IJHSR
ijhighschoolresearch.org
■ RESEARCH ARTICLE

The Adoption of Green Programming Languages as a Promising Approach to Improve Computing's Sustainability

Arya Kashyap

Interlake High School, 16245 NE 24th St, Bellevue, WA, 98005, USA, kashyapgba@gmail.com

ABSTRACT: As the environmental impact of computing worsens, transitioning to more sustainable computing practices, also known as green computing, has become crucial. This literature review examines the concept of green computing, focusing on the environmental impacts of software languages and how millions of developers worldwide interact with popular yet energy-intensive languages such as Python and Perl daily without realizing their negative environmental consequences. The review elucidates how such impacts are measured and the potential advantages and drawbacks of adopting more green programming languages as new tools instead of energy-intensive ones. The paper further argues that green computing, while challenging to implement now, is critical for creating a more sustainable future for our society.

KEYWORDS: Systems Software; Languages and Operating Systems; Sustainability; Energy-Efficiency; Green Computing.

■ Introduction

Most energy today is produced unsustainably by burning fossil fuels such as coal, oil, and natural gas.[1] This process releases greenhouse gases into the atmosphere, which leads to numerous environmental problems, including climate change, ozone depletion, and more.[2] As of 2020, computing-related emissions have reached over 2.1% of the world's total emissions, a figure that has nearly doubled over the last decade.[3] This amounts to a staggering 52.4 gigatons of CO_2 emissions. The computing industry now emits CO_2 on a similar level as the airline industry.[3] Estimates suggest that 1.3% of global emissions were related to software development and programming alone, also in 2020. This means that software development and programming independently account for over 60% of global computing emissions.[4] These numbers make it critical to consider the impact computers and technology have on the environment as they continue to play an essential role in our daily lives. They also call for taking action to reduce the carbon footprint of computing,[5] especially when it comes to programming and software development.

Programming languages are a fundamental part of the modern technology landscape and an essential part of many people's lives worldwide. These languages enable us to create software and applications that power everything, from mobile phones to complex artificial intelligence (AI) systems. Currently, there are hundreds of programming languages in use by developers, including older established languages that have been around for many decades, such as C and Fortran, as well as newer, specialized languages, such as Swift and Kotlin.[6]

Green computing seeks to use computing resources more sustainably. It has become increasingly crucial given the grave challenges the world faces surrounding climate change and environmental sustainability.[7] Since programming and software development are the key components of computing, more green languages have been designed to reduce the energy consumption of software applications to reduce their environmental impact.[8] Green programming languages are, therefore, an essential aspect of moving towards sustainable computing practices.[9]

Many factors can influence the energy consumption of programming languages. This includes the efficiency of the underlying hardware, code optimization, and energy-efficient algorithms, which reduce memory usage and the number of computations to solve a problem.[10] Green programming languages aim to significantly reduce energy consumption and carbon emissions in multiple ways. The current state of adoption of green programming languages is an area of significant interest, as many new green languages have been developed in the last decade, and there have been increased publications on the topic in recent years.

Some green programming languages have been widely used for decades, while others are less well-known and have yet to gain widespread adoption. Even as the development of these languages has drastically increased in recent years, the use of green programming languages has declined. Therefore, it is essential to understand the barriers to adopting these languages to identify ways to accelerate the shift towards greener programming languages and practices.

This paper makes three contributions. First, it offers an overview of the current developments and adoptions of green programming languages. Second, by examining the differences in energy efficiency across many programming languages, the paper will explore the difference between "green" languages and other languages, including their relative energy consumption and environmental impact. This includes both the direct effect of reducing energy consumption and the potential for broader environmental benefits. Finally, the review will highlight the concerns and limitations of green programming languages, such as the potential impact on performance, the need for additional training and resources, and the challenges of measuring and evaluating the environmental impact.

DOI: 10.36838/v6i5.4

Background information:

In this section, we first explain the concept of green computing, tying it back to a system used to "measure" how "green" a programming language is. Second, we show the energy efficiency of 28 different programming languages and explain how this connects directly to how sustainable these languages are. Third, we argue that green programming languages in our coding environment on a global scale are of the utmost importance by providing examples of negative repercussions that society will face if this downward spiral continues.

Green Computing and its Metrics:

Green programming languages are a class of software languages designed to support environmentally sustainable computing practices. These programming languages incorporate a range of methods to promote cleaner computing practices. However, a widely accepted definition of the specific criteria that define what makes a programming language "green" is yet to be agreed upon.[11]

Programming and software development can consume significant amounts of energy, especially in contexts such as data centers.[12] Memory allocation is a key aspect of programming that contributes to energy consumption.[13] Memory allocation refers to reserving memory space for variables, data structures, objects, etc. This lengthy process involves many steps, including finding available memory blocks, updating the memory management system, and initializing the memory for the program.[14] All these systems use large amounts of energy, particularly in programs with large memory capacities or high usage rates. Another aspect of computing that creates high energy demand is CPU utilization.[15] The CPU, or central processing unit, is a key part of a computing system responsible for executing instructions and performing computations. In programming, CPU utilization refers to the percentage of the time that the CPU is active and performing calculations.[16] Programming languages that require high CPU utilization often lead to large amounts of energy waste, as the CPU consumes a large amount of power when active.[15]

A second key factor contributing to computing emissions is the choice of programming language.[17] Some programming languages are known to be less energy-efficient than others, depending on their design and implementation. For example, some programming languages are designed for concurrent programming or parallelism, meaning that the language can process and execute multiple tasks simultaneously.[18] This essentially reduces energy consumption by enabling more efficient use of CPU resources.

Yet another factor that contributes to low energy efficiency in programming languages is handling inefficient code.[19] Although this can occur in every programming language, some languages are designed to tackle this issue to reduce energy consumption. For example, greener languages implement strategies such as automatic code optimization, refactoring, and testing to help promote energy efficiency and reduce energy waste.

Various approaches have been developed to optimize green programming languages. Some of these approaches include analyzing and optimizing the language's runtime performance to minimize energy consumption. Another strategy includes incorporating internal support for energy-efficient programming practices, such as minimizing the number of memory allocations. Other approaches involve implementing energy-related metrics to help programmers identify the most energy-intensive sections of their programs and provide them with solutions to further optimize these sections for energy efficiency.[20]

Despite the growing interest in green programming languages in the recent past, many of the most popular programming languages used today were developed many years, often decades ago, and need to incorporate the same tactics to optimize energy efficiency as modern languages, with the exception of C and C++. This makes it challenging to measure the energy efficiency of these older programming languages, which is directly linked to their environmental friendliness.[21]

The Energy Efficiency of 28 Programming Languages:

Many developers have created Computer Language Benchmarks (CLBG) on open-source platforms to assess the energy efficiency of programming languages. This initiative provides a framework for running, testing, and comparing different solutions for many programming problems that coders may have to develop while programming.[22] Currently, the CLBG has a framework for solutions for 13 common types of benchmark problems and compares the energy use of each solution possible in up to 28 programming languages.[23] Some problems are incompatible with specific programming languages, so these results are summarized in the following data table.[4]

Table 1: The energy consumption of 28 programming languages, expressed as a ratio to the greenest programming language C.[4]

Programming Language	Energy
C	1.00
Rust	1.03
C++	1.34
Ada	1.70
Java	1.98
Pascal	2.14
Chapel	2.18
Lisp	2.27
Ocaml	2.40
Fortran	2.52
Swift	2.79
Haskell	3.10
C#	3.14
Go	3.23
Dart	3.83
F#	4.13
JavaScript	4.45
Racket	7.91
TypeScript	21.50
Hack	24.02
PHP	29.30
Erlang	42.23
Lua	45.98
Jruby	46.54
Ruby	69.91
Python	75.88
Perl	79.58

DOI: 10.36838/v6i5.4

There are many situations where a programmer or software engineer must choose a particular language to implement an algorithm according to some functional or non-functional requirements. Applicable requirements are concerned with specific functions or features that software provides, while non-functional requirements are concerned with the overall quality attributes of software, such as performance and reliability. However, based on the problem at hand, the developer may prioritize one of these three variables: energy, time, and memory.[4] For instance, if developing software for wearables, it would be vital to apply energy-aware techniques that help save batteries. Another example would be if a developer implemented tasks running in a program's background. In this case, most of the time, execution time may be less of a concern to the developer than memory or energy use.

Green programming languages are directly tied with energy efficiency - meaning a less energy-efficient language would have a negative impact on the environment. In contrast, one that is more energy efficient would be more sustainable, having less of an impact on our environment.[21] This paper examines the environmental effects of different programming languages, which are tied directly to the energy efficiency of various software languages. This is why we will only look at energy efficiency (CLBG). C has a value of 1.00 (Table 1), which will be considered a "perfect value" or "target value" as it is the most energy-efficient programming language surveyed.[24] The other values for different languages are expressed as a ratio of the energy used by C, the most energy-efficient language. By this metric, a lower ratio indicates a language with an energy-efficiency level closer to that of C, meaning it is more energy-efficient. As the values get higher, the languages become less energy efficient. For example, the programming language Rust has an energy use ratio of 1.03 (Table 1), signifying it is nearly equal in energy consumption to C. Similarly, C++ has a ratio of 1.34 (Table 1), indicating it consumes 34% more energy than C. As we continue this investigation, we will regard C as the ideal language regarding energy efficiency. Although languages such as Pascal and Go were found to utilize less memory space to execute solutions to the benchmark suite[4], we will be primarily examining energy use, for which C is the ideal language.

Emerging programming languages (e.g., Ada, Julia, and Chapel) are more energy efficient. However, because they are not widespread, the study above does not cover them. This paper will cover some emerging languages, explicitly explaining how they decrease energy use.

The Importance of Green Programming Languages:

Given the planetary crisis humanity faces, implementing green programming practices on a global scale is crucial. Yet, despite the growing recognition of sustainable computing practices,[7] there needs to be more awareness and action.

If the transition to green languages does not occur swiftly, several negative consequences may follow, including:

- Increased energy consumption and carbon emissions from the computing sector exacerbate the already significant environmental impact of the tech sector. Computing emissions currently contribute over 2% of the world's emissions,[5] and are expected to increase by 40% by 2030.[26]
- Greater risk of energy supply chain disruptions and higher energy costs for organizations that rely heavily on computing resources.[27]
- Potential loss of competitive advantage for organizations that fail to adopt sustainable practices, as customers and investors increasingly demand environmentally responsible business practices.[27]
- Long-term sustainability concerns for the technology sector, as the availability of resources such as energy and rare earth metals become increasingly uncertain.[28]

In addition to the significant changes needed regarding the transition to greener programming languages, steps can also be taken to make popular programming languages more energy-efficient and, therefore, more sustainable. For example, popular languages like Java and Python only need 25-57% of the energy they consume while running to provide the same functions.[29] This missed potential is mainly due to such languages needing to be used efficiently and effectively. However, below are a few of the primary ways through which these languages are misused and how this can have an impact on the programming language's energy efficiency and environmental friendliness:

- Inefficient Data Structures and Algorithms:
    - These structures and algorithms can require more computational resources and increase energy consumption. For example, a programmer might use a simple sorting algorithm with a more significant time complexity (such as bubble sort) instead of a more efficient algorithm with a lower time complexity (such as quicksort). Similarly, using a data structure not optimized for a particular application or workload can lead to unnecessary memory allocation and increased energy consumption.[30]
- Poor Memory Management:
    - Poor memory management practices, such as failing to release memory when it is no longer needed or allocating too much memory, can lead to increased energy consumption. This is because allocating and deallocating memory consumes energy, particularly in systems with large memory capacities or high usage rates.[31]
- Lack of Optimization in Programs
    - Failing to optimize code for a particular platform or hardware architecture can lead to increased energy consumption. For example, a programmer might write code that could be optimized for a specific CPU architecture, leading to efficient use of CPU resources and increased energy consumption.[32]
- Excessive Network Use:
    - Applications that require excessive network usage, such as frequently making requests to remote servers or using large amounts of bandwidth, can lead to increased energy consumption. This is because network usage consumes energy, particularly in wireless communication systems that require the use of radio transceivers.[33]

- Inefficient Use of Parallelism in Languages
  - Programming languages that support parallelism or concurrency can enable more efficient use of CPU resources and reduce energy consumption. However, failing to take advantage of these features can lead to increased energy consumption. For example, a programmer might write code that does not take advantage of parallelism or concurrency, leading to inefficient use of CPU resources and increased energy consumption.[18]

Research suggests that if the above aspects are addressed to a small extent, the software language component of computing emissions can be reduced by up to 40%, which essentially cuts back on the environmental impact of the computing sector.[33]

Current State:

This section of the paper examines the current adoption state of programming languages, both green and non-green, based on data from the TIOBE index.[34] It further identifies trends in the usage rates of these languages to gauge whether their adoption has been increasing or decreasing over the past few years.

The TIOBE index, created in 2001, is widely cited as one of the most comprehensive sources for tracking the adoption of various programming languages across the globe. TIOBE stands for "The Importance of Being Earnest" and was named to emphasize the need for everyone to have more information on computing.[34] The TIOBE index tracks the popularity of around 200 programming languages based on various factors, including the number of skilled engineers worldwide, search engine results, third-party vendors supporting the language, courses, and more.[34]

While this index is one of the most widely used metrics for measuring the popularity of programming languages, it has potential drawbacks and inaccuracies. Its algorithm, which calculates the popularity of programming languages, has been criticized for not considering many other important factors, such as the number of job postings related to a software language, the number of open-source projects, and the growth rate of the language.

Additionally, the index relies heavily on search engine usage, which could be influenced by external factors such as media coverage rather than reflecting actual usage. Regarding search engine tracking, it only tracks in English, while many popular search engines use other languages, such as Chinese (Mandarin).[35] Based on the wide range of research papers using TIOBE, it does address the need to calculate the popularity of programming languages.

For this paper, the definition of "current" programming language popularity will be as of February 2023 (Table 2). However, the data on the TIOBE index continues to change month-by-month.[34] As of February 2023, below are the ten most popular programming languages, along with their usage percentages based on TIOBE's algorithm:

Table 2: The ten most popular programming languages in Feb 2023, their usage rates, and their change over the last year.[34]

Rank (Feb 2023), Name of Language	Usage Rate, Change in rate from Feb 2022
1, Python	15.49%, +0.16%
2, C	15.39%, +1.31%
3, C++	13.94%, +5.93%
4, Java	13.21%, +1.07%
5, C#	6.38%, +1.01%
6, Visual Basic	4.14%, -1.09%
7, JavaScript	2.52%, +0.70%
8, SQL	2.12%, +0.58%
9, Assembly Language	1.38%, -0.21%
10, PHP	1.29%, -0.49%

Despite the significant growth in the development of the green programming industry and the adoption of green programming languages over the past decade, most popular programming languages still need to be designed with sustainability in mind.[34] This means that the current state of adoption for green programming languages is relatively small compared to the overall usage of all programming languages worldwide.

However, there has been a significant increase in the development of green programming languages in recent years (Table 3).[23] Many new programming languages have been specifically developed to incorporate efficient energy practices, focusing on optimizing runtime performance to reduce energy consumption. However, the overall goal of a green programming language is to reduce the total energy consumption of computing and information technology (IT) systems.[4] Some of these programming languages include the following:

Table 3: Examples of green programming languages, attributes, and usage rates over the last year.[34]

Name of Language, Rank (Feb 2023), Change in rate from Feb 2022	Description
Go, 11, -0.12%	An open-source language developed by Google, designed to support concurrent programming, which makes it ideal for multicore processors.[36]
Swift, 15, -0.25%	Developed by Apple and used to develop software for macOS, iOS, watchOS, etc.[37] Focus on modern programming practices.
Rust, 20, +0.16%	Developed by Mozilla, popular for system-level programming and is used primarily in web browsers and video games.[38]
Julia, 34, -0.05%	Developed for numerical and scientific computing. Designed to be fast, dynamic and easy to use.[39]
Elixir, 62, -0.03%	A functional programming language that is designed to support fault-tolerant applications, primarily used in web development and real-time messaging.[40]

Unfortunately, the overlying trend among these programming languages is that they are reducing in popularity, while more prominent languages that are not as energy efficient and sustainable are growing in popularity, as shown in Table 2.[34] This shows that while the development of green languages, such as the ones discussed above has increased in recent years, the use of these languages has not been growing at nearly the same rate as development. The potential future effects of this trend will be discussed in the following sections.

DOI: 10.36838/v6i5.4

Despite more green programming languages being developed, their adoption is still relatively small compared to the overall usage of programming languages worldwide. However, as awareness of sustainability computing practices increases, there may be an increased interest in these languages, leading to more widespread adoptions in the years to come.

Differences between "green" languages and other languages:

Computing and programming have revolutionized how humans live, work, and communicate. However, the energy required to power and cool these digital devices has a significant carbon impact contributing to climate change. The exact carbon impact of programming is difficult to estimate as it depends on several factors, such as the programming language used, the complexity of the code, and the efficiency of the hardware running the program.[3] Nevertheless, studies have examined the energy consumption of specific programming languages and found that certain languages are more energy-efficient than others.[4] In this section of the paper, we show how many Joules (energy units) the Computer Languages Benchmark Game solutions consume in different programming languages. This value is the average amount of Joules consumed across the 13 different solutions in the CLBG across the 28 different languages. We aim to highlight the stark difference between green languages and non-green languages. We then isolate energy use from the programming itself - programming is only 1 part of the computing sector and isn't responsible for all the computing emissions - showing how it has a larger environmental impact than expected.

Comparing energy use (Joules) by the average program in different software languages:

The following figure (Figure 1) uses the Computer Languages Benchmark Games introduced earlier to estimate and visualize the energy use of C to run a program. The figure starts with C since it is the most energy-efficient programming language,[4] and can therefore act as a benchmark. The average C program across evaluated workloads in the benchmark suite requires 57 joules to run.[22] This energy comes from various sources, including the CPU, memory allocation processes, processing power, data use, and more.

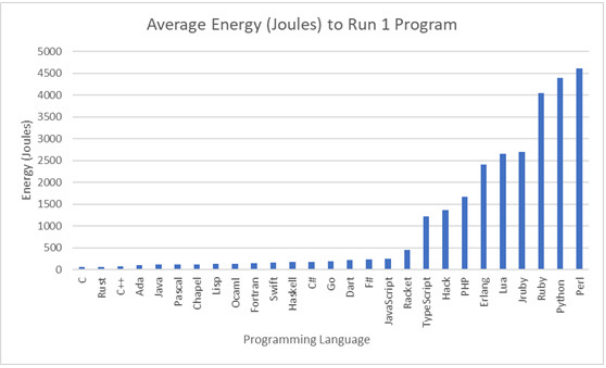

Figure 1: Visualizing the average energy (joules) to run one program across various programming languages.

A significant difference in the amount of energy consumed per program is seen when comparing the energy use of different programming languages. For example, the most energy-efficient language, C, only uses 57 joules per program, whereas the least energy-efficient language, Perl, consumes 4,604 joules per program. Figure 1 visualizes the significant differences in energy consumption between non-green languages such as Python and Perl and green languages such as C.

It is important to note that there are also efforts to optimize existing, popular programming languages for energy efficiency. For example, the LLVM compiler infrastructure is a popular project used to maximize the performance of C, C++, and other programming languages. LLVM includes features such as profile-guided optimization and in-time compilation, improving runtime performance while reducing energy consumption.[41]

A key trend noticed is that the world continues to move forward and increase its computing emissions; the transition to languages such as Python and JavaScript is only growing.[22] These languages, which are not green, consume hundreds of times more energy per program run than green software languages such as C and modern languages specifically designed to combat these pressing issues, as mentioned earlier.[4]

This staggering trend highlights the growing need to develop and adopt green programming languages. In addition, as the computing industry continues to expand, it's crucial to prioritize energy-efficient and sustainable practices to mitigate programming's negative impact on the environment.

Concerns and Limitations

Finally, in this section of the paper, we describe some of the concerns and limitations that may reduce the feasibility of switching to green programming languages while still arguing that it is the correct choice in most scenarios.

Green programming languages are designed to support sustainable computing practices, mainly focusing on optimizing energy efficiency to reduce the carbon footprint of computing systems. While the development and adoption of green programming languages is an important goal for many organizations and developers, there are many concerns and limitations to be considered when considering the switch to green programming languages.[42]

One of the most significant concerns regarding green programming languages is compatibility with existing systems and applications. An organization relying on systems or applications built long ago with non-green-oriented programming languages will find transitioning to a green programming language costly (resources and fund-wise) and time-consuming. More importantly, this transition process may also have a large carbon footprint, which should be considered. In some cases, it may not even be feasible to migrate these systems, as they may rely on software or hardware components that are not compatible with the components related to current green programming languages.

Performance is another potential limitation of green programming languages. While most green programming languages are developed to optimize energy efficiency, this may come at the cost of the overall performance of systems. For example, some green programming languages may effectively improve energy efficiency for smaller projects. However, it may require more memory or processing power to achieve the same

discussed in this paper, similar scaling issues could impact user experiences or even the overall efficiency of various systems. This is an important consideration, especially for applications needing high-performance levels or real-time processing.

Another potential limitation of green programming languages is the expertise required to develop and maintain applications that use them. Since these green software languages are relatively new compared to older, well-established ones, fewer resources may be available to help developers learn and use them most effectively. This may also make hiring and retaining talent harder for companies and organizations that rely on green programming languages, especially if there is a shortage of skilled developers in the hiring market.

Cost is also important when transitioning from older, popular languages to green ones. High costs may be associated with retraining developers, acquiring new hardware or software tied to green software languages, and maintaining existing systems during transition periods. In some cases, the cost of transitioning to green programming languages may outweigh the potential benefits of reduced energy consumption or improved sustainability. This is particularly important for smaller-sized businesses, which may need more money/resources to invest in the transition to green programming languages.

Finally, there may be better choices than green programming languages for all use cases or applications. In some cases, the trade-offs associated with using a green programming language (including reduced performance or increased complexity) may not be worth the potential benefits of improved energy efficiency or sustainability, especially while more research still needs to be done in this field. For example, financial trading algorithms or gaming applications prioritize performance over efficiency. They may be slower to adopt these green systems because they do not run as fast as non-green systems. Therefore, it is important to consider the specific needs and requirements when deciding whether to adopt green programming languages.

In many cases, green programming is clearly the best route for an organization to reduce its carbon footprint and increase sustainability. However, this is not the sole solution to reducing a programmer's or organization's footprint. Facility-level optimizations, platform-level policies, and virtualization technologies can also improve energy efficiency, highlighting the need to consider a holistic approach that incorporates a combination of these strategies to achieve meaningful reductions in energy consumption.

Discussion & Conclusion

Green programming languages are a promising approach for slowing or even reversing the worsening energy consumption and growing environmental impact from the computing sector, which could slow major environmental issues, including climate change. This argumentative review helps explain the adverse effects of the continued use of non-green programming languages such as Python and Perl and the use of these languages in energy-consuming environments.

The switch to greener programming languages should be aggressively pursued, with more funding to help large systems that heavily rely on non-green programming languages switch to greener languages. However, it is essential to distinguish between programming languages themselves and their specific implementations or runtime environments. While certain programming languages may be associated with higher energy consumption, such assessments should account for the variability in energy efficiency across different language implementations and runtimes. For example, languages like Python and Java have multiple implementations, each with its own energy efficiency characteristics (ex. CPython, IronPython, Jython, PyPy). Comparing these can also give us insights into the relative energy efficiency of each option.

As explained in the paper, while many more green software languages have been developed and popularized over the last decade, coders worldwide continue migrating to non-green languages as their primary tool when coding. There are many reasons for this to occur, including the potential concerns and limitations green programming languages bring to software systems and corporations, as discussed in section 4 of the paper. In contrast, awareness of green languages only continues to grow.

While we have shown the average amount of energy it takes to run a program in 28 different languages based on energy efficiency, it still needs to be determined how much energy a single programming language consumes in terms of total world energy use. However, it is pretty clear which programming languages consume more energy than others, which enables us to determine which languages are "green" and which are "non-green."

Recent advancements and developments regarding green programming languages should function by stopping or reversing total energy use in the computing sector, relieving pressure on the environment and computing systems worldwide.

Acknowledgments

I want to give special thanks to my amazing mentor, Dr. Samar Sabie, an assistant professor at the University of Toronto. She helped me navigate the process of writing my first research paper and supported me while guiding me through researching developments in green computing. I also wanted to thank my parents, sister, and dog for their continued support and dedication to me!

References

1. US, G. Electricity production and distribution. https://afdc.energy.gov/fuels/electricity_production.html#:~:text=According%20to%20the%20U.S.%20Energy,power%2C%20biomass%2C%20and%20geothermal/.
2. Berkeley, D. Burning of fossil fuels. https://ugc.berkeley.edu/background-content/burning-of-fossil-fuels/
3. Master, F. Computing industry CO2 emissions in the spotlight. https://www.reuters.com/article/us-computing-carbon-emissions/computing-industry-co2-emissions-in-the-spotlight-idUSTRE50E5QO20090115.
4. Pereira, R.; Couto, M.; Ribeiro, F.; Rua, R.; Cunha, J.; Fernandes, J. P.; Saraiva, J. Energy Efficiency across Programming Languages : How Do Energy, Time, and Memory Relate? *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering* 2017, 256–267.
5. Li, Q.; Zhou, M. The Survey and Future Evolution of Green Computing. *2011 IEEE/ACM International Conference on Green Computing and Communications 2011*.

DOI: 10.36838/v6i5.4

6. Online historical encyclopedia of programming languages. https://hopl.info/.
7. Saha, B. Green Computing. *International Journal of Computer Trends and Technology* 2014, *14* (2), 46–50.
8. Raza, K.; Patle, V. K.; Arya, S. A Review on Green Computing for Eco-Friendly and Sustainable It. *Journal of Computational Intelligence and Electronic Systems* 2012, *1* (1), 3–16.
9. Soomro, T. R.; Sarwar, M. Green Computing: From Current to Future Trends. *World Academy of Science, Engineering and Technology International Journal of Humanities and Social Sciences* 2012, *6* (3), 1–4.
10. Georgiou, S.; Kechagia, M.; Louridas, P.; Spinellis, D. What Are Your Programming Language's Energy-Delay Implications? *Proceedings of the 15th International Conference on Mining Software Repositories* 2018, 303–313.
11. Mahadevappa, S.; Figueira, S. Energy-Efficient Programming Languages for Mobile Applications. *2021 IEEE Global Humanitarian Technology Conference (GHTC)* 2021
12. Dayarathna, M.; Wen, Y.; Fan, R. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys & Tutorials* 2015, *18* (1), 732–794.
13. Liao, X.; Jin, H.; Yu, S.; Zhang, Y. A Novel Memory Allocation Scheme for Memory Energy Reduction in Virtualization Environment. *Journal of Computer and System Sciences* 2015, *81* (1), 3-15.
14. Serrano, M.; Boehm, H.-J. Understanding Memory Allocation of Scheme Programs. *Proceedings of the fifth ACM SIGPLAN international conference on Functional Programming* 2000, 245–256.
15. Yao, F.; Demers, A.; Shenker, S. A Scheduling Model for Reduced CPU Energy. *Proceedings of IEEE 36th Annual Foundations of Computer Science* 1995.
16. Bui, D.-M.; Nguyen, H.-Q.; Yoon, Y. I.; Jun, S. I.; Amin, M. B.; Lee, S. Gaussian Process for Predicting CPU Utilization and Its Application to Energy Efficiency. *Applied Intelligence* 2015, *43* (4), 874–891.
17. Cassel, D. Which programming languages use the least electricity? https://thenewstack.io/which-programming-languages-use-the-least-electricity/.
18. Porterfield, A. K.; Olivier, S. L.; Bhalachandra, S.; Prins, J. F. Power Measurement and Concurrency Throttling for Energy Reduction in OpenMP Programs. *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum* 2013, 884–891.
19. Capra, E.; Francalanci, C.; Slaughter, S. A. Is Software "Green"? Application Development Environments and Energy Efficiency in Open Source Applications. *Information and Software Technology* 2012, *54* (1), 60–71.
20. Herelius, S. Green Coding: Can We Make Our Carbon Footprint Smaller through Coding? *Independent* 2022, *39*.
21. Szydlowska, N. Green Software Development: Energy Efficient programming languages, tools and practices in coding. https://curiosum.com/blog/green-coding-software-development-energy-efficient-programming-languages/.
22. Shun, J.; Blelloch, G. E.; Fineman, J. T.; Gibbons, P. B.; Kyrola, A.; Simhadri, H. V.; Tangwongsan, K. Brief Announcement. *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in Algorithms and Architectures* 2012, 68–70.
23. Couto, M.; Pereira, R.; Ribeiro, F.; Rua, R.; Saraiva, J. Towards a Green Ranking for Programming Languages. *Proceedings of the 21st Brazilian Symposium on Programming Languages* 2017, No. 7, 1–8.
24. Lott, C. C is the greenest programming language. https://hackaday.com/2021/11/18/c-is-the-greenest-programming-language/.
25. Teschler, L. The top ten "green" programming languages. https://www.designworldonline.com/the-top-ten-green-programming-languages/.
26. IEA, I. E. A. Digitalization and energy – analysis. https://www.iea.org/reports/digitalisation-and-energy.
27. Wang, D. Meeting Green Computing Challenges. *2008 10th Electronics Packaging Technology Conference* 2008, 121–126.
28. Jones, N. Scarcity of Rare Metals Hinders Green Technologies. *Global Warming - So What?* 2013, 1–5.
29. Lima, L. G.; Soares-Neto, F.; Lieuthier, P.; Castor, F.; Melfe, G.; Fernandes, J. P. Haskell in Green Land: Analyzing the Energy Behavior of a Purely Functional Language. *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)* 2016.
30. Jung, C.; Rus, S.; Railing, B. P.; Clark, N.; Pande, S. Brainy: effective selection of data structures. *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation* 2011, 46 (6), 86–97.
31. Issenin, I.; Brockmeyer, E.; Miranda, M.; Dutt, N. DRDU: A Data Reuse Analysis Technique for Efficient Scratch-Pad Memory Management. *ACM Transactions on Design Automation of Electronic Systems* 2007, 12 (2), 15.
32. Babu, S. Towards Automatic Optimization of MapReduce Programs. *Proceedings of the 1st ACM Symposium on Cloud Computing* 2010, 137–142.
33. Farhan, L.; Kharel, R.; Kaiwartya, O.; Hammoudeh, M.; Adebisi, B. Towards Green Computing for Internet of Things: Energy Oriented Path and Message Scheduling Approach. *Sustainable Cities and Society* 2018, *38*, 195–204.
34. Tiobe index. https://www.tiobe.com/tiobe-index/.
35. Morales-Arroyo, M. A.; Spink, A. Multimedia Search Capabilities of Chinese Language Search Engines. *Information Processing & Management* 2010, *46* (3), 308–319.
36. Meyerson, J. The Go Programming Language. *IEEE Software* 2014, *31* (5), 104–104.
37. Goodwill, J.; Matlock, W. The Swift Programming Language. *Beginning Swift Games Development for iOS* 2015, 219–244.
38. Matsakis, N. D.; Klock, F. S. The Rust Language. *Proceedings of the 2014 ACM SIGAda annual conference on High Integrity Language Technology* 2014, 34 (3), 103–104.
39. Bezanson, J.; Chen, J.; Chung, B.; Karpinski, S.; Shah, V. B.; Vitek, J.; Zoubritzky, L. Julia: Dynamism and Performance Reconciled by Design. *Proceedings of the ACM on Programming Languages* 2018, *2* (OOPSLA), 1–23.
40. Vegi, L. F.; Valente, M. T. Code Smells in Elixir. *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension* 2022, 580–584.
41. Lattner, C.; Adve, V. The LLVM Compiler Framework and Infrastructure Tutorial. *Lecture Notes in Computer Science* 2005, 15–16.
42. Nardi, B.; Tomlinson, B.; Patterson, D. J.; Chen, J.; Pargman, D.; Raghavan, B.; Penzenstadler, B. *Computing within limits* 2018, *61* (10), 86–93.

Author

Arya Kashyap is a high school sophomore currently enrolled in the IB diploma program in Seattle, USA. He has always been interested in coding and excited by environmental sustainability. He was intrigued to find a way to tie these concepts together, which led to his interest in learning and investigating the field of green computing.