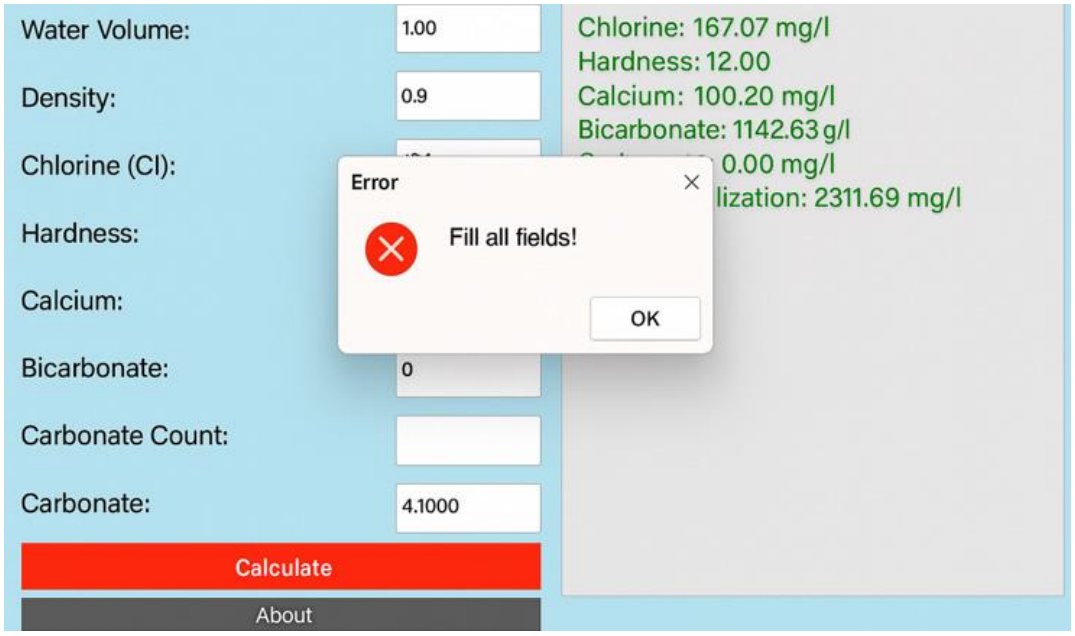

Figure 6. The state of the program when one of the fields is left empty.

This means that the program returned an error because we left a field empty, which confirms that our program is functioning properly.

Now we can proceed to calculate the practical results. For this, we need the data obtained from chemical analyses.

After entering all the necessary data into our program, it automatically performs the calculations and provides the corresponding results.

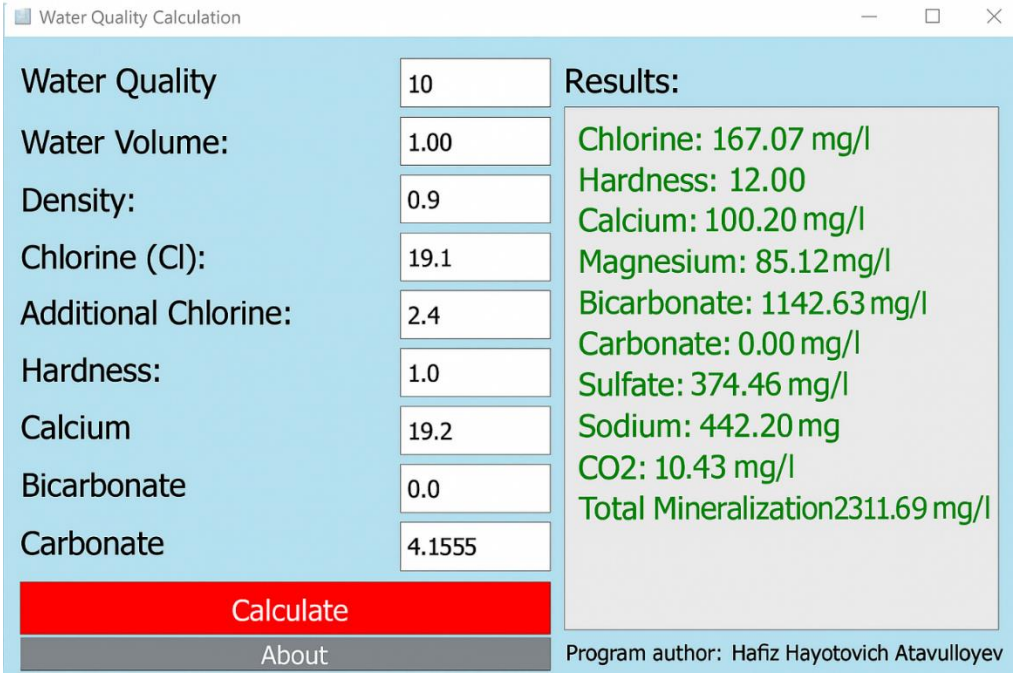

Figure 7. The data displayed in the program when all the required information is entered.

CONCLUSION

The developed program can be effectively used in industrial laboratories and educational processes. Nowadays, most similar programs are either online or paid versions, and working with them can be somewhat complicated. Therefore, such programming approaches significantly increase work efficiency.

Today, many young people are interested in programming and often consider other sciences unnecessary. However, many scientific problems can, in fact, be solved through programming. Programs like this one contribute to strengthening the interdisciplinary connection between programming and other scientific fields.

REFERENCES

1. Ryzhkov, F.V.; Ryzhkova, Y.E.; Elinson, M.N. Python in Chemistry: Physicochemical Tools. Processes 2023, 11, 2897. https://doi.org/10.3390/pr11102897
2. Menke, E. J. (2020). Series of Jupyter notebooks using Python for an analytical chemistry course. Journal of Chemical Education, 97(10), 3899–3903. https://doi.org/10.1021/acs.jchemed.9b01131
3. Lafuente, D., Cohen, B., Fiorini, G., García, A. A., Bringas, M., Morzan, E., & Onna, D. (2021b). A gentle Introduction to Machine Learning for Chemists: an undergraduate workshop using Python notebooks for visualization, data processing, analysis, and modeling. Journal of Chemical Education, 98(9), 2892–2898. https://doi.org/10.1021/acs.jchemed.1c00142
4. Ayeni, O. M., Migliaro, I., Omary, M. A., & Atkinson, M. B. (2025). SYMmSPEC: an interactive Python tool for predicting IR and RAMAN activity for undergraduate inorganic chemistry. Journal of Chemical Education. https://doi.org/10.1021/acs.jchemed.4c01137
5. Hutchison, G. R. (2021). Integrating Python into an Undergraduate Mathematics for Chemists Course. In ACS symposium series (pp. 123–134). https://doi.org/10.1021/bk-2021-1387.ch009
6. Dong, J., Yao, Z., Zhang, L., Luo, F., Lin, Q., Lu, A., Chen, A. F., & Cao, D. (2018). PyBioMed: a python library for various molecular representations of chemicals, proteins and DNAs and their interactions. Journal of Cheminformatics, 10(1). https://doi.org/10.1186/s13321-018-0270-2
7. Boguslawski, K., Leszczyk, A., Nowak, A., Brzęk, F., Żuchowski, P. S., Kędziera, D., & Tecmer, P. (2021). Pythonic Black-box Electronic Structure Tool (PyBEST). An open-source Python platform for electronic structure calculations at the interface between chemistry and physics. Computer Physics Communications, 264, 107933. https://doi.org/10.1016/j.cpc.2021.107933
8. Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.
9. Moore, A. D. (2018). Python GUI Programming with Tkinter: Develop responsive and powerful GUI applications with Tkinter. Packt Publishing

Ltd.

10. Meier, B. (2019). Python GUI Programming Cookbook: Develop functional and responsive user interfaces with tkinter and PyQt5. Packt Publishing Ltd.