RESEARCH ARTICLE OPEN ACCESS

Comparative Review of Python and C: Choosing the Right Language for Beginners

Mr. Pankaj Jain [1], Hemant Soni [2], Aditya Pratap Singh [3], Puneet Mehta [4], Neeraj Babani [5]

[1] Department of Computer Science and Engineering, Global Institute of Technology Jaipur (Rajasthan), India,
[2] Department of Computer Science and Engineering, Global Institute of Technology Jaipur (Rajasthan), India,
[3] Department of Computer Science and Engineering, Global Institute of Technology Jaipur (Rajasthan), India,
[4] Department of Computer Science and Engineering, Global Institute of Technology Jaipur (Rajasthan), India,
[5] Department of Computer Science and Engineering, Global Institute of Technology Jaipur (Rajasthan), India,

ABSTRACT
The choice of programming language for beginners is a critical decision that influences their learning journey and skill development trajectory. This paper presents a comprehensive comparative review of Python and C, two prominent programming languages with distinct characteristics, to assist beginners in selecting the most suitable language for their educational endeavors. The comparison encompasses various dimensions including syntax simplicity, readability, learning curve, versatility, performance, community support, and practical application domains. Through an analysis of these key metrics, this research aims to provide valuable insights into the strengths and weaknesses of Python and C, facilitating informed decision-making for novice programmers. Additionally, the paper discusses the pedagogical implications of language selection, considering factors such as educational accessibility, programming paradigms, and career opportunities. By addressing the nuanced differences between Python and C, this study contributes to enhancing the understanding of programming language selection for beginners and fosters a more informed approach to computer science education.
Keywords: Python, C, Comparison, Programming Language, Beginner.

I. INTRODUCTION

Programming languages play an important role in software development by providing a variety of tools and methods to solve various computational problems. Among the many programming languages available, Python and C stand out as important choices due to their unique quality and features. Deciding to choose between Python and C, especially for those new to programming, should have a similar analysis, considering ease of learning, diversity, quality of feasibility, ecosystem support, and relevance to data research issues such as gender. Known for its

simplicity and readability, Python has become a favorite among beginners and experienced developers alike. The syntax is inspired by readability-centered principles and is designed to reduce the skills of programmers, making it an attractive choice for learning content. Python's versatility covers many areas including web development, data analysis, machine learning, computing, automation, and more. The extensive language framework library and large ecosystem of third-party software packages provide many tools and resources to speed up the development and troubleshooting process. In addition, the dynamically typed and defined nature of Python enables rapid prototyping, interactive development, and easy debugging, thus making overall development easier. C is for standard programming, embedded system

development and basic functionality. C is known for its high performance, direct hardware control, and low performance, giving developers unparalleled control and optimization. C's compilation features enable fast execution and memory management, making it the first choice for critical tasks.

Although C's syntax is more explicit and unforgiving than Python's, its emphasis on memory management, evaluation, and interaction between processes allows for a better understanding of computer architecture and programming fundamentals. Choosing the right programming language requires careful consideration, especially for beginners who are starting to research a thesis topic. Python's ease of learning, rich library, and general usability make it an ideal choice for beginners entering many fields such as data science, technology web development, and specialization. User-friendly syntax and community support create a learning environment that allows beginners to focus on problem solving and application development. In contrast, C refers to low-level processing, memory management and hardware interaction. Although the learning curve is steep, mastering C provides an in-depth understanding of computer architecture, memory management, and optimization necessary for high performance. In this comparative review we take a look at the advantages, disadvantages and disadvantages of C. Python and C are suitable for beginners. They are research topics in various fields. By analyzing factors such as learning curve, ecosystem support, performance, and specific requirements, our goal is to guide beginners in their decision-making when choosing the right programming language for their education and research.

II. OBJECTIVE

The purpose of this comparative review is to help beginners search for research papers in various fields Identify and evaluate Python and C programming languages to help them choose the right language. This review is designed to give beginners who are starting out programming and working on research-based projects a better understanding of the strengths, weaknesses, and advantages of Python and C. Learnability, versatility, performance, ecosystem support and specific This review is designed to ensure Beginners gain the knowledge and understanding they need to make the right programming language choices. to decide. their research works.

1. Rate the difficulty of learning:

Compare the learning curve of Python and C for beginners without any programming experience. Analyze problems caused by Python's easy-to-read, easy-to-use and user-friendly C syntax, manual memory management and low-level concepts for a fresh start.

2. Explore the possibilities and uses:

Web development, data science, machine learning, programming, etc. Explore a wide range of areas and applications where Python and C.Examine the suitability of Python and C. Both are used for specific research applications, including data processing, algorithm implementation, performance, and interaction needs.

3. Performance and Performance Analysis:

Evaluate the performance differences between Python and C in terms of execution speed, memory usage, and resource usage.Examine the performance of Python's interpreting and writing quality compared to C's compiled nature, manual memory management, and direct hardware access.

4. See supporting ecosystem and tools:

Evaluate the availability and richness of libraries, frameworks, and development tools for Python and C, including building model libraries, third-party packages, IDEs, and debugging tools.Explore community support, information, tutorials, and online resources for Python and C beginners to support their learning and problem solving.

5. Consider specific registration requirements:

Identify specific case studies and areas where Python and C have advantages or limitations.Analyze case studies or research project examples using Python and C to show their advantages and disadvantages in practical applications.

6. Provide recommendations and guidance:

Based on comparative analysis, provide recommendations and guidance for beginners to choose Python or C as their main programming language research project.Recommends best practices, learning resources, and ideas for beginners to gain skills, overcome challenges, and harness the power of Python and C.

In terms of design and purpose, this comparative analysis aims to bridge. The gap between technical knowledge and the C language. Beginners gain the knowledge, skills and understanding they need to explore the complexities of programming languages and conduct successful research.

III. OVERVIEW

In the field of programming languages, Python and C reveal a especially important feature, advantage and application option. This comparative analysis of Python and C is designed to guide beginners in making decisions about choosing the right language to explore research topics in various fields. This maintenance provides basic training in two languages, delving into the complexity of Python's user interface, extensive libraries, and more, and C's performance, low-level control, and process-level programming.

Python is known for its readability, simplicity and ease of learning, making it an excellent entry point for beginners into the world of programming. Syntax is inspired by readability-centered principles, reducing grammatical complexity and creating a learning environment. Python's extensive standards library and vast ecosystem of third-party products help beginners tackle many aspects of data science, from data analysis, machine learning, and web development to computing, automation, and artificial intelligence. Dynamic authoring, translation, and interactive development can help quickly prototype, test, and iterate problems; This is important for beginners to explore various types of research.

In contrast, C is a basic language known for efficiency, effectiveness and direct hardware access. Although C's syntax can be confusing to beginners, its emphasis on memory management, scaling, and low-level control provides great insights into computer architecture, systems programming, and performance. C's compilation features speed up execution, making it ideal for critical operations, systems development, and inter-process communication. Although there is a learning curve, knowing the C language can give beginners a deeper understanding of programming, memory management, and computer systems that are still a tool for solving complex research problems.

This comparison between Python and C includes a few important points to help beginners choose the language:

1. Learning Curve and Accessibility:

Python's user-friendly syntax, readability, and extensive resources make it suitable for beginners with different experience levels.C Learning curve, memory management, and little content can cause difficulties, but it provides easy information and resource management.

2. Versatility and Applicability:

Python's versatility spans multiple domains and allows beginners to explore different topics in data science, web development, machine learning, and more.The applicability of C shines in systems programming, embedded systems, performance optimization, and direct hardware interactions, providing an in-depth understanding of computer architecture and performance levels.

3. Performance and Efficiency:

-Python prioritizes ease of use and development speed over performance, making it suitable for development modeling, data analysis, and the use of advanced algorithms

-. C’s compiled nature, high-performance memory management, and direct hardware access result in better performance and resource utilization, making it ideal for critical and low-performance applications.

4. Ecosystem support and community resources:

- Python's extensive standards libraries, third-party software packages, and community support provide beginners with tools, frameworks, and learning resources to solve research problems. .

- C's ecosystem provides access to core libraries, development tools, and low-level programming; It is complemented by a community focused on programming, good business programming, and hardware interaction.

5. Special considerations:

- Set options for specific research documents and records by relating them to the project's requirements, constraints, and needs for results.

- Review of case studies, examples, and best practices of research projects in Python and C provide insight into the strengths, limitations, and usability of each language.

Featuring a general introduction to Python and C, this comparative review provides beginners with the information, considerations, and guidance they need to make decisions when choosing a format. Good comments for research papers. By understanding the nuances, trade-offs, and applicability of Python and C to various fields, beginners can begin their journey with confidence and clarity and lay the foundation for successful research.

3.1 Overview of C:

C Programming Language Generally accepted It has an important place in software development, operation and important studies, forming the basis of today's programming language. Compared to Python, C stands out for its performance, hardware handling, and low workload, making it the best choice for beginners in programming and research across topics. Understanding the nuances, advantages, and applicability of C is crucial for beginners who want to understand the fundamentals of programming, delve into system interactions, and solve problems.

1. Efficiency and Performance Optimization:

C's integrated structure and direct interaction with hardware contribute to its efficiency and best performance. Unlike defined languages such as Python, C is compiled into machine code, allowing for faster execution and reduced memory overhead. Beginners looking for data science that requires high performance, real-time rendering, or low resolution can leverage the power of C to implement powerful and usable solutions. Good results.

2. Manual memory management and pointers:

One of the features of C is memory management using pointer support as well as functions such as malloc and free. The memory management guide provides developers with detailed control over memory allocation, release, and data structure, while revealing complexities and potential pitfalls such as memory leaks and segmentation errors. Beginners who understand the scientific concepts of memory management, data structure and optimization can gain insight into memory enhancement practices and performance level.

3. Direct Hardware Access and System Programming:

C's ability to go directly to hardware, interact with drivers, and perform system programming makes it ideal for business, the interaction of technology and tools is important. Beginners interested in data communications, driver development, real-time operating systems (RTOS), or low-level programming can use the power of C to get involved in computer architecture and software systems.

4. Portability and Platform Independence:

Despite the low level of the C language, it provides portability and platform independence, allowing the code to be written in C Compile and run on various operating systems, processors and architectures. This flexibility makes C suitable for cross-platform development, hardware and software that require integration with different environments. Beginners looking for data science on multiple platforms or needing cross-platform solutions can benefit from C's portability and flexibility.

In summary, the C programming language provides a powerful platform for beginners to understand the operation, performance and low-level software development process; because performance is the best choice for work that requires hardware-level management. . and interactions in the system. By understanding the advantages, challenges, and practical applications of C, beginners can gain a better understanding, master the concepts in the study, and improve the research process in the field of computer science and software development.

3.1.1 Applications of C:

1. System Software:
C is widely used to develop software such as operating systems (such as Linux kernel, Windows kernel), device drivers, firmware, and electronic devices. Its low performance and direct access to equipment make it ideal for stage operations.

2. Embedded Systems:
C is a popular choice for developing systems including microcontrollers, IoT devices, and real-time devices. Its performance, compact footprint, and ability to interface with hardware make it ideal for limited spaces.

3. Programming Language Development:
Many programming languages and compilers are used in C or C++. C's efficiency, control structure, and ability to interact with assembly code make it suitable for developing programming languages and compilers.

4. Game Development:
C is used in game development to implement game engines, graphics libraries (such as OpenGL, DirectX), and core components. Its speed, low entry level, and ability to handle complex tasks make it easy to create great games.

5. Database Systems:
C is used in database management systems (DBMS) to implement important functions such as query processing, indexing, transaction management, and memory management. Its high performance and low maintenance help improve data performance.

Overall, C's versatility, efficiency, and hardware management make it the best choice for many applications, from standard programming to embedded systems, visualization and games. Its low operating costs and high performance make it a powerful tool for developing critical software and optimizing resources.

3.1.2 Advantages of C:

1. Efficiency: C is highly efficient and provides low-level management of system resources, making it suitable for critical and operational tasks.

2. Portability: C programs can be compiled and executed on multiple platforms with few modifications this provides portability between different programs and hardware.

3. Flexibility: C supports both high- and low-level programming, allowing developers to work at as many levels of abstraction as necessary.

4. Extensive libraries: C provides a rich set of libraries containing basic functions for data manipulation, input/output, string operations, and mathematical calculations.

5. Memory Management: While memory management can be complex, it also provides greater control over memory allocation and location, resulting in better memory utilization in your application.

3.1.3 Disadvantages of C:

1. Complex syntax: C has a more complex syntax compared to advanced languages like Python or Java. It requires explicit memory management and manipulation of mathematical expressions and data structures, which can be difficult for beginners and lead to errors.

2. Memory Management: In C, developers need to manage memory allocation and allocation using functions like malloc and free. If it is not working properly, it may cause memory leaks, drooping indicators, and poor memory.

3. Pointers and Memory Access: C's use of pointers makes memory access efficient and effective, but it can also cause a segmentation fault and cannot be bypassed if pointers are not used.

4. String processing: C's processing of strings as strings can be error prone, especially when processing untrimmed strings. Non-overflow and string manipulation errors are common pitfalls in C programming.

5. Limited standard library: Compared to high-level languages, C's standard library is also limited; requires developers to rely on third-party libraries or write custom code to get the job done, O and data I/O Activities like networking.

C is still a powerful and widely used language, especially in areas that require high-level functionality, performance and direct use of hardware. Many of these difficulties can be alleviated through careful study, experimentation, and understanding of the nuances of Type C.

3.2 Overview of Python:

Python for its simplicity Ease of use , readability and versatility make it a programming language in many fields, from web development and data search research to machine learning and automation. Compared to C, Python appears to be an important language for its ease of learning, rapid development, and widespread use, making it a good choice for beginners. Understanding the power, ecosystem, and usability of Python is crucial for beginners who want to understand the basics of programming, solve complex problems, and complete research.

1. Easy to learn and read:

Python's user-friendly syntax is inspired by readability-oriented principles, reducing syntactic complexity and improving beginners' access to legal processes. Unlike C's low-level constructs and manual memory management, Python's high-level abstraction, dynamic typing, and automatic memory management simplify the learning curve, making it easier for beginners with no programming experience to get started. Those who are just starting to research scientific topics can focus on problem solving and algorithmic thinking rather than getting stuck on compound words and memory problems.

2. Versatility and Applicability:

Python's versatility covers a wide range of fields, including web development, data analysis, machine learning, scientific computing, automation, and more. Its extensive standards library and large ecosystem of third-party packages provide beginners with ready-to-use tools and resources to tackle different research topics. Python is suitable for rapid prototyping, testing, and refactoring; This makes it ideal for those just starting to explore new technologies, AI-driven applications, and research data.

3. Community Support and Support:

Python has a great community of developers, teachers, and supporters who provide in-depth information, tutorials, networking talks, and news to encourage beginners. The Python community creates a collaborative learning environment that encourages knowledge sharing and provides education to beginners throughout their journey. Those new to exploring data science using Python can benefit from community support, collaboration on open-source projects, and access to best practices and coding standards recognized by the Python community.

4. Extensive libraries and third-party packages:

Python's comprehensive library provides built-in frameworks and functions for common workflows. For example, data I/O, communications, data serialization, array operations, and arithmetic. Additionally, Python's vast ecosystem of third-party packages provided by repositories such as PyPI (Python Package Index) includes many sites including web (such as Django, Flask), data science libraries (such as NumPy, Pandas, Matplotlib). , machine learning methods (such as Tensor Flow, PyTorch, Scikit-learn), automation tools, etc. Beginners looking for scientific topics can take advantage of these libraries and models to quickly develop and analyze data, use algorithms, and find useful results.

5. Interpreter and Rapid Development:

The Python interpreter simplifies development by allowing interactivity, fast design, and easy debugging. Beginners can speed up the development cycle by writing and running the script line by line, testing the code line in an interactive shell (Python REPL), and repeating the speed of number change. Python improves performance by supporting concepts and expressions by emphasizing code readability, indentation-based syntax, and language constructs such as list comprehensions, constructors, and lambda functions.

Python programming language provides a good place for beginners to explore research topics, practical ideas and create creative solutions to problems in various fields. Beginners can master the intricacies of programming languages, solve complex problems, and succeed in their research by taking advantage of Python's ease of learning, versatility, community support, extensive libraries, and rapid development capabilities. Python continues to be influential in academia, business, and science as a beginner-friendly, powerful, and versatile language, making it an excellent choice for beginners to begin their journey.

3.2.1 Applications of Python:

1. Web Development: Python is widely used in web development, and frameworks like Django and Flask offer a great way to create powerful and useful web applications. Python's simplicity, readability, and rich library make it popular for backend development, API development, and server-side scripting.

2. Data Science and Analysis: Python's rich library ecosystem (including NumPy, Pandas, Matplotlib, and SciPy) makes it the first choice for data science, machine learning, and data analysis. Python is used for data management, statistical analysis, machine learning development, and data visualization.

3. Machine Learning and Artificial Intelligence: Python is an important language for machine learning and artificial intelligence. Education model. Ownership. Python's simplicity and readability enable rapid prototyping and testing in AI projects.

4. Scientific Research: Python; It is widely used in computing, including simulation, numerical analysis, and mathematical modeling. Libraries such as SciPy and SymPy provide tools for visualization, optimization, numerical simulation, and solving differential equations.

5. Automation and Scripting: Python's ease of use and

cross-platform compatibility make it ideal for automation, scripting, and systems administration. Python scripts can perform repetitive tasks, manipulate data, interact with data, and manage resources.

6. Game Development: Python is used to develop games, especially standalone games and prototypes. Libraries such as Pygame provide tools for creating 2D games, while other frameworks such as Panda3D and Unity Python API support 3D games.

Overall, Python's performance, useful libraries and ease of use make it suitable for many applications such as web development, site, data science, machine learning, computational science, automation, game development. Its popularity and community support continues to drive innovation and adoption in the industry.

3.2.2 Advantages of Python:

1. Easy to Learn: Python's simple and readable syntax makes it easy for beginners to learn and understand.

2. Function library: Python reduces development time by providing a large function library and many third-party libraries for various tasks.

3. Versatility: Python is suitable for web development, data analysis, machine learning, automation, etc. It can be used to make it very versatile.

4. Definitions: Python definitions enable rapid design, testing, and debugging.

5. Community Support: Python has a large and vibrant community that provides resources, tutorials, and support to developers.

6. Platform Independence: Python code is portable and can run on different platforms without modification.

7. High-level abstraction: Python provides high-level information and abstraction, simplifying complex tasks.

8. Scalability: Python is scalable and suitable for small and large scripts.

3.2.3 Disadvantages of Python:

1. Slower Execution: Python can be slower to interpret compared to programming languages like C.

2. Memory consumption: Compared to memory-managed languages, Python uses more memory, resulting in higher usage.

3. Global Interpreter Lock (GIL): In parallel applications, GIL limits functionality by allowing only one thread to execute Python bytecode at a time.

4. Not good for high performance: Due to its slowness and memory consumption, Python may not be the best choice for high performance work.

5. Dependency management: Managing dependencies and modules in a Python project can be difficult, especially if the views are conflicting.

6. Mobile Development: Although Python has a framework like Kivy for mobile development, it is not widely used as a language in mobile app development.

7. Security Issues: Python's dynamic typing feature and flexibility can lead to security vulnerabilities if not used correctly.

8. No compiled binaries: Python code is usually distributed as raw data or bytecode; this may not be as secure or optimized as compiled binaries.

IV. SYNTAX COMPARISON BETWEEN C AND PYTHON:

In our comparative analysis of C and Python syntax, we examined various fundamental programming constructs and language features. Below, we elaborate on each topic in detail:

1. Variables:

- C: Variables in C require explicit declaration with a specified data type before use.

- Python: Variables in Python are dynamically typed, allowing assignment without explicit declaration.

2. Casting:

- C: Type casting is commonly used in C for converting data from one type to another, often with explicit casting syntax.

- Python: Implicit type conversion is prevalent in Python, but explicit type casting can be performed using built-in functions like int (), float (), and str ().

3. Print Functions:

- C: Printing in C typically involves using printf() function from the standard library, with format specifiers for different data types.

- Python: Python provides the print () function for displaying output to the console, which supports printing multiple values separated by commas.

4. Commenting:

- C: Comments in C are denoted using /* */ for block comments and // for single-line comments.

- Python: Comments in Python are indicated using # symbol, which spans the entire line for single-line comments.

5. Separators:

- C: Statements in C are terminated by a semicolon; and blocks of code are enclosed within curly braces {}.

- Python: Python uses indentation to denote blocks of code, and statements are separated by line breaks, eliminating the need for explicit terminators.

6. Operators:

- C: C provides a wide range of operators for arithmetic, logical, bitwise, and relational operations.

- Python: Python supports similar operators as C but also introduces additional operators such as ** for exponentiation and // for floor division.

7. Strings:

- C: Strings in C are represented as arrays of characters terminated by a null character \0, and manipulation often involves using standard library functions.

- Python: Strings in Python are immutable sequences of characters, allowing easy manipulation with built-in string methods and operators.

8. User Inputs:

- C: User inputs in C are typically obtained using functions like scanf() for formatted input.

- Python: Python provides the input () function for accepting user inputs, which returns a string that can be converted to other data types as needed.

9. Conditional Statements:

- C: Conditional statements in C include if, else if, and else constructs for decision making.

- Python: Python's conditional statements are similar to C but use indentation to denote code blocks instead of braces.

10. Loops:

- C: C supports for, while, and do-while loops for iterative operations.

- Python: Python provides for and while loops, with a versatile for loop capable of iterating over various iterable objects.

11. Programming Paradigms (OOPS, Procedural):

- C: C primarily follows procedural programming paradigm, although object-oriented programming (OOP) can be implemented using structs and function pointers.

- Python: Python supports both procedural and object-oriented programming paradigms natively, with classes and inheritance for OOP.

12. Functions and Methods:

- C: Functions in C are defined using a return type, name, parameters, and body enclosed within curly braces.

- Python: Functions in Python are defined using the def keyword, and methods are functions associated with objects, invoked using dot notation.

13. Exception Handling:

- C: Error handling in C often involves using conditional statements and error codes returned by functions.

- Python: Python's exception handling mechanism utilizes try, except, finally, and raise keywords for robust error management.

14. File Handling:

- C: File handling in C requires using functions like fopen(), fclose(), fread(), and fwrite() for file operations.

- Python: Python simplifies file handling with built-in functions like open (), read (), write (), and context managers for automatic resource management.

While this comparison covers a wide range of topics, it's important to note that both languages have unique strengths and weaknesses, making them suitable for different contexts and preferences. Further exploration and experimentation are recommended for a deeper understanding of each language's syntax and capabilities.

V. FUTURE SCOPE

1. C Language:

- System Programming: C's efficiency and low-level control make it indispensable for systems programming, including operating systems, device drivers, and embedded systems.

-Performance-Critical Applications: C's speed and memory management capabilities are important for the development of performance-critical applications such as real-time systems, high-frequency trading, and scientific computing.

-Legacy System Maintenance: Many legacy systems are written in C, creating a constant need for C developers to maintain and extend existing systems.

-IoT and Embedded Systems: Since IoT devices and systems are under development, C is also suitable for firmware development, interoperability between IoT protocols and low-end hardware.

2. Python Language:

-Data Science and Machine Learning: Python's rich library ecosystem (including NumPy, Pandas, and TensorFlow) makes it an excellent choice for data science, machine learning, and artificial intelligence. application form. important words.

-Web Development: Python frameworks like Django and Flask continue to gain popularity in web development due to their productivity, scalability, and ease of use.

-Automation and Scripting: Python's simplicity and versatility make it the first choice for automation, scripting, and project management.

-Education and Training: Python's easy-to-learn programming language and comprehensive curriculum make Python a popular language choice for teaching programming and computer science.

-Research Trends: Python's use in simulation, data analysis and computational modeling research is expected to increase due to its ease of use and powerful libraries.

CONCLUSION

In conclusion, both C and Python are powerful programming languages with unique benefits and applications. C is good at systems programming, embedded systems, performance-critical applications, and legacy system maintenance, providing low maintenance and high performance. Python, on the other hand, shines in the fields of data science, machine learning, artificial intelligence, web development, automation, scripting and education and takes advantage of its library. It is rich, simple and versatile. While C is important for low-level hardware and real-time interaction, Python's popularity and ease of use on modern hardware make it the first choice for many applications. Ultimately, the choice between C and Python depends on the specific needs of the project, and developers often use both languages to solve different problems.

REFERENCES

[1] Selina, Moirangthem, Borishphia, Jaeson, Tekcham "Comparative Analysis of Python and Java for Beginners"

[2] Kasurinen, Jussi. "Python as a programming language for the introductory programming courses." (2007).

[3] https://www.simplilearn.com/tutorials/c-tutorial/use-of-c-language

[4] Bogdanchikov, A., M. Zhaparov, and R. Suliyev. "Python to learn programming." Journal of Physics: Conference Series. Vol. 423. No. 1. IOP Publishing, 2013.

[5] Helminen, J., Ihantola, P., Karavirta, V., & Malmi, L., 2012. How Do Students Solve Parsons Programming Problems? - An Analysis of Interaction Traces. *ICER '12 Proceedings of the ninth annual international conference on international computing education research.* (Sep. 9--11, 2012) Auckland, New Zealand. 119--126

[6] Srinath, K. R. "Python–The Fastest Growing Programming Language." International Research Journal of Engineering and Technology (IRJET) 4.12 (2017): 354-357.

[7] https://www.interviewbit.com/blog/applications-of-python/

[8] C Programming Absolute Beginner's Guide by Perry and Miller https://usermanual.wiki/Pdf/CProgrammingAbsoluteBeginnersGuide3rdEditio.424140197.pdf

[9] Lars Ole Andersen "Program Analysis and Specialization for the C Programming Language"

[10] https://www.analyticsinsight.net/top-10-python-communities-everyone-should-join-in-2021/

[11] http://pypl.github.io/PYPL.html

[12] Adawadkar, Kalyani. "Python Programming Applications and the Future." International Journal of Advanced Engineering and Research Development. http://ijaerd. com/papers/special_papers/IT032. pdf (2017).

[13] Dr. R. Nageswara Rao. "Core Python Programming". New Delhi: Dreamtech Press:2018

[14] Donaldson, Toby. "Python as a first programming language for everyone." Western Canadian Conference on Computing Education. Vol. 232.

[15] G.K. Soni, A. Rawat, S. Jain and S.K. Sharma, "A Pixel-Based Digital Medical Images Protection Using Genetic Algorithm with LSB Watermark Technique", Springer Smart Systems and IoT: Innovations in Computing. Smart Innovation Systems and Technologies, vol. 141, pp 483–492, 2020.

[16] Gaurav Kumar Soni, Himanshu Arora and Bhavesh Jain, "A Novel Image Encryption Technique Using Arnold Transform and Asymmetric RSA Algorithm", Springer International Conference on Artificial Intelligence: Advances and Applications

2019 Algorithm for Intelligence System, pp. 83-90, 2020. https://doi.org/10.1007/978-981-15-1059-5_10

[17] S. Gour and G. K. Soni, "Reduction of Power and Delay in Shift Register using MTCMOS Technique," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI), pp. 202-206, 2020.

[18] H. Arora, G. K. Soni, R. K. Kushwaha and P. Prasoon, "Digital Image Security Based on the Hybrid Model of Image Hiding and Encryption," IEEE 2021 6th International Conference on Communication and Electronics Systems (ICCES), pp. 1153-1157, 2021.

[19] Babita Jain, Gaurav Soni, Shruti Thapar, M Rao, "A Review on Routing Protocol of MANET with its Characteristics, Applications and Issues", International Journal of Early Childhood Special Education, Vol. 14, Issue. 5, pp. 2950-2956, 2022.

[20] Pradeep Jha, Deepak Dembla & Widhi Dubey , "Implementation of Transfer Learning Based Ensemble Model using Image Processing for Detection of Potato and Bell Pepper Leaf Diseases", International Journal of Intelligent Systems and Applications in Engineering, 12(8s), 69–80, 2024.

[21] Pradeep Jha, Deepak Dembla & Widhi Dubey, "Deep learning models for enhancing potato leaf disease prediction: Implementation of transfer learning based stacking ensemble model", Multimedia Tools and Applications, Vol. 83, pp. 37839–37858, 2024.

[22] P. Upadhyay, K. K. Sharma, R. Dwivedi and P. Jha, "A Statistical Machine Learning Approach to Optimize Workload in Cloud Data Centre," 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2023, pp. 276-280, doi: 10.1109/ICCMC56507.2023.10083957.

[23] Pradeep Jha, Deepak Dembla & Widhi Dubey , "Crop Disease Detection and Classification Using Deep Learning-Based Classifier Algorithm", Emerging Trends in Expert Applications and Security. ICETEAS 2023. Lecture Notes in Networks and Systems, vol 682, pp. 227-237, 2023.

[24] P. Jha, D. Dembla and W. Dubey, "Comparative Analysis of Crop Diseases Detection Using Machine Learning Algorithm," 2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2023, pp. 569-574, doi: 10.1109/ICAIS56108.2023.10073831.

[25] P. Jha, R. Baranwal, Monika and N. K. Tiwari, "Protection of User's Data in IOT," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 1292-1297, doi: 10.1109/ICAIS53314.2022.9742970.

[26] P. Jha, T. Biswas, U. Sagar and K. Ahuja, "Prediction with ML paradigm in Healthcare System," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2021, pp. 1334-1342, doi: 10.1109/ICESC51422.2021.9532752.

[27] Mehra, M., Jha, P., Arora, H., Verma, K., Singh, H. (2022). Salesforce Vaccine for Real-Time Service in Cloud. In: Shakya, S., Balas, V.E., Kamolphiwong, S., Du, KL. (eds) Sentimental Analysis and Deep Learning. Advances in Intelligent Systems and Computing, vol 1408. Springer, Singapore. https://doi.org/10.1007/978-981-16-5157-1_78

[28] Gaur, P., Vashistha, S., Jha, P. (2023). Twitter Sentiment Analysis Using Naive Bayes-Based Machine Learning Technique. In: Shakya, S., Du, KL., Ntalianis, K. (eds) Sentiment Analysis and Deep Learning. Advances in Intelligent Systems and Computing, vol 1432. Springer, Singapore. https://doi.org/10.1007/978-981-19-5443-6_27

[29] P. Jha, D. Dembla and W. Dubey, "Implementation of Machine Learning Classification Algorithm Based on Ensemble Learning for Detection of Vegetable Crops Disease", International Journal of Advanced Computer Science and Applications, Vol. 15, No. 1, pp. 584-594, 2024.