BMC Bioinformatics

SOFTWARE **Open Access**

PyChelator: a Python-based Colab and web application for metal chelator calculations

Emrulla Spahiu[1†], Esra Kastrati[2†] and Mamta Amrute-Nayak[1*]

†Emrulla Spahiu and Esra Kastrati have contributed equally to this work.

*Correspondence: amrute.mamta@mh-hannover.de

[1] Institute of Molecular and Cell Physiology, Hannover Medical School, Carl-Neuberg-Str. 1, 30625 Hannover, Germany
[2] Lassonde School of Engineering, York University, Toronto M3J 1P3, Canada

Abstract

Background: Metal ions play vital roles in regulating various biological systems, making it essential to control the concentration of free metal ions in solutions during experimental procedures. Several software applications exist for estimating the concentration of free metals in the presence of chelators, with MaxChelator being the easily accessible choice in this domain. This work aimed at developing a Python version of the software with arbitrary precision calculations, extensive new features, and a user-friendly interface to calculate the free metal ions.

Results: We introduce the open-source PyChelator web application and the Python-based Google Colaboratory notebook, PyChelator Colab. Key features aim to improve the user experience of metal chelator calculations including input in smaller units, selection among stability constants, input of user-defined constants, and convenient download of all results in Excel format. These features were implemented in Python language by employing Google Colab, facilitating the incorporation of the calculator into other Python-based pipelines and inviting the contributions from the community of Python-using scientists for further enhancements. Arbitrary-precision arithmetic was employed by using the built-in Decimal module to obtain the most accurate results and to avoid rounding errors. No notable differences were observed compared to the results obtained from the PyChelator web application. However, comparison of different sources of stability constants showed substantial differences among them.

Conclusions: PyChelator is a user-friendly metal and chelator calculator that provides a platform for further development. It is provided as an interactive web application, freely available for use at https://amrutelab.github.io/PyChelator, and as a Python-based Google Colaboratory notebook at https://colab.research.google.com/github/AmruteLab/PyChelator/blob/main/PyChelator_Colab.ipynb.

Keywords: Metal, Chelator, PyChelator, Web application, Python, Calcium, EGTA

Background

The crucial role of metal ions and their ligands, known as chelators, extends across a myriad of biological phenomena, ranging from hemoglobin interacting with iron to chlorophyll binding magnesium. The term "chelate" originating from the Greek word "chela" (the great claw of lobsters [1]), refers to the complex formation between the metal ions and chelators using coordinate bonding.

Maintaining defined free metal ion concentrations in experimental procedures is often critical due to the strong regulatory effects on many isolated protein functions and intracellular systems. Divalent metals like calcium, magnesium, and zinc are commonly encountered in biological studies with chelators like EGTA (ethylene glycol-bis(β-aminoethyl ether)-*N*,*N*,*N*′,*N*′-tetraacetic acid) also known as Egtazic acid, Adenosine triphosphate (ATP), and Ethylenediaminetetraacetic acid (EDTA), which are frequently employed to buffer these metals. However, direct measurement of free metal ions often presents a challenge due to the limited availability of ion-specific electrodes in many laboratories. Consequently, several software calculators have been developed to estimate the free metal concentrations, such as 'SPECS' by Fabiato [2], 'Chelator' by Schoenmakers et al. [3], 'Bound and determined' by Brooks and Storey [4], 'Calcium' by Föhr et al. [5], and 'MaxChelator' by Bers et al. [6]. The distinguishing factors among these tools lie in their availability, user interface, and the stability constants utilized, with minor variations in calculation methodologies. These calculations take into account the affinity of chelators to metal ions and protons at specific temperature, ionic strength, and pH. As most stability constants are typically measured at standard conditions (e.g., 20 or 25 °C, 0.1 M ionic strength), apparent stability constants are initially computed for the user-defined experimental conditions, followed by the determination of the distribution of species in the solution. Variations in results among different calculators arise from the use of different protonation and metal-affinity stability constants. While older calculators were programmed in languages less accessible today, MaxChelator developed in JavaScript by Chris Patton in 2010 [6] is readily available through modern browsers. However, the numerous calculators present in the literature do not allow the selection of constants from other sources, while some allow only manual editing of the existing constants. Therefore, development of a software that allows easy selection of the available constants and entry of user-defined constants, alongside improvements in user interface and accessibility, is expected to be a valuable advancement.

Metal chelator calculations involve multiple arithmetic operations on numbers with many decimal places, potentially introducing rounding errors, as mentioned for JavaScript-based MaxChelator [6]. JavaScript implicitly converts between strings and floating-point numbers and it relies on the standard double-precision 64-bit binary format, accurate up to 15 digits for integers and 17 for decimals [7, 8]. This inherent limitation in precise representation of the decimals as binary can result in accumulated rounding errors and inaccuracies in computed results. In contrast, Python, a contemporary programming language that achieved high popularity in the scientific community, offers several advantages. Python's accessibility, extensive scientific libraries and the flexibility in project integration make it an attractive choice [9]. Notably, Python supports a rich variety of numeric data types (integer, float, and complex), and includes a built-in Decimals module, facilitating precise handling of the decimals [10]. User-defined precision in Python yields more reliable results, particularly in scientific computations. Metal-chelator calculators could benefit from such arbitrary-precision arithmetic with the aim to minimize rounding errors.

To address these needs and concerns, we developed PyChelator web application (https://amrutelab.github.io/PyChelator/), an open source program, based on the well-established and widely used Maxchelator framework. PyChelator offers enhanced user experience and customization options. The Python code in a Google Colaboratory notebook makes the PyChelator functionalities readily available to the Python-using scientific community for further development.

Implementation

PyChelator web application uses JavaScript for calculations and utilizes HTML & CSS for the frontend interface. In Python format, it is accessible as an interactive research notebook (.ipynb) within Google Colaboratory, where users can enter the input parameters via form fields even without coding experience. Upon initiation of the web application, the setup() method is invoked to load the selected chelators and constants, with default constants sourced from the National Institute of Standards and Technology (NIST) Database [11]. A schematic representation of the flow of functions to perform the calculations is depicted in Fig. S1.

Results and discussion

The graphical user interface (GUI) of the PyChelator web application is designed for simplicity of use and comprises three panels. In the left panel users select the calculation mode (Free or Total metal concentration), enter the values for different environmental parameters (i.e., temperature, pH, and ionic equivalence), specify the unit for concentration input, and customize the content of the final report. The top panel features the fields for users to input the concentrations of chelators and metals, and the middle panel contains the buttons to do the calculations that are appended one after the other. Finally, a download option enables users to export the results as a single Excel file. An easy-to-follow tutorial of an example calculation and manual entry of constants are included inside PyChelator, and shown in Fig. 1.

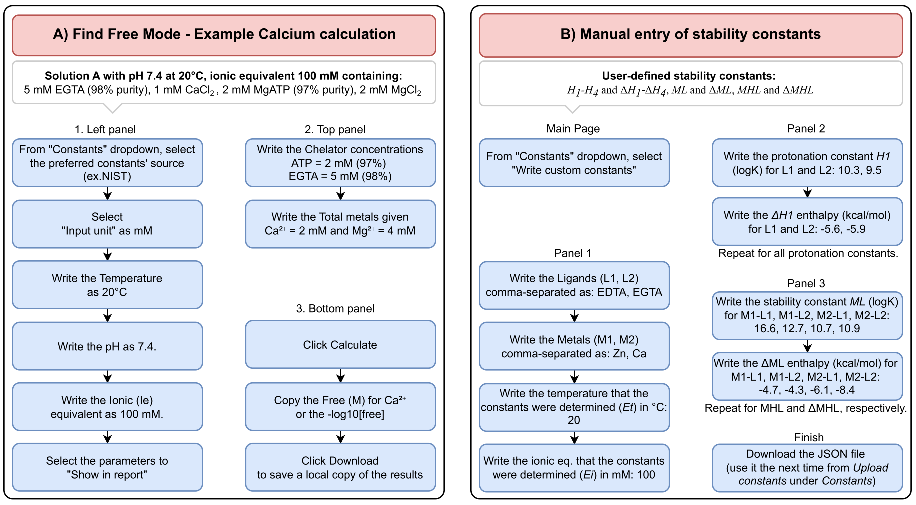
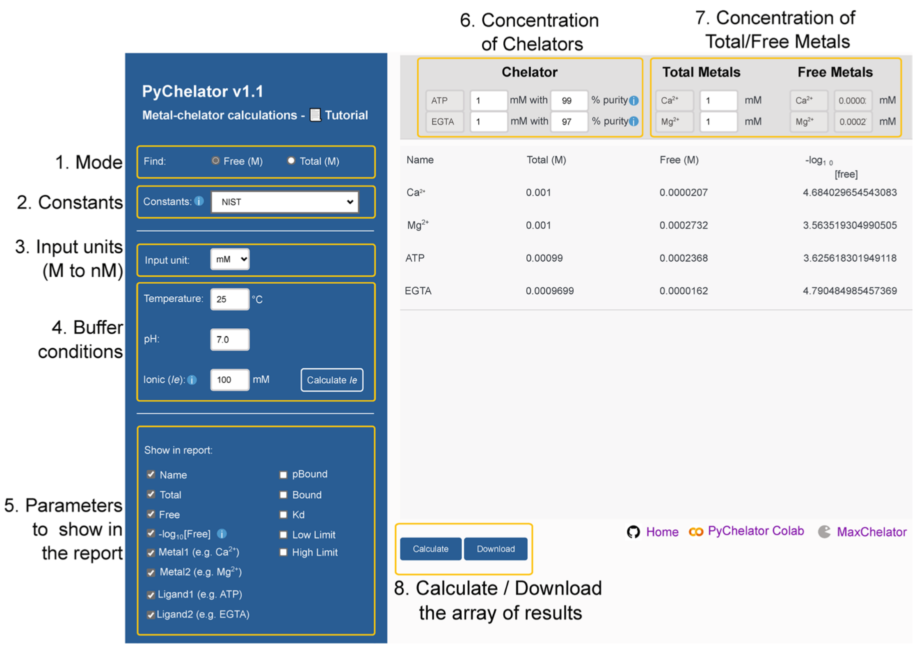

A) Find Free Mode - Example Calcium calculation
Solution A with pH 7.4 at 20°C, ionic equivalent 100 mM containing:
5 mM EGTA (98% purity), 1 mM $CaCl_2$, 2 mM MgATP (97% purity), 2 mM $MgCl_2$
1. Left panel
From "Constants" dropdown, select the preferred constants' source (ex.NIST)
Select "Input unit" as mM
Write the Temperature as 20°C
Write the pH as 7.4.
Write the Ionic (Ie) equivalent as 100 mM.
Select the parameters to "Show in report"
2. Top panel
Write the Chelator concentrations ATP = 2 mM (97%) EGTA = 5 mM (98%)
Write the Total metals given Ca^{2+} = 2 mM and Mg^{2+} = 4 mM
3. Bottom panel
Click Calculate
Copy the Free (M) for Ca^{2+} or the -log10[free]
Click Download to save a local copy of the results
B) Manual entry of stability constants
User-defined stability constants:
H_1-H_4 and ΔH_1-ΔH_4, ML and ΔML, MHL and ΔMHL
Main Page
From "Constants" dropdown, select "Write custom constants"
Panel 1
Write the Ligands (L1, L2) comma-separated as: EDTA, EGTA
Write the Metals (M1, M2) comma-separated as: Zn, Ca
Write the temperature that the constants were determined (*Et*) in °C: 20
Write the ionic eq. that the constants were determined (*Ei*) in mM: 100
Panel 2
Write the protonation constant *H1* (logK) for L1 and L2: 10.3, 9.5
Write the *ΔH1* enthalpy (kcal/mol) for L1 and L2: -5.6, -5.9
Repeat for all protonation constants.
Panel 3
Write the stability constant *ML* (logK) for M1-L1, M1-L2, M2-L1, M2-L2: 16.6, 12.7, 10.7, 10.9
Write the ΔML enthalpy (kcal/mol) for M1-L1, M1-L2, M2-L1, M2-L2: -4.7, -4.3, -6.1, -8.4
Repeat for MHL and ΔMHL, respectively.
Finish
Download the JSON file (use it the next time from *Upload constants* under *Constants*)

Fig. 1 Example uses of PyChelator. **A** An example shows how to calculate the free calcium concentration from the total calcium in the assay buffer used in the in vitro motility assay. **B** The manual entry of stability constants for other metals and chelators. In this case, constants for Zinc and Calcium buffered using EDTA and EGTA were used [11]

User-friendly input and structured output

PyChelator facilitates the entry of smaller concentrations of chelators and metals through a dropdown menu, enabling entry of values from molar (M) to nanomolar range (M, mM, μM, and nM). Users can utilize the top panel to input the desired values for calculations in the selected unit. Notably, all results are consistently expressed in molar units. Temperature is to be written in degrees Celsius (°C) and ionic strength as equivalence of ions ($Ie = 0.5\sum Ci|zi|$), as explained by Smith and Miller, rather than the standard ionic strength [12]). A function to calculate the ionic equivalence inside PyChelator is also included. Regarding the output, checkboxes were implemented in the "Show in report" section in the left panel, in order to introduce user control over the comprehensive parameters and the metals/ligands to be included in the report. Additionally, PyChelator includes an option for the log-transformed values of the free metal concentrations ($-\log_{10}$[free]), corresponding to the pX values (for example pCa for free calcium). In the top panel, a field for purity of chelators was introduced, aiming at addressing the reported significant impact of the chelator purity on calculations [6, 13]. Results are handled in a convenient way. The subsequent calculations are dynamically appended to the middle panel upon clicking the "Calculate" button, and all results can finally be downloaded as a single Excel file. A screenshot of the PyChelator GUI, accompanied by annotations, is presented in Fig. 2. Collectively,


6. Concentration of Chelators
7. Concentration of Total/Free Metals
1. Mode
2. Constants
3. Input units (M to nM)
4. Buffer conditions
5. Parameters to show in the report
8. Calculate / Download the array of results
PyChelator v1.1
Metal-chelator calculations - Tutorial
Calculate Ie
Calculate
Download
Home
PyChelator Colab
MaxChelator

Find: [✓ Free (M)] Total (M)

Constants: NIST

Input unit: mM

Temperature: 25 °C

pH: 7.0

Ionic (Ie): 100 mM

Show in report:

✓ Name	☐ pBound
✓ Total	☐ Bound
✓ Free	☐ Kd
✓ $-\log_{10}$[Free]	☐ Low Limit
✓ Metal1 (e.g. Ca^{2+})	☐ High Limit
✓ Metal2 (e.g. Mg^{2+})	
✓ Ligand1 (e.g. ATP)	
✓ Ligand2 (e.g. EGTA)	

Chelator				
ATP	1	mM with	99	% purity
EGTA	1	mM with	97	% purity

Total Metals			Free Metals		
Ca^{2+}	1	mM	Ca^{2+}	0.0000:	mM
Mg^{2+}	1	mM	Mg^{2+}	0.0002	mM

Name	Total (M)	Free (M)	$-\log_{10}$ [free]
Ca^{2+}	0.001	0.0000207	4.684029654543083
Mg^{2+}	0.001	0.0002732	3.563519304990505
ATP	0.00099	0.0002368	3.625618301949118
EGTA	0.0009699	0.0000162	4.790484985457369

Fig. 2 Screenshot of PyChelator graphical user interface (GUI), illustrating the intuitive layout and functionality. The left panel offers the selection of the desired mode of calculation (step 1), followed by the choice of the constants to be utilized (step 2). Four different sources of constants are available. Additionally, user-defined constants can be entered. Users can specify the unit for concentration input of metal and chelators (step 3), the buffer conditions (step 4), and the parameters for inclusion in the final report (step 5). Subsequently, concentration values of chelators together with the measured or supplier-indicated purity should be specified (step 6), followed by the metal concentration (step 7). Finally, results can be calculated through the "Calculate" button and downloaded as a single Excel file (step 8)

these enhancements contribute to an improved user experience for a more efficient and precise use of the calculator, compared to the pre-existing calculators.

A wide selection and manual input of stability constants

MaxChelator has two sources of absolute stability constants, i.e., the Chelator program by Schoenmakers et al. [3], and NIST database [11]. The user has to choose the calculator and do the calculations using one of these sources. PyChelator is a single page application where users can switch constants by the use of a dropdown menu. Additionally, there are two new constant sets added. One is sourced from Fabiato and Fabiato [2, 14–16], and the other from the Calcium program by Föhr et al. [5]. The equilibrium constants of ATP were corrected for a temperature of 20 °C, using the standard way by employing the enthalpy of reactions, as explained in Bers et al. [6]. Results obtained using the four sets of stability constants were compared using buffers with varying total calcium concentration, keeping other parameters constant (pH 7.0, temperature of 20 °C, ionic equivalent (*Ie*) of 100 mM, 2 mM ATP, 1 mM EGTA, 6 mM $MgCl_2$) as shown in Fig. 3. Results obtained using different stability constants show substantial differences. With a narrow range of pX values playing a significant role, as is the case in muscle physiology, where a pCa range of 5–6 is critical in muscle activation, and with results obtained using different constants, it becomes necessary to report the used set of stability constants during such calculations.

PyChelator further enhances user customization by allowing manual input of user-defined stability constants. This feature is accessible under the "Constants" dropdown menu and labeled "Write custom constants". Users can input their data and download it in JSON format for future use. This feature makes it possible to use PyChelator as "Any Metal Any Chelator Calculator". Some commonly used metal-chelator pairs are included

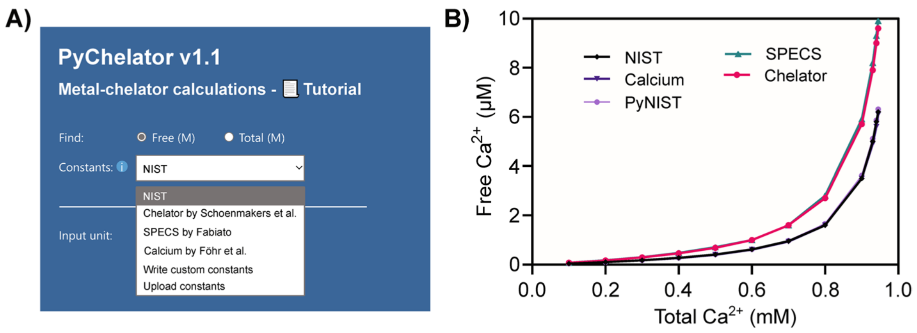

PyChelator v1.1
Metal-chelator calculations - Tutorial
Find: ○ Free (M) ● Total (M)
Constants: NIST
Input unit:

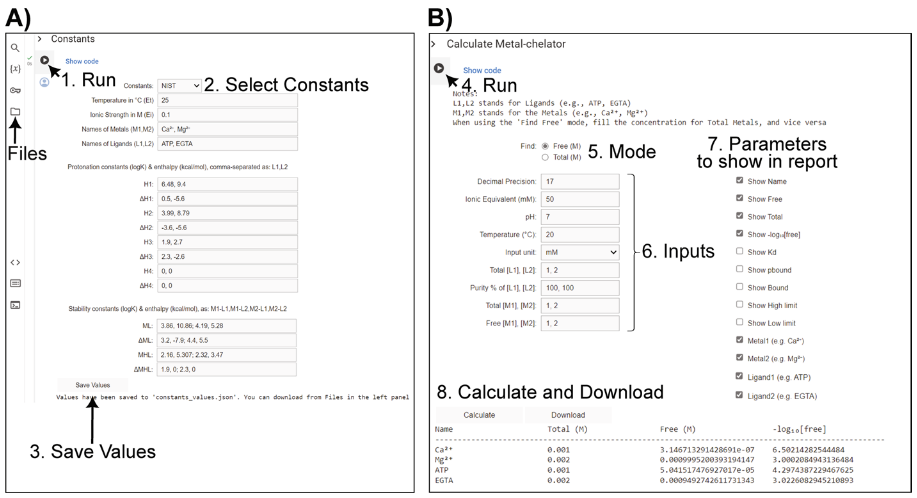
Fig. 3 Comparison of PyChelator results obtained from different stability constants. **A** PyChelator offers the selection of stability constants from four sources, and allows the input of user-defined constants. **B** Calculations done in PyChelator using the four sets of stability constants, sourced from the National Institute of Standards and Technology 'NIST' [11], 'SPECS' by Fabiato [2], 'Chelator' by Schoenmakers et al. [3], and 'Calcium' by Föhr et al. [5]. Buffers composed of varying total calcium concentration and constant pH of 7.0, temperature of 20 °C, Ionic equivalence (*Ie*) of 100 mM, 2 mM ATP, 1 mM EGTA, and 6 mM $MgCl_2$. Results represented in a connecting-line plot show that constants sourced from NIST and Calcium software yield similar results to each other and differ from those generated using the constants from SPECS and Chelator program. Notably, the high-precision calculations performed in PyChelator Colab employing the NIST constants and arbitrary precision of 50 decimals in calculations (PyNIST), were similar to those obtained from PyChelator web application using the same constants (NIST)

together with references under the “Stability Constants” folder in PyChelator Github repository.

PyChelator Colab

The aforementioned features of the calculator were composed in a Google Colab Notebook. It is available in the GitHub Repository for download and as a link in the web application. PyChelator Colab offers a cloud-based interactive Python environment that lets users write and execute code in their browsers. Unlike Flask and Django, which are designed to handle server-side operations and require managing server hosting and domain registration (which can be costly), PyChelator Colab and PyChelator web applications do not require server hosting. They can be easily hosted on free platforms like GitHub Pages without the need for managing server hosting and domain registration.

The main steps to do the metal chelator calculations in this environment are summarized in Fig. 4. The displayed results can be further employed by experienced users as a part of a pipeline, integrating into a larger project where other data are incorporated. PyChelator Colab is also modified to use the built-in Python Decimal module, introducing user-defined precision in the calculations, which are otherwise limited by double precision in the floating-point arithmetic of Python and JavaScript. The use of arbitrary precision arithmetic gives a higher precision in calculations. Nevertheless, the results obtained from the PyChelator web application, which utilizes JavaScript in calculations, compared to PyChelator Colab, did not show a notable difference (Fig. 3B, NIST vs PyNIST). This implies that any calculations done using MaxChelator algorithms are not limited by precision issues related to the employed JavaScript language. However, although the calculated buffers may offer a good approximation with the measured values, challenges like chelator purity or pH can introduce errors even to the best



Constants: NIST
Temperature in °C (Et): 25
Ionic Strength in M (Ei): 0.1
Names of Metals (M1,M2): Ca^{2+}, Mg^{2+}
Names of Ligands (L1,L2): ATP, EGTA

Protonation constants (logK) & enthalpy (kcal/mol), comma-separated as: L1,L2

H1:	6.48, 9.4
ΔH1:	0.5, -5.6
H2:	3.99, 8.79
ΔH2:	-3.6, -5.6
H3:	1.9, 2.7
ΔH3:	2.3, -2.6
H4:	0, 0
ΔH4:	0, 0

Stability constants (logK) & enthalpy (kcal/mol), as: M1-L1,M1-L2,M2-L1,M2-L2

ML:	3.86, 10.86; 4.19, 5.28
ΔML:	3.2, -7.9; 4.4, 5.5
MHL:	2.16, 5.307; 2.32, 3.47
ΔMHL:	1.9, 0; 2.3, 0

```
Values have been saved to 'constants_values.json'. You can download from Files in the left panel
```

```
Notes:
L1,L2 stands for Ligands (e.g., ATP, EGTA)
M1,M2 stands for the Metals (e.g., Ca²⁺, Mg²⁺)
When using the 'Find Free' mode, fill the concentration for Total Metals, and vice versa
```

Find: [✓ Free (M)] Total (M)

Decimal Precision:	17
Ionic Equivalent (mM):	50
pH:	7
Temperature (°C):	20
Input unit:	mM
Total [L1], [L2]:	1, 2
Purity % of [L1], [L2]:	100, 100
Total [M1], [M2]:	1, 2
Free [M1], [M2]:	1, 2

- ✓ Show Name
- ✓ Show Free
- ✓ Show Total
- ✓ Show $-\log_{10}$[free]
- ☐ Show Kd
- ☐ Show pbound
- ☐ Show Bound
- ☐ Show High limit
- ☐ Show Low limit
- ✓ Metal1 (e.g. Ca^{2+})
- ✓ Metal2 (e.g. Mg^{2+})
- ✓ Ligand1 (e.g. ATP)
- ✓ Ligand2 (e.g. EGTA)

```
Name        Total (M)      Free (M)                 -log₁₀[free]
------------------------------------------------------------------------
Ca²⁺        0.001          3.146713291428691e-07    6.50214282544484
Mg²⁺        0.002          0.0009995200393194147    3.0002084943136484
ATP         0.001          5.041517476927017e-05    4.2974387229467625
EGTA        0.002          0.0009492742611731343    3.0226082945210893
```

Fig. 4 PyChelator Colab GUI. **A** To run a calculation in PyChelator Colab, the user has to run the first code cell "Constants" which enables the dropdown for selection/entry of constants, followed by saving the options through clicking "Save Values". The user can download the constants from "Files" in the left panel. **B** Next, the user has to run the code in the following “Calculate Metal-chelator” section, to input values related to the buffer and to select the output parameters to be included in the report. A field for decimal precision is included to specify the level of precision in decimal places, employing arbitrary precision provided by the Decimals module of Python

Table 1 Comparison of PyChelator to other software

	Programming language	Constants	Input	Output	Refs
PyChelator (2024)	JavaScript, Python	NIST, Schoenmakers, Fabiato, Chelator, User-defined entry	M, mM, μM, nM	Choose parameters, Append, Excel	This work
MaxChelator (2010)	JavaScript	NIST, Schoenmakers, Editable constants file	M	One at a time	[6]
Calcium (1993)	N/A	Martell and Smith, Editable constants	mM	One at a time	[5]
Chelator (1992)	Turbo Pascal	Schoenmakers, Editable constants file	M	One at a time	[3]
SPECS (1988)	Fortran	Fabiato and Fabiato, Editable using WordStar	M	One at a time	[2]

calculations. Thus, it is always recommended to measure the free metal ions whenever possible [5, 6, 14]. A general comparison of the features offered by PyChelator compared to other calculator software is given in Table 1.

Limitations

PyChelator validates the input values for temperature (0–40 °C), ionic equivalence (0–500 mM), and pH (0–14). No validation could be introduced to the entry of parameters for the manual entry of constants. PyChelator was prepared to facilitate improvements in these calculations. Future versions may incorporate features such as the possibility to use it for multiple metal-chelator calculations, generation of buffer series with a given composition, calculation of the composition of all complexes in the solution, and downloadable stability constants calculations under different buffer conditions.

Conclusions

The PyChelator delivers a significant improvement over currently available web applications by offering a user-friendly metal chelator calculator that allows, among others, selection out of four sets of stability constants, input in smaller units, the option to specify user-defined constants, and the convenient ability to download results as a single Excel file. The Python-based PyChelator Colab offers easy customization in the modern programming language with user-defined precision in decimal calculations. Overall, this study not only expands accessibility to metal-chelator calculations but also paves the way for further advancements.

Abbreviations

NIST	National institute of standards and technology
EGTA	Ethylene glycol bis(b-aminoethylether)-N,N,N0,N0-tetraacetic acid
EDTA	Ethylenediaminetetraacetic acid
ATP	Adenosine triphosphate

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-024-05858-8.

Supplementary Material 1.

Acknowledgements
We thank Chris Patton for his kind support and feedback during the preparation of the software. We thank Ante Radocaj for his critical comments on the manuscript.

Author contributions
ES and MA conceived the project. ES and EK developed the software. ES, EK and MA wrote and edited the manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL. This work was partly supported by a grant from Deutsche Forschungsgemeinschaft (DFG) to MA (AM/507/2-1).

Availability of data and materials
Project name: PyChelator. Project home page: https://amrutelab.github.io/PyChelator/. Source code: https://github.com/AmruteLab/PyChelator. Archived version: https://doi.org/10.5281/zenodo.10674753. Operating system(s): Platform independent. Programming language: Python, JavaScript. Other requirements: None. License: GNU GPL v2. Any restrictions to use by non-academics: E.g., license needed.

Declarations

Ethics approval and consent to participate
Not Applicable.

Consent for publication
Not applicable.

Competing interests
Authors declare no competing interests.

Received: 18 April 2024 Accepted: 9 July 2024
Published online: 16 July 2024

References
1. Morgan GT, Drew HDK. CLXII.—Researches on residual affinity and co-ordination. Part II. Acetylacetones of selenium and tellurium. J Chem Soc Trans. 1920;117:1456–65.
2. Fabiato A. [31] Computer programs for calculating total from specified free or free from specified total ionic concentrations in aqueous solutions containing multiple metals and ligands. In: Methods in enzymology. Academic press; 1988. pp. 378–417.
3. Schoenmakers TJM, Visser GJ, Flik G, Theuvenet APR. CHELATOR: an improved method for computing metal ion concentrations in physiological solutions; 1992.
4. Brooks SPJ, Storey KB. Bound and determined: a computer program for making buffers of defined ion concentrations. Anal Biochem. 1992;201:119–26.
5. Föhr KJ, Warchol W, Gratzl M. Calculation and control of free divalent cations in solutions used for membrane fusion studies. In: Methods in enzymology. Academic Press; 1993. pp. 149–57.
6. Bers DM, Patton CW, Nuccitelli R. A practical guide to the preparation of Ca($^{2+}$) buffers. Methods Cell Biol. 2010;99:1–26.
7. JavaScript Numbers. https://www.w3schools.com/js/js_numbers.asp. Accessed 28 Feb 2024.
8. The Floating-Point Guide—Floating-point cheat sheet for JavaScript. https://floating-point-gui.de/languages/javascript/. Accessed 12 Mar 2024.
9. Ekmekci B, McAnany CE, Mura C. An introduction to programming for bioscientists: a python-based primer. PLOS Comput Biol. 2016;12:e1004867.
10. Decimal—decimal fixed point and floating point arithmetic. Python documentation. https://docs.python.org/3/library/decimal.html. Accessed 28 Feb 2024.
11. Smith RM, Martell A, Motekaitis R, Smith R, Motekaitis R. NIST critically selected stability constants of metal complexes database; 2004.
12. Smith GL, Miller DJ. Potentiometric measurements of stoichiometric and apparent affinity constants of EGTA for protons and divalent ions including calcium. Biochim Biophys Acta BBA Gen Subj. 1985;839(3):287–99.
13. McGuigan JAS, Kay JW, Elder HY. Ionized concentrations in Ca^{2+} and Mg^{2+} buffers must be measured, not calculated. Exp Physiol. 2020;105:427–37.
14. McGuigan JAS, Kay JW, Elder HY. Ionised concentrations in calcium and magnesium buffers: standards and precise measurement are mandatory. Progress Biophys Mol Biol. 2017;126:48–64.
15. Fujishiro N, Kawata H. A program for calculating the total concentrations of ligands and metals at any temperature, ionic strength and pH for solutions with a controlled metal concentration. Comput Biol Med. 1995;25:61–80.
16. Fabiato A, Fabiato F. Calculator programs for computing the composition of the solutions containing multiple metals and ligands used for experiments in skinned muscle cells. J Physiol. 1979;75:463–505.