International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

The Role of Python in Developing Artificial Intelligence Applications: A Review of Tools, Libraries, and Learning Accessibility

Kartik Agarwal

Abstract

Python has emerged as a leading programming language for artificial intelligence because of its readable syntax, extensive library ecosystem, open-source nature, and accessibility. This review examines the factors driving Python's adoption in AI, its principal libraries and frameworks, major applications, and limitations compared with alternative languages. Tools such as NumPy, Pandas, Scikit-learn, TensorFlow, PyTorch, Keras, and OpenCV support data processing, machine learning, deep learning, and computer vision. Python is widely applied in natural language processing, healthcare, bioinformatics, education, finance, automation, and emerging technologies. The review finds that Python's primary strength lies in simplifying development, enabling rapid prototyping, supporting collaboration, and reducing barriers to AI learning. Although C++, Java, and R may provide advantages in execution speed, enterprise performance, or statistical analysis, Python compensates through optimized libraries and strong community support. Despite challenges involving memory use, security, scalability, and package dependencies, Python is likely to remain central to AI development.

Keywords: Python; artificial intelligence; machine learning; deep learning; data science; AI libraries; programming accessibility

CHAPTER 1: INTRODUCTION

1.1 Background of the Study

Over the past decades, artificial intelligence has witnessed significant growth in its applications across various sectors, mainly, healthcare, business, engineering, science, agriculture, defence, and weather forecasting (Pannu, A. 2015; Jindal, H. et al. 2021). Programming languages are the vital tools in the development of artificial intelligence systems, enabling the simulation of intelligent processes (Neumann. G, 2002) while also playing a significant role in determining the execution speed and efficiency in machine learning tasks (Adetiba, E. 2021).

Python has emerged as a dominant programming language in artificial intelligence due to its ease of use, flexibility, and strong ecosystem of libraries, thus making it a preferred choice among researchers and developers. Python facilitates rapid prototyping and iteration so that AI researchers can try new models with ease (Talati, D. 2021)

1.2 Problem Statement

Despite the widespread popularity of Python in artificial intelligence, there is limited critical evaluation of its limitations in high-performance AI systems. Many developers adopt Python due to its ease of use and extensive ecosystem of libraries without comparing it with other programming languages such as C, C++ or Java. This creates a lack of understanding whether Python's dominance

is because of its technical superiority or its accessibility. Furthermore, the increasing dependence on Python raises concerns regarding scalability and long-term sustainability in complex AI systems.

1.3 Research Objectives

- To examine the role of Python in the development of AI applications across industries.
- To analyze major Python tools and libraries used in machine learning and deep learning.
- To evaluate the accessibility of Python for beginners and non-technical users.
- To identify challenges, limitations, and future prospects of Python in AI.

1.4 Significance of the Study

This study provides guidance to students and researchers in selecting appropriate tools for artificial intelligence development. It helps developers understand the strengths and weaknesses of Python in real-world applications. Additionally, it contributes to academic literature by offering a structured review of Python's role in artificial intelligence and assists policymakers and educators in designing AI-focused curricula.

1.5 Research Question

The central research question guiding this study is: Why has Python emerged as the dominant programming language in artificial intelligence development, and how do its tools, libraries, and accessibility influence its adoption compared to alternative programming languages?

CHAPTER 2: EVOLUTION OF PYTHON IN COMPUTING

2.1 Origin and Development of Python

Python is a popular open source programming language developed by Guido van Rossum in the late 1980s and was released in 1991 with the first version named as 0.9.0 with a focus on code readability and simplicity. Python was created as a better substitute of ABC language as it closes the gap with its exceptions handling and its ability to interact with the Amoeba's operating system which was also created by the former. With the release of version 2.0 in the year 2000, it brought a progressive change in Python as it introduced numerous new features with it like garbage collection, list comprehension, Unicode support, reference counting tool and cycle detecting. The version 3.9 is security wise and capable.

The name "Python" was derived from the British comedy group Monty Python, as Guido van Rossum had fun while developing the language. Python has features like dynamic typing, dustbin for memory management. Python is a multi-paradigm programming language that accommodates multiple programming paradigms which comprises procedural, object-oriented, and functional programming. Python comes with various modules and packages that makes the developer's work easier and simpler (Elhalid,O et al. 2023;Fatoba, et al. 2025).

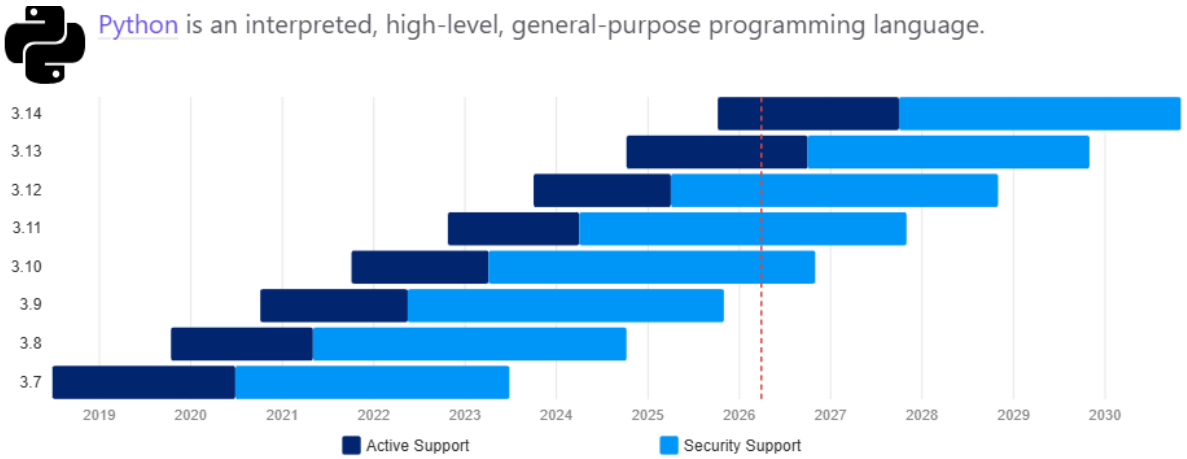

Figure 1

Python Software Release Life Cycle and Support Schedule Timeline

Note. Reprinted from *Python Development Cycle Status*, by endoflife.date, 2026 (https://endoflife.date/). Licensed under CC BY-NC-SA 4.0.

2.2 Growth of Python in Software Development

Software development is a collection of technical and problem-solving skills along with creativity that results in forming, writing, and testing software systems. Its aim is to fulfil the needs of the modern digital world and is used in industries and organisations by creating efficient and user-oriented software systems. Python is used in web development,computer automation, artificial intelligence and data science (Chaturvedi,V.2023)

Over the years, Python has witnessed significant growth in its popularity among other programming languages, mainly due to its strong and supportive community. From Youtube Tutorials to Python resource websites, there is enough material online to improve the knowledge of the language (Saabith,A.2019). Python has the ability to easily add analytics and machine learning to present systems via frameworks like Django and Flask that help reduce significant application development time. They can be read and customised, making it a strong tool for startups and large companies (Chaturvedi,V.2023).

Frameworks are a collection of libraries and modules to create an effective web application in a less amount of time; Python is used in creation of web frameworks such as CherryPy, Django, TurboGears, Bottle, Flask, and more. It provides standard libraries and modules that help in completing tasks such as content management and interfacing for different internet protocols. Because of Python's readability and regular maintenance, it is very popular among developers. There are three web development frameworks for Python:

Full Stack Frameworks	**Non-Full Stack Frameworks**	**Asynchronous Framework**
It gives full support from the human-machine interface to the storehouse of data.	These are light frameworks as it does not provide as many features of full stack frameworks.The user have to enter a lot of code manually	These frameworks allow large bundles of simultaneous connections.
Django-This framework is	**Flask**-It has a built-in debugger and is a	**AIOHTTP**-It has pluggable

secure,fast and versatile. **Web2Py**-This framework is secure and works cross platform	lightweight framework **CherryPy**-It has a flexible plugin system and can easily be run on many HTTP servers simultaneously.	routing and supports both Client WebSockets and Server WebSockets.

2.3 Python's Entry into AI and Data Science

With the growing interest in data science over the last decade, Python has become one of the most widely used languages for handling large-scale datasets efficiently (Fanchon, E. et al. 2025). Python has emerged as the cornerstone programming language in the rapidly evolving fields of Artificial Intelligence and Data Science, and its combination of accessibility, robust ecosystem, and versatile capabilities has made it the preferred choice for researchers, developers, and organizations pushing the boundaries of machine intelligence (Mantrala, S. 2026). The study analyzes Python's rich ecosystem of libraries and frameworks, including NumPy, Pandas, Scikit-learn, TensorFlow, PyTorch, Keras, OpenCV, and Hugging Face Transformers, which collectively provide comprehensive support for data processing, machine learning, deep learning, computer vision, natural language processing, and predictive analytics (Mantrala, S. 2026). Python's adoption in AI and ML has revolutionized the field, enabling rapid development, prototyping, and deployment of intelligent systems (Fatoba et al. 2025). Evidence from recent academic literature and industry reports demonstrates that Python has substantially reduced development time, improved collaboration between multidisciplinary teams, and democratized access to advanced AI technologies by lowering technical barriers for students, researchers, and practitioners (Mantrala, S. 2026). With user-friendly libraries such as Scikit-learn and the growing availability of educational resources and online platforms, the entry barriers for understanding and applying machine learning have significantly decreased (Fatoba et al. 2025).

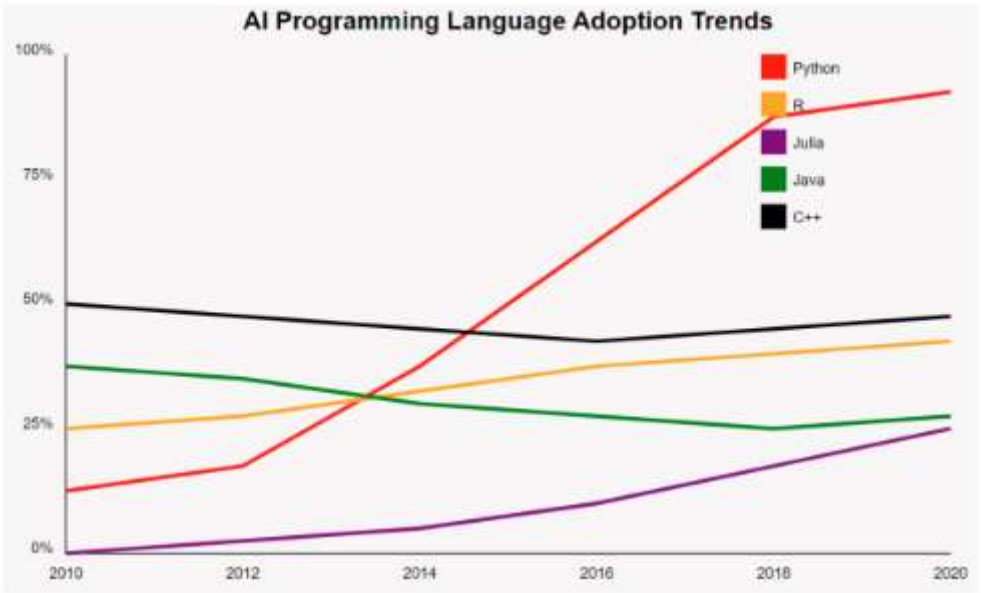

Figure 2

AI Programming Language Adoption Trends

Note. Reprinted from "Python in the Evolution of AI: A Comparative Study of Emerging Technologies," by H. Patil, V. Mahandule, and A. Gunjal, 2024, *SSRN Electronic Journal*, p. 2 (ssrn.com).

CHAPTER 3: FACTORS DRIVING PYTHON'S ADOPTION IN AI

3.1 Simplicity and Readability

One of the most powerful features of Python is that it has English-like syntax, which makes Python extremely readable and easy to use. Unlike other low-level programming languages like C or Java that need complex syntax rules, Python provides developers the freedom to develop readable and brief code. This enhances code readability and maintainability as well as collaboration, particularly in large-scale projects with various developers (Talati, D. 2021).

Python's simplicity and readability, which makes it easy for beginners to learn while still being powerful enough for advanced research and industrial applications. In classroom teaching, Python allows students to focus more on understanding concepts rather than struggling with complex syntax. This ease of learning significantly reduces the entry barrier for students exploring AI and Data Science (Akhtar, T. Ranjan, A. 2026).

The ease of learning and coding efficiency directly affect adoption rates among developers and researchers. Python and R excel in this category due to their simple syntax and high level abstractions. Julia also offers an intuitive syntax similar to MATLAB, making it easier to learn. In contrast, C++ has a steep learning curve due to complex memory management, and Java requires more boilerplate code, making rapid prototyping less efficient (Müller, M. 2025).

Python's simplicity, readability, and robust ecosystem of libraries make it particularly suited for complex computations required in AI, ranging from basic data manipulation to advanced deep learning frameworks like TensorFlow and PyTorch Additionally, Python's syntax is intuitive, making it an ideal teaching tool in AI and software engineering courses, even for novice programmers (Müller, M. 2025).

Another peculiarity of Python is that it has extremely simple syntax and compact coding structure. In comparison with other popular programming languages that need heaps of code in order to do something, Python does the same but with less code. It makes debugging and testing easier, which makes the programmers concentrate more on logic and solution methods rather than on the complex syntax rules (Talati, D. 2021).

3.2 Extensive Libraries and Frameworks

Python's dominance is its rich ecosystem of libraries and frameworks that support data analysis, machine learning, and artificial intelligence (Akhtar, T. Ranjan, A. 2026).

Python is the most widely used programming language for AI and ML due to its simplicity, readability, and vast ecosystem of libraries such as TensorFlow, PyTorch, Scikit-learn, and Keras. Its dynamic typing and high-level syntax make it accessible for researchers and developers, enabling rapid prototyping and experimentation 2025 (Müller, M. 2025).

Libraries such as NumPy, SciPy, and Scikit-learn allow developers to build sophisticated models with relatively little code (Müller, M. 2025).

- **TensorFlow**: TensorFlow is an open-source library for machine learning and deep learning developed by Google Brain in 2015. The most popular framework for developing and deploying artificial intelligent models , because of its flexibility and scalability. TensorFlow is a preferred choice for large scale artificial intelligence projects due to its data processing capability on CPU and GPU also (Patil, H. et al. 2024).
- **PyTorch**: Designed by Facebook's AI Research lab, PyTorch provides deep learning users with an easy-to-use interface and dynamic computation graphs. PyTorch has grown in popularity since its 2016

release due to its robust support for research applications and ease of usage. Researchers love it because of its speedy prototyping and debugging capabilities (Patil, H. et al. 2024).

- **Scikit-learn**: Scikit-learn is a complete library of classical machine learning techniques that offers tools for clustering, regression, classification, and data preprocessing . Its widespread use in AI applications can be attributed to its simplicity and ease of interaction with other libraries (Patil, H. et al. 2024).

3.3 Community Support and Open-Source Nature

Python is freely available for everyone. It is available on its official website www.python.org and has a large community across the world that is dedicatedly working towards making new python modules and functions. Anyone can contribute to the Python community. Open-source means, "Anyone can download its source code without paying any penny" (Saabith,A.2020).

Python comes with an open-source license that has been approved by the Open Source Initiative (OSI). Python can hence be freely downloaded, altered, and redistributed by anybody. The development of Python is ensured with the massive open-source community that keeps on contributing to it (Talati, D. 2021). It has the largest developer community with vast web resources, tutorials, and forums. Whether a beginner or an expert, programmers can always get assistance and problem-solving tips from Python's huge user base (Talati, D. 2021). This community support provides access to a wide range of courses, events and third-party tools. (Patil, H. et al. 2024). The robust Python community continues to drive innovation, with regular updates to existing libraries and the constant emergence of new tools addressing cutting-edge AI challenges. (Mantrala, S. 2026).

Aug 2026	Aug 2025	Change	Programming Language	Ratings	Change
1	1		Python	18.53%	-7.61%
2	3	^	C	11.10%	+2.07%
3	2	v	C++	8.62%	-0.56%
4	4		Java	8.25%	-0.34%
5	5		C#	4.09%	-1.43%
6	6		JavaScript	2.83%	-0.52%
7	7		Visual Basic	2.18%	-0.15%
8	12	^^	SQL	1.88%	+0.16%
9	14	^^	R	1.56%	+0.19%
10	18	^^	Rust	1.45%	+0.31%
11	10	v	Delphi/Object Pascal	1.37%	-0.46%
12	17	^^	Scratch	1.27%	+0.12%
13	15	^	PHP	1.11%	-0.16%
14	8	vv	Go	1.07%	-1.04%
15	11	vv	Fortran	0.99%	-0.77%
16	26	^^	Ruby	0.98%	+0.23%
17	25	^^	Swift	0.96%	+0.19%
18	9	vv	Perl	0.94%	-1.13%
19	22	^	COBOL	0.87%	+0.02%
20	20		Assembly language	0.86%	-0.16%

Figure 2

TIOBE Programming Community Index Top 20 Rankings for August 2026

Note. Reprinted from *TIOBE Index for August 2026*, by TIOBE, 2026 (tiobe.com). Copyright 2026 by TIOBE.

3.4 Cross-Platform Compatibility

Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, and so we can say that Python is a portable language, as it enables programmers to develop software for several competing platforms by writing a program only once (Saabith, A. et al. 2020).

Python's integration with cloud computing platforms has ushered in a new era of IT solutions, enabling organizations to leverage the full potential of cloud-based technologies. Python's popularity in cloud computing rests on its compatibility with major cloud platforms such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), and its libraries and frameworks provide developers with the tools they need to seamlessly integrate with cloud services, manage resources, and deploy scalable solutions.

Python's support for automation and scripting makes it well-suited for orchestrating cloud infrastructure and managing complex workflows, and with libraries like boto3 for AWS, Azure-SDK for Azure, and Google-Cloud-Python for GCP, developers can interact with cloud services programmatically, enabling efficient resource provisioning, monitoring, and management. Python's dominance in data science and machine learning also contributes to its significance in cloud computing, as many data scientists and machine learning practitioners prefer Python for developing and deploying models in cloud environments due to its rich ecosystem of libraries such as NumPy, Pandas, TensorFlow, and PyTorch (Swaminathan, K. Kaavya, K. 2024).

Python's ability to run across different operating systems ensures that AI applications can be developed and deployed seamlessly across diverse environments, from development laptops to cloud-based production systems, and this flexibility has proven particularly valuable as AI workloads increasingly span heterogeneous computing environments (Mantrala, S. 2026).

CHAPTER 4: PYTHON LIBRARIES FOR AI AND MACHINE LEARNING

4.1 Core Machine Learning Libraries

Python seems to be winning the battle as the preferred language of Machine Learning. The availability of libraries and open source tools make it an ideal choice for developing ML models. Python has been the go-to choice for Machine Learning and Artificial Intelligence developers for a long time. It offers some of the best flexibility and features to developers that not only increase their productivity but the quality of the code as well, not to mention the extensive libraries helping ease the workload (Saabith, A. 2020).

Python is the most widely used programming language for AI and ML due to its simplicity, readability, and vast ecosystem of libraries such as TensorFlow, PyTorch, Scikit-learn, and Keras. Its dynamic typing and high-level syntax make it accessible for researchers and developers, enabling rapid prototyping and experimentation (Singh, S. 2025).

TensorFlow: Coming from developers at Google, it is an open-source library of data flow graphs computations, which are sharpened for Machine Learning. It was designed to meet the high-demand requirements of Google environment for training Neural Networks and is a successor of DistBelief, a Machine Learning system, based on Neural Networks. However, TensorFlow isn't strictly for scientific use in the borders of Google. It is also general enough to use it in a variety of real-world applications. The key feature of TensorFlow is their multi-layered nodes system that enables quick training of artificial

neural networks on large datasets. This powers Google's voice recognition and object identification from pictures. (Saabith,A.2020)

SciKit-Learn: Scikits are additional packages of SciPy Stack designed for specific functionalities like image processing and machine learning facilitation. In the regard of the latter, one of the most prominent of these packages is scikit-learn. The package is built on the top of SciPy and makes heavy use of its math operations. The scikit-learn exposes a concise and consistent interface to the common machine learning algorithms, making it simple to bring ML into production systems. The library combines quality code and good documentation, ease of use and high performance and is the de-facto industry standard for machine learning with Python. (Saabith,A.2020)

Keras: It is an open-source library for building Neural Networks at a high-level of the interface, and it is written in Python. It is minimalistic and straightforward with a high level of extensibility. It uses Theano or TensorFlow as its backends, but Microsoft makes its efforts now to integrate CNTK (Microsoft's Cognitive Toolkit) as a new back-end. The minimalistic approach in design aimed at fast and easy experimentation through the building of compact systems. Keras is really eased to get started with and keep going with quick prototyping. It is written in pure Python and high-level in its nature. It is highly modular and extendable. The general idea of Keras is based on layers, and everything else is built around them. Data is prepared in tensors, the first layer is responsible for input of tensors, the last layer is responsible for output, and the model is built in between. (Saabith,A.2020)

PyTorch: It is one of the few machine learning libraries for Python. Apart from Python, PyTorch also has support for C++ with its C++ interface if you're into that. Considered among the top contenders in the race of being the best Machine Learning and Deep Learning framework, PyTorch faces tough competition from Tensor Flow. PyTorch is a popular open-source Machine Learning library for Python based on Torch, which is an open-source Machine Learning library which is implemented in C with a wrapper in Lua. It has an extensive choice of tools and libraries that support Computer Vision, Natural Language Processing (NLP) and many more ML programs. It allows developers to perform computations on Tensors with GPU acceleration and also helps in creating computational graphs. (Saabith,A.2020)

4.2 Data Processing Libraries

Machine learning and scientific computing applications commonly utilize linear algebra operations on multidimensional arrays, which are computational data structures for representing vectors, matrices, and tensors of a higher order. NumPy is a multidimensional array library with basic linear algebra routines, and the SciPy library adorns NumPy arrays with many important primitives, from numerical optimizers and signal processing to statistics and sparse linear algebra. While both NumPy and Pandas [10] (Figure1) provide abstractions over a collection of data points with operations that work on the dataset as a whole, Pandas extends NumPy by providing a data frame-like object supporting heterogeneous column types and row and column metadata. In recent years, Pandas library has become the de-facto format for representing tabular data in Python for extract, transform, load" (Kumar, P. et al. 2022).

NumPy (Numerical Python) is the fundamental package for numerical computation in Python; it contains a powerful N-dimensional array object. It has around 18,000 comments on GitHub and an active community of 700 contributors. It's a general-purpose array-processing package that provides high-performance multidimensional objects called arrays and tools for working with them. NumPy also addresses the slowness problem partly by providing these multidimensional arrays as well as providing functions and operators that operate efficiently on these arrays. Key features of NumPy are: Provides fast, precompiled functions for numerical routines, Array-oriented computing for better efficiency, Supports

an object-oriented approach, and Compact and faster computations with vectorization. NumPy is particularly useful for Extensively used in data analysis, creates a powerful N-dimensional array,

Pandas: Pandas (Python data analysis) is a must in the data science life cycle. It is the most popular and widely used Python library for data science, along with NumPy in matplotlib. With around 17,00 comments on GitHub and an active community of 1,200 contributors, it is heavily used for data analysis and cleaning. Pandas provide fast, flexible data structures, such as data frame CDs, which are designed to work with structured data very quickly and intuitively. Key features of Pandas are Eloquent syntax and rich functionalities that gives you the freedom to deal with missing data, enables you to create your function and run it across a series of data, High-level abstraction, and Contains highlevel data structures and manipulation tools. SciPy is particularly useful forGeneral data wrangling and cleaning, ETL (extract, transform, load) jobs for data transformation and data storage, as it has excellent support for loading CSV files into its data frame format, Used in a variety of academic and commercial areas, including statistics, finance, and neuroscience, Time-series-specific functionality, such as date range generation, moving window, linear regression, and date shifting

SciPy (Scientific Python) is another free and open-source Python library extensively used in data science for high-level computations. SciPy has around 19,000 comments on GitHub and an active community of about 600 contributors. It's widely used for scientific and technical computations because it extends NumPy and provides many user-friendly and efficient routines for scientific calculations. Key features of SciPy are Collection of algorithms and functions built on the NumPy extension of Python, High-level commands for data manipulation and visualization, Multidimensional image processing with the SciPy.ndimage submodule, and Includes built-in functions for solving differential equations. SciPy is particularly useful for Multidimensional image operations, Solving differential equations and the Fourier transform, Optimization algorithms, and Linear algebra (Saabith, A. 2020).

4.3 Visualization Libraries

Data visualization is an essential tool for data analysis, al- lowing data analysts and scientists to explore and communicate data insights effectively. Python has become one of the most widely used programming languages for data analysis, data visualization, and machine learning due to its rich library ecosystem that provides powerful tools for data visualization. Several Python data visualization libraries have emerged recently, providing various options for data analysts and scientists (Lavanya, A. et al. 2023).

Data visualization is the graphical representation of information and data using visual elements like charts, graphs, and maps. It is critical in decision-making processes and used in various fields such as healthcare, finance, marketing, and more. Python is an ideal choice for data visualization due to its numerous visualization libraries (Lavanya, A. et al. 2023).

It is often useful to visualize the characteristics of a model's learned parameters and the interpretation of its interactions with a set of data. Feature importance and attribution scores can provide more useful insights when analyzed in a visual form, exposing patterns that would be otherwise difficult to discern. In the Python machine learning community, Matplotlib, Seaborn , Bokeh , and Altair are widely used for visualization data in plots and charts (Kumar, P. et al. 2022).

Matplotlib: The most well-known library for data visualization in Python. It is a low-level library that provides a wide range of customizable 2D and 3D plots, including scatter plots, line plots, histograms, and more. Matplotlib is built on NumPy arrays and is highly compatible with other Python libraries, such as Pandas, NumPy, and Scikit-learn. It also has an interactive environment that can be used across multiple platforms. Matplotlib has been widely used in research as a powerful data visualization tool, providing a

wide range of visualization options that allow researchers to communicate complex data and trends easily. Its interactive visualizations can be manipulated in real time, and it provides a way to create publication-quality outputs that can be easily reproduced in research papers and presentations (Lavanya, A. et al. 2023).

Seaborn: Another popular library for data visualization in Python. It is built on Matplotlib and provides more advanced statistical visualization features. Seaborn has meaningful default themes, offering different colour palettes defined around best practices. It interfaces well with Pandas data frames, provides data mapping onto visualizations, and can transform data as part of plot creation. Seaborn has helped researchers across various fields with exploratory data analysis by providing a high-level interface for creating attractive and informative statistical graphics, including scatterplots, bar plots, and heatmaps that allow researchers to explore their data and identify patterns and relationships (Lavanya, A. et al. 2023).

Bokeh: A Python library for creating interactive visualizations. It provides a flexible and powerful toolset for creating interactive plots and can handle large and complex datasets, including streaming data and real-time updates. Bokeh is a popular data visualization library that provides interactive and responsive web browser plots. It has been widely used in research for exploratory data analysis, enabling researchers to visualize patterns and relationships in their data quickly. Bokeh's interactive tools allow researchers to zoom, pan, and select data points, providing a powerful way to gain insights into complex datasets (Lavanya, A. et al. 2023).

Altair: A declarative library for creating interactive visualizations in Python. It is built on Vega-Lite and provides a simple and intuitive syntax for creating visualizations. Altair supports various data formats and can be used with Pandas data frames, CSV files, JSON files, and more. It allows for declarative visualization concisely and intuitively, and has gained popularity in the research community due to its ability to produce high-quality visualizations with minimal coding. Altair helps research by providing a simple and effective way to create interactive visualizations for data exploration and analysis, and can reveal complex patterns and trends in data -especially useful in fields such as data science and machine learning (Lavanya, A. et al. 2023).

4.4 Deep Learning Packages Comparison

The selection of an appropriate deep learning framework is a critical decision in any AI development project, as each framework offers distinct strengths and limitations depending on the nature of the task. A comprehensive review of Python-based deep learning packages by (Mohialden et al. 2024) evaluated five major frameworks -TensorFlow, PyTorch, Keras, Theano, and Caffe -across key parameters including ease of use, performance, flexibility, community support, documentation, and compatibility.

PyTorch, originally developed by Facebook's AI Research Lab and now part of the Linux Foundation, is distinguished by its dynamic computational graph, which allows for greater flexibility and real-time code testing. It supports over 200 mathematical operations and provides robust tools for data loading, preprocessing, and model optimization. PyTorch has experienced significant growth in research environments due to its ease of experimentation and rapid prototyping capabilities (Mohialden et al., 2024).

Theano, developed by the Montreal Institute for Learning Algorithms (MILA), was one of the earliest Python-based deep learning libraries and contributed significantly to the foundation of modern AI frameworks. It is particularly effective at optimizing mathematical computations through symbolic differentiation and integrates well with NumPy. However, active development of Theano ceased in 2017,

and it has since been succeeded by the Aesara project, limiting its relevance in contemporary AI development (Mohialden et al., 2024).

Caffe, developed by Berkeley AI Research (BAIR), is optimized for speed and efficiency in image classification and convolutional models. Its configuration-based programming model enables straightforward CPU-GPU switching and simplified deployment. However, its restricted interaction with other frameworks and the requirement to define models through configuration files limit its flexibility compared to TensorFlow and PyTorch (Mohialden et al., 2024).

A comparative analysis of these frameworks reveals clear patterns in their suitability for different use cases. TensorFlow and PyTorch lead in terms of flexibility, performance, and community adoption, making them the dominant choices for both industry and research. Keras offers the most accessible entry point for new practitioners due to its simplicity and streamlined workflow. Theano and Caffe, while less commonly used in current development, retain value for specific computational applications. As noted by Mohialden et al. (2024), the optimal framework selection depends on project-specific factors including task complexity, required computational resources, speed requirements, and the level of flexibility needed in model design.

CHAPTER 5: APPLICATIONS OF PYTHON IN AI AND DATA SCIENCE

5.1 Natural Language Processing (NLP)

Python is widely used in Natural Language Processing (NLP) research for a variety of applications, including text mining, sentiment analysis, and language modelling. Tokenization, the tagging of parts of speech, named entity recognition, and machine translation are all possible because of the resources provided by libraries like NLTK, spaCy, and Transformers. Research in NLP contributes to a better knowledge of and ability to analyze data relating to human language (Ranjan, M. et al. 2023).

Sentiment analysis examines the text for positive, negative, or neutral sentiment. It can also consider the meaning of the words as well as the context in which they were spoken. Customers are increasingly expressing their needs, ideas, preferences, and complaints online, which is why sentiment analysis is becoming popular among retailers and technology companies. Businesses can use sentiment analysis and natural language processing to learn what their consumers are saying and how they feel about the products they're selling (Sharma, A. 2021).

Python is widely used in Natural Language Processing (NLP) research for a variety of applications, including text mining, sentiment analysis, and language modelling. Tokenization, the tagging of parts of speech, named entity recognition, and machine translation are all possible because of the resources provided by libraries such as NLTK, spaCy, and Transformers. Research in NLP contributes to a better knowledge of and ability to analyze data relating to human language (Ranjan, M. et al. 2023).

Apple Siri, Amazon Alexa, and Google Duplex are among the most well-known instances of NLP in action, delivering outstanding human-machine interactions in the form of chatbots and virtual assistants. By 2023, there were estimated to be eight billion digital voice assistants in use globally, owing to their widespread popularity. With such a large-scale use, the data generated from these interactions is also immense, and can be used to further research and development of Natural Language Processing across industries such as healthcare, technology, and business (Sharma, A. 2021).

5.2 Computer Vision

Computer vision is the construction of explicit, meaningful descriptions of physical objects from their image. The output of computer vision is a description or an interpretation of structures in a 3D scene.

Python Library for Computer Vision

1. OpenCV: OpenCV is a popular and open-source computer vision library that is focussed on real-time applications. The library has a modular structure and includes several hundreds of computer vision algorithms. OpenCV includes a number of modules including image processing, video analysis, 2D feature framework, object detection, camera calibration, 3D reconstruction.
2. PyTorchCV: It is a PyTorch-based framework for computer vision tasks. The framework is a collection of image classification, segmentation, detection, and pose estimation models. There are a number of implemented models in this framework, including AlexNet, ResNet, ResNeXt, PyramidNet, SparseNet, DRN-C/DRN-D and more (Saabith, A. et al. 2020).

Computer vision is the construction of explicit, meaningful descriptions of physical objects from their image, with the output being a description or interpretation of structures in a three-dimensional scene (Saabith et al., 2020). Python has emerged as one of the most widely used languages for computer vision tasks due to its rich ecosystem of libraries and deep learning frameworks that together enable efficient image processing, object detection, and real-time visual analysis across a wide range of domains (Saabith, A. et al. 2020).

Python supports image processing and object detection through a comprehensive set of dedicated libraries. Image processing using Python has become increasingly prevalent due to the availability of powerful libraries such as OpenCV, Pillow (PIL), and scikit-image, which together enable tasks including image enhancement, restoration, segmentation, feature extraction, and pattern recognition, with applications spanning medical imaging, remote sensing, computer vision, and multimedia (Batta, V. 2024). Among these, OpenCV remains the most widely adopted library for computer vision in Python, providing a modular structure with hundreds of algorithms covering image processing, video analysis, object detection, camera calibration, and 3D reconstruction (Saabith, A. et al. 2020). Complementing OpenCV for more advanced tasks, PyTorchCV provides a PyTorch-based framework for computer vision tasks including image classification, segmentation, detection, and pose estimation, with implementations of well-established models such as AlexNet, ResNet, and ResNeXt (Saabith et al., 2020).

The integration of deep learning within Python's ecosystem has significantly improved the accuracy of facial recognition systems. A real-time facial recognition system built in Python using Convolutional Neural Networks and the VGGFace deep learning architecture, trained through transfer learning on a dataset of 7,500 images of 26 distinct individuals, achieved meaningful improvements in recognition accuracy over conventional machine learning approaches (Singh, A. et al. 2023).

These capabilities have been applied extensively in surveillance systems. (Singh, A. et al. 2023) demonstrated a Python-based real-time system for detecting and identifying individuals in live and recorded surveillance feeds, illustrating the practical deployment of deep learning facial recognition in security contexts. A Python and OpenCV-based live face recognition monitoring system for CCTV-based person tracking in residential and commercial settings further demonstrates how Python's computer vision tools can be directly deployed in real-world surveillance infrastructure (Mishra, S. et al. 2023). Python-implemented facial recognition systems built using OpenCV also hold significant potential in national security contexts, such as identifying individuals with criminal records in surveillance footage, while also enhancing individual security by eliminating password-based vulnerabilities through biometric identification (Tyagi, S. et al. 2023).

Beyond surveillance, Python-based computer vision has found critical applications in healthcare imaging and industrial automation. Deep learning algorithms, predominantly implemented in Python, have

significantly advanced medical image analysis and transformed diagnostic processes, with convolutional neural networks enabling accurate and efficient analysis across imaging modalities and supporting computer-aided diagnosis at a sensitivity comparable to experienced radiologists for conditions such as lung nodule detection in chest CT scans (Matsuzaka, Y. Iyoda, M. 2026). In industrial automation, a real-time image processing framework developed in Python using the YOLOv5 model and integrated directly into a manufacturing production line demonstrated that Python-based deep learning enables seamless communication between machine vision systems and mechanical components, confirming its scalability and applicability in smart manufacturing environments (Ozer, A. Cinar, I. 2025).

5.3 Healthcare and Bioinformatics

Python has emerged as a key language in the fields of medical chemistry, biochemistry, and bioinformatics for tackling complex research challenges. (Ryzhkov, F. et al. 2025) confirm in their review published in *Molecular Diversity* that Python is widely used to solve tasks such as drug discovery, high-throughput and virtual screening, protein and genome analysis, and predicting drug efficacy - presenting a comprehensive list of Python tools, libraries, and ready-made programs that serve as essential resources for scientists seeking to apply automation and optimization to routine tasks in medical chemistry and bioinformatics.

In the field of bioinformatics, the Python library PyBioMed has made a significant contribution by providing a comprehensive platform for molecular representation and analysis.PyBioMed is a feature-rich and highly customizable Python library capable of calculating 775 chemical descriptors and 19 kinds of chemical fingerprints, 9,920 protein descriptors based on protein sequences, and more than 6,000 DNA descriptors from nucleotide sequences that enables researchers to build full data analysis pipelines from molecular data acquisition and pretreatment through to the construction of machine learning models. The library has demonstrated direct applicability in drug discovery, covering applications such as Caco-2 cell permeability, aqueous solubility, drug-target interaction data, protein subcellular location, and nucleosome positioning in genomes (Dong, J.et al. 2018).

Python-based AI models have also demonstrated strong capabilities in predicting diseases from large-scale healthcare datasets. In a study published in *Scientific Reports*, collected sample datasets on 5,145 cases, including 326,686 laboratory test results, and investigated a total of 39 specific diseases based on ICD-10 codes. These datasets were used to construct LightGBM and XGBoost machine learning models and a DNN model built using Python's TensorFlow library (Park et al., 2021). The optimized ensemble model achieved an F1-score of 81% and a prediction accuracy of 92% for the five most common diseases (Park, D. et al. 2021).

5.4 AI in Education

Artificial intelligence has increasingly been adopted in educational settings to personalize learning experiences, support intelligent tutoring, automate assessment, and enhance accessibility (Kamalov, F. et al. 2023) Python, with its accessible syntax and rich ecosystem of AI and NLP libraries, has been widely adopted as a foundational tool for developing and implementing such systems, serving as both a teaching language and a development platform for AI-driven educational applications (Akhtar, T. Ranjan, A. 2026). Python's versatility and simplicity have made it the language of choice for both industry professionals and educators (Müller, 2025). The study reveals that Python is not only the preferred language for AI development but also an effective teaching tool in software engineering education (Müller, 2025). Python's syntax is intuitive, making it an ideal teaching tool in AI and software engineering courses, even for novice programmers (Müller, 2025). Python's simplicity, coupled with its powerful AI libraries, makes it an ideal platform for introducing students to AI concepts early in their academic careers (Müller, 2025). At the

school level, software tools such as Scratch and Python, which help to develop the computational thinking of AI algorithms, can be introduced to both primary and secondary schools (Yim, I. Su, J. 2024). The data suggests that Python-based AI tools significantly improve student engagement and learning outcomes, particularly in introductory programming courses (Müller, 2025).

Zabala and Narman (2024) present the development of an AI-based Python programming education system that integrates a Chatbot for student assistance, an automated grading system for feedback, and an entrance exam feature that suggests chapters and sections for review (Zabala & Narman, 2024). Users can take quizzes, receive grades, obtain personalized feedback, and get course recommendations (Zabala & Narman, 2024). The grading system achieved 28/30 consistency with its output while the course recommendation system achieved 26/30 consistency with its outputs (Zabala & Narman, 2024). While current AI systems are not yet capable of supplanting human educators, the continuous evolution of APIs holds considerable promise for future applications (Zabala & Narman, 2024).

5.5 Business and Finance Applications

Python has emerged as a preferred language for implementing predictive risk assessment due to its versatility, open-source nature, and robust ecosystem of libraries. Key libraries such as Pandas and NumPy enable efficient data manipulation and numerical computation, while scikit-learn and TensorFlow support machine learning and deep learning models . Python also facilitates integration with big data platforms and cloud-based solutions, allowing financial institutions to process large-scale transaction and market data for real-time risk analysis. Studies demonstrate the use of Python for credit risk assessment, market risk forecasting, liquidity risk monitoring, and operational risk evaluation (Agu, M. et al. 2025).

Predictive analytics plays a pivotal role in data-driven decision-making across various industries, and Python, with its extensive library ecosystem, provides powerful tools for forecasting, anomaly detection, and risk assessment. Leveraging libraries such as prophet for time series forecasting, scikit-learn for anomaly detection, and classification algorithms like Logistic Regression for risk assessment, Python empowers organizations to extract actionable insights from data, mitigate risks, and drive strategic decision-making (Basireddy, M. 2023).

(Condori, P. et al. 2026) evaluated several supervised learning models including XGBoost, neural networks, Random Forest, decision trees, and logistic regression on an imbalanced dataset of credit card transactions. The analysis was implemented in Python (version 3.12.13) using scikit-learn, XGBoost, TensorFlow, and imbalanced-learn. The main methodological contribution of the study lies in integrating machine learning techniques into an economic evaluation framework, enabling model selection based not only on predictive performance but also on financial impact, which reduces the gap between academic research and its practical application in financial fraud management (Condori, P. et al. 2026).

With the increasing application of Python and big data technology, companies now have access to a broader range of technical solutions for processing, storing, and analyzing financial data, and these advancements enable faster extraction of valuable insights, providing robust support for decision-making. Results highlight the significant advantages of Python and its libraries such as Pandas, NumPy, and scikit-learn in enhancing analysis efficiency, accuracy, and decision support. Python's applications in financial analysis include data preprocessing, financial statement analysis, predictive modeling, and risk management. Looking ahead, as technology evolves, Python and big data will play an increasingly vital role in financial analysis, empowering businesses to tackle complex market dynamics and boost competitiveness (Yan, Q. 2025).

CHAPTER 7: ADVANTAGES OF PYTHON IN AI DEVELOPMENT

7.1 Rapid Prototyping

Python facilitates rapid prototyping and iteration so that AI researchers can try new models with ease (Talati, D. 2021). The dynamic typing and interpreted nature of Python enable researchers to quickly develop prototypes and test ideas, which plays a crucial role in the iterative AI and ML development process. This feature facilitates rapid experimentation and hypothesis validation, significantly reducing the time required to gain insights into research projects. Additionally, tools like Jupyter Notebooks enhance this workflow by offering an interactive platform for iterative development and real-time feedback (Fatoba, et al. 2025)

(Akhtar, T. Ranjan, A. 2026) states in their paper that python is an excellent choice for rapid prototyping in comparison to other programming languages. With its high-level data structures, object-oriented approach, and elegant syntax, Python allows for efficient scripting and rapid application development across different platforms (Elhalid, O. et al. 2023).

Beyond speeding up individual experiments, Python's rapid prototyping capability allows AI teams to move quickly from a rough idea to a working proof of concept without committing significant development resources upfront. This lowers the barrier to testing unconventional approaches, since researchers can discard a failed prototype and pivot to a new architecture within hours rather than days, a flexibility that compiled, statically typed languages generally cannot match (Fatoba, et al. 2025). The ability to write and execute code in small, testable chunks also means errors surface early in the development cycle, rather than after an entire system has been built out.

This same quality has made Python the default language for AI research communities that rely heavily on shared code and reproducible experiments. Because researchers can quickly adapt existing prototypes to new datasets or problem statements, Python has become the common language through which academic findings move into practical, testable implementations (Akhtar, T. Ranjan, A. 2026).

7.2 Large Standard Libraries

The language comes with a large number of built-in methods as a part of standard library Comes with a large number of libraries as a part of standard installer. (Cutting, V. Stephen, N. 2021). Python is that it possesses a rich set of in-built libraries and frameworks that make AI development easy. Libraries like NumPy, SciPy, Matplotlib, NLTK, and SimpleAI offer the basic tools for data processing activities, machine learning activities, and natural language processing activities. These libraries greatly add to the functionality of Python, which makes it a perfect language for AI development.

Libraries make AI development easier by offering pre-existing functions and models, lowering the amount of coding that must be done manually. Python's strong AI ecosystem makes it easy to test and deploy machine learning solutions. The other significant benefit of Python is its comprehensive number of built-in libraries, especially for AI and machine learning applications. These libraries offer pre-coded scripts for sophisticated functionality, saving time and effort in creating intelligent AI-based applications. Python also provides a huge collection of AI-oriented libraries like TensorFlow, PyTorch, Scikit-learn, and Keras, which have pre-implemented functions for machine learning, neural networks, and natural language processing (Talati, D. 2021).

Beyond the well-known machine learning libraries, Python's standard library itself, covering tasks like file handling, data serialization, and basic mathematical operations, reduces the amount of boilerplate code developers need to write before they can even begin building an AI model. This built-in functionality

means a developer does not need to rely on third-party tools for many foundational tasks, which shortens the setup time for new projects considerably (Cutting, V. Stephen, N. 2021).

The open-source nature of Python's broader library ecosystem also means these libraries are under constant, community-driven development, with contributors regularly releasing updates that improve performance, add new features, and fix bugs. This continuous improvement cycle allows AI developers to benefit from cutting-edge techniques almost as soon as they are published, without needing to build the underlying implementation themselves (Talati, D. 2021).

7.3 Versatility Across Domains

Python was a general-purpose language that could accommodate a range of paradigms such as procedural, objectoriented, and functional programming. From its discovery, it has continued developing and growing and is now one of the world's most applied languages. Python has grown into a programming language with diversity in its use in various areas, including software development, web development, desktop GUI Applications, education and research, and more (Talati, D. 2021).

It is widely adopted in industries that include startups, academia, and large tech compared to other programming languages (Akhtar, T. Ranjan, A. 2026). Python is the most versatile language in the programming world, and it is applicable in almost every domain of software development (Saabith et al., 2020).

This versatility extends into how Python integrates with other systems and languages, since it can be embedded within larger applications or used to glue together components written in faster, lower-level languages like C or C++. This interoperability allows AI developers to prototype in Python while still taking advantage of the performance benefits of compiled code where it matters most, giving teams the flexibility to optimize only the parts of a system that actually require it (Elhalid, O. et al. 2023).

Because the same language can be used across research, backend development, and deployment, organizations often find it easier to move an AI model from an experimental notebook into a production environment without needing to rewrite it in a different language altogether. This consistency across the development pipeline reduces both the technical overhead and the communication gap between research teams and engineering teams (Saabith et al., 2020).

7.4 Accessibility for Beginners

Python is easy to learn as compared to other programming languages. Its syntax is straightforward and much the same as the English language. There is no use of the semicolon or curly-bracket, the indentation defines the code block. It is the recommended programming language for beginners (Saabith et al., 2020). Python today is considered as one of the easiest programming languages to begin with because of its super user-friendly coding style (Cutting, V. Stephen, N. 2021). Python seems to be a choice more and more common at high school for students taking computing classes (Pellet, J. et al. 2019).

Its simplicity and efficiency make it an ideal choice for beginners entering the world of programming. It has gained recognition as a beginner-friendly language that can be used for tasks such as web development, data analysis, machine learning, and artificial intelligence. Python's easy-to-learn nature enables beginners to quickly start programming while being supported by its free availability across multiple platforms. Platforms like Code Academy, Tutorial Point , official documentation (Python.org), Udemy and Coursera are great sources for beginners learning python (Elhalid, O. et al. 2023).

Python's beginner-friendly design also lowers the barrier to entry for non-programmers working in adjacent fields, such as biologists, economists, and social scientists, who increasingly rely on AI tools for their own research but may not have a formal computer science background. Its readable syntax allows

these users to understand and modify existing code with relatively little training, extending the practical reach of AI methods well beyond traditional software engineering circles (Pellet, J. et al. 2019).

This accessibility is further reinforced by the sheer size and activity of Python's beginner-focused online community, where new learners can find troubleshooting help, code reviews, and project ideas with minimal friction. The combination of an active support community and free, high-quality learning resources has made Python one of the fastest languages for a complete beginner to become productive in, particularly within AI and data-focused coursework (Elhalid, O. et al. 2023).

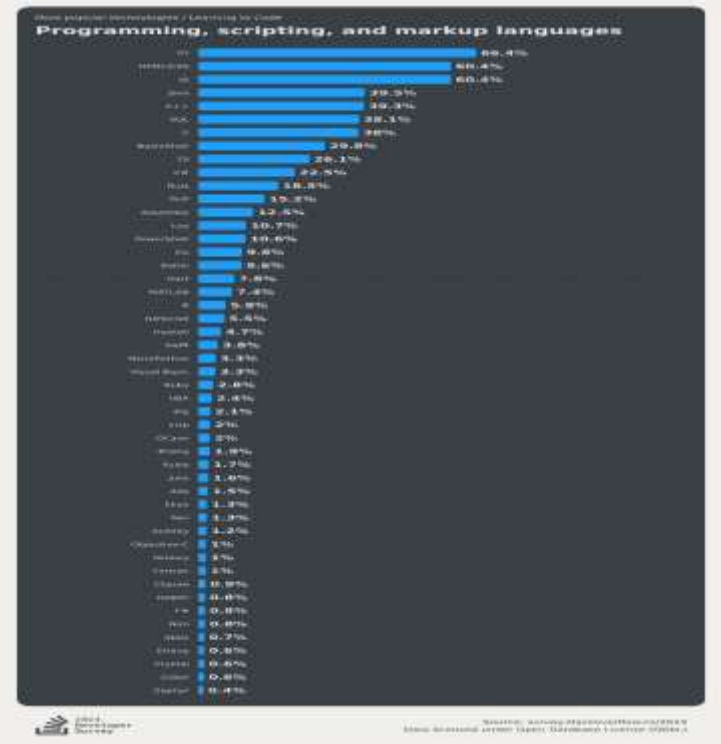

Figure 3

Programming, Scripting, and Markup Languages Used by Individuals Learning to Code

Note. Reprinted from *2024 Developer Survey: Technology Section*, by Stack Overflow, 2024 (stackoverflow.co). Licensed under the Open Database License (ODbL).

CHAPTER 8: CHALLENGES AND LIMITATIONS

8.1 Performance Limitations

Python code is executed line by line, which causes it to slow down (Rana, Y. 2019). Since Python is an interpreted language, it generally performs slower than compiled languages such as C or C++. In data intensive and computation-heavy AI applications, this can become a concern. From a teaching standpoint, students often observe longer execution times when working with large datasets or complex models. While optimized libraries and parallel processing techniques can reduce this issue, Python may still not be the first choice when raw performance and low-level optimization are critical (Akhtar, T. Ranjan, A. 2026). Python is dynamically typed and interpreted, hence some run-time processing. Any variable data types are referenced during run-time, adding extra computational burden (Talati, D. 2021).

This performance gap becomes especially noticeable in AI applications that involve heavy numerical computation, such as training deep neural networks from scratch without relying on optimized backend

libraries. In such cases, raw Python loops can take significantly longer to execute than equivalent code written in a compiled language, which is why most production-grade AI frameworks delegate their heaviest computations to underlying C or C++ implementations rather than running them in pure Python (Rana, Y. 2019).

For students and early-career developers, this limitation often becomes apparent only once they move beyond small classroom datasets into larger, real-world problems. What runs instantly on a sample dataset of a few hundred rows can take considerably longer once scaled to millions of records, a gap that highlights the importance of understanding when Python's convenience is worth its performance tradeoff and when a different tool may be more appropriate (Talati, D. 2021).

8.2 Memory Consumption

Python was never a great option for tasks that are memory intensive. Mainly, because of the flexibility of its data-types, its memory consumption is large (Nagpal, A. Gabrani, G. 2019). Large memory consumption makes python less favorable for mobile devices (Cutting, V. Stephen, N. 2021). For any memory intensive tasks, Python is not a good choice. Python’s memory consumption is also high due to the flexibility of the data types (Rana, Y. 2019).

Python programs tend to consume more memory compared to some other programming languages. This is mainly due to its dynamic typing and highlevel data structures. In Data Science and AI applications, where large datasets and multiple models are handled simultaneously, memory usage can become a limitation. In academic labs, students sometimes face hardware constraints that highlight this issue, especially when working with deep learning models or high-dimensional data (Akhtar, T. Ranjan, A. 2026).

This memory overhead is partly a consequence of how Python stores even simple data types internally, wrapping basic values like integers and floating-point numbers in full Python objects rather than storing them as raw bytes the way lower-level languages do. While this design contributes to Python's flexibility and ease of use, it also means that even simple data structures consume noticeably more memory than their equivalents in a language like C++ (Nagpal, A. Gabrani, G. 2019).

In practical AI development, this becomes a real constraint when working with large-scale datasets or high-resolution image and video data, where memory usage can quickly exceed the limits of standard consumer hardware. Developers working under these constraints often need to rely on techniques such as batch processing or data generators to avoid loading entire datasets into memory at once, adding an extra layer of complexity that would not be necessary in a more memory-efficient language (Cutting, V. Stephen, N. 2021).

8.3 Security Concerns

Security Concerns Python’s popularity in web applications and data processing exposes it to security vulnerabilities. Ensuring secure coding practices, protecting against common threats, and maintaining up-to-date libraries are crucial to safeguard applications. Python’s flexibility and ease of scripting, while advantageous for rapid development, can inadvertently introduce security risks if best practices are not rigorously followed (Shakrum, A. 2025).

Security is another area where Python requires careful handling. While Python itself provides standard security features, applications developed using Python can be vulnerable if best practices are not followed. In AI and Data Science projects that involve sensitive data, such as healthcare or financial information, improper data handling or insecure libraries can pose risks. From an educational perspective, this

limitation emphasizes the need to teach secure coding practices alongside technical skills (Akhtar, T. Ranjan, A. 2026).

Python's package ecosystem, while a major strength for development speed, also introduces a security surface that is difficult to fully audit, since applications often rely on dozens or even hundreds of third-party libraries, each maintained independently and each a potential entry point for vulnerabilities. A single outdated or poorly maintained dependency can expose an otherwise well-built AI application to risks that have nothing to do with the developer's own code (Shakrum, A. 2025).

This risk is compounded in AI-specific contexts, where models are sometimes trained on or deployed using data pulled directly from external sources, including scraped web content or user-submitted inputs. Without careful validation, such data can be used to introduce malicious inputs designed to manipulate a model's behavior or exploit weaknesses in the surrounding application code, making secure data handling as important a skill for AI developers as model design itself (Akhtar, T. Ranjan, A. 2026).

8.4 Dependency and Package Issues

Python's extensive ecosystem, while a strength, can also introduce dependency and version management challenges. Conflicting library versions and inconsistent package behavior across environments may complicate development, deployment, and maintenance. Tools such as pip, virtualenv, and Anaconda mitigate these issues, but dependency management remains a common pain point, particularly in large-scale projects (Shakrum, A. 2025).

The major reason for its under supported libraries is that most of these libraries are made by volunteers that have very little time to spend on supporting and documenting a library. It is better to see if a library is supported actively before we use it for making an application or else we have to spend time debugging the code (Nagpal, A. Gabrani, G. 2019).

Frequent updates to these packages often cause their latest versions to have breaking changes for applications using them. Large Python applications risk their historical builds becoming unreproducible due to the widespread usage of Python dependencies, and the lack of uniform practices for dependency version specification (Mukherjee, S, et al. 2021).

CHAPTER 9: COMPARATIVE ANALYSIS WITH OTHER LANGUAGES

9.1 Python vs C++

Python has long been favored for its ease of use and strong ecosystem of AI libraries, while C++ remains preferred in performance-critical applications due to its low-level memory management and runtime efficiency (Brown, L. 2025). The syntax used in Python is simple and this simplicity inspired beginners to learn and comprehend different programming concepts. Python also has a great collection of modules that make it simplistic to perform tasks such as data analysis, web scraping, and machine learning.

C++ is the powerful programmable language that is widely applied in system programming, games and other programs which require high performance. It is more complex than Python and can take longer to learn, but it provides a better understanding of computer architecture, memory management, and other low-level concepts (Asghar, J. et al. 2025). Python supports rapid prototyping in application development, as shown by tools that let developers build prototype interfaces inside a Jupyter Notebook and then carry that same interface forward into a complete, deployable web-based application (Christensen, S. et al. 2022).

This tradeoff between ease of use and raw performance often plays out directly within the same AI project, where developers write the overall logic and experimentation pipeline in Python while relying on C++-

based backends, such as those used in TensorFlow or PyTorch, to handle the computationally intensive operations under the hood. This hybrid approach allows development teams to benefit from Python's readability and fast iteration speed without fully sacrificing the performance that C++ provides at the execution level (Brown, L. 2025).

For students transitioning between the two languages, the shift in mindset can be considerable, since C++ requires explicit management of memory allocation and deallocation, concepts that Python abstracts away entirely through automatic garbage collection. While this makes Python considerably more forgiving for beginners, learning C++ alongside Python is often recommended in computer science curricula specifically because it forces students to understand what is happening beneath Python's simplified surface, a foundation that can make them more effective programmers even when they return to writing in Python (Asghar, J. et al. 2025).

9.2 Python vs Java

Java is somewhat more complex than Python in structure, it provides a better understanding of memory management and is more secure. Python is short, simple, and easy. A novice can easily understand a Python program because it is written in simple English. In Python since indentation is compulsory, it makes the code more readable. However, this is not the case for Java as there is no effect for indentation and the whole program can be written in one line to make it look short. The use of semicolon which indicates the end of the line in Java is sometimes overlooked which leads to a major compilation error Python being dynamically typed leads to longer execution time as the variable type is checked during run time whereas Java is statically typed so the exact datatype for variables is known during compilation leading to faster execution than Python (Khoirom, S. et al. 2020).

For big AI projects, Java's strict typed and direct memory handle work best, while Python's broad-rangeness is great for exploring ideas fast Reusability of software and code is encouraged by object-oriented programming frameworks, as the extensive collection of class libraries found in the Java language attests to. Network connectivity, multimedia capabilities, windowing, and graphical user interface programming are all supported by the Java foundation class libraries (Object-Oriented Programming and Java, n.d.). When combined, they show off the useful and effective work done with Java.

Python offers a variety of approaches that make programming on both local and big scales simple and understandable. Its design approach puts a strong emphasis on effectively readable code, particularly when employing large amounts of whitespace. Most importantly, Python's large, feature-rich standard library makes it simple to combine imperative, functional, procedural, and object-oriented programming paradigms. Python's exceptionally simple syntax is among its most appealing features. Python was created to be user-friendly, enabling developers to convey their ideas in a manner that mimics a normal language, in contrast to other programming languages with intricate symbols and cryptic syntax. Java's syntax is somewhat long. Therefore, it could be quite challenging for beginners (Albatera, J. et al. 2024).

9.3 Python vs R

R is software designed to run statistical analyses and output graphics. It can run on virtually any operating system, and is open source When beginning to use R the programmer reads their data into a data frame, uses a built-in model by using R’s formula language, and then can later look back at the model summary output. When getting started with Python, the programmer has many more choices to make. These can include choosing how they would like to read their data, what kind of structure they should use to store their data in, what machine learning package they should use, and what type of objects does the package even allow to be in the input (Ozgur, C. et al. 2017).

Python, known for its simplicity and versatility, has gained significant popularity due to its ability to integrate seamlessly with various technologies. R, on the other hand, was designed specifically for statistical computing and data visualization, making it a preferred language among statisticians and researchers. Python excels in scalability, integration with other technologies, and machine learning capabilities, R remains the preferred choice for statistical analysis and visualization.

Python has gained widespread adoption due to its general-purpose nature and extensive support for artificial intelligence and machine learning. Studies suggest that Python's libraries, such as Pandas, NumPy, Scikit-learn, TensorFlow, and PyTorch, make it a preferred choice for developing machine learning models and handling large datasets. Additionally, Python's ability to integrate with web development, cloud computing, and big data frameworks has further solidified its dominance in the industry.

Conversely, R has been widely acknowledged for its superior statistical analysis capabilities. Researchers have highlighted R's specialized libraries, such as ggplot2, dplyr, and caret, which enable advanced statistical modeling and visualization. R's built-in support for statistical tests, hypothesis testing, and time-series analysis makes it an essential tool for academic research and statistical computing. Many studies also suggest that R's visualization capabilities surpass those of Python, as it offers more detailed and customizable plotting options. Machine learning is an area where Python outperforms R.

The vast majority of machine learning frameworks are developed in Python, making it the preferred language for AI and deep learning application Visualization is another key area of comparison. R is often preferred for its advanced and highly customizable plotting capabilities. ggplot2 allows users to create complex visualizations with minimal code, making it ideal for academic and research applications.

Python, while offering visualization libraries such as Matplotlib and Seaborn, often requires more code to achieve similar results. However, recent developments in visualization libraries, such as Plotly, have enhanced Python's capabilities in this domain. The performance evaluation of both languages indicates that Python is generally faster in executing machine learning tasks due to optimized libraries and better memory management. However, for statistical analysis, R demonstrates superior efficiency, particularly when performing complex statistical computations (Abiola, B. 2024).

9.4 Toolkit Comparison

Performance plays a crucial role in AI/ML applications, especially for computationally intensive tasks such as deep learning and real-time inference. Languages like C++ and Julia offer high execution speed due to their low-level memory management and compiled nature. Python, despite being slower in raw execution, compensates with optimized libraries like NumPy and TensorFlow, which leverage C++ backends. Java provides stable performance, particularly for enterprise applications, while R is relatively slower but efficient for statistical computations.

A strong library ecosystem accelerates AI/ML development by providing pre-built tools for common tasks. Python leads in this area with frameworks like TensorFlow, PyTorch, and Scikit-learn. R is well-equipped for statistical modeling with libraries like Caret and mlr. C++ offers performance oriented libraries like OpenCV and TensorRT, whereas Java has enterprise-focused tools such as Weka and Deeplearning4j. Julia, though emerging, is gaining traction with Flux.jl and MLJ.jl

Scalability is crucial for deploying AI/ML models in production environments. Java and C++ are widely used in large-scale, high-performance applications due to their speed and robustness. Python, while dominant in research, requires additional optimizations for deployment, such as TensorFlow Serving or ONNX. Julia offers performance benefits but lacks the extensive deployment infrastructure of other

languages. R is primarily used for research and statistical analysis rather than large-scale production systems (Singh, S. 2025).

Chapter 10

10.1 Role in Emerging Technologies

Python has established itself as one of the go-to language options for emerging fields like quantum computing, edge AI, and federated learning because of its versatility and rich ecosystem of libraries.The convergence of AI/ML with big data and cloud computing is another emerging trend where Python is at the forefront. Python's integration with big data frameworks like Apache Spark (via PySpark) and its compatibility with cloud services such as AWS, Google Cloud, and Azure, allow for scalable AI/ML solutions RL is a significant focus in machine learning research and is increasingly applied in fields such as robotics, autonomous systems, and gaming. Python is the primary language for conducting RL experiments and implementations, largely due to its compatibility with RL libraries such as OpenAI Gym, Stable Baselines, and RLlib, which are driving progress in the field (Fatoba, et al. 2025).

10.2 Growth in Deep Learning and Automation

Python, a prominent and celebrated programming language, has emerged as a pivotal choice for deep learning study and development. Python has become a leading language in deep learning due to its simplicity and the vast array of libraries available for developers and researchers (Mohialden, Y. et al. 2024).

Python-based frameworks have largely powered the explosion in deep learning research. TensorFlow and PyTorch provide the tools necessary to build and train complex neural networks, while higher-level APIs like Keras make implementation more accessible (Mantrala, 2026).

These libraries offer tools for developing and training neural networks and support advanced architectures like transformers, GANs (Generative Adversarial Networks), and reinforcement learning models" (Fatoba, et al. 2025). Beyond modelling, Python’s role in automation from data pipelines to robotic process automation (RPA) and test automation continues to expand (Akhtar, T. Ranjan, A. 2026).

10.3 Python in AI Education

Python is playing a pivotal role in making artificial intelligence (AI) and machine learning (ML) more accessible to a wider audience. With user-friendly libraries such as Scikit-learn and the growing availability of educational resources and online platforms, the entry barriers for understanding and applying ML have significantly decreased (Fatoba, et al. 2025).

In computer science education, Python was commonly introduced as a first programming language due to its clarity and ease of implementation. Python has become an essential part of modern computing education, serving as a bridge between foundational programming concepts and advanced intelligent systems. Python is increasingly applied in the education sector for academic analytics and intelligent learning systems (Akhtar, T. Ranjan, A. 2026).

Python leads in AI education because of its simplicity and readability. Its natural lan-guage–like syntax makes it accessible to both students and educators with limited programming experience.Python leads in AI education because of its simplicity and readability. Its natural language–like syntax makes it accessible to both students and educators with limited programming experience.

A wide range of libraries, frameworks, and development tools has been shown to contribute to teaching and learning innovation. Core machine learning tools such as Scikit-learn and TensorFlow provide the computational foundation, while education-oriented tools like EduBlocks and PyTutor extend Python’s

applicability to classroom settings. Visualization dashboards (such as Streamlit, Taipy) and natural language processing frameworks (like LangChain, OpenAI, spaCy, and NLTK) support interactive learning environments, auto-mated feedback, and communication of results.

Domain-specific libraries such as BioPython, PsychoPy, and Librosa illustrate how Py-thon's ecosystem can serve specialized educational and research purposes. Collectively, these resources enable personalized learning, automated grading, adaptive testing, student analytics, emotional support, and intelligent educational dashboards. Teachers and learners alike can benefit from Python's accessibility, readability, and community support, which foster practical skills applicable to modern educational challenges (Papadimitriou, A. 2025).

10.4 Sustainability and Long-Term Relevance

Python has a very vast ecosystem of libraries such as Scikit-learn, TensorFlow, Keras, Numpy, Pytorch, and Cython which are powerful tools for building and deploying machine learning models, making Python indispensable for cutting-edge technology development (Fatoba, et al. 2025). AI and ML continue to advance, it would leverage Python's features to achieve its expansion; it would shape the emerging trends and play a crucial role in defining the future trajectory of these domains (Fatoba, et al. 2025).

Python has a large community and many machine learning resources. This community support provides access to a wide range of courses, events and third-party tools. Python continues to be a leader in AI libraries and tools, providing extensive support for a wide variety of AI applications. Python's simplicity, readability, and strong community support solidify its position as the preferred choice for AI projects. Its integration with big data technologies and role in explainable AI tools further emphasize its versatility. (Patil, H. et al. 2024).

References

1. Alhumaidi, N. H., Dermawan, D., Kamaruzaman, H. F., & Alotaiq, N. (2025). The use of machine learning for analyzing real-world data in disease prediction and management: Systematic review. *JMIR Medical Informatics, 13*, e68898. https://doi.org/10.2196/68898
2. Bogdanchikov, A., Zhaparov, M., & Suliyev, R. (2013). Python to learn programming. *Journal of Physics: Conference Series, 423*, 012027. https://doi.org/10.1088/1742-6596/423/1/012027
3. Dong, J., Yao, Z.-J., Zhang, L., Luo, F., Lin, Q., Lu, A.-P., & Chen, A. F. (2018). PyBioMed: A Python library for various molecular representations of chemicals, proteins and DNAs and their interactions. *Journal of Cheminformatics, 10*, Article 16. https://doi.org/10.1186/s13321-018-0270-2
4. Matsuzaka, Y., & Iyoda, M. (2026). Applications, image analysis, and interpretation of computer vision in medical imaging. *Frontiers in Radiology, 5*, 1733003. https://doi.org/10.3389/fradi.2025.1733003
5. Mukherjee, S., Almanza, A., & Rubio-González, C. (2021). Fixing dependency errors for Python build reproducibility. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp. 439–451). https://doi.org/10.1145/3460319.3464797
6. Ozer, A. S., & Cinar, I. (2025). Real-time and fully automated robotic stacking system with deep learning-based visual perception. *Sensors, 25*(22), 6960. https://doi.org/10.3390/s25226960
7. Singh, A., Bhatt, S., Nayak, V., & Shah, M. (2023). Automation of surveillance systems using deep learning and facial recognition. *International Journal of System Assurance Engineering and Management, 14*(Suppl. 1), 236–245. https://doi.org/10.1007/s13198-022-01844-6

8. Park, D. J., Park, M. W., Lee, H., Kim, Y.-J., Kim, Y., & Park, Y. H. (2021). Development of machine learning model for diagnostic disease prediction based on laboratory tests. *Scientific Reports, 11*, Article 7567. https://doi.org/10.1038/s41598-021-87171-5
9. Ozgur, C., Colliau, T., Rogers, G., Hughes, Z., & Myles, L. (2018). MatLab vs. Python vs. R. *Journal of Data Science, 16*(1), 17–42. https://doi.org/10.6339/JDS.201801_16(1).0002
10. Javaid, M., Haleem, A., Singh, R. P., & Ahmed, M. (2024). Computer vision to enhance healthcare domain: An overview of features, implementation, and opportunities. *Intelligent Pharmacy, 2*(6), 792–803. https://doi.org/10.1016/j.ipha.2024.05.007
11. Olveres, J., González, G., Torres, F., Moreno-Tagle, J. C., Carbajal-Degante, E., Valencia-Rodríguez, A., et al. (2021). What is new in computer vision and artificial intelligence in medical image analysis applications. *Quantitative Imaging in Medicine and Surgery, 11*(8), 3830–3853. https://doi.org/10.21037/qims-20-1151
12. Elyan, E., Vuttipittayamongkol, P., Johnston, P., Martin, K., McPherson, K., & Moreno-García, C. F. (2022). Computer vision and machine learning for medical image analysis: Recent advances, challenges, and way forward. *Artificial Intelligence Surgery, 2*, 22–45. https://doi.org/10.20517/ais.2021.15
13. Fernandez-Vega, M., Alfaro-Viquez, D., Zamora-Hernandez, M., Garcia-Rodriguez, J., & Azorin-Lopez, J. (2025). Transforming robots into cobots: A sustainable approach to industrial automation. *Electronics, 14*, 2275. https://doi.org/10.3390/electronics14112275
14. Choi, Y. W., Lee, J., Lee, Y., Lee, S., Jeong, W., Lim, D. Y., & Lee, S. W. (2025). A vision-guided adaptive and optimized robotic fabric gripping system for garment manufacturing automation. *Robotics and Computer-Integrated Manufacturing, 92*, 102874. https://doi.org/10.1016/j.rcim.2024.102874
15. Dhanda, M., Rogers, B. A., Hall, S., Dekoninck, E., & Dhokia, V. (2025). Reviewing human-robot collaboration in manufacturing: Opportunities and challenges in the context of Industry 5.0. *Robotics and Computer-Integrated Manufacturing, 93*, 102937. https://doi.org/10.1016/j.rcim.2024.102937