

THE INTERSECTION OF PROGRAMMING AND DATA PROTECTION: SECURE DEVELOPMENT WITH JAVA AND PYTHON

UDC: 004.43:004.056.5

DOI: <https://doi.org/10.53486/tids2025.05>

ȚURCAN AURELIA

Academy of Economic Studies of Moldova

cce.turcan@gmail.com

ORCID ID: 0009-0003-2512-2231

CERBU OLGA

Moldova State University

olga.cerbu@gmail.com

ORCID ID: 0000-0002-6278-7115

Abstract. In the context of accelerated digitization and strict data protection regulations, programming can no longer be analyzed exclusively from a technical perspective, but also as a tool for legal and ethical compliance. The article explores the intersection between programming and data protection, focusing on the role of Java and Python languages in secure software application development. The particularities of the two languages in implementing the principles of "privacy by design" and in complying with international regulations are highlighted. The comparative analysis emphasizes the advantages of Java in enterprise applications, where robustness and security are critical, and the advantages of Python in data-driven projects, where flexibility and a rich ecosystem of libraries facilitate data processing and protection. At the same time, the paper presents examples of tools and applications developed in both languages.

The conclusion emphasizes the need for convergence between technology and regulations to ensure both innovation and respect for fundamental rights regarding data privacy and security.

Keywords: Java, Python, cybersecurity, data protection, privacy by design, secure programming.

JEL Classification: C88, L86, O33

INTRODUCTION

The profound transformations brought about by global digitalization have led to a paradigm shift in the way software applications are developed. While in the early stages of programming the emphasis was placed on the *code first* principle—meaning the fastest possible implementation of technical functionalities—today the dominant paradigm is *privacy by design*. This concept, initially introduced by Ann Cavoukian and later enshrined in European and international legislation, entails the integration of data protection and privacy principles from the very design phase of applications. In such a context, programming languages are no longer merely technical tools but become vehicles for implementing legal and ethical principles [1].

Major international regulations—such as the European Union's General Data Protection Regulation (GDPR) [4], the European NIS2 Directive on the security of networks and information systems [5], or the California Consumer Privacy Act (CCPA) [19]—impose strict standards regarding the protection of personal data, cybersecurity, and the transparency of information processing. In practice, the application of these standards is achieved through a combination of organizational policies and technical solutions, with programming languages used in application development playing a crucial role.

In this regard, **Java and Python** occupy a central position. Today, they are among the most widely used languages in enterprise and data-driven application development, covering a broad spectrum of needs: from banking applications, distributed systems, and e-commerce platforms to applications based on artificial intelligence, big data analytics, and machine learning [23]. Java, with its robustness and object-oriented nature, is preferred in projects that require reliability, scalability, and compliance with strict security standards. **Python**, with its simplicity and flexibility, is favored in the rapid development of prototypes and data-centered applications, where the analysis, processing, and interpretation of information are essential [6].

Thus, the transition from *code first* to *privacy by design* is closely linked to the adoption of programming languages that allow not only the implementation of functionalities, but also the guarantee of fundamental principles such as:

- ✓ **security by default**, through the use of well-established libraries and frameworks (e.g., Spring Security for Java, Django and Flask with security extensions for Python);
- ✓ **transparency and accountability of coding processes**, through audit and logging tools integrated into the Java and Python ecosystems;
- ✓ **adaptability to regulations**—through the development of modules and APIs that implement GDPR requirements (e.g., the right to be forgotten, data portability), as well as compliance with regulations such as NIS2 or CCPA [4,5].

That is why the choice of Java and Python as the focus of this research is not accidental, as they represent the meeting point between technological demands and the legal and ethical imperatives of the information age. The purpose of this research is to analyze the role that the Java and Python programming languages play in contemporary information society, with emphasis on their technological, educational, and economic impact. The importance of the subject derives from the fact that both languages have become fundamental tools of digitalization, being used both in academia and in industry, in fields such as enterprise application development, data science, artificial intelligence, and cybersecurity. The practical relevance of this research lies in identifying the ways in which Java and Python contribute to the development of digital skills and the stimulation of technological innovation. On the theoretical level, the comparative analysis provides a framework for understanding the evolution of programming paradigms and their convergence in a globalized and digitalized society. Studying the role of these programming languages not only reflects the current level of science and technology but also offers benchmarks for the sustainable development of the digital ecosystem in accordance with the fundamental principles of data protection and cybersecurity.

2. Comparative Analysis of the Java and Python Languages

In the Tiobe global ranking [20], the Python programming language holds the first place, with a market share of approximately 13.97% in January 2024. An increase of nearly 12.7 times in 20 years is truly impressive. Java is also in the TOP 5 [22].

Next, we decided to compare these popular languages.

Java, officially launched in 1995 by Sun Microsystems, became established through the principle of “*Write Once, Run Anywhere*”, offering portability and robustness. According to Statista data, in 2023, it was used by over 30% of developers worldwide [9].

By contrast, Python, created by Guido van Rossum in 1991, focused on simplicity, readability, and a user-friendly syntax, which accelerated its popularity. In 2023, nearly 50% of developers

worldwide used Python, making it one of the most popular languages in the industry. Its appeal lies in simplicity, readability, and an extensive ecosystem of libraries [9].

Both languages have evolved significantly, adapting to the needs of the information society: Java toward complex and scalable applications (banking systems, Android applications), and Python in areas such as data analysis, machine learning, and rapid prototype development.

To conduct a comparative analysis of the Java and Python programming languages, we will refer to the main differences between these languages, which are broadly presented in the following table.

Table 1. The Main Differences Between Java and Python

Parameter	Java	Python
What is a programming language?	Java is a multi-platform, object-oriented, and network-centered programming language. It is among the most widely used programming languages. It is also used as a computing platform and was first released by Sun Microsystems in 1995. Later, it was acquired by Oracle Corporation.	Python is a high-level, object-oriented programming language. It has built-in data structures combined with dynamic binding and typing, which makes it an ideal choice for rapid application development. Python also provides support for modules and packages, allowing for system modularity and code reuse. It is one of the fastest programming languages, as it requires very few lines of code. Its emphasis on readability and simplicity makes it an excellent choice for beginners.
Compilation	Java is a compiled language.	Python is an interpreted language.
Static or dynamic	Java is statically typed.	Python is dynamically typed.
String operations	Provides limited string-related functions.	Provides a wide range of string-related functions.
Learning curve	Complex learning curve.	Easy to learn and use.
Multiple inheritance	Multiple inheritance is partially achieved through interfaces.	Provides both single and multiple inheritance.
Braces vs. Indentation	Uses braces to define the beginning and end of each function and class definition.	Python uses indentation to separate code into code blocks.
Speed	Java programs run faster compared to Python.	Python programs run slower than Java.
Portability	Any computer or mobile device that can run the Java Virtual Machine can run a Java application.	Python programs require an interpreter installed on the target machine to translate Python code. Compared to Java, Python is less portable.
File reading	Java requires 10 lines of code to read from a file.	Python requires only 2 lines of code.
Architecture	The Java Virtual Machine provides the runtime environment to execute code and convert bytecode into machine language.	For Python, the interpreter translates the source code into machine-independent bytecode.
Famous companies using this technology	Airbnb, Netflix, Spotify and Instagram.	Uber Technologies, Dropbox and Google.
Best features	• Great libraries • Widely used • Excellent tools	• Readable code • Rapid development • Elegant code

Parameter	Java	Python
	A huge amount of documentation available	
Best use cases	Java is the best for desktop GUI applications, embedded systems, web application services, etc.	Python: Excellent for scientific and numerical computing, machine learning applications, and more.
Database support	Provides stable connectivity.	Provides weak connectivity.

Source: elaborated based on [9], [10], [15].

Which is better, Python or Java?

The simplicity and readability of Python make it an excellent option for beginners and for developers in fast-growing environments, while Java's static typing and object-oriented features make it ideal for large-scale applications. The choice of the "better" language depends on your project requirements and personal preferences. Although Python is expected to run slower than Java, its development takes less time. Thanks to its built-in high-level data types as well as dynamic typing, Python programs are usually shorter than equivalent Java programs, which makes them simpler and faster to develop. Since Java requires more code and everything must be predefined, developers need more time to check everything and fix potential errors. Naturally, the more code there is, the more complex it becomes. However, the attention required to write good code can also lead to the creation of more stable and reliable software.

Simply put, Python runs slower but starts up faster. In contrast, Java starts up more slowly but stands out by running faster. Ultimately, the best programming language is the one that fits the type of software the developer wants to create. Ideally, as already mentioned, it would be useful for developers to learn both languages [15].

Considering that one of the essential dimensions in the comparative analysis of programming languages is typing and execution mode, we can make the following summary:

Java is a statically typed and compiled language, which means that type checking takes place at compile time, and the resulting code is transformed into optimized bytecode, executed on the Java Virtual Machine (JVM). This characteristic enhances the robustness of applications, reduces the risk of runtime errors, and provides superior performance in complex and critical environments (e.g., banking systems, government infrastructures). However, in terms of string manipulation, Java offers a relatively limited set of native functionalities, which often requires the use of additional libraries.

In contrast, Python is a dynamic and interpreted language, where type checking takes place at runtime. This model provides a high degree of flexibility, reducing development time and enabling rapid prototyping. Moreover, Python offers a vast range of functions and libraries dedicated to string manipulation, which makes it extremely efficient for data processing tasks, text analysis, or natural language processing. This orientation places it at the core of data-driven applications and artificial intelligence projects.

For a better study aimed at choosing between Java and Python and deciding which of these programming languages to use, it is necessary to focus on the characteristics and disadvantages of Java and Python, presented in the following table.

Table 2. Characteristics and Disadvantages of the Java and Python Languages

Characteristics of the Java	Characteristics of the Python
• Write code once and run it on almost any computing platform. • It is designed for building object-oriented applications. • It is a multithreaded language with automatic memory management. • It facilitates distributed computing as network-centered.	• Easy to learn, read, and maintain. • Can run on various hardware platforms using the same interface. • You can include low-level modules in the Python interpreter. • Python offers an ideal structure and supports large programs. • Python provides support for automatic garbage collection. • Supports an interactive mode for testing and debugging. • Offers high-level dynamic data types and also supports dynamic type checking. • The Python language can be integrated with Java, C, and C++ programming code.
Disadvantages of the Java	Disadvantages of the Python
Over the time I have used Java, I have encountered the following disadvantages: • The JIT compiler makes the program relatively slow. • Java has high memory and processing requirements. Therefore, hardware costs increase. • It does not provide support for low-level programming constructs such as pointers. • You have no control over garbage collection, since Java does not provide functions such as delete() or free().	

Source: elaborated based on [9], [10].

The decision to use Java or Python in a software development project must be based on the nature, objectives, and constraints of the project.

Python is often recommended for beginners due to its simple syntax, close to natural language, which facilitates the learning process. In addition, its open-source nature has led to the accelerated development of a vast ecosystem of tools and libraries covering areas such as statistical analysis, machine learning, and data visualization. Initial development costs are generally lower, making it attractive for startups and exploratory research projects.

Java, on the other hand, is designed as a general-purpose language, oriented toward portability through the principle of “write once, run anywhere”. This characteristic makes it suitable for large-scale projects where stability, security, and performance are priorities. Moreover, the Java ecosystem (Spring, Hibernate, Jakarta EE) provides strong support for enterprise applications, cloud infrastructures, and mobile applications (Android).

Therefore, the selection of the language depends on the balance between development costs, project complexity, and performance requirements. For applications focused on data exploration and rapid prototyping, Python is the optimal choice. Conversely, for critical applications requiring robustness, scalability, and adherence to strict security standards, Java remains the preferred solution.

2. Programming with Java and Python and Data Protection

In the era of strict regulations on privacy and digital security, programming can no longer be separated from the issue of data protection. According to the principle of “privacy by design”, enshrined in the General Data Protection Regulation (GDPR), software development must integrate from the design phase both technical and organizational measures intended to ensure the protection of personal information [11].

The role of Java and Python in implementing the principles of GDPR, NIS2, and CCPA

Both Java and Python, through their extensive ecosystems, provide libraries and frameworks dedicated to ensuring compliance with security and privacy standards. In environments regulated by GDPR or NIS2, enterprise applications developed in Java rely on robust security architectures such as Spring Security and the Java Cryptography Architecture (JCA), which implement mechanisms for multi-factor authentication, encryption, and access auditing [8], [12].

At the same time, Python, due to its flexibility and popularity in data processing, integrates packages such as PyCryptodome or Fernet, used for the encryption and anonymization of personal data [2]. In the context of the California Consumer Privacy Act (CCPA), Python is frequently employed in building analysis and reporting tools that help companies demonstrate compliance with requests for access to or deletion of personal data [16].

Thus, both languages are not only “vehicles” for software development but also technical means of applying legislation in the field of data protection.

Table 3. Examples of tools and applications for security and data protection in Java and Python

a		Java
Monitoring and log analysis	- Scripts for parsing security logs (SIEM) with pyparsing, pandas. - Event visualization with matplotlib, seaborn. - Detection of multiple failed authentication attempts.	- Microservices that collect logs with logback / log4j and forward them via Kafka. - Real-time alerting rules (divergent geolocation, suspicious entries).
Traffic and packet analysis (detections)	- Scapy for packet capture and analysis. - Detection of data leaks through traffic analysis.	- Network analysis libraries in enterprise applications. - Integration with MLOps systems for intrusion detection at the gateway level.
Application-level data protection	- cryptography, pyca/cryptography libraries for encrypting PII and medical data. - Data masking in APIs/UI to protect sensitive information.	- javax.crypto, java.security for symmetric/asymmetric key encryption. - Tokenization of sensitive data in large databases.

Compliance and access rights management	- Scripts for audit and automated compliance reports. - Validation of RBAC/ABAC access policies on APIs.	- Access control through OAuth2, OpenID Connect in security gateways. - Privacy policies in enterprise workflows.
Automation of attachment management and data migration	- Scripts for migrating sensitive data between databases, with validation and in-transit encryption. - DLP (Data Loss Prevention) for detecting data in the cloud.	- Enterprise services for migration with logging, rollback, and full audit. - Integration with commercial/open-source DLP solutions via APIs.
Security and vulnerability testing	- SAST/DAST scripts for source code scanning. - Automated penetration testing with pytest.	- OWASP Dependency-Check for Maven/Gradle. - Automated evaluation of configurations (CSRF, XSS, CORS).
Data portability and migrations	Export/import in CSV, JSON, Parquet with schema validations.	Export/import in JSON, XML with XSD/JSON Schema validations.

Source: elaborated based on [2], [8], [12], [16].

Examples of real projects

- Security pipeline for a web application: prototype in Python, then ported to Java for production at scale.
- DLP tool for monitoring data in the cloud: analysis in Python, integration API in infrastructure in Java.
- Data protection in databases: Transparent Data Encryption (TDE) in Java at the JDBC driver level, monitoring and alerting in Python.

Table 4. Errors and consequences in the Java and Python languages

Common mistake	Consequence
Logging personal data	Accidental exposure in files, infrastructure, or cloud.
Lack of encryption at rest	Sensitive data stored in plain text in databases or files.
Lack of access control	Any user can access data that does not belong to them.
Storing passwords in plain text	Easy exploitation in case of a breach.

Source: elaborated based on [2], [3], [8].

Secure programming: security by design

Programming with Java and Python in the field of data protection involves using these languages to develop security solutions such as log analysis automation, packet analysis, and other specific tasks. Secure programming means not only writing functional code, but also adhering to fundamental principles: data minimization, encryption in transit and at rest, access control, audit and traceability, anonymization, and pseudonymization [2], [11], [12].

Java, through its security-oriented infrastructure, provides integrated solutions for implementing these principles in larger and more complex applications. Python, due to its simplicity and rich ecosystem, allows for the rapid development of prototypes and tools for compliance auditing and monitoring. Both languages can be used for developing data protection tools and systems, confirming the necessary convergence between technology and data protection legislation [3], [16].

CONCLUSIONS

Contemporary trends in the field of programming languages indicate a clear direction in the evolution of technologies, centered on innovation, security, and productivity. In a digital era defined by the rapid expansion of services and software applications, programming languages emerge not only as technical tools but also as structural foundations of the modern information society. Beyond functionality, the dimension of security becomes an essential criterion, while efficiency and productivity serve as benchmarks for evaluating the suitability of a language to the specifics of a project [14].

The research results confirm that Java and Python go beyond the status of mere programming languages, becoming true driving forces of digital transformation. In Eastern Europe, and especially in the Republic of Moldova, a clear distribution of preferences can be observed: Java is used predominantly in complex, enterprise-type applications, while Python dominates in data analysis and the field of artificial intelligence. Considering the complexity of learning C++, beginner programmers usually choose between Java and Python. This complementarity fosters not only technological innovation but also the sustainable development of the digital ecosystem, providing a framework for cooperation between industry, education, and research.

Looking to the future, three major directions of evolution are emerging:

- ✓ Strengthening the complementarity between Java and Python in interdisciplinary and large-scale projects;
- ✓ Expanding the educational role, with emphasis on training new generations of digital specialists;
- ✓ Deepening ethical debates, through the integration of security and privacy principles into programming practice.

Overall, the convergence of Java and Python creates the premises for collaborative, ethical, and innovative development, in which security and responsibility play a central role. This confirms the fact that programming languages constitute the pillars of the modern information society, and continuous research on their role and influence is an indispensable condition for technological progress and for strengthening an interconnected, resilient, and efficient digital world.

The security of programming languages has become one of the main issues, while their efficiency and productivity are key criteria in choosing the appropriate language for a project. Future-oriented programming languages have been defined as those that will shape technology in the coming years, and education and training in programming languages are considered vital for preparing future IT specialists.

Ultimately, it is evident that programming languages form the foundation of the modern information society, and their continuous study and the understanding of their influence across various domains are necessary for technological development and continuous innovation. It is important to remain open to change and adapt to new trends and technologies, in order to contribute to building an interconnected and efficient world.

REFERENCES

1. Cavoukian Ann. Privacy by Design. The 7 Foundational Principles Implementation and Mapping of Fair Information Practices. Disponibil la: <https://privacy.ucsc.edu/resources/privacy-by-design---foundational-principles.pdf>.
2. CodeVisionZ. (2023). *Python Cryptographic Libraries*. Disponibil la: <https://codevisionz.com/lessons/python-cryptographic-libraries/>
3. Dev.to. (2024). *Java Security: Protecting Your Applications with Secure Coding, Cryptography, Access Control*. Disponibil la: <https://dev.to/adityabhuyan/java-security-protecting-your-applications-with-secure-coding-cryptography-access-control-26ej>
4. European Union (2016). *General Data Protection Regulation (GDPR)*. Regulation (EU) 2016/679. Disponibil la: <https://gdpr-info.eu/>
5. European Union (2022). *Directive (EU) 2022/2555 on measures for a high common level of cybersecurity across the Union (NIS2 Directive)*. Disponibil la: <https://eur-lex.europa.eu/eli/dir/2022/2555/oj/eng>
6. GetWidget (2023). *Python vs Java: A Comprehensive Comparison*. Disponibil la: <https://www.getwidget.dev/blog/python-vs-java/>
7. Gosling J., Joy B., Steele G., Bracha G. (2005). *The Java Language Specification*. Addison-Wesley. Disponibil online: https://www.researchgate.net/publication/2290452_The_Java_Language_Specification
8. *Java Cryptography Architecture*. Disponibil la: https://en.wikipedia.org/wiki/Java_Cryptography_Architecture
9. *Java vs Python - Comparison Guide* (2025) Disponibil online: <https://brightdata.com/blog/web-data/java-vs-python?ysclid=meokkuwsfd379795768>
10. *Java vs Python – Difference Between Them* Disponibil online: <https://www.guru99.com/java-vs-python.html>
11. Moldstud. (2023). *Integrating Security and Privacy Considerations in Software Architecture*. Disponibil la: <https://moldstud.com/articles/p-integrating-security-and-privacy-considerations-in-software-architecture>
12. Oracle. *Java Cryptography Architecture (JCA) Reference Guide* (2023).. Disponibil la: <https://docs.oracle.com/en/java/javase/21/security/java-cryptography-architecture-jca-reference-guide.html>
13. Oracle. *Java SE Documentation*(2023). Disponibil online: <https://docs.oracle.com/en/java/>
14. Pirlog A., Turcan A. *Programming languages in the information society era: evolution and analysis* (2024).Disponibil online: x-natsionalnaya-nauchno-prakticheskaya-konferentsiya-problemy-i-vyzovy-ekonomiki-regiona-v-usloviyakh-globalizatsii-2024.pdf
15. *Python and Java: key differences, performance, and use cases* (2025) Disponibil online: <https://www.imaginarycloud.com/blog/python-vs-java>
16. Python Central. (2023). *How Python and Cybersecurity Services Help with Compliance and Regulatory Requirements*. Disponibil la: <https://www.pythoncentral.io/how-python-and-cybersecurity-services-help-with-compliance-and-regulatory-requirements/>
17. Python Software Foundation *Python Documentation* (2023). Disponibil online: <https://docs.python.org/3/>
18. Python Tutorial. Disponibil online: <https://www.w3schools.com/python/default.asp>
19. State of California (2018). *California Consumer Privacy Act (CCPA)*. Assembly Bill No. 375. Disponibil la: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375
20. TIOBE Index (2025). *Programming Community Index*. <https://www.tiobe.com/tiobe-index/>
21. Van Rossum G., Drake F. (2009). *The Python Language Reference*. Python Software Foundation. Disponibil online: <https://archive.org/details/pythonlanguage0000vanr>
22. *Which path should a beginner programmer take: Python vs Java* / Habr (2024) Disponibil online: <https://habr.com/ru/articles/788348/>
23. *Why enterprises rely on JavaScript, Python, and Java*. Disponibil la: <https://www.infoworld.com/article/2336936/why-enterprises-rely-on-javascript-python-and-java.html>