RJC

RASĀYAN *J. Chem.*
Vol. 17 | No. 4 |1460-1472| October - December | 2024
ISSN: 0974-1496 | e-ISSN: 0976-0083 | CODEN: RJCABP
http://www.rasayanjournal.com
http://www.rasayanjournal.co.in

PYTHON PROGRAM FOR STRUCTURE AND MOLECULAR FORMULA OF ORGANIC COMPOUNDS: AN ALTERNATIVE THEORETICAL APPROACH

Hemant Verma[1], Jyoti Singh[2] and Sarita Passey[2,✉]
[1]Department of Chemistry, Hindu College; University of Delhi, Delhi-110007, India
[2]Department of Chemistry, Zakir Husain Delhi College, University of Delhi, Delhi-110002 India
✉Corresponding Author: spassey@zh.du.ac.in

ABSTRACT

A chemist always desires the automation of a chemistry laboratory as it reduces their workload by minimizing the time, energy, and effort. Automation when coupled with artificial intelligence is the need in today's era as it gives more flexibility and power to the chemist. Using computer programs for a large number of such operations would further ease out these tasks as the desired objectives may be achieved with the click of a mouse. In this research article, the Python program has been used to arrive at the empirical and molecular formulas of some compounds. Python code has also been written for some compounds and their 2D and 3D structures drawn henceforth. With the introduction of Python language at the undergraduate level, this paper would allow chemistry students to make use of Python in organic chemistry as well. As a result, the chemistry students will be able to use the Python program for verification and classroom-based structure determination of various organic compounds. The students thus gain access and become equipped with computational skills that will be helpful in the modern world and can utilize the Python code as an alternative theoretical approach for these operations. While Python provides valuable computational tools, integrating thematic chemistry and new methodologies is crucial for advancing structural analysis. This approach enhances the depth and applicability of theoretical models beyond basic calculations. This paper is a perfect amalgamation of computer programming with organic chemistry and is an attempt to equip the chemists to do the above-mentioned operations theoretically from their desks.

Keywords: Python, Theoretical Approach, Alternate Method, Empirical and Molecular Formula, 2D, and 3D Structure, Computational Tool.

RASĀYAN *J. Chem.*, Vol. 17, No.4, 2024

INTRODUCTION

Python is a general-purpose, open-source computer programming language and is seeing increasing use nowadays.[1] It is becoming very popular and its user base is growing by the day.[2] Python owes its growing popularity to its relative ease as a language and thus it is comfortably used by non-programmers, accountants, scientists, researchers, students etc. for their everyday use in organizing finances and data. This is because of its simple and readable syntax which makes it easy to learn, read, and write. It is also a portable language and thus same code can be used on different machines. Its data structure is easy to use and thus makes data management and data manipulation simple and manageable. Since it is a general-purpose and versatile language, it can be used in different domains and different fields interchangeably which helps in interdisciplinary learning. Its capabilities extend to high-resolution simulations and detailed coordinate-based analyses, making it ideal for precise scientific computations. The ease of working with Python is also enhanced by its easy code readability along with reliable reproducibility and transparency. A very big advantage of using Python is that it has a very vast collection of libraries which eliminates the need to write complete codes and thus reduces coding errors. The presence of libraries increases the efficiency and speed of developers since they can work with pre-existing codes. These libraries are used for various domains like web development, data management, data visualization, data manipulation, solving equations, numerical, and so on. Another advantage is the large open-source community of Python which includes developers, software engineers, and beginners. This works like a boon for beginners and those who want to contribute to the language thus improving and expanding it. Here too, in-depth knowledge of the subject combined with Python can create numerous benefits, bridging both fields and enhancing

capabilities in chemical analysis and computational simulations. As a result of the numerous benefits of Python, it is being used in a large number of fields.[3] Python is increasingly being used by data analysts for collecting processing and analyzing data which can be further used in industry, business, academic, or scientific arenas. Python is also progressively being used for Data visualization which is an important component of data analysis that may be done at any stage of analysis in the raw, modified, or analyzed form. The popularity of Python is understandable because of its role in website development. Though languages like JavaScript, HTML, and CSS are commonly used for the front end of a website, for the back end, Python is the most popular choice because of its easy usability, wide availability, and high adaptability amongst other already mentioned features. Though many languages are used for software development, Python is fast becoming the language of choice because it doesn't require one to use a complex syntax and thus allows programmers to develop applications quickly. In today's day and age, the whole world is moving towards artificial intelligence and machine learning. Not surprising that Python is the most used computer programming language in machine learning and it owes its popularity to the large number of libraries it has, to its easy readability since it has an easy-to-read syntax, strong community support, compatibility with many other computer languages, and scalability. Along with this the extensive visualization capability and low barrier to entry of Python make it an excellent choice for machine learning professionals to use this powerful tool for use in ML and AI purposes. Python is also particularly a favorite programming language for task automation or scripting. Automation is the buzzword in today's age and Python is a great tool to facilitate that. Python can be used to create, delete, rename, modify, merge, and split files along with other file management operations. At the workplace, Python is often used to automate the scheduling and reminders for meetings via emails and texts to increase productivity and time management at work. Apart from these, Python can also automate jobs like sending emails, scheduling tasks, searching and downloading information from the internet, completing and submitting online forms, and many other tasks which when done manually would be monotonous and boring, consume time and energy, and may induce some amount of error. In a chemistry laboratory too, automation would largely help the chemists as it would allow them to do innovative work rather than doing the repetitive and time-consuming jobs involved in running the reactions as well as doing strenuous calculations.[4] Attempts are being made to use robots to do these tedious tasks.[5,6] Artificial Intelligence coupled with automation is the key in this situation as this would open up various experimental avenues for the chemists.[7,8] Lab automation has already surrounded us in many ways from automated lab equipment like magnetic stirrers to automated flash chromatographer, liquid chromatography, mass spectrometry, the NMR autosampler carousel to automated software tasks. Though laboratory hardware and software automation has been around for many years now but challenge is using automation for doing organic synthesis or titrations or study of mechanisms, finding new reaction paths, better catalysts, and the like.[9] Python, the very popular and versatile computer programming language, may be the solution to a large number of such tasks in the chemistry laboratory in the future. It is being used for several operations in varied fields and looking at its versatility it holds immense potential to automate chemistry labs. Normally structure determination of an unknown molecule is a very long process. It involves the isolation and purification of the compound followed by determining its empirical and molecular formula. This is followed by the identification of functional groups and then followed by the determination of structure and stereochemistry. Even if the empirical and molecular formulas are known, the structure is determined by using a number of methods together such as X-ray crystallography, cryo-electron microscopy etc. Theoretically, also, the structure can be determined by quantum mechanical calculations via the Schrodinger equation and also by computational modeling.[10,11] When chemical research is combined with computational tools, a systematic framework is created that improves the workflow for structure determination overall and supplements conventional chemical processes. Once the structure is known, a chemist may use various software like ChemDraw and Chemsketch to draw these structures. While many software options are available, Python offers a unique advantage with its versatility and open-source nature. Comparative studies show that Python-based tools can integrate with experimental data effectively, providing a robust framework for structure determination and visualization. Here in this paper an alternate method to draw the 2D and 3D structures using Python program has been developed. The Python program is also used to arrive at the empirical and molecular formulas of many organic compounds without doing any calculations manually. For this certain

information about the compound needs to be provided as input as shown in Table-1. This work can be extended to a large number of organic compounds if the required information is stored in a dictionary and then the Python code is applied to obtain the empirical and molecular formula in a short span of time, thus eliminating the calculations that are involved in obtaining these formulas. For obtaining the 2D structure of a given compound, certain information about the compound needs to be stored by the program as a dictionary and the program keeps on iterating over the dictionary. The requirements are that the SMILES of organic compounds are stored in a dictionary and the name of the organic compound is provided as input. This has been shown in the Input Table-2. We have run the Python program for 15 compounds and obtained their 2D structures successfully as shown in the output Table-3. Finally, we applied the Python code to draw the 3D structure of 15 organic compounds. The information requirement for input is the same as for the 2D program i.e. the name of the organic compound and the SMILES of that compound which would be stored in the dictionary. The 3D structures obtained as output by the program are shown in Table-4. 2D and 3D structures have been drawn for simple cyclic compounds like glucose and complex ones like cholesterol along with certain acyclic compounds like lycopene and polynuclear compounds like coronene and corannulene. Structures have also been drawn for simple heterocyclic compounds like nicotine and caffeine and complex heterocyclic compounds like sulfasalazine, porphyrin ring, codeine, cocaine, etc. The aim is to build a library containing the SMILES of all organic compounds so that the 2D and 3D structure of any compound is obtained by the click of a single button.

EXPERIMENTAL

The Experimental Work is Divided into Three Sections A, B, and C with Each Section Containing the PYTHON PROGRAM SPECIFIC to its Aim

A. **Determination of the Empirical and Molecular Formula of Any Given Organic Compound** is done by writing the Python program. Python program is written in such a way that atomic masses of the elements are stored in a dictionary and the following information would be provided as input while running the program:

1. Number of elements in the compound.
2. Percentages of each element in the compound.
3. The molecular mass of the compound.

Python Program for Determining the Empirical and Molecular Formula of a Compound

```
elements = []
percentages = []
atomic_masses = {'C': 12, 'H': 1, 'O': 16, 'N': 14,
            'S': 32, 'P': 31, 'Cl': 35.5,'F': 19,
             'Br': 79.9,'I': 126.9}
n = int(input('Number of different elements: '))
order = 1
for i in range(n):
   elem = str(input(f'Element number {order}: '))
   elements.append(elem)
   proportion = float(input(f'Percentage of element {elem} in the molecule: '))
   percentages.append(proportion)
   order += 1
ratios = []
for i in range(len(elements)):
   ratio = percentages[i] / atomic_masses[elements[i]]
   ratios.append(ratio)
print("Ratios:")
for i, element in enumerate(elements):
   print(f"{element}: {ratios[i]}")
least = min(ratios)
least_integer_coefs = []
for i in ratios:
   least_int = i / least
```

```
    least_integer_coefs.append(least_int)
print("Least Integer Coefficients:")
for i, element in enumerate(elements):
    print(f"{element}: {least_integer_coefs[i]}")
integer = 1
while True:
    score = 0
    for i in least_integer_coefs:
        if (integer * i) - round(i * integer) < 0.09 and (integer * i) - round(i * integer) > -0.09:
            score += 1
    if score == len(least_integer_coefs):
        break
    integer += 1
leastintegermols = []
for i in least_integer_coefs:
    leastintegermols.append(int(integer * i))
empirical_formula = ''
for i in range(len(elements)):
    empirical_formula += elements[i]
    empirical_formula += str(leastintegermols[i])
print(f'The empirical formula of the molecule is {empirical_formula}')
at_masses = [atomic_masses[element] for element in elements]
molar_mass_emp = sum([at_masses[i] * leastintegermols[i] for i in range(len(elements))])
molar_mass_comp = float(input('Molecular mass of the compound: '))
ratio = molar_mass_comp / molar_mass_emp
Molecular_formula = ''
for i in range(len(elements)):
    Molecular_formula += elements[i]
    Molecular_formula += str(int(round(leastintegermols[i] * ratio)))
print(f'The Molecular formula of the molecule is {Molecular_formula}')
```

Table-1: Input and Output Table for Determination of Empirical and Molecular Formula

S.No.	Input Data					Output Data			
	Name of Compound	Number of elements	Percentages of elements		Molecular Mass of Compound	Ratio of elements		Empirical Formula of the Compound	Molecular Formula of the Compound
1.	Nicotine	3	C	74.00	162	C	6.16	C5H7N1	C10H14N2
			H	8.70		H	8.70		
			N	17.27		N	1.23		
2.	Caffeine	4	C	49.38	194.19	C	4.11	C4H5N2O1	C8H10N4O2
			H	5.16		H	5.16		
			N	28.76		N	2.05		
			O	16.70		O	1.04		
3.	Efavirenz	6	C	53.00	315.67	C	4.41	C58462H37857Cl4191F11906N4198O9398	C14H9Cl1F3N1O2
			H	2.86		H	2.86		
			Cl	11.24		Cl	0.31		
			F	17.09		F	0.89		
			N	4.44		N	0.31		
			O	11.36		O	0.71		
4.	Sulfasalazine	5	C	54.10	398.34	C	4.50	C2164H1680N480O600S123	C18H14N4O5S1
			H	3.54		H	3.50		
			N	14.06		N	1.00		
			O	20.05		O	1.25		
			S	8.24		S	0.25		
5.	Lycopene	2	C	89.40	536.87	C	7.45	C5H7	C40H56

			H	10.43		H	10.43		
6.	D-Glucose	3	C	40.00	180	C	3.33	C1H2O1	C6H12O6
			H	6.67		H	6.67		
			O	53.33		O	3.33		
7.	β-Lactose	3	C	42.06	342.3	C	3.50	C70H131O65	C12H22O11
			H	6.42		H	6.48		
			O	51.46		O	3.21		
8.	Coronene	2	C	96	300.35	C	8.00	C2H1	C24H12
			H	4		H	4.00		
9.	Corannulene	2	C	95.88	250.3	C	7.98	C2H1	C20H10
			H	3.99		H	3.99		
10.	Porphyrin ring	3	C	77.33	310.35	C	6.44	C10H7N2	C20H14N4
			H	4.51		H	4.51		
			N	18.04		N	1.28		
11.	Methylene Blue	5	C	60.00	319.85	C	5.00	C32H37Cl2N6S2	C16H18Cl1N3S1
			H	5.78		H	5.78		
			Cl	11.09		Cl	0.31		
			N	13.13		N	0.93		
			S	10.00		S	0.31		
12.	Thiamine (Vitamin B_1)	6	C	47.81	300.81	C	3.98	C71H102Cl6N24O6S6	C12H17Cl1N4O1S1
			H	5.70		H	5.70		
			Cl	11.79		Cl	0.33		
			N	18.63		N	1.33		
			O	5.32		O	0.33		
			S	10.66		S	0.33		
13.	Cholesterol	3	C	83.87	386.27	C	6.98	C27H46O1	C27H46O1
			H	11.91		H	11.91		
			O	4.14		O	0.25		
14.	Codeine	4	C	72	299.3	C	6.0	C2005H2367N112O338	C18H21N1O3
			H	7.08		H	7.08		
			N	4.69		N	0.33		
			O	16.22		O	1.01		
15.	Cocaine	4	C	67.12	303.35	C	5.59	C2694H3392N159O639	C17H21N1O4
			H	7.04		H	7.04		
			N	4.62		N	0.33		
			O	21.22		O	1.32		

B. Drawing 2D Structure of any Given Organic Compound

Python program is written to draw the 2D structure of any organic compound with the following specifications:

1. SMILES of various organic compounds should be stored in the dictionary.
2. The program asks for the name of the organic compound as input.
3. rdkit-pypi and matplotlib.pyplot libraries are used.

Python Program to Draw the 2D Structure of Compound

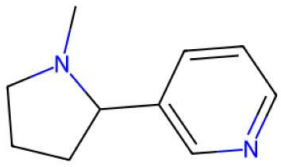
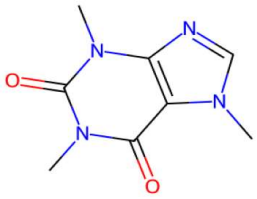
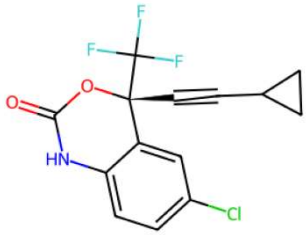
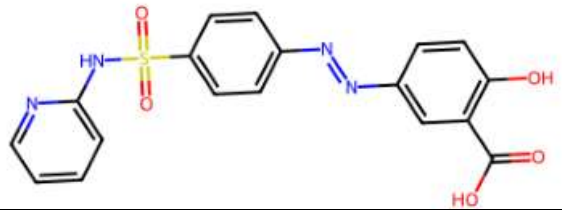
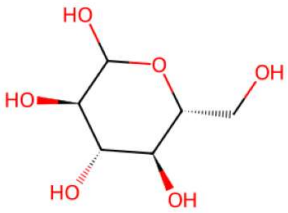
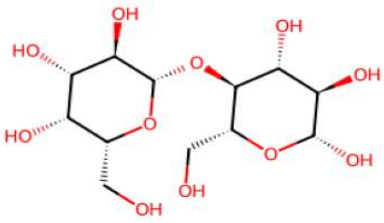
```
!pip install rdkit-pypi
from rdkit import Chem
from rdkit.Chem import Draw
import matplotlib.pyplot as plt
smiles = {'Caffeine': 'CN1C=NC2=C1C(=O)N(C(=O)N2C)C', 'Methylamine': 'CN',
'Nicotine': 'CN1CCCC1C2=CN=CC=C2',  'Lycopene':
'CC(=CCC/C(=C/C=C/C(=C/C=C/C(=C/C=C/C=C(/C=C/C=C(/C=C/C=C(/CCC=C(C)C)\C)\C)\C)/C)/C)/C)C',
    'Beta carotene':
'CC1=C(C(CCC1)(C)C)/C=C/C(=C/C=C/C(=C/C=C/C=C(/C=C/C=C(/C=C/C2=C(CCCC2(C)C)C)\C)\C)/C)/C',
    'Sulfasalazine': 'C1=CC=NC(=C1)NS(=O)(=O)C2=CC=C(C=C2)N=NC3=CC(=C(C=C3)O)C(=O)O',
```

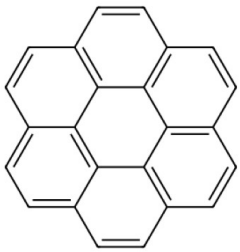
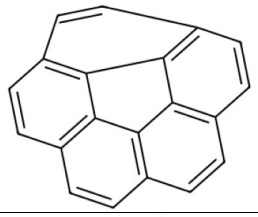
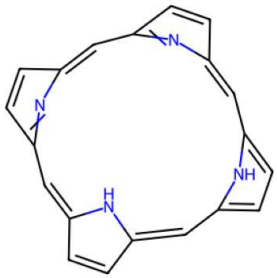
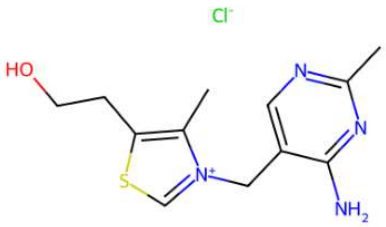
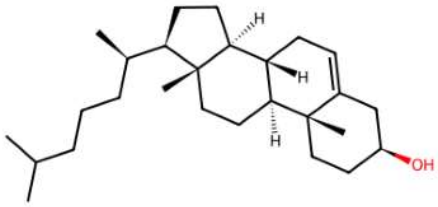
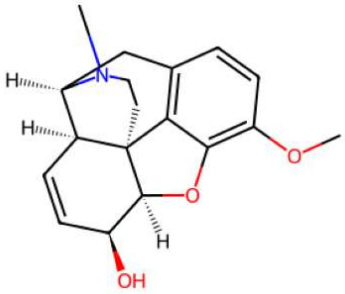
```
        'Efavirenz': 'C1CC1C#C[C@]2(C3=C(C=CC(=C3)Cl)NC(=O)O2)C(F)(F)F',
        'D-Glucose': 'C([C@@H]1[C@H]([C@@H]([C@H](C(O1)O)O)O)O)O',
        'Beta lactose'
:'C([C@@H]1[C@@H]([C@@H]([C@H]([C@@H](O1)O[C@@H]2[C@H](O[C@H]([C@@H]([C@H]2O)O)O)
CO)O)O)O)O',
         'Corannulene':'C1=CC2=C3C4=C1C=CC5=C4C6=C(C=C5)C=CC(=C36)C=C2',
        'Coronene': 'C1=CC2=C3C4=C1C=CC5=C4C6=C(C=C5)C=CC7=C6C3=C(C=C2)C=C7',
        'Porphyrin ring':'C1=CC2=CC3=CC=C(N3)C=C4C=CC(=N4)C=C5C=CC(=N5)C=C1N2',
        'Methylene blue': 'CN(C)C1=CC2=C(C=C1)N=C3C=CC(=[N+](C)C)C=C3S2.[Cl-]',
        'Thiamine': 'CC1=C(SC=[N+]1CC2=CN=C(N=C2N)C)CCO.[Cl-]', 'Nicotine': 'CN1CCCC1C2=CN=CC=C2',
'Codeine':'[H][C@@]12OC3=C(OC)C=CC4=C3[C@@]11CCN(C)[C@]([H])(C4)[C@]1([H])C=C[C@@H]2O',
        'Morphine': 'CN1CC[C@]23[C@@H]4[C@H]1CC5=C2C(=C(C=C5)O)O[C@H]3[C@H](C=C4)O',
'Ephedrine': 'CN[C@H]([C@@H](c1ccccc1)O)C',
        'Cocaine': 'CN1[C@H]2CC[C@@H]1[C@H]([C@H](C2)OC(=O)C3=CC=CC=C3)C(=O)OC',
        'Cholesterol':
'C[C@H](CCCC(C)C)[C@H]1CC[C@@H]2[C@@]1(CC[C@H]3[C@H]2CC=C4[C@@]3(CC[C@@H](C4)O)C
)C'}
compound_name = input('Enter the name of the compound: ')
if compound_name in smiles:
    smiles_notation = smiles[compound_name]
    mol = Chem.MolFromSmiles(smiles_notation)
    # Check if the molecule was successfully created
    if mol is not None:
        img = Draw.MolToImage(mol, size=(300, 300))
        plt.imshow(img)
        plt.axis('off')
        plt.show()  # Display the image using Matplotlib
    else:
        print('Invalid SMILES notation.')
else:
    print ('Compound not found in the dictionary
```

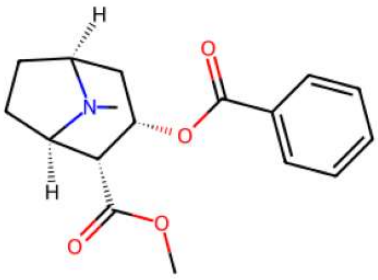
Table-2: Input Table for determination of 2D and 3D Structure of an Organic Compound

S.No.	Name of Compound	SMILES in Dictionary
1.	Nicotine	CN1CCCC1C2=CN=CC=C2
2.	Caffeine	CN1C=NC2=C1C(=O)N(C(=O)N2C)C
3.	Efavirenz	C1CC1C#C[C@]2(C3=C(C=CC(=C3)Cl)NC(=O)O2)C(F)(F)F
4.	Sulfasalazine	C1=CC=NC(=C1)NS(=O)(=O)C2=CC=C(C=C2)N=NC3=CC(=C(C=C3)O)C(=O)O
5.	Lycopene	CC(=CCC/C(=C/C=C/C(=C/C=C/C(=C/C=C/C=C(/C=C/C=C(/CCC=C(C)C)\C)\C)\C)/C)/C)C
6.	D-Glucose	C([C@@H]1[C@H]([C@@H]([C@H](C(O1)O)O)O)O)O
7.	β-Lactose	C([C@@H]1[C@@H]([C@@H]([C@H]([C@@H](O1)O[C@@H]2[C@H](O[C@H]([C@@H]([C@H]2O)O)O)CO)O)O)O)O
8.	Coronene	C1=CC2=C3C4=C1C=CC5=C4C6=C(C=C5)C=CC7=C6C3=C(C=C2)C=C7
9.	Corannulene	C1=CC2=C3C4=C1C=CC5=C4C6=C(C=C5)C=CC(=C36)C=C2
10.	Porphyrin ring	C1=CC2=CC3=CC=C(N3)C=C4C=CC(=N4)C=C5C=CC(=N5)C=C1N2
11.	Methylene Blue	CN(C)C1=CC2=C(C=C1)N=C3C=CC(=[N+](C)C)C=C3S2.[Cl-]
12.	Thiamine (Vitamin B_1)	CC1=C(SC=[N+]1CC2=CN=C(N=C2N)C)CCO.[Cl-]
13.	Cholesterol	C[C@H](CCCC(C)C)[C@H]1CC[C@@H]2[C@@]1(CC[C@H]3[C@H]2CC=C4[C@@]3(CC[C@@H](C4)O)C)C
14.	Codeine	[H][C@@]12OC3=C(OC)C=CC4=C3[C@@]11CCN(C)[C@]([H])(C4)[C@]1([H])C=C[C@@H]2O
15.	Cocaine	CN1[C@H]2CC[C@@H]1[C@H]([C@H](C2)OC(=O)C3=CC=CC=C3)C(=O)OC

Table-3: Output Table for 2D Structure of the Organic Compound

S.No.	Name of Compound	2D Structure of the Compound
1.	Nicotine	
2.	Caffeine	
3.	Efavirenz	
4.	Sulfasalazine	
5.	Lycopene	
6.	D-Glucose	
7.	β-Lactose	

8.	Coronene	
9.	Corannulene	
10.	Porphyrin ring	
11.	Methylene Blue	
12.	Thiamine (Vitamin B_1)	
13.	Cholesterol	
14.	Codeine	

15.	Cocaine	

C. Drawing 3D structure of any given Organic Compound

Finally, Python code is also used to draw the 3D structure of organic compounds in the following situations:

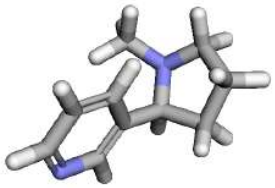
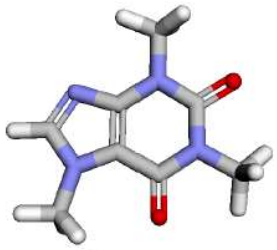
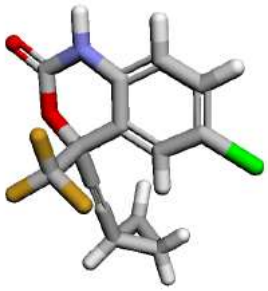
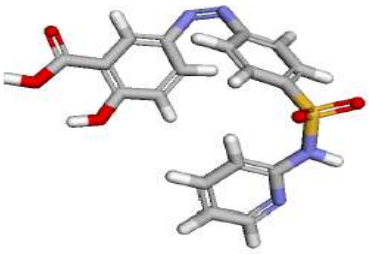
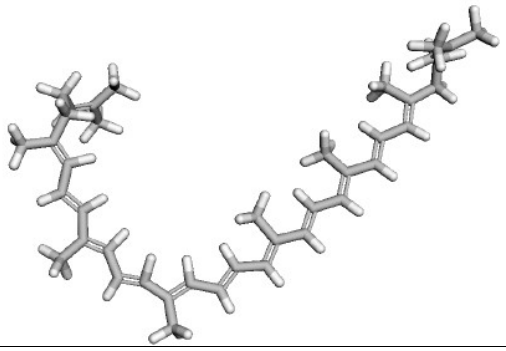
1. The program stores some information as a dictionary and iterates over a dictionary.
2. The key is the name of the organic compound to be asked/provided as input.
3. The value is SMILES of various organic compounds stored in the dictionary.

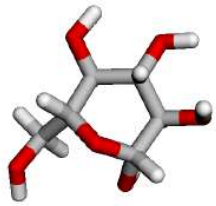
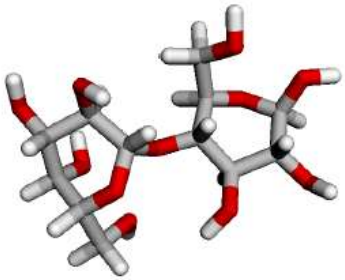
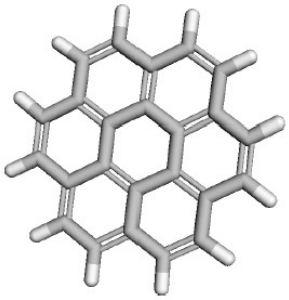
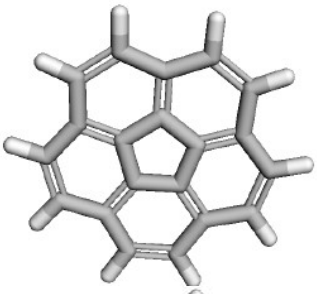
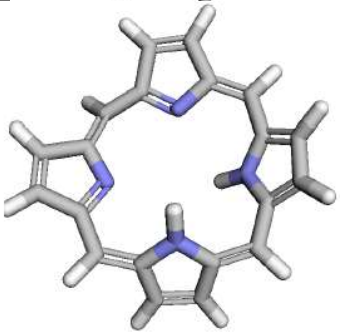
Python Program to Draw 3D Structure of the Compound is:

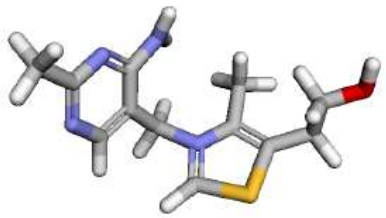
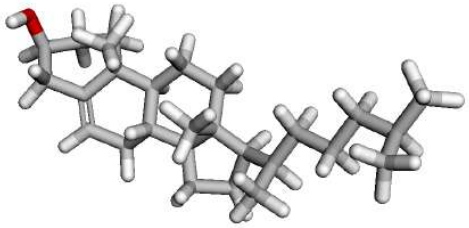
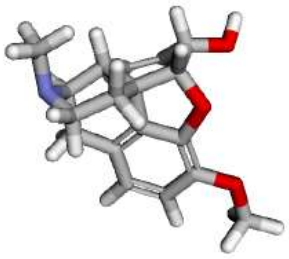
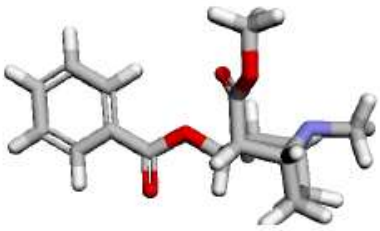
```
 !pip install rdkit-pypi
!pip install py3Dmol
from rdkit import Chem
from rdkit.Chem import AllChem
import py3Dmol
smiles = {'Caffeine': 'CN1C=NC2=C1C(=O)N(C(=O)N2C)C', 'Methylamine': 'CN', 'Nicotine':
'CN1CCCC1C2=CN=CC=C2', 'Lycopene':
'CC(=CCC/C(=C/C=C/C(=C/C=C/C(=C/C=C/C=C(/C=C/C=C(/C=C/C=C(/CCC=C(C)C)\C)\C)\C)/C)/C)/C)C',
        'Beta carotene':
'CC1=C(C(CCC1)(C)C)/C=C/C(=C/C=C/C(=C/C=C/C=C(/C=C/C=C(/C=C/C2=C(CCCC2(C)C)C)\C)\C)/C)/C',
        'Sulfasalazine': 'C1=CC=NC(=C1)NS(=O)(=O)C2=CC=C(C=C2)N=NC3=CC(=C(C=C3)O)C(=O)O',
        'Efavirenz': 'C1CC1C#C[C@]2(C3=C(C=CC(=C3)Cl)NC(=O)O2)C(F)(F)F',
        'D-Glucose': 'C([C@@H]1[C@H]([C@@H]([C@H](C(O1)O)O)O)O)O',
        'Beta lactose'
:'C([C@@H]1[C@@H]([C@@H]([C@H]([C@@H](O1)O[C@@H]2[C@H](O[C@H]([C@@H]([C@H]2O)O)O)
CO)O)O)O)O',
         'Corannulene':'C1=CC2=C3C4=C1C=CC5=C4C6=C(C=C5)C=CC(=C36)C=C2',
        'Coronene': 'C1=CC2=C3C4=C1C=CC5=C4C6=C(C=C5)C=CC7=C6C3=C(C=C2)C=C7',
        'Porphyrin ring':'C1=CC2=CC3=CC=C(N3)C=C4C=CC(=N4)C=C5C=CC(=N5)C=C1N2',
        'Methylene blue': 'CN(C)C1=CC2=C(C=C1)N=C3C=CC(=[N+](C)C)C=C3S2.[Cl-]',
        'Thiamine': 'CC1=C(SC=[N+]1CC2=CN=C(N=C2N)C)CCO.[Cl-]', 'Nicotine': 'CN1CCCC1C2=CN=CC=C2',
'Codeine':'[H][C@@]12OC3=C(OC)C=CC4=C3[C@@]11CCN(C)[C@]([H])(C4)[C@]1([H])C=C[C@@H]2O',
        'Morphine': 'CN1CC[C@]23[C@@H]4[C@H]1CC5=C2C(=C(C=C5)O)O[C@H]3[C@H](C=C4)O',
'Ephedrine': 'CN[C@H]([C@@H](c1ccccc1)O)C',
        'Cocaine': 'CN1[C@H]2CC[C@@H]1[C@H]([C@H](C2)OC(=O)C3=CC=CC=C3)C(=O)OC',
        'Cholesterol':
'C[C@H](CCCC(C)C)[C@H]1CC[C@@H]2[C@@]1(CC[C@H]3[C@H]2CC=C4[C@@]3(CC[C@@H](C4)O)C
)C'}
compound_name = input('Enter the name of the compound: ')
if compound_name in smiles:
   smiles_notation = smiles[compound_name]
   mol = Chem.MolFromSmiles(smiles_notation)
   if mol is not None:
     mol = Chem.AddHs(mol)
     AllChem.EmbedMolecule(mol, randomSeed=42, useRandomCoords=True)
     from rdkit.Chem import AllChem
     block = Chem.MolToMolBlock(mol)
     viewer = py3Dmol.View(width=300, height=300)
     viewer.addModel(block, format='sdf')
```

```
    viewer.setStyle({'stick': {}})
    viewer.zoom to()
    viewer.show()
  else:
    print('Invalid SMILES notation.')
else:
  print('Compound not found in the dictionary.
```

Table-4: Output table for the 3D structure of the organic compound

S. No.	Name of Compound	3D Structure of the Compound
1.	Nicotine	
2.	Caffeine	
3.	Efavirenz	
4.	Sulfasalazine	
5.	Lycopene	

6.	D-Glucose	
7.	β-Lactose	
8.	Coronene	
9.	Corannulene	
10.	Porphyrin ring	
11.	Methylene Blue	

12.	Thiamine (Vitamin B_1)	
13.	Cholesterol	
14.	Codeine	
15.	Cocaine	

CONCLUSION

In this paper, the authors have used the Python program to arrive at the empirical and molecular formula accurately for a large number of compounds for which the atomic mass of the elements present was stored in the dictionary. The number of elements present in the compound along with their percentages and the molecular mass of the compound needed to be provided as input. The Python code was further used to draw the 2D and 3D structures with complete precision for a number of compounds. For both structures, only the name of the compound was needed and the SMILES of that compound should be stored in the dictionary. In the same manner and using these results the 2D and 3D structures of other compounds can be determined. The aim is to build a complete library where the structures of any organic compound can be reproduced with just a click. The authors of this article hope that the Python program may be further extended to inorganic compounds and complexes as well. These results demonstrate the versatility of the Python program and opens up a plethora of opportunities for the organic chemists without doing any calculations, just by the click of a mouse. With the introduction of the Python Program in the teaching of chemistry students at the undergraduate level this paper would provide the right opportunity to students as well. We also hope that the Python codes can be further used in a number of ways thus saving the time and energy of the chemists as well as integrating computer programming and organic chemistry closely together.

ACKNOWLEDGEMENTS

Dr. Hemant Verma would like to thank Hindu College, University of Delhi; Prof. Sarita Passey and Dr. Jyoti Singh would like to thank Zakir Husain Delhi College, University of Delhi, for the support.

CONFLICT OF INTERESTS

The authors declare that there is no conflict of interest.

AUTHOR CONTRIBUTIONS

All the authors contributed significantly to this manuscript, participated in reviewing/editing, and approved the final draft for publication. The research profile of the authors can be verified from their ORCID IDs, given below:

Sarita Passey http://orchid.org/0000-0001-9856-9163
Hemant Verma http://orchid.org/0009-0008-8713-3450
Jyoti Singh http://orchid.org/0009-0001-2640-6549

REFERENCES

1. A. Dwivedi, A. Jaiswal, Python: The Versatile Language, *Recent Trends in Programming Languages*, **8**, 24(2021)
2. R. Prajapati, A. Agrahari, A. Shahi, *International Journal of Innovative Research in Technology*, **8**, 151859(2021), https://ijirt.org/Article?manuscript=151859
3. https://allusesof.com/technology/15-uses-of-python-programming-language/
4. Y. Shen, J. E. Borowski, M.A. Hardy, *et. al., Nature Reviews Methods Primers*, **1**, 23(2021), https://doi.org/10.1038/s43586-021-00022-5
5. Matthew Sparkes, https://www.newscientist.com/article/2305188-human-and-robot-chemists-work-better-together-than-alone/
6. J. Boyd, Robotic Laboratory Automation, *Science,* **295**, 517(2002), https://doi.org/10.1126/science.295.5554.517
7. James Mitchell Crow, https://www.chemistryworld.com/features/the-robots-revolutionising-chemistry/4016798.article
8. A. Saeki and K. Kranthiraja, *Japanese Journal of Applied Physics,* **59**, SD0801(2020), https://doi.org/10.7567/1347-4065/ab4f39
9. A. Milo, *Israel Journal of Chemistry*, **58**, 131(2018), https://doi.org/10.1002/ijch.201700148
10. M. Elyashberg, *TrAC Trends in Analytical Chemistry*, **69**, 88(2015), https://doi.org/10.1016/j.trac.2015.02.014
11. Q. Zhu, S. Hattori, *Journal of Materials Research*, **38**, 19(2023), https://doi.org/10.1557/s43578-022-00698-9.

[RJC-8848/2024]