# Python for Chemistry

*An introduction to Python algorithms, Simulations, and Programing for Chemistry*

**Dr. M. Kanagasabapathy**

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

**UK | UAE | INDIA | SINGAPORE**

ISBN 978-93-55517-975

www.bpbonline.com

# About the Author

**Dr. M. Kanagasabapathy** is working as the Assistant Professor, Department of Chemistry, Rajapalayam Rajus' College, affiliated to Madurai Kamaraj University, Rajapalayam, Tamil Nadu, India. He pursued his Ph.D. at Central Electrochemical Research Institute, Council for Scientific & Industrial Research, New Delhi, India. His fields of research interests are fabrication of novel electrode materials for supercapacitors, batteries, electrochemical biosensors via electrochemical deposition techniques and crystallographic data analysis by powder X-ray diffraction. He is pursuing electrochemical research works / funded research projects in collaboration with National and International Research Institutes and Universities. He received funds from UGC and MSME, India for industrial electrochemical research project works. He has about 28 years of teaching experience in both chemistry as well as in chemical engineering disciplines. To date, he has published 27 research papers, in peer-reviewed research journals and designed 23 computer simulation programs coded in Python, MATLAB, Visual Studio and wxMaxima for electrochemical modeling, X-ray diffraction crystal data simulation as well as extraction of graphical data. He has published 5 books in the theme like electrochemistry of rechargeable batteries, electrochemical supercapacitors, simulating cyclic voltammograms and symbolic computations with wxMaxima. He is a peer-reviewer for many high-impact electrochemical research journals. He is also serving as the Technical Consultant for Energe Capacitors Pvt. Ltd., Rajapalayam (TN) India to design the EDL supercapacitor electrodes. His computer simulation programs were published in research journals and indexed in renowned software publishers and international Universities' web databases.

## About the Reviewer

**Sivakumar V.** is working as an Assistant Professor at the College of Rajapalayam Rajus' College for two years. He received his Ph.D. from Manonmaniam Sundaranar University. He is proficient in the areas of computer networking, cloud computing, and Python programming. He has twelve years of teaching experience at the National Engineering College, Kovilpatti, and four years of experience at the PSR Engineering College, Sivakasi. He worked on many research projects developed in Python programming. He and his teammates have developed an R&D internship project for a US ECRIO company.

# Acknowledgement

I  express my heartfelt gratitude to **Dr. S. Singaraj**, Secretary, College Governing Council, Rajapalayam Rajus' College, and **Dr. D. Venkateswaran**, Principal, Rajapalayam Rajus' College for granting their consents and for their constant encouragements and motivations to publish my work.

I am grateful to the peer-reviewers, **Dr. Alfred Noel** and **Dr. V. Sivakumar** for their suggestions, which enhances the publishing standards of the book.

My earnest appreciations to the entire BPB Publications team for formatting my work into a reputed book.

# Preface

Python is a versatile and powerful computer language without a steep learning curve. It can be deployed to simulate various physico-chemical parameters or to analyze complex molecular, bio-molecular, crystalline structures. It can be used as a graphical tool or as a proficient calculator for numerical as well as for the symbolic computations in the interdisciplinary fields of chemistry.

The objective of this book is to give a gentle introduction to the Python programing with relevant algorithms, iterations and basic simulations in terms of a Chemist's perspective. This book outlines the fundamentals of Python coding through the built-in functions, libraries, modules as well as with few selected external packages for physical / materials / inorganic / analytical / organic / nuclear chemistry in terms of numerical, symbolic, structural and graphical data analysis using the default, Integrated Development and Learning Environment. Outlines on the structural elucidation of organic molecules and inorganic complexes with specific cheminformatics modules are also included.

Chemical data analyzes with Numpy is given with illustrations. Similarly, SymPy for symbolic computations with CAS is included. Plotting tools with Matplotlib is explained with appropriate examples. Algorithm blended with coding, based on real-time calculations, simulations and iterations in diversified and fundamental topics of chemistry is presented in a lucid style.

*Gist of chapterwise contents are given below:*

**Chapter 1: Understanding Python Functions for Chemistry-** This chapter covers the basic Python functions with reference to deploying dictionaries to fetch chemical data, estimating atomic percentage from a molecular formula, reading and writing .csv files, and determination of thermodynamic, photochemical as well as chemical kinetics parameters. It also covers the error handlers with reference to electron transfer redox reactions. Algorithm to record the rate of the reaction with timer function is discussed. Deploying loops and operators in the estimation of various chemical parameters or for simulations are also explained with examples. By using the math module, algorithm for pH metric titrations, determination of energy of activation and spin coupling for NMR spectral data can also be discoursed.

**Chapter 2: Computations in Chemistry with NumPy-** This chapter encompasses essential NumPy functions for numerical analysis and array functions in a chemist's perspective. It includes the key functions of numerical python for the computation of entropy, distribution coefficient and association factor for phenol. Algorithms for balancing the chemical equation by matrix based row echelon form, predicting the concentration for equilibrium reactions, Lagrange & polynomial interpolation for the viscosity of glycerol and fetching the data for amino acids from .csv files and other essential tools for numerical analysis are also discussed.

**Chapter 3: Interpolation, Physico-chemical constants, and Units with SciPy -** This chapter encloses the fundamental functions of Scientific Python with reference to built-in physical & chemical constants, interconversion of scientific units and integral calculus functions using quad & Romberg methods. Programs to predict the number of H atoms from the intensity of NMR spectral data, cubic spline interpolation to predict the viscosity of glycerol, II order reaction kinetics from system of linear equations and curve fitting techniques are also discussed. Algorithm for matrix based, balancing the combustion chemical equations is elucidated. Important statistical functions are enlisted.

**Chapter 4: SymPy for Symbolic Computations in Chemistry-** This chapter outlines the significant functions of SymPy for symbolic computations for chemistry. It covers higher order derivatives, definite integrals and solving linear as well as non-linear equations. Programs include rate of a formation of acetic acid by fermentation, estimation of coulombic charge in an electrochemical cell and determination of the stoichiometric coefficients via matrices. It covers the estimation of concentration of components in equilibrium reactions based on the roots of quadratic equation. Matrix operations and binomial functions are also covered in this chapter.

**Chapter 5: Interactive plotting of physico-chemical data with Matplotlib-** This chapter focusses the essence of graph plotting functions and it encloses GUI tools for plots as well as subplots. Optimizing the plot style in terms of font, legend, marker, tick marks and essential plotting functions for bar & pie charts based on thermodynamic and electrochemical parameters are given.

**Chapter 6: Introduction to Cheminformatics with RDKit-** This chapter gives a gentle introduction to RDKit, the cheminformatics package. It briefs about the vital structural aspects of chemical compounds, fetching the molecular structures from SMILES data and from .mol file, interconversion of SMILES to .mol formats and drawing / exporting the molecular structures. Coding to fetch the number of atoms, bond nature, ring size and structural data from Structural Data File (.sdf) from chemical suppliers of a molecule are included. Drawing the stereochemical notation of molecules is also included.

**Chapter 7: ChemFormula for atomic and molecular data-** This chapter covers the important functions of the ChemFormula package. It mainly focuses on printing molecular formula with hill, unicode, LaTex and .html formats. Functionalities such as checking the radioactivity, charge of a molecule and determination of atomic mass percent from the molecular formula are also covered.

**Chapter 8: Chemlib for physico-chemical parameters-** This chapter outlines the essential built-in functions and data bases of the package chemlib. It includes fetching elemental data such as atomic mass, atomic radius, electronegativity, ionization potential, specific heat, isotopes and the like. Determination of molar mass, atomic percentage, empirical formula and coding to determine stoichiometric coefficients, limiting reagent, pH, pOH and molality are also discussed. Despite that, determination of electrochemical parameters such as electrode potential, cathodic current efficiency is enclosed. Codes to compute the frequency and wavelength of the electromagnetic radiations and energy of an electron in Bohr orbital is also summarized.

**Chapter 9: ChemPy for computations in chemistry-** This chapter summarizes the indispensable functions of ChemPy package. It has many built-in functions for specific parameters of physical chemistry and it covers the molar mass calculations, stochiometric mole fraction data for reactants and products. Display functions for LaTex, unicode and .html formats for chemical reactions are given. It covers balancing the chemical equations and reactions in ionic equilibria. Algorithms for the estimation of reaction rates, activation energy and Arrhenius factor are also summarized.

**Chapter 10: Mendeleev package for atomic and ionic data-** This chapter outlines the important functions associated with Mendeleev package, which mainly contains database for various atomic parameters of elements such as atomic radius, phase transitions, lattice constant, molar heat capacity, electron affinity, electronegativity, dipole polarizability, oxidation states and the like. Codes to fetch the data for possible ionization energies of elements, ionic & crystal radii, radio isotopic parameters such as mass number, half-life period, g-factor, quadrupole moment, spin and the like are included. Algorithms to estimate of effective nuclear charge based on slater's rule and electronegativity data in Pauling, Allred-Rochow and Mulliken scales are also discussed.

**Chapter 11: Computations of parameters of electrolytes with pyEQL-** This chapter gives a gentle introduction to key functions of pyEQL package, which is deployed for the estimation of various parameters related to electrolytes. It gives an overview of density of the electrolyte solutions at different temperatures and concentrations. Estimation of specific conductance, ionic concentration, ionic strength, activity coefficients and diffusion coefficients are given. The algorithm for the ionic conductance simulation by incorporating electrolytes is also discussed. Despite these, determination of transport number, osmotic pressure, kinematic and dynamic viscosities of electrolytes are summarized. Codes to fetch the ionic mobilities and dielectric constants data for electrolytes are also included.

**Chapter 12: stk module for molecular structures-** This chapter summarizes the key functions of the stk module, which is a Python library to design or to manipulate and to view the 3-dimensional complex molecular structures. It covers drawing molecular structures with specific functional groups from SMILES and .mol followed by exporting the structures. Codes to construct the polymeric reaction from monomers, cage structures, covalent organic molecular framework and topology graph for metal-ligand complexes are given.

## Code Bundle and Coloured Images

Please follow the link to download the
***Code Bundle*** and the ***Coloured Images*** of the book:

**https://rebrand.ly/jejzdz8**

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Python-for-Chemistry**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

## Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

**errata@bpbonline.com**

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

## Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

**https://discord.bpbonline.com**

# Table of Contents

CHAPTER 1

# Understanding Python Functions for Chemistry

## Introduction

Python is the most preferred language for scientific computing since it has not a steep learning curve. It was developed by Guido van Rossum, in 1991. It can be deployed from simple numerical computing or data analysis to complex symbolic computations along with 2D, 3D graphical representations. It can also be used for web-based applications, and it can be installed in Windows, Linux and Mac operating systems. Syntax of the Python is easily understandable. For example, the following syntax shows, addition of two numbers and the syntax used is like the plain English.

```
b = 2; x = 3

print("Sum of 'b' and 'x' is", (b+x))

>>>

Sum of 'b' and 'x' is 5
```

Though python language has many built-in functions for scientific computations, this chapter outlines the basic Python functions for computing chemical data such as deploying dictionaries to fetch the atomic mass and atomic number of the chemical elements, estimating atomic percentage from the molecular formula, reading and writing .csv files, computation of thermodynamic, photochemical and chemical kinetics parameters. This chapter also covers the error handlers with reference to electron transfer redox reactions. Algorithm to record the rate of the reaction with timer function is also discussed. Deploying loops and operators in the estimation of various chemical parameters or for simulations are also explained with examples. By using the math module, algorithm for pH metric titrations, determination of energy of activation and spin coupling for NMR spectral data can also be discoursed.

## Structure

- Dictionary for atomic numbers and atomic masses
- Adding elements to the dictionary
- Updating elements in the dictionary
- Deleting elements from the dictionary
- Atomic mass percentage from molecular formula
- Data module for physical and chemical constants
- Molar gas constant from Boltzmann's constant
- Estimation of volume of an ideal gas
- Quantum efficiency of photochemical reactions
- Fetching Rf data of amino acids from .csv file
- Fetching selective data for amino acids from .csv file
- Converting 'amino_acids.csv' to dictionary
- Estimation of rate constant with data as a list
- Exporting rate constant data to .csv
- Math module
- Power of 10 (e)
- pH metric acid-base titration
- R.M.S and average velocity of ideal gas molecules
- Rate constant from activation energy
- Calculating sine (angle) from radians
- Estimating bond length from the bond angle
- Priority for arithmetical operators
- Quotient and Modulo operators
- Assignment operators
- Comparison operators

- Logical operators
- Identity operators
- Membership operators
- `cmath` module
- Scrutinizing user input data
- Tackling of errors in user inputs
- Number of electrons transferred in a redox reaction
- Conditions and Loops
- if ... elif ... else statements
- Error handling with if ... else loops
- Nested loops
- while loops
- for loops
- range() function
- Fetching selective rows of amino_acid.csv file
- timer() function
- Recording concentration with time (reaction rate)
- Recursion
- Predicting spin–spin coupling in NMR spectra
- Lambda function

## Dictionary for atomic numbers and atomic masses

Dictionary is a collection of data (str, int, float formats) and is used to store data as key : values pairs within curled braces, { }. It is ordered, changeable and do not allow duplicates. Based on the key, data values can be fetched. But, in a dictionary new key : values can be added or existing key : values can be updated and can be deleted. Dictionaries can be created for any types of data sets and can be deployed for fetching the required values on program execution. This program demonstrates the creation of a data dictionary for atomic number and atomic masses of the chemical elements based on their symbols as keys.

```
Syntax: dictionary_name = {"key": [value1, value2]}
```

Following code demonstrates the creation of a dictionary for chemical elements.

```
# dictionary_name = {"symbol": [atomic number, atomic mass]}
atomic_number_mass = {
"H" : [1, 1.007],   # "H" is key and [1, 1.007] are values.
"He" : [2, 4.003],
"Li" : [3, 6.941],
"Be" : [4, 9.012],
"B" : [5, 10.812],
"C" : [6, 12.011],
"N" : [7, 14.007],
"O" : [8, 15.999],
"F" : [9, 18.998],
"Ne" : [10, 20.18],
"Na" : [11, 22.99],
"Mg" : [12, 24.305],
"Al" : [13, 26.982],
"Si" : [14, 28.086],
"P" : [15, 30.974],
"S" : [16, 32.066],
"Cl" : [17, 35.453],
"Ar" : [18, 39.948],
"K" : [19, 39.098],
"Ca" : [20, 40.078],
"Sc" : [21, 44.956],
"Ti" : [22, 47.867],
"V" : [23, 50.942],
"Cr" : [24, 51.996],
"Mn" : [25, 54.938],
"Fe" : [26, 55.845],
"Co" : [27, 58.933],
"Ni" : [28, 58.693],
"Cu" : [29, 63.546],
"Zn" : [30, 65.382],
"Ga" : [31, 69.723],
"Ge" : [32, 72.631],
"As" : [33, 74.922],
"Se" : [34, 78.963],
"Br" : [35, 79.904],
"Kr" : [36, 83.798],
"Rb" : [37, 85.468],
"Sr" : [38, 87.621],
"Y" : [39, 88.906],
"Zr" : [40, 91.224],
"Nb" : [41, 92.906],
"Mo" : [42, 95.962],
"Tc" : [43, 98],
"Ru" : [44, 101.072],
"Rh" : [45, 102.906],
"Pd" : [46, 106.421],
"Ag" : [47, 107.868],
"Cd" : [48, 112.412],
"In" : [49, 114.818],
"Sn" : [50, 118.711],
"Sb" : [51, 121.76],
"Te" : [52, 127.603],
"I" : [53, 126.904],
"Xe" : [54, 131.294],
"Cs" : [55, 132.905],
"Ba" : [56, 137.328],
"La" : [57, 138.905],
"Ce" : [58, 140.116],
"Pr" : [59, 140.908],
"Nd" : [60, 144.242],
"Pm" : [61, 145],
"Sm" : [62, 150.362],
"Eu" : [63, 151.964],
"Gd" : [64, 157.253],
"Tb" : [65, 158.925],
"Dy" : [66, 162.5],
"Ho" : [67, 164.93],
"Er" : [68, 167.259],
"Tm" : [69, 168.934],
"Yb" : [70, 173.055],
"Lu" : [71, 174.967],
"Hf" : [72, 178.492],
"Ta" : [73, 180.948],
"W" : [74, 183.841],
"Re" : [75, 186.207],
"Os" : [76, 190.233],
"Ir" : [77, 192.217],
"Pt" : [78, 195.085],
"Au" : [79, 196.967],
"Hg" : [80, 200.592],
"Tl" : [81, 204.383],
"Pb" : [82, 207.21],
"Bi" : [83, 208.98],
"Po" : [84, 209],
"At" : [85, 210],
"Rn" : [86, 222],
"Fr" : [87, 223],
"Ra" : [88, 226],
"Ac" : [89, 227],
"Th" : [90, 232.038],
"Pa" : [91, 231.036],
"U" : [92, 238.029],
"Np" : [93, 237],
"Pu" : [94, 244],
"Am" : [95, 243],
"Cm" : [96, 247],
"Bk" : [97, 247],
"Cf" : [98, 251],
"Es" : [99, 252],
"Fm" : [100, 257],
"Md" : [101, 258],
"No" : [102, 259],
"Lr" : [103, 266],
"Rf" : [104, 267],
"Db" : [105, 268],
"Sg" : [106, 269],
"Bh" : [107, 270],
"Hs" : [108, 277],
"Mt" : [109, 278],
"Ds" : [110, 281],
"Rg" : [111, 282],
"Cn" : [112, 285],
"Nh" : [113, 286],
"Fl" : [114, 289],
"Mc" : [115, 290],
"Lv" : [116, 293],
"Ts" : [117, 294],
"Og" : [118, 294]
}
x = input ("Enter symbol: ")     #user input for symbol (key)
print("Atomic number for ", x, "is: ")
print(atomic_number_mass.get(x)[0]) #index[0] atomic number
print("Atomic mass for ", x, "is: ")
print(atomic_number_mass.get(x)[1]) #index[1] atomic mass
>>>
Enter symbol: F
Atomic number for F is:
9
Atomic mass for F is:
18.998
```

From the user input for the given key value (as **'x'** for symbol of elements), respective atomic number as well as atomic mass can be fetched. It must be noted that the index values for this dictionary, (**atomic_number_mass**) is having only two values [**0**] for the first item of the list, which is atomic number and [**1**] for the second item of the list, which is atomic mass. Such dictionaries are used to create modules of data sets of various chemical parameters. It must be noted that if any duplicate key is added, it will be removed.

## Adding elements to the dictionary

New data can be added, and existing data can be updated or deleted in the dictionary via key : values.

Adding new elements (keys) into the existing dictionary.

```
atomic_number_mass = {
"H" : [1, 1.007],
"He" : [2, 4.003],
"Li" : [3, 6.941],
"Be" : [4, 9.012],
"B" : [5, 10.812],
"C" : [6, 12.011],
"N" : [7, 14.007]                # intentionally limited to "N"
}
print(len(atomic_number_mass))   # len function to know the number of elements
atomic_number_mass["O"] = [8, 15.999] # new key : values
print(atomic_number_mass) # display new dictionary
print(len(atomic_number_mass))
>>>
7
{'H': [1, 1.007], 'He': [2, 4.003], 'Li': [3, 6.941], 'Be': [4, 9.012],
'B': [5, 10.812], 'C': [6, 12.011], 'N': [7, 14.007], 'O': [8, 15.999]}
8
```

## Updating elements in the dictionary

Modifying  the keys and their values in the existing keys of the dictionary and their values is depicted as following:

```
atomic_number_mass = {
"H" : [1, 1.007],
"He" : [2, 4.003],
"Li" : [3, 6.941],
"Be" : [4, 9.012],
"B" : [5, 10.812],
"C" : [6, 12.011],
"N" : [7, 14.007]                      # intentionally limited to "N"
}
atomic_number_mass["C"] = [6, 13.013]  # updating the key "C" values.
print(atomic_number_mass)
>>>
{‘H’: [1, 1.007], ‘He’: [2, 4.003], ‘Li’: [3, 6.941], ‘Be’: [4, 9.012],
‘B’: [5, 10.812], ‘C’: [6, 13.013], ‘N’: [7, 14.007]}
```

## Deleting elements from the dictionary

With **`del name_of_dictionary[“key”]`** the given key and its values can be deleted.

```
atomic_number_mass = {
"H" : [1, 1.007],
"He" : [2, 4.003],
"Li" : [3, 6.941],
"Be" : [4, 9.012],
"B" : [5, 10.812],
"C" : [6, 12.011],
"N" : [7, 14.007]                      # intentionally limited to 'N'
}
del atomic_number_mass[“N”]            # removes 'N' and its values
print(atomic_number_mass)
>>>
{'H': [1, 1.007], 'He': [2, 4.003], 'Li': [3, 6.941], 'Be': [4, 9.012],
'B': [5, 10.812], 'C': [6, 12.011]}
```

Deleting a key can also be executed by pop command as: **`atomic_number_mass.pop ("N")`** also gives the same output. Using the keyword, `del` the dictionary can be deleted.

```
del atomic_number_mass    # This deletes the dictionary
```

## Atomic mass percentage from molecular formula

From a dictionary, key: value data sets of string, float, integer, Boolean for various chemical parameters can be fetched for real-time calculations. Hence the designed dictionary can be deployed for real-time applications.

Based on the dictionary in program # 1, it is possible to calculate various basic data such as molar mass or number of moles of compounds.

This program demonstrates the calculation of molar mass and atomic mass percentage of individual elements of the given compound based on the molecular formula as user input.

```
# Creating a dictionary for each element with atomic numbers and atomic
masses.

# With RegEx module, segregating atoms and their counts  for the given
compound.

# Splitting of atoms as string from the user input of molecular formula
via Uppercase.

# Counting of individual atoms followed by multiplication with its
atomic mass fetched from dictionary.

# Interconversion of float and string followed by summation of the
individual atomic masses.

# Loop to calculate atomic mass and atomic mass % of atoms.
```

```
loop_value = 0                        # for infinite loop
while loop_value == 0:                # indentation for loop
    atomic_number_mass = {            # dictionary
    "H" : [1, 1.007],
    "He" : [2, 4.003],
    "Li" : [3, 6.941],
    "Be" : [4, 9.012],
    "B" : [5, 10.812],
    "C" : [6, 12.011],
    "N" : [7, 1.007],
```

# Dictionary of elements with atomic number and the relevant atomic mass:

```
    "O" : [8, 15.999],
    "F" : [9, 18.998],
    "Ne" : [10, 20.18],
    "Na" : [11, 22.99],
    "Mg" : [12, 24.305],
    "Al" : [13, 26.982],
    "Si" : [14, 28.086],
    "P" : [15, 30.974],
    "S" : [16, 32.066],
    "Cl" : [17, 35.453],
    "Ar" : [18, 39.948],
    "K" : [19, 39.098],
    "Ca" : [20, 40.078],
    "Sc" : [21, 44.956],
    "Ti" : [22, 47.867],
    "V" : [23, 50.942],
    "Cr" : [24, 51.996],
    "Mn" : [25, 54.938],
    "Fe" : [26, 55.845],
    "Co" : [27, 58.933],
    "Ni" : [28, 58.693],
    "Cu" : [29, 63.546],
    "Zn" : [30, 65.382],
    "Ga" : [31, 69.723],
    "Ge" : [32, 72.631],
    "As" : [33, 74.922],
    "Se" : [34, 78.963],
    "Br" : [35, 79.904],
    "Kr" : [36, 83.798],
    "Rb" : [37, 85.468],
    "Sr" : [38, 87.621],
    "Y" : [39, 88.906],
    "Zr" : [40, 91.224],
    "Nb" : [41, 92.906],
    "Mo" : [42, 95.962],
    "Tc" : [43, 98],
    "Ru" : [44, 101.072],
    "Rh" : [45, 102.906],
    "Pd" : [46, 106.421],
    "Ag" : [47, 107.868],
    "Cd" : [48, 112.412],
    "In" : [49, 114.818],
    "Sn" : [50, 118.711],
    "Sb" : [51, 121.76],
    "Te" : [52, 127.603],
    "I" : [53, 126.904],
    "Xe" : [54, 131.294],
    "Cs" : [55, 132.905],
    "Ba" : [56, 137.328],
    "La" : [57, 138.905],
    "Ce" : [58, 140.116],
    "Pr" : [59, 140.908],
    "Nd" : [60, 144.242],
    "Pm" : [61, 145],
    "Sm" : [62, 150.362],
    "Eu" : [63, 151.964],
    "Gd" : [64, 157.253],
    "Tb" : [65, 158.925],
    "Dy" : [66, 162.5],
    "Ho" : [67, 164.93],
    "Er" : [68, 167.259],
    "Tm" : [69, 168.934],
    "Yb" : [70, 173.055],
    "Lu" : [71, 174.967],
    "Hf" : [72, 178.492],
    "Ta" : [73, 180.948],
    "W" : [74, 183.841],
    "Re" : [75, 186.207],
    "Os" : [76, 190.233],
    "Ir" : [77, 192.217],
    "Pt" : [78, 195.085],
    "Au" : [79, 196.967],
    "Hg" : [80, 200.592],
    "Tl" : [81, 204.383],
    "Pb" : [82, 207.21],
    "Bi" : [83, 208.98],
    "Po" : [84, 209],
    "At" : [85, 210],
    "Rn" : [86, 222],
    "Fr" : [87, 223],
    "Ra" : [88, 226],
    "Ac" : [89, 227],
    "Th" : [90, 232.038],
    "Pa" : [91, 231.036],
    "U" : [92, 238.029],
    "Np" : [93, 237],
    "Pu" : [94, 244],
    "Am" : [95, 243],
    "Cm" : [96, 247],
    "Bk" : [97, 247],
    "Cf" : [98, 251],
    "Es" : [99, 252],
    "Fm" : [100, 257],
    "Md" : [101, 258],
    "No" : [102, 259],
    "Lr" : [103, 266],
    "Rf" : [104, 267],
    "Db" : [105, 268],
    "Sg" : [106, 269],
    "Bh" : [107, 270],
    "Hs" : [108, 277],
    "Mt" : [109, 278],
    "Ds" : [110, 281],
    "Rg" : [111, 282],
    "Cn" : [112, 285],
    "Nh" : [113, 286],
    "Fl" : [114, 289],
    "Mc" : [115, 290],
    "Lv" : [116, 293],
    "Ts" : [117, 294],
    "Og" : [118, 294]
    }

    x = input("\nEnter molecular formula:  ")
                          # User input - molecular formula of the compound
    import re             # RegEx module
    y = re.findall('[a-zA-Z][^A-Z]*', x)
    # This separates the molecular formula by Uppercase.
    # It leads to create key values.
    z = [re.split(r'(\d+)', s)[0:2] for s in (y)]
    # Lists for each atom as str and its count in float.
    # Number of each element is counted.
    n = 0   # for scrutinizing each atom one by one
    molar_mass = 0
    while len(z) > n:
        for formula in (z[n]):
            a = formula.split(',')     # conversion to list
            b = " ".join(str(x) for x in a)
            try:
                c = float(b)
            except ValueError:   # Error handler str to float
                atom = str(b)
                c = 1
                d = atomic_number_mass.get(atom)[1]
        molar_mass = molar_mass + (c*d)
        n = n + 1   # go to next atom
        c = 1; d = 0
    print(“Molar mass of ", x, "is ", round(molar_mass,3))
    n = 0
```

```
    while len(z) > n:
        for formula in (z[n]):
            a = formula.split(',')     # conversion to list
            b = " ".join(str(x) for x in a)
            try:
                c = float(b)
            except ValueError:   # Error handler str to float
                atom = str(b)
                c = 1
                d = atomic_number_mass.get(atom)[1]
        elemental_mass =(c*d)
        elemental_mass = (elemental_mass*100/molar_mass)
        print("Atomic mass % of ", atom, "is ", round(elemental_mass,3))
        n = n + 1
        c = 1; d = 0
>>>
Enter molecular formula:    C8H9NO2                  # $HOC_6H_4NHCOCH_3$
                                                 # acetaminophen
Molar mass of  C8H9NO2 is  151.156
Atomic mass % of  C is  63.569
Atomic mass % of  H is  5.996
Atomic mass % of  N is  9.267
Atomic mass % of  O is  21.169

>>>
Enter molecular formula:  Mo2Ti2C3           # $Mo_2Ti_2C_3$
Molar mass of  Mo2Ti2C3 is  323.691
Atomic mass % of  Mo is  59.292
Atomic mass % of  Ti is  29.576
Atomic mass % of  C is  11.132

>>>
Enter molecular formula:  K4FeC6N6H6O3        # $K_4Fe(CN)_6.3H_2O$
Molar mass of  K4FeC6N6H6O3 is  422.384
Atomic mass % of  K is  37.026
Atomic mass % of  Fe is  13.221
Atomic mass % of  C is  17.062
Atomic mass % of  N is  19.897
Atomic mass % of  H is  1.43
Atomic mass % of  O is  11.363

>>>
Enter molecular formula:  C10H18N2Na2O10      # Na2EDTA.2H2O
Molar mass of  C10H18N2Na2O10 is  372.22
Atomic mass % of  C is  32.269
Atomic mass % of  H is  4.87
Atomic mass % of  N is  7.526
Atomic mass % of  Na is  12.353
Atomic mass % of  O is  42.983

>>>
Enter molecular formula:  C17H18FN3O3        # Ciprofloxacin
Molar mass of  C17H18FN3O3 is  331.329
Atomic mass % of  C is  61.627
Atomic mass % of  H is  5.471
Atomic mass % of  F is  5.734
Atomic mass % of  N is  12.683
Atomic mass % of  O is  14.486
Enter molecular formula:  CoMn2CdSe3O4H6S21

>>>                                          # CoMn2CdSe3O4H6S21
Molar mass of  CoMn2CdSe3O4H6S21 is  1261.534
Atomic mass % of  Co is  4.672
Atomic mass % of  Mn is  8.71
Atomic mass % of  Cd is  8.911
Atomic mass % of  Se is  18.778
Atomic mass % of  O is  5.073
Atomic mass % of  H is  0.479
Atomic mass % of  S is  53.378
```

This demonstrates, fetching of the relevant data from arrays for further computations. This type of dictionaries can be used to build larger data sets for data analyses.