International Journal of Tropical Medicine
ISSN: 1818-779X
OPEN ACCESS

Original Article

Python In Pharmacy: Its Background, Significance, And Application

Pawan Vishwakarma[1*], Abdul Quaiyoom[2], Navneet Kumar Verma[3], Ashish Pal[4]
[1-4]Suyash Institute of Pharmacy, Hakkabad, Gorakhpur, Uttar Pradesh, India-273016.

***Corresponding Author**
Pawan Vishwakarma
Suyash Institute of Pharmacy, Hakkabad, Gorakhpur, Uttar Pradesh, India-273016
Email- pv838616@gmail.com

Article History
Received on: 25-08-2025
Accepted on: 20-09-2025
Published on: 04-10-2025

Copy Right: © This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution and reproduction in any medium, provided the original author and source are credited

Abstract: Over the past few years, Python has become more and more popular. Python has been rapidly gaining popularity in recent years and is frequently taught as the first programming language to students at several colleges. Guido van Rossum created Python at the National Research Institute for Mathematics and Computer Science in the Netherlands in the late 1980s and early 1990s. Python is very easy to read and learn because it is quite similar to the English language. Python's history is a dynamic narrative that combines the vision of its founder, the cooperative efforts of a broad community, and the language's adaptation to changing technical landscapes. It is not just a chronological development of versions and features. Python's beginnings date back to the late 1980s, when Guido van Rossum developed a language intended for readability and user-friendliness. Pharmacy education is changing quickly to incorporate data-driven and digital technology. Python programming and artificial intelligence (AI) have become crucial skills for contemporary pharmacists, enabling them to handle complicated datasets, simulate drug behavior, and improve patient outcomes.

Keywords: *Python, History of Python, Biopython, Application of Python in Pharmacy, Application of Python in Drug Development*

INTRODUCTION

PYTHON

Over the past few years, Python has become more and more popular [1]. Python has been rapidly gaining popularity in recent years and is frequently taught as the first programming language to students at several colleges [2]. Guido van Rossum created Python at the National Research Institute for Mathematics and Computer Science in the Netherlands in the late 1980s and early 1990s. Numerous other languages, such as ABC, Modula-3, C, C++, Algol-68, Small Talk, Unix shell, and other scripting languages, are the ancestors of Python. Python is protected by copyright. Python source code is currently distributed under the GNU General Public License (GPL), just like Perl. Although a core development team at the institute currently maintains Python, Guido van Rossum continues to play a crucial role in steering its advancement. November 1994 saw the release of Python 1.0. Python 2.0 was released in 2000. The most recent version of Python 2 is 2.7.11. In the meantime, Python 3.0 came out in 2008. Python 2 and Python 3 are incompatible. In Python 3, the focus was on eliminating redundant programming structures and modules so that "There should be one-and preferably only one-obvious way to do it." The most recent version of Python 3 is 3.7.4 [3].

Python is a strong programming language that is simple to learn. It features effective high-level data structures and a straightforward yet powerful object-oriented programming methodology. Python is a perfect language for scripting and quick application development across a wide range of platforms due to its sophisticated syntax, dynamic typing, and interpreted nature [4]. Numerous scientific and technical jobs can benefit from the usage of the Python programming language. It can be used for data analysis and plotting. Additionally, it can be used to solve scientific and engineering issues numerically that are challenging or even impossible to answer analytically [5].

Python is a high-level general-purpose programming language that is interpreted. Python's design philosophy places a strong emphasis on code readability by using a lot of indentation. Its object-oriented methodology and language elements are designed to assist programmers in writing logical, understandable code for both small and large-scale projects [6]. Both

inexperienced programmers and highly experienced professional developers utilize Python. It is utilized in financial organizations, big businesses, academic institutions, and web enterprises. Python's versatility allows it to be used in nearly every area of software development. Any area of development can make use of Python as a whole [7,8,9]. The Python programming language is growing in popularity for the following reasons:

- **Simple and Easy to learn**: Python is very easy to read and learn because it is quite similar to the English language.
- **Supportive Community**: Since Python has been around for thirty years, there has been ample time for a sophisticated and encouraging community to emerge around the language. Python learners of all ages and abilities can obtain the assistance they require to advance their language skills through official documentation and YouTube lessons.
- **Web Development**: Python offers several options for web programming because it supports a variety of frameworks for creating websites. Python has a plethora of frameworks, including Django, Flask, Pylon, and many more. The majority of web development uses Python.
- **Use In Big Data and Machine Learning**: Two of the most popular computer science topics at the moment are big data and machine learning, which are assisting businesses in changing their workflows and procedures. A large portion of this research and development is done in Python. Python is the second most widely used tool for data science and analytics, and it powers numerous data processing tasks in businesses all around the world. In the meantime, thousands of machine learning projects use Python libraries like TensorFlow for neural networks and OpenCV for computer vision every day.
- **Efficiency:** Compared to more traditional programming languages like Java and C++, Python represents a distinct programming paradigm. But this "Pythonic" approach frequently allows developers to accomplish more with less effort—often in just a few lines of code. Additionally, Python's adaptability makes it possible to utilize the language in a wide range of contexts, from desktop programs and hardware programming to web and mobile development [10].
- Python is an interactive, object-oriented, and interpreted programming language. Lists, tuples, sets, associative arrays (also known as dictionaries), dynamic typing and binding, modules, classes, exceptions, intelligent memory management, and more are among the high-level data structures it offers. It is a sophisticated programming language that is also utilized for parallel computing systems and has a very straightforward coding syntax. The Java interpreter for Python, called JPython, is comparable to the C interpreter. In terms of machine learning, Python outperforms C++, Java, JavaScript, and other languages, according to a GitHub study conducted between January and December of 2018 [11].
- Its use as a scripting language in large software suites, its active development community, its free availability, its widespread use in scientific disciplines like bioinformatics, and the numerous publicly accessible scientific libraries (like BioPython) [12].

The significance of licenses in scientific software development should be briefly mentioned in any discussion of libraries, modules, and extensions. One scientist's development work will almost certainly support the research endeavors of others, either in the short or long term, in subfields that may be closer to one's own or perhaps farther away (and unexpected), as demonstrated by the widespread usefulness of existing software libraries in contemporary research communities. By promoting open interchange, transparency, communicability, and reproducibility of research initiatives, free software licenses support the unrestricted advancement of scientific research. A program must enable the user to read and modify the source code (for any purpose), share the code to others, and distribute modified copies of the code to others in order to be considered free software. Lists of licenses that, to varying degrees, adhere to the open-source criteria are alphabetized and categorized by the Open-Source Initiative [13].

Two fundamental characteristics of computer programs are (i) algorithms, or, more broadly, the "programming logic," and (ii) data structures, or how data is represented within the program, including whether particular components are manipulable, inerrable, etc. [14]. Python is especially well-suited to the object-oriented programming (OOP) paradigm, which views these two aspects of a program as interdependent. There are numerous comprehensive approaches to OOP, including language-independent texts [15].

History of Python

Python's history is a dynamic narrative that combines the vision of its founder, the cooperative efforts of a broad community, and the language's adaptation to changing technical landscapes. It is not just a chronological development of versions and features. Python's beginnings date back to the late 1980s, when Guido van Rossum developed a language intended for readability and user-friendliness. Mark Lutz and David Ascher's 1999 book "Learning Python" serves as a key resource for comprehending the fundamentals of Python. "Python Tutorial" (1995) is essential to comprehending the introduction and communication of Python to its user base. Jason R. Briggs' "Python for Kids" (2012) serves as a focal

point for evaluating Python's contribution to educational endeavors. A Brief History of Development of Python Programming Language:

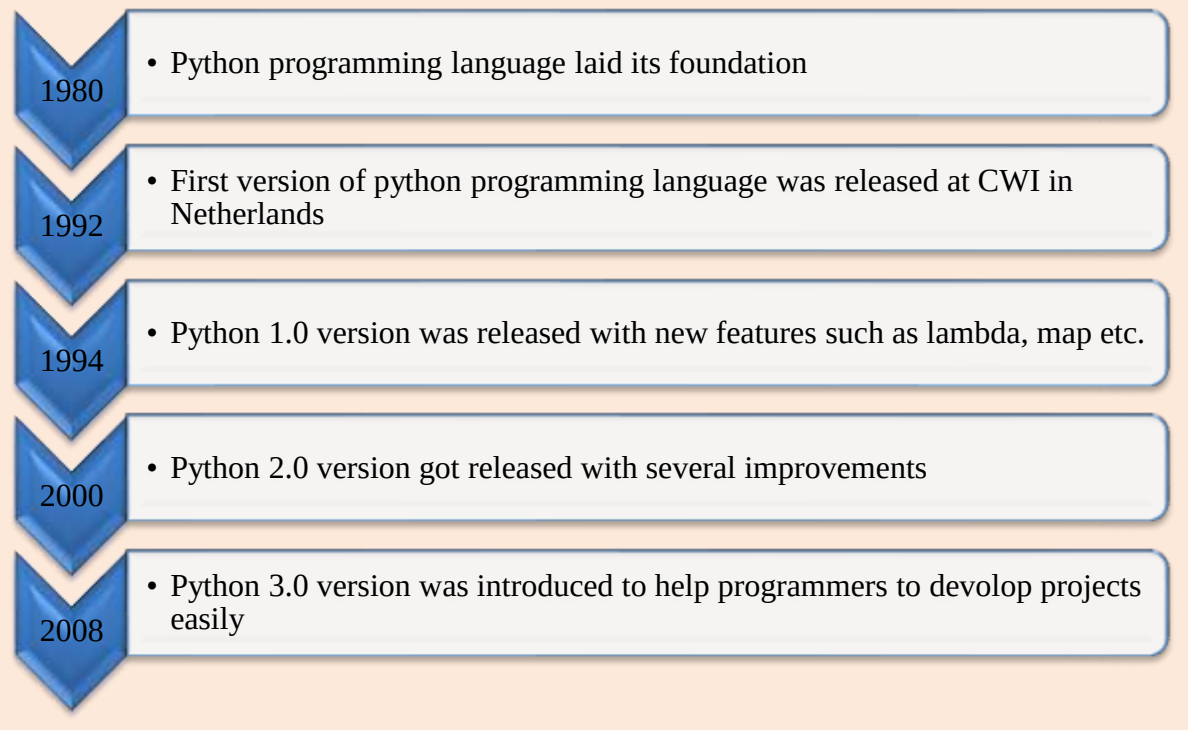


Figure: 1. The Evolution of Python Programming Language [16].

Third Party of Python

There are now a lot of well-known third-party Python libraries. These established projects are often (i) well-documented, (ii) readily accessible as stable (production) versions, (iii) constantly being developed to add new features, and (iv) distinguished by sizable user bases and vibrant communities (mailing lists, etc.). The SciPy repository offers a helpful set of these tools [17, 18]. It serves as a framework for the distribution and upkeep of a number of well-known packages, including NumPy, which is crucial for calculations involving matrices [19]. PANDA, which makes data import, management, and organization easier [20], MATPLOTLIB, a top codebase for plotting and general-purpose visualization [21], and SciPy, which offers routines from linear algebra, signal processing, statistics, and a multitude of other numerical tools.

Machine learning and statistical analysis features are added to SciPy by the package SCIKIT-LEARN [22]. Lastly, there are other more specialized packages, such Py Brain [23]. as well as DEAP [24].

Python In Biological Research

As a language to learn, Python is a great option [25]. The computational challenges that come up when conducting biological research will probably be similar to those that have already been resolved and for which software libraries are already available. This is mostly due to the interdisciplinary character of biological research, which applies very well-established formalisms and methods from mathematics, computer science, and physics to biological systems. For example, (i) The simulated annealing method was created as a physically inspired combinatorial optimization technique, and it quickly became a key component in the refining of biomolecular structures as determined by X-ray crystallography or NMR spectroscopy [26]. (ii) Before being widely used in sequence alignment techniques and other bioinformatics domains, dynamic programming was developed as an optimization technique in operations research. (iii) The algorithms employed in applications ranging from phylogenetic tree estimation to protein structure prediction are based on the Monte Carlo method, which was developed as a sampling technique in physics [27]. All of the computational techniques mentioned above can be implemented in Python.

Biopython

In order to create Python modules and applications that meet the demands of present and future bioinformatics work, Biopython is a distributed, cooperative effort [28]. It offers resources for handling biological sequences, parsers for widely used file formats in bioinformatics (FASTA, COMPASS [29], GenBank, PIR [30], PDB [31], BLAST output [32], Inter

Pro [33], Locus Link [34], PROSITE [35], Phred [36], Phrap [37], data retrieval from biological databases Swiss-Prot [38], PubMed [39], GenBank [40], a wrapper for bioinformatics programs (BLAST, ClustalW [41], EMBOSS [42], Primer3 [43], and more), functions to estimate DNA and protein properties such as isoelectric points [44,45].

This study only presents a small piece of the larger picture because a review of Biopython functions would take up a much larger amount of material. The first example demonstrates how to extract and report only the necessary features from a BLAST output. Writing a BLAST report parser is a fundamental bioinformatics activity because BLAST is the most widely used program in the field [46]. Sequence alignment, motif analysis, clustering algorithms, structural biology modules, phylogenetic analysis modules, NCBI, and ExPASy online services are all accessible through Biopython, a bioinformatics processing service that is a veritable gold mine of online resources [47, 48].

For editing, visualizing, and analyzing numerous sequence comparisons, bioinformatics databases and tools such as Jalview and the SIB Swiss Institute of Bioinformatics are essential. In the fields of proteins and proteomes, glycomics, structural biology, genes and genomes, and evolutionary and systems biology, they provide researchers with an abundance of resources. These databases and software tools offer strong support for processing and analyzing the ever-increasing amount of biometric data because they have built-in capabilities for visualizing and analyzing DNA, RNA, and protein sequences and structures as well as linked views of aligned DNA and protein products. This makes life sciences research and development easier, allowing researchers to more effectively evaluate data, promote scientific discovery, and spark innovation in the field of bioinformatics [49].

Basic Feature of Biopython

With its rich feature set and user-friendly design, Biopython has become an indispensable programming tool in the field of bioinformatics, winning over developers all over the world [50]. By introducing feature-rich sequence objects, Biopython goes beyond the constraints of traditional string manipulation [51]. By offering a common interface for reading and writing a range of sequence file formats, the SeqIO module in Biopython functions as a flexible data processing tool, similar to a Swiss Army knife [52].

Reliable sequencing results depend on Biopython's capacity to filter and execute quality control on sequencing data. It guaranties data accuracy, providing a strong basis for further analysis and research projects. This trait is crucial in the era of widely used high-throughput sequencing technology. Because of its many capabilities and easy-to-use interface, Biopython has established itself as a vital resource for bioinformatics researchers. It not only increases the effectiveness of research but also propels scientific advancements. Because of its adaptability and extensibility, Biopython is well-positioned to develop with the rapidly expanding area of bioinformatics, offering countless possibilities for further research projects [53].

Because of its adaptability and forward-thinking outlook, Biopython has become an indispensable tool for academics working in the dynamic and revolutionary field of bioinformatics [54]. The continuous development of Biopython has accelerated new discoveries and greatly accelerated the scientific research process [55]. Biopython is well-positioned to continue serving as a catalyst for scientific advancement and technological innovation as the life sciences develop and expand, offering future researchers better support and promoting deeper and more expansive exploration in the field [56].

Importance of Python Programming in Pharmacy

Pharmacy education is changing quickly to incorporate data-driven and digital technology. Python programming and artificial intelligence (AI) have become crucial skills for contemporary pharmacists, enabling them to handle complicated datasets, simulate drug behavior, and improve patient outcomes [57]. By bridging computational intelligence and pharmaceutical sciences, AI & Python Programming for Pharmacy–II develops students become analytically motivated professionals. Learners can model pharmacokinetic processes and analyze complex clinical data by using mathematics-based AI techniques, such as matrices, differentiation, and neural-network processing. Students who are exposed to AI-enabled pharmacy operations and CDSS systems are better equipped to optimize therapy, guaranty medication accuracy, and advance patient safety [58].

The computational foundation of AI algorithms used in drug modeling and process optimization is explained to students using mathematical concepts like matrices, calculus, and differential equations [59]. Programs created in Python may operate on a variety of operating systems, including Windows, macOS, and Linux, because to its cross-platform interoperability. In healthcare settings, where several platforms and systems may be used, this flexibility is beneficial. Generally, Python code written for one platform can run on another with little to no changes. The vast Python community offers a wealth of learning and problem-solving options. Programmers of all skill levels can get help via online tutorials,

documentation, forums, and community-contributed code libraries. This community help facilitates troubleshooting and skill improvement for students learning Python.

Python's dynamic typing enables variables to store many kinds of data without the need for explicit type definitions. This feature lowers the overhead of variable management and makes writing code easier. Beginners can learn programming more easily by concentrating on data values and operations rather than type specifications [60].

Application of Python in Drug Development

The process of finding new drugs is expensive and time-consuming. The use of molecular modeling and in silico computational chemistry for computer-aided drug creation has been increasingly popular recently. The use of free and/or open-source software in the drug discovery process is examined in this paper. Applications written in Python, Perl, and Java as well as resources such software libraries are among the software packages that have been examined. Cheminformatics approaches to drug discovery, such as QSAR studies, energy minimization, and docking studies in drug design attempts, may benefit from these programs. This evaluation also looks at ways to use open-source computer modeling software into drug discovery initiatives. Since then, cheminformatics has advanced dramatically thanks to developments in computer hardware and software that compute chemical characteristics in order to find molecules that could be effective medications [61].

Application of AI In Pharmacy Operation

By automating tedious and error-prone activities, artificial intelligence (AI) transforms pharmacy operations. AI-powered automated dispensing systems use robotic arms, barcode scanning, and image recognition to guaranty precise medicine selection, packing, and labeling [62]. AI systems can use machine learning algorithms to predict trends in drug demand and expiration. Time-series models, for example, examine past prescription data to forecast seasonal increases in the use of vaccines or antibiotics. This lowers wastage from outdated stock and aids pharmacists in maintaining ideal inventory levels [63].

In order to identify any drug interactions, allergies, or dosage problems, AI-based software continuously scans electronic prescriptions. Before dispensing, for instance, machine learning algorithms incorporated into hospital information systems identify possible adverse occurrences [64]. Sensors, Bluetooth, and cloud analytics are used by AI-enabled smart pillboxes and mobile health applications to monitor medication compliance. For example, the system notifies the pharmacist or sends customized reminders if a patient forgets to take a dose [65].

Application of Python in Pharmacy

Pharmacy students frequently work with huge datasets, such as stability data, dissolution profiles, or clinical outcomes. They can effectively store and handle such data thanks to Python's lists and tuples [66]. Functions promote reusable, modular code. For pediatric dosing calculations, for example, a function calculate dose (Weight, mg per kg) might be used frequently. In a similar vein, functions can convert storage temperatures or compute medication release, improving the efficiency and organization of code [67]. Digital record management is made possible via file handling (open, read, write, close). Batch records, like Batch F2 – pH: 5.8 – Release: 78%, can be automatically written by students and retrieved at a later time. This ability is in line with Good Manufacturing Practice (GMP) criteria for digital documentation [68].

Future Prospective of Python

Programming and data science are becoming increasingly important in the pharmaceutical sciences. Python's status as a top language for automation, machine learning, and data analysis indicates that its significance in pharmacy and healthcare environments will only increase. Professionals with both clinical domain expertise and computational skills are in greater demand as a result of the healthcare industry's digital revolution. Graduates in pharmacy who are proficient in programming will be better suited for new positions that integrate these skills. Software development and data analysis skills are necessary for telemedicine platforms, customized medicine applications, and clinical decision support systems. Python is appropriate for creating and managing these systems because of its adaptability [69].

Data production, reduction/processing, and integration are at least three aspects of the complex data-analysis requirements of contemporary research initiatives. Technological and engineering advancements, like commodity equipment for next-generation DNA sequencing, are major drivers of data production [70], and robotics for structural genomics [71,72].

The most recent generation of highly scalable, parallel MD codes can produce data faster than it can be sent to local workstations for processing via standard computer network backbones. Tools for "on-the-fly" trajectory analysis have been developed as a result of these demands [73,74]. The Protein Data Bank (PDB) will probably certainly be used in any type

of 3D structural investigation [75]. The modeling and simulation of inter-neuronal connections at the tissue system level have emerged as a new area called "connectomics" [76,77].

It is commonly acknowledged that one of science's main challenges is producing knowledge from massive databases [78]. Biologists with the ability to gather, organize, process/reduce, and interpret large-scale datasets—both quantitatively and visually—are desperately needed. The issues are more basic than, say, merely "data-wrangling," or changing the format of data files. Fortunately, it is possible to quickly grasp the fundamentals of the required computational procedures. A working knowledge of a programming language is one of the two main components of computational fluency, and (ii) an understanding of fundamental computer science concepts, such as data structures and sorting techniques. A seemingly rudimentary understanding of programming fundamentals will frequently be sufficient to comprehend the workings of extremely complex code because all programming projects are built upon the same set of fundamental principles. As specific research questions arise, one can become familiar with more advanced topics (such as graph algorithms, computational geometry, numerical methods, etc.). Early scientific training is the ideal time to start developing computational skills. Interdisciplinary education seems to be useful in bridging the knowledge gap between young scientists in the life sciences and computer science, as revealed by recent educational studies [79,80].

CONCLUSION

In contemporary pharmacy, Python has become a game-changing instrument. In this review paper, we concentrated on how Python and AI are crucial to the pharmacy industry and how they facilitate medical research. Both students and teachers can learn the fundamentals of how and where Python can be used in the medical or pharmacy fields by reading this article. It also sheds light on how Python helps with the structural design and formulation of novel drugs. Python is a game-changer in modern pharmacy because it makes programming languages simpler and easier to understand. With its assistance, pharmacists can rapidly analyze and store data, speed up research on new drugs, and lower the risk of prescription errors.

REFERENCES

1. Dyer R, Chauhan J. An exploratory study on the predominant programming paradigms in Python code. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering 2022 Nov 7 (pp. 684-695).
2. Robert M. Siegfried, Diane Liporace, and Katherine G. Herbert-Berger. 2019. WhatCan the Reid List of First Programming Languages Teach Us About TeachingCS1?. In Proceedings of the 50th ACM Technical Symposium on Computer ScienceEducation (Minneapolis, MN, USA) (SIGCSE '19). Association for ComputingMachinery, New York, NY, USA, 1256ś1257.
3. https://www.geeksforgeeks.org/history-of-python/
4. Van Rossum G, Drake FL. An introduction to Python. Bristol: Network Theory Ltd.; 2003 Apr.
5. Pine D. Introduction to Python for science.
6. Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises". Section 1.1. Archived from the original (PDF) on 23 June 2012.
7. K. R. Srinath, "Python – The Fastest Growing Programming Language," International Research Journal of Engineering and Technology (IRJET), vol. 4, Issue 12 pp. 354–357, December 2017.
8. Kalyani Adawadkar, "Python Programming-Applications and Future," International Journal of Advance Engineering and Research Development(IJAERD), Special Issue SIEICON-2017, pp. 1–4, April -2017.
9. QuantInsti, Python Basics With Illustrations from the Financial Markets, QuantInsti Quantitative Learning Pvt. Ltd.
10. Saabith AS, Fareez MM, Vinothraj T. Python current trend applications-an overview. International Journal of Advance Engineering and Research Development. 2019 Oct;6(10).
11. Dhruv AJ, Patel R, Doshi N. Python: the most advanced programming language for computer science applications. Science and Technology Publications, Lda. 2021;1:292-9.
12. Cock PJA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, et al. BioPython: Freely Available Python Tools for Computational Molecular Biology and Bioinformatics. Bioinformatics. 2009 Mar;25(11):1422–1423. pmid:19304878.
13. Open Source Initiative: Licenses & Standards;. Available from: http://opensource.org/licenses.
14. Wirth N. Algorithms + Data Structures = Programs. Prentice-Hall Series in Automatic Computation. Prentice Hall; 1976.
15. Budd T. An Introduction to Object-Oriented Programming. 3rd ed. Pearson; 2001.
16. Maan R, Sharma A. The Evolution of Python Programming Language. International Journal of Gender, Science and Technology. 2020;9(2).

17. Jones E, Oliphant T, Peterson P, et al. SciPy: Open-source Scientific Tools for Python; 2001-. [Online; accessed 2015-06-30]. Available from: http://www.scipy.org/.
18. Scientific Computing Tools for Python;. Available from: http://www.scipy.org/about.html.
19. van der Walt S, Colbert SC, Varoquaux G. The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering. 2011 Mar;13(2):22–30.
20. Python Data Analysis Library;. Available from: http://pandas.pydata.org/.
21. Hunter JD. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering. 2007;9(3):90–95.
22. scikit-learn: machine learning in Python;. Available from: http://scikit-learn.org/.
23. PyBrain. Journal of Machine Learning Research. 2010;11:743–746.
24. Fortin FA, Rainville FMD, Gardner MA, Parizeau M, Gagné C. DEAP: Evolutionary Algorithms Made Easy. Journal of Machine Learning Research. 2012 jul;2171–2175(13).
25. Chou PH (2002) Algorithm education in Python. Available: http://www.python10.org/p10-papers/index.htm. Accessed 4 October 2007.
26. Brunger AT, Adams PD. Molecular Dynamics Applied to X-ray Structure Refinement. Accounts of Chemical Research. 2002 Jun;35(6):404–412. pmid:12069625.
27. Langtangen HP. A Primer on Scientific Programming with Python. Texts in Computational Science and Engineering. Springer; 2014.
28. (2007) Biopyton Version 1.43. Available: http://www.biopython.org/. Accessed 4 October 2007.
29. Sadreyev R, Grishin N (2003) COMPASS: A tool for comparison of multiple protein alignments with assessment of statistical significance. J Mol Biol 326: 317–336.
30. Wu CH, Yeh LS, Huang H, Arminski L, Castro-Alvear J, et al. (2003) The Protein Information Resource. Nucleic Acids Res 31: 345–347.
31. Hamelryck T, Manderick B (2003) PDB file parser and structure class implemented in Python. Bioinformatics 19: 2308–2310.
32. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, et al. (1997) Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. Nucleic Acids Res 25: 3389–3402.
33. Mulder NJ, Apweiler R, Attwood TK, Bairoch A, Bateman A, et al. (2005) InterPro, progress and status in 2005. Nucleic Acids Res 33: D201–D205.
34. The Centre for Applied Genomics (2007) NCBI Locus Link in BioXRT Database. Available: http://projects.tcag.ca/bioxrt/locuslink/. Accessed 4 October 2007.
35. Hulo N, Bairoch A, Bulliard V, Cerutti L, De Castro E, et al. (2006) The PROSITE database. Nucleic Acids Res 34: D227–D230.
36. Ewing B, Hillier L, Wendl M, Green P (1998) Basecalling of automated sequencer traces using phred. I. Accuracy assessment. Genome Res 8: 175–185.
37. Laboratory of Phil Green (2007) Phred, Phrap, Consed. Available: http://www.phrap.org/phredphrapconsed.html#block_phrap. Accessed 4 October 2007
38. Bairoch A, Boeckmann B, Ferro S, Gasteiger E (2004) Swiss-Prot: Juggling between evolution and stability. Brief Bioinform 5: 39–55.
39. PubMed (2007) Available: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi. Accessed 4 October 2007.
40. Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Wheeler DL (2006) GenBank. Nucleic Acids Res 34: D16–D20.
41. Higgins DG, Thompson JD, Gibson TJ (1996) Using CLUSTAL for multiple sequence alignments. Methods Enzymol 266: 383–402.
42. Rozen S, Skaletsky H (2000) Primer3 on the WWW for general users and for biologist programmers. Methods Mol Biol 132: 365–386.
43. Toldo L (2007) PI EMBL WWW Gateway to Isoelectric Point Service. Available: http://www.embl-heidelberg.de/cgi/pi-wrapper.pl. Accessed 4 October 2007.
44. Bjellqvist B, Hughes GJ, Pasquali C, Paquet N, Ravier F, et al. (1993) The focusing positions of polypeptides in inmobilized pH gradients can be predicted from their amino acid sequences. Electrophoresis 14: 1023–1031.
45. Stajich JE, Lapp H (2006) Open source tools and toolkits for bioinformatics: Significance, and where are we? Brief Bioinform 7: 287–296.
46. McGinnis S (2005) NCBI communication to BioPerl Team. Available: http://www.bioperl.org/w/index.php?title=NCBI_Blast_email&oldid=5114. Accessed 14 June 2007
47. Shajii, A., Numanagić, I., Leighton, A. T. et al. (2021). A Python-based programming language for highperformance computational genomics. Nat Biotechnol, 39: 1062–1064.
48. Cock, P., Antao, T., Chang, J. T. et al. (2009). Biopython: Freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11): 1422–1423. DOI: 10.1093/bioinformatics/btp163.

49. Wei J. Biopython: important application tools in the field of bioinformatics and computational molecular biology. InІнноватика в освіті, науці та бізнесі: виклики та можливості 2024 Nov 15. Київський національний університет технологій та дизайну.
50. Chapman, B., Chang, J. (2000). Biopython: Python tools for computational biology. ACM SIGBIO Newsletter, 20(2): 15–19.
51. Chapman, B. A., Chang, J. T. (2011). Biopython: an enhanced suite of Python libraries for computational biology. ACM SIGBIO Newsletter, 51–52.
52. Druce, M., Cock, P. J. A. (2012). SeqIO: a Biopython module for handling sequence data. Proceedings of the 12th Python in Science Conference.
53. Chang, J., Chapman, B., Friedberg, I., Hamelryck, T., de Hoon, M., Cock, P., ... Wilczynski, B. (2010). Biopython tutorial and cookbook. Update, 15–19.
54. Le Cun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. Nature, 521(7553): 436–444.
55. Chapman, B. A., Chang, J. T. (2011). Biopython: an enhancing tool for bioinformatics. Methods in Molecular Biology, 694: 301–320.
56. Cock, P. J. A., Grierson, D. S. (2013). Biopython as a tool for computational molecular biology and bioinformatics education. Briefings in Bioinformatics, 14(4): 453–457.
57. Aziz, M. H. A., AlShammari, T. M., & Elnaem, M. H. (2024). A scoping review of artificial intelligence within pharmacy. American Journal of Pharmaceutical Education, 88(4), 100615.
58. Patel B, Mahato TK. World Journal Medical and Pharmaceutical Research. World. 2025;1(07):01-13.
59. Simpson, M. D., Lin, S., & Kang, H. (2025). Clinical and operational applications of artificial intelligence in pharmacy practice. Frontiers in Pharmacology, 16, 1193220.
60. Nagda P. Book: PYTHON PROGRAMMING FOR PHARMACEUTICAL AND HEALTHCARE APPLICATIONS.
61. Geldenhuys WJ, Gaasch KE, Watson M, Allen DD, Van der Schyf CJ. Optimizing the use of open-source software applications in drug discovery. Drug discovery today. 2006 Feb 1;11(3-4):127-32.
62. Simpson, M. D., Lin, S., & Kang, H. (2025). Clinical and operational applications of artificial intelligence in pharmacy practice. Frontiers in Pharmacology, 16, 1193220.
63. Kandhare, P., Inamdar, M., & Patil, A. (2025). Artificial intelligence in pharmaceutical sciences: A review. Journal of Pharmaceutical Sciences & Technology, 12(2), 89–104.
64. Mortlock, R., Naunton, M., & Peterson, G. (2024). Generative artificial intelligence in pharmacy: Opportunities and challenges. Exploratory Research in Clinical and Social Pharmacy, 6, 100178.
65. Aziz, M. H. A., AlShammari, T. M., & Elnaem, M. H. (2024). A scoping review of artificial intelligence within pharmacy. American Journal of Pharmaceutical Education, 88(4), 100615.
66. Abdel Aziz, M. H., Rowe, C., Southwood, R., Nogid, A., Berman, S., & Gustafson, K. (2024). A scoping review of artificial intelligence within pharmacy education. American Journal of Pharmaceutical Education, 88(1)l: 100615.
67. Mortlock, R., & Lucas, C. (2024). Generative artificial intelligence (Gen-AI) in pharmacy education: Utilization and implications for academic integrity: A scoping review. Exploratory Research in Clinical and Social Pharmacy, 15: 100481
68. Simpson, M. D., & Qasim, H. S. (2025). Clinical and operational applications of artificial intelligenceand machine learning in pharmacy: A narrative review of real-world applications. Pharmacy (Basel), 13(2): 41.
69. Nagda P. Book: PYTHON PROGRAMMING FOR PHARMACEUTICAL AND HEALTHCARE APPLICATIONS.
70. Ekmekci B, McAnany CE, Mura C. An introduction to programming for bioscientists: a Python-based primer. PLoS computational biology. 2016 Jun 7;12(6):e1004867.
71. Klock HE, White A, Koesema E, Lesley SA. Methods and Results for Semi-automated Cloning Using Integrated Robotics. Journal of Structural and Functional Genomics. 2005;6(2–3):89–94. pmid:16211504.
72. Viola R, Carman P, Walsh J, Frankel D, Rupp B. Automated Robotic Harvesting of Protein Crystals—Addressing a Critical Bottleneck or Instrumentation Overkill? Journal of Structural and Functional Genomics. 2007 Dec;8(4):145–152. pmid:17965947.
73. Ramanathan A, Agarwal PK, Kurnikova M, Langmead CJ. An Online Approach for Mining Collective Behaviors from Molecular Dynamics Simulations. Journal of Computational Biology. 2010 Mar;17(3):309–324. pmid:20377447
74. Tribello GA, Bonomi M, Branduardi D, Camilloni C, Bussi G. PLUMED 2: New Feathers for an Old Bird. Computer Physics Communications. 2014 Feb;185(2):604–613.

75. Bernstein FC, Koetzle TF, Williams GJ, Meyer EF, Brice MD, Rodgers JR, et al. The Protein Data Bank: A Computer-based Archival File for Macromolecular Structures. Journal of Molecular Biology. 1977 May;112(3):535–542. pmid:875032
76. Lichtman JW, Pfister H, Shavit N. The Big Data Challenges of Connectomics. Nature Neuroscience. 2014 Nov;17(11):1448–1454. pmid:25349911
77. Engert F. The Big Data Problem: Turning Maps into Knowledge. Neuron. 2014 Sep;83(6):1246–1248. pmid:25233305.
78. Rudin C, Dunson D, Irizarry R, Ji H, Laber E, Leek J, et al. Discovery with Data: Leveraging Statistics with Computer Science to Transform Science and Society. American Statistical Association; 2014. Available from:
79. Rubinstein A, Chor B. Computational Thinking in Life Science Education. PLoS Comput Biol. 2014 Nov;10(11):e1003897. pmid:25411839
80. Goodman AL, Dekhtyar A. Teaching Bioinformatics in Concert. PLoS Comput Biol. 2014 Nov;10(11):e1003896. pmid:25411792.