The Origins of Python and Its Use in Current Technologies

Author: Sarah Parker

Department of Computer Science, University of Tennessee, Chattanooga

CPSC5100: Theory of Programming Languages

December 5, 2024

The Origins of Python and its Use in Current Technologies

Programming languages are the heart of computer science. High-level languages like Fortran, COBOL, and LISP paved the way into the Information Age by allowing people to interact with computers using simple terms. Without such languages, people would still be forced to operate computers using low-level languages such as binary or machine language. These low-level languages, although practical, had limitations and were very cumbersome to write and debug. While over 8,000 programming languages have been created since the 1940s, less than 2,000 are still in use, and only 10-15 make up the majority of languages used today (Jansen, 2024).

Python quickly became the most popular programming language, overtaking Java in 2018 (“PYPL Popularity,” 2024). It accounts for 22% of programming tutorial searches, followed by C++ at 11% and Java at 10.51% (TIOBE, 2024). Python was created to be programmer-friendly with its user-friendly semantics, low learning curve, duck typing, and ease of debugging. Although easy to use, it still provides a robust, safe framework that rivals its predecessors, such as C, FORTRAN, COBOL, C++, and Java. Python also received the Programming Language of the Year award 4 times between 2007 and 2020 ("Python Programming Language," 2024. This award is given to languages with the highest rise in rankings, and Python is the only language to have received this award more than 3 times. Python is used in several large organizations’ tech stacks, such as Wikipedia, Google, NASA, Amazon, and Reddit ("Python Programming Language," 2024. This paper explores Python's origin and future prospects by evaluating its’ semantics, advantages and disadvantages, libraries and extensions, and uses in current technologies.

History of Python

Origin

Guido van Rossum created Python in 1989 (Python, 2024). In Amsterdam, he worked on the Amoeba project with Centrum Wiskunde & Informatica (CWI). The primary goal of the Amoeba project was to make several computer networks appear as a single computer on a distributed Kernel (Sevrance,

2015). However, van Rossum and his team encountered overwhelming challenges when writing several user-level tools in the C language. Guido van Rossum decided he could build a new language that would better meet their needs and take less time than continuing to write the utilities in the C language. Van Rossum drew from his prior experience with the ABC, an imperative language developed at Centrum Wiskunde & Informatica ("ABC Programming Language," 2024). ABC had several limitations, such as being unable to communicate with the OS (operating system) or file systems ("ABC (programming language)," 2024). This prevented it from being used in systems programming, like the Amoeba project. However, some features of ABC were straightforward to use, such as variable types not needing to be declared, and van Rossum drew heavily from these features when creating Python. He started working on it in December 1989, and within three months, he demonstrated Python's first rudimentary programming capabilities to his colleagues (Severance, 2015). This garnered immediate interest, and others joined in to help make the language fully functional. A year after starting the project, the first version (0.9.0) of Python was released ("Python Programming Language," 2024).

Evolution of Python

Van Rossum saw the potential for widespread use and released Python as free software on Usenet (Severance, 2015). Interest and suggestions for improvement continued to be gathered as more users joined the open-source project. Arguably, one of Python's most critical initial modifications was adjusting it to work seamlessly on different computer architectures, not just Unix machines (Severance, 2015). Python's next milestone came in 1994 when van Rossum was offered a job from the CNRI (Corporation for National Research Initiatives) to specifically keep developing and expanding Python (Severance, 2015). Although there were several updates and releases over the years, Python's next biggest release would come in 2000 with the release of Python 2.0. This version included many vital new features and updates, including "list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support" ("Python (programming language)," 2024). Python 3.0 was released in

2008 with the addition of a utility that would automatically translate code from Python 2 to Python 3. The current version of Python is 3.13. Over the past four years, numerous vulnerabilities have been found within the language, including potential remote code execution, web cache poisoning, and denial of service. This accelerated several versions of Python 3.1x releases to resolve these issues and improve security ("Python Programming Language," 2024). Van Rossum officially retired in 2018; he was fondly called Python's "benevolent dictator for life" by the Python community in recognition of his long-standing commitment as a lead developer ("Python (programming language)," 2024). He came out of retirement in 2020 and joined Microsoft in the developer division (Lardinois, 2024). A non-profit organization now maintains Python called the Python Software Foundation.

Technical Aspects of Python

Definitions

In its simplest definition, Python is a dynamic, object-oriented, duck-typed, and strongly typed language. The following sections will explore these definitions more thoroughly. When translating a program from a high-level language to machine language, most programming languages fall into two categories: interpreted or compiled. Compiled languages require a compiler to evaluate the program before execution, while interpreted languages evaluate the program line by line during execution.

Object-Oriented Language

Python is an object-oriented language. Geeks for Geeks defines the primary goal of object-oriented programming as "to bind the data and the functions that work together as a single unit so that no other part of the code can access this data" ("Python OOP," 2024). Python utilizes modules and classes to write maintainable, scalable code, preserves encapsulation, promotes code reuse, and employs polymorphism, inheritance, and data abstraction ("Python OOP," 2024). Most languages commonly used for programming can support object-oriented programming ("Object-oriented

programming," 2024). Van Rossum's Python design stemmed from creating a highly extensible language via modules, interfaces, and an extensive standard library ("Python Programming Language," 2024). The vastness of libraries in Python makes it versatile and easy to use for all types of programs, applications, data evaluation, and emerging technologies. Python's libraries and extensions will be discussed later in this paper.

Typing system

Python is dynamic and duck-typed. Dynamic languages assign data types to the interpreter during program execution. Statically typed languages assign variable types during compilation and are often (but not always) compiled. Dynamic typing can lead to more significant runtime errors as type errors are not caught until execution. Some feel that debugging can be easier with dynamic languages as the interpreter stops the program and shows the line of code where the error occurred (Khoirom et al., 2024). Therefore, only one line of code is debugged at a time. However, others feel this is more challenging as it can be difficult to trace the error back to the source if it occurs further away from the root problem (Lewis, 2021). Moreover, this type of debugging can be tedious and time-consuming as the program must be rerun every time an error is fixed to see if there are any other errors.

Additionally, interpreted and dynamic languages are much slower to execute than static and compiled ones. However, some tools are available to help improve performance. Cython is an extension of Python that allows types to be explicitly declared and converted into a C variable. The returning C variable is converted into a Python object when the operation is completed. Additionally, the code is statically typed and compiled first (Cythonized) and then executed. Although this may seem like the additional steps of converting would take the program even longer to run, the opposite is true. In an analysis by a company called Future Learn, they executed two high-performance computing codes, one in pure Python and one that was Cythonized. The analysis between the two showed that the pure

Python code took 0.57s to run, while the Cythonized code took 14ms. When analyzing performance between programs, the speedup is calculated at the original time divided by the new time. In the analysis above, this equates to a speedup of approximately 40x faster ("*Using Static,"* n.d.). This is a significant improvement in program performance. Therefore, programmers should explore these optimizing extensions if performance is an issue.

Typing of a language, assigning a data type to a value, can occur explicitly (programmer-specified) or implicitly. Python employs a method called duck typing, a form of implicit typing. This is a powerful feature of Python, as it frees the programmer from having to specify and write unnecessarily verbose code. Duck typing allows polymorphisms in the language as it cares less whether the object is a "duck" as long as it "quacks" (Myrianthous, 2024). Python introduced type hints in version 3.5. With this feature, the programmer can explicitly state variable types in the code. However, the actual data type is still implicitly assigned when executing the program. This function is more for readability and clarity of code as it does not affect the actual data type ("Support for," n.d.). Mypy is another optional extension that adds static typing with a compiler (Mypy, 2024). This allows programmers to add static types to Python code that can be checked during compilation. It also boasts the feature of mixing duck typing and static typing. Programmers can write one function using static typing and another using duck typing. For example, you can write a function with static typing as

```
def adding (a: int , b :int) -> int:
    return a + b


adding (3, 6) -→ Correct // outputs 9
adding (3, "6") -→ Incorrect// cannot mix string and int
adding ("Hello", "World") → Incorrect // use of strings not allowed
```

Mypy will check the function, ensuring only integers are passed as parameters. Otherwise, a type error will result if a different data type (such as a string) is passed in as a parameter. A function with pure duck typing would be written as

```
def adding (a, b):
    return a + b
adding (3, 6) → Correct // outputs 9
adding (“Hello”, “ World”) → Correct // outputs “Hello World”
adding (3, “Hello”) → Incorrect // error
```

Since duck typing implicitly assigns the type, there is more flexibility with the data types passed in as parameters. Strings can be passed in as parameters because there are rules for concatenation with strings using the + symbol. Therefore, in our second example, the resulting output is “Hello World.” However, since no rules allow adding or concatenating a string and an integer, the third example results in an error. This last example also demonstrates that Python is a strongly typed language.

Strongly typed languages refer to the concept that a language has strict rules regarding types and the operations that can be performed. A strongly typed language can enforce these rules during the compilation or execution of a program. Weakly typed languages indicate looser rules and implicit type conversions may happen more frequently. This can increase the chance of incurring runtime errors or inaccurate results ("Strong and Weak," 2024). As Python is strongly typed, there are minimal implicit conversions among data types, and the rules regarding types and operations are strongly enforced during runtime. Some implicit conversions may occur for numeric data types, such as integers, floats, and doubles ("Python Syntax,” 2024). In the last example, <adding (3, “Hello”)> would result in an error for two reasons. The first was described above. The second is that Python does not implicitly convert data types. Therefore, it does not convert the int 3 to a string “3”. JavaScript is an example of a weakly

typed language that allows implicit conversion. If the preceding example were written in JavaScript, it would be written as

```
function adding (a, b) {
        return a + b;
}
console.log(adding(3,6) → correct //outputs 9
console.log(adding("3", 6)→ correct//outputs 36
```

In the second output, six was implicitly converted to a string, and then the concatenation of "3" and "6" occurred. This results in the output "36." This exemplifies how inaccurate results can be produced using weakly typed languages.

Syntax and Semantics

Python was meant to be easy to read by using keywords and removing punctuation such as curly braces and semicolons to delimit blocks and statements that its predecessors (such as ABC) used. Instead, it relies on whitespace and indentation to delineate blocks of code. Python requires using four spaces (or an indentation if set appropriately) to set apart control flow blocks, unlike other languages, such as Java, which require curly braces ("Python Programming Language," 2024). While the lack of extra symbols makes Python visually appealing, it can be challenging for beginning developers as a simple extra space on an indentation line can be exceptionally difficult to locate. However, several extensions for use within an interactive development environment (IDE) can help check the syntax structure of a code. Python for VS Code provides code formatting, syntax checking, and debugging features, while Python Indent will give the accurate number of spaces/indentations when hitting the enter button ("Top 10," 2024). These extensions can provide valuable and time-saving tools that allow programmers to focus on the functionality and logic of their programs.

Python uses a “call by object reference” model for passing, changing, and accessing data types ("Python Syntax," 2024). This means if a mutable variable (i.e. (dictionaries, sets, lists) is passed through to a function, it is referencing the actual variable, and any changes made within the function can alter the original variable as well. However, if an immutable object (i.e., string, tuple, or whole number) is passed through, it cannot be altered inside the function (“Is Python,” 2023). This is termed call by value. This model is essential for programmers to understand when writing code to avoid accidentally changing variables that need to be used in their original value in other program parts. Unlike other languages, Python does not have a way to make variables constant or immutable (Hunner, 2021). Defining a variable that should remain unchanged by naming it in all capital letters is considered good practice. However, this does not prevent the variable from being changed during execution (Hunner, 2021). Instead, it acts as a signal to programmers working on that program. This is only relevant to mutable objects such as dictionaries.

Advantages and Disadvantages

Python has numerous advantages, including flexibility, ease of learning, a massive open-source community, code maintainability, code reuse, and beginner-friendly syntax (Khoirom et al., 2020). However, Python’s biggest strengths lie in its extensive and comprehensive libraries and extensions. The Python standard library contains over 200 core modules primarily written in C (“Libraries in Python” 2024). However, it is estimated that there are over 137,000 total libraries for Python (Parashar, 2024). Although there are too many libraries to explore in this paper, some more popular ones will be quickly mentioned. Often used in machine learning and deep learning algorithms, PyTorch, PyBrain, and TensorFlow are libraries for high-level computations, artificial learning, and neural networks. Data scientists use the Pandas library extensively to evaluate and analyze data structures (“Libraries in Python,” 2024). PyGame is used to help develop video game graphics and audio. Multiple other libraries

help optimize code, provide data visualization, perform web scraping, and provide operating system interfaces.

Despite all the advantages Python offers, it is vital to understand its limitations. Since Python is interpreted, it can take much longer to execute a program when compared to other languages that are compiled, such as Java and C++ (Gupta, 2024). On average, Python code (without optimizations) is 10- 40 times slower than Java. However, this is where extensions such as Cython can help drastically improve performance. Pypy is another implementation that provides a just-in-time compiler feature ("Python Programming Language," 2024). Sudra (2024) also found that "optimizing code using PyPy has shown that code performance exceeds Java by up to 1.8x". The data structures built in Python take up considerable amounts of memory.

Because of its dynamic typing, Python uses a lot of memory. It tends to over-allocate memory space for objects, especially data structures (Barasa. 2024). Python uses a garbage collection system to free up memory from objects no longer used in the program. However, this collection system is not the most robust, and memory leaks occur if not used efficiently. This results in allocated but not actively used memory in the program (Barsa, 2024). One user on Stack Overflow (2019) found that executing a program that made a list of prime numbers took nine times more memory in Python than the same program written in C. There are several memory profiling tool extensions, dependencies, and ways to optimize Python code to manage memory more efficiently, but that is beyond the scope of this paper.

Safety is another concern with Python. Developers should be aware of known security issues with Python and ways to mitigate these risks. The two main issues are database vulnerabilities such as SQL and directory traversal attacks. The first issue concerns Python and its use of databases such as SQL. Van Der Made (2022) stated that "a command injection, SQL injection, is an attacker's ability to run any commands or code on a target machine" through user inputs passed in a function. Compared to other

languages, Python's database security layers are considered underdeveloped and less robust when compared to other Languages and, therefore, are less likely to be used by large businesses with extensive databases (Khoirom et al., 2022). Another security issue is a directory traversal attack. This issue is also caused by incorrect user input and can result in confidential files being exposed (Van Der Made, 2022). Given that Python's popularity is due primarily to its libraries, running programs with outdated libraries can unintentionally leave programs vulnerable to bugs and attackers. Developers must ensure they update libraries and packages regularly to ensure safe code. Broken access control is a vulnerability within a program that allows unauthorized users to read or write parts of a program that they should not be able to access. In his article, Vand Der Made (2022) reported that "approximately 94% of applications had some form of broken access control". This can be better secured with appropriate validation and verification methods.

Another limitation of Python is that it can only run a program on a single processor. The Global Interpreter Lock (GIL) allows only one thread to execute at a time (Khoirom et. al., 2020). Therefore, while a program can execute several threads, the threads will run concurrently, not parallelized, and performance will not improve. This limits its usefulness for programs that require hyperthreading or parallelization using multi-processing methods. Some libraries, such as NumPy and SciPy, utilize C/C++ and Fortran to bypass Python's limitations and parallelize applicable code as specified (Khoirom et al., 2020). Parallelization is not inherent to the standard Python language and must be linked to an appropriate library to optimize code in this fashion.

Current Technologies

Applications and Websites

Python is often the language for advanced technologies and applications, including quantum computing, data visualization, machine learning and AI (artificial intelligence), audio and visual

applications, and game development (JayDevs, 2023). Several well-known, large companies use Python heavily in their tech stack, including Google, Netflix, Dropbox, Reddit, Spotify, Datadog, NASA, and Amazon. Google was one of the first companies to implement Python into their tech stack. This was primarily due to van Rossum's career at Google in the early 2000s. His most notable contributions during that time were building Mondrian, an internal code review tool, and the App Engine (van Rossum, n.d.). Python is currently used at Google for analytical algorithms in conjunction with search engines for video processing and data analytics (JayDevs, 2023). Reddit (n.d.) is a popular social site that boasts being a "home to thousands of communities…and authentic human connections" and currently has about 430 million monthly users. When Reddit saw increased users and web traffic, it moved its entire tech stack from Lisp to Python as they needed a more robust system to handle the increased burden. Python manages the Reddit website's infrastructure, caching system, search engine, and message queue (JayDevs, 2023).

Data Science and analytics

In 2019, Python was ranked as the number one programming language for use in analytics, data science, and machine learning (Raschka et al., 2020). Some fundamental data analysis concepts are gathering, processing, and visualization. The vastness of Python libraries significantly assists data scientists in accomplishing these goals. Libraries such as BeatifulSoup and Scrapy can gather data from websites and create structured data sets from this information. Processing data is made more accessible using Python libraries that perform complex computations, such as Pandas or NumPy. Furthermore, data scientists must be able to share the insights from their analysis accurately, often using visualization tools. Matplotlib is a popular library for visualization as it transforms data into pie charts, graphics, and diagrams ("Why Do," n.d.). The ability to handle and correctly analyze extensive data is why data science is intricately linked to machine learning, neural networks, deep learning, and artificial intelligence.

Machine Learning and Artificial Intelligence

More recently, Python has been extensively used in machine learning (ML) and artificial intelligence (AI). Raschka et al. (2020) define machine learning as “the study and development of approaches that automate complex decision-making, as it enables computers to discover predictive rules from patterns in labeled data without explicit instructions.” Although machine learning is intimately interconnected with AI, it is important to distinguish their differences. Raschka et al. (2020) define AI as “computer programs and machines capable of performing tasks that humans are naturally good at, including natural language understanding, speech comprehension, and image recognition.” The distinction may seem small, but where ML focuses on the algorithms used to automate processes, AI focuses on using algorithms to provide context (understanding) of the problems and generate solutions. Although AI was recognized as a scientific discipline in 1956, it was not largely successful until the 2010s when there was a sudden increase in computing power (use of graphic processing units (GPUs)) and the extensive amount of data now available for use in training models ("Artificial intelligence," 2024). The extension CPython is often extensively used for ML and AI. However, Python cannot execute parallel threads, significantly limiting its speed and usefulness. This is where Python’s libraries are of extreme benefit. Significant time and energy have been spent developing NumPy and SciPy. These libraries utilize C/C++ and Fortran code to use parallelization with multi-processors and threads effectively (Raschka et al., 2020). This highly effective solution allows code to be written in Python but works around Python’s limitations with parallelization

Quantum Computing

Quantum computing can be thought of as the next generation of computers. It uses a different architecture from what currently exists and seeks to solve extensive computations that are impossible (due to time limitations) with current computers ("Quantum computing," 2024). Current computer

architectures operate on a binary level: a state is either a 1 or 0 (aka. Bit). Currently, all personal computers are built using binary operations on these bits. Quantum computing is based on a scientific phenomenon where physical matter can exhibit both properties of particles and waves simultaneously ("Quantum computing," 2024). Unlike traditional computers, quantum computing uses the qubit, which can consist of the traditional 1 or 0; however, it is notated as |0> and |1>. It can also exist in a superposition state where it is in both states simultaneously by applying a vector constant and then added together ("Quantum computing," 2024). This fundamental difference in quantum computing allows computations to execute exponentially faster than traditional computer architecture. Quantum computing is still highly experimental but has significantly progressed in the last ten years. One of the most significant advancements, yet widely argued, came in 2019 when Google claimed it had achieved quantum supremacy ("Quantum computing," 2024). Google stated that using quantum computing, they completed a calculation in three minutes, which would have taken 10,000 years to accomplish with a typical supercomputer (Roush, 2020). IBM, Google's biggest rival in quantum computing, refuted this claim, stating it would have only taken a supercomputer a few days to complete the calculation (Roush, 2020). They have not, however, produced evidence for their claim.

Python has quickly become the language of choice in the development of quantum computing. Python has been used to develop quantum software and hardware such as computing simulators (i.e., reference-qvm), compilers (i.e., PyZX), and full-stack libraries (i.e., Project Q, XACC, and Qubiter) (Fingerhuth et al., 2018). Since IBM enabled public cloud access to their five qubit quantum processor in 2016, they have garnered a large community of developers for their quantum software library, Qiskit, written in Python (Fingerhuth et al., 2018). According to IBM, Qiskit has become the most popular software library in quantum computing due to its open-source contributions and ability to "build circuits, leverage Qiskit functions, transpile with AI tools, and execute workloads in an optimized runtime environment." ("Qiski*t," n.d.)* The usefulness of quantum computing strives to have real-world

applications that can improve cybersecurity, data analytics, artificial intelligence, optimization, and simulation ("Quantum computing," 2024). With hopes of exponential economic growth, quantum information science received over 2 billion dollars from private investors in 2022 (Bogobowicz et al., 2023). Additionally, several agencies geared toward quantum technologies allocate over 200 million dollars of federal funding (“Annual Report,” 2023). In fact, due to this funding, the National Science Foundation awarded $800,000 to the University of Tennessee, Chattanooga, towards the Quantum Information Science and Engineering (QISE) program. This funding will enable the QISE program to research a “novel theoretical and experimental scheme for demonstrating distributed quantum network in downtown Chattanooga” (Stafford, 2024).

Conclusion

The future of Python remains bright. It is frequently the language of choice in all categories of technology. Furthermore, it has become the most popular choice for new developers seeking to learn programming and advanced programmers seeking a robust language that can handle heavy computations. Although Python is still considered a newer language, it has firmly established itself as one of the most versatile, easy-to-use, and robust programming languages of modern times. Despite its humble beginnings as a hobby project, it is now used in websites, applications, and advanced technologies such as machine learning, AI, and quantum computing**.** While other languages outperform Python in speed and security, its success is primarily due to its vast libraries and solid open-source community that works diligently to create new tools for Python developers. For this reason, Python will likely remain one of the most popular programming languages as technologies continue to advance.

References

ABC (programming language). (2024, July 29). In *Wikipedia*. https://en.wikipedia.org/wiki/ABC_(programming_language)

Annual report on the NQI program budget. (2023, December 1). National Quantum Initiative. https://www.quantum.gov/

Artificial intelligence. (2024, November 1). In *Wikipedia*. https://en.wikipedia.org/wiki/Artificial_intelligence

Barasa, M. (2024, November 1). *Your guide to reducing python memory usage.* Honeybadger. https://www.honeybadger.io/blog/reducing-your-python-apps-memory-footprint/

Bogobowicz, M., Gao, S., Masiowski, M. Mohr, N., Soller, H., Zemmel, R., and Zesko, M. (2023, April 24). *Quantum technology sees record investments, progress on talent gap.* McKinsey Digital. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/quantum-technology-sees-record-investments-progress-on-talent-gap

Fingerhuth, M., Babej, T., & Wittek, P. (2018). *Open source software in quantum computing*. PLOS *ONE*.*13*(12), e0208561. https://doi.org/10.1371/journal.pone.0208561

Hunner, T. (2021, April 14). *Does Python have constants?*. Python Morsels. Retrieved October 28, 2024, from https://www.pythonmorsels.com/python-doesnt-have-constants/

Is Python a call by reference or a call by value? (2023, December 13). Geeks for Geeks. from https://www.geeksforgeeks.org/is-python-call-by-reference-or-call-by-value/

Gupta, G. (2024, May 12). *Advantages and disadvantages of Python* – make a favorable decision. Squareboat. https://squareboat.com/blog/advantages-and-disadvantages-of-python

JayDevs (2023, May 23). Top 30 companies that use Python for success and profit. *JayDevs*. https://jaydevs.com/top-companies-that-use-python/

Khoirom, S., Sonia, M., Laikhuram, B., Laishram, J., & Singh, T. D. (2020). Comparative analysis of Python and Java for beginners. *Int. Res. J. Eng. Technol*, *7*(8), 4384-4407.

Lardinois, F. (2020, November 12). *Python creator Guido van Rossum joins Microsoft*. TechCrunch. https://techcrunch.com/2020/11/12/python-creator-guido-van-rossum-joins-microsoft/

Lewis, M. (2021, December 9). *The struggle of dynamically typed languages*. Medium. https://drmarkclewis.medium.com/the-struggle-of-dynamically-typed-languages-ef91a87164a1

Libraries in Python (2024, August 1). Geeks for Geeks. https://www.geeksforgeeks.org/libraries-in-python/

Munro, A. (2024, September 21). Python. Encyclopedia Britannica. https://www.britannica.com/technology/Python-computer-language

Mypy (2024) mypy. Retrieved October 28, 2024, from https://mypy-lang.org/

Myrianthous, G. (2024, July 8). *Understanding duck typing in Python*. Builtin. https://builtin.com/articles/python-duck typing#:~:text=Duck%20typing%20is%20a%20term%20used%20to%20describe%20the%20polymorphic,cares%20about%20whether%20it%20quacks.

Object-oriented programming. (2024, October 6). In *Wikipedia*. https://en.wikipedia.org/wiki/Object-oriented_programming

Parashar, A. (2024, July 17). *Top 10 Python libraries*. Code 360 by coding ninjas. https://www.naukri.com/code360/library/python-libraries

Python (programming language). (2024, October 23). In *Wikipedia*. https://en.wikipedia.org/wiki/Python_(programming_language)

PYPL PopularitY of programming language. (2024, October 24). PYPL. https://pypl.github.io/PYPL.html

Support for type hints. (n.d.). Python. Retrieved October 25, 2024 from https://docs.python.org/3/library/typing.html

Python OOPs concepts. (2024, September 5). Geeks for Geeks. https://www.geeksforgeeks.org/python-oops-concepts/

Python syntax and semantics. (2024, October 28). In *Wikipedia*. https://en.wikipedia.org/wiki/Python_syntax_and_semantics

Qiskit (n.d.). IBM. Retrieved 11/2/2024 from https://www.ibm.com/quantum/qiskit

Quantum computing. (2024, November 2). In *Wikipedia*. https://en.wikipedia.org/wiki/Quantum_computing

Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in Python: main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, *11*(4), 193. https://doi.org/10.3390/info11040193

Reddit (n.d.) *in Reddit: About Us.* Retrieved November 2, 2024, from https://redditinc.com/

Roush, W. (2020, February 26). *The Google-IBM "quantum supremacy" feud.* MIT Technology Review. https://www.technologyreview.com/2020/02/26/905777/google-ibm-quantum-supremacy-computing-feud/#:~:text=Researchers%20at%20IBM%2C%20one%20of,the%20significance%20of%20quantum%20supremacy.

Severance, C. (2015, February). Guido van Rossum: The early years of Python. *Computer*, 48, 7–9. https://doi.org/10.1109/MC.2015.45

Stafford, G. *(2024, October 22). Leading the quantum frontier: NSF funding accelerates UTC's QISE program*. University of Tennessee Chattanooga. https://blog.utc.edu/news/2024/10/leading-the-quantum-frontier-nsf-funding-accelerates-utcs-qise-program/?_gl=1*b32u0h*_gcl_aw*R0NMLjE3MjcxOTM0MzAuQ2owS0NRand4c20zQmhEckFSSXNBTXRWejZOcFY2Q24wLVFtWFh0bC11RlR2X2V4cmRHVlFQVGhBU3VPMHJQV21VOGZQU3JNei1BLTRrSWFBb2ViRUFMd193Y0I.*_gcl_au*ODQyODEzNjU4LjE3Mjg2MDc4NTI.

Strong and weak typing. (2024, May 13). In *Wikipedia*. https://en.wikipedia.org/wiki/Strong_and_weak_typing

Sudra, A. (2024, January 8). *Java vs. Python: Which one will be more leveraged in 2024.* iCoderz. Retrieved November 3, 2024 from https://www.icoderzsolutions.com/blog/java-vs-python/#:~:text=Difference%20between%20Python%20and%20Java%20at%20Performance%20Levels&text=Higher%20memory%20usage%20ranging%20from%202-4x%20compared%20to%20Python.

TIOBE. (2024, October 1). *TIOBE Index for October 2024*. TIOBE: the software quality company. https://www.tiobe.com/tiobe-index/

Top 10 VS code extensions for Python [2024]. (2024, September 25). Geeks for Geeks. from https://www.geeksforgeeks.org/top-vs-code-extensions-for-python/

Using static typing (n.d.). Future Learn. Retrieved October 25, 2024, from https://www.futurelearn.com/info/courses/python-in-hpc/0/steps/65121#:~:text=Python%20is%20both%20a%20strongly,is%20determined%20only%20during%20runtime

Van Der Made, C. (2022, March 22). *5 Python security traps you need to avoid*. CISCO. From https://blogs.cisco.com/developer/pythonvulnerabilities01

Van Rossum, G. (n.d.). *Resume*. Guido Van Rossum. Retrieved November 2, 2024, from https://gvanrossum.github.io/Resume.html

Why do data analysts use python? (n.d.). Professional Academy (n.d.) Retrieved November 3, 2024 from https://www.ucd.ie/professionalacademy/resources/why-do-data-analysts-use-python/#:~:text=Data%20analysts%20can%20also%20use,%20with%20large%20tables%20of%20data.

Why does python implementation use 9 times more memory than C? (2021). Stack Overflow. Retrieved November 3, 2024 from https://stackoverflow.com/questions/55367167/why-does-python-implementation-use-9-times-