Escuela Politécnica Nacional

Facultad de Ingeniería de Sistemas

Tema:

Comparativa de 2 lenguajes de programación emergentes enfocados en Aplicativo.

Integrantes:

Enrique Pérez

Martin Zambonino

Titulo

Lenguajes de programación emergentes

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SOFTWARE

Objetivos:

- Entender los lenguajes de programación enfocados en el aplicativo para la industria.
- Comprender las funciones y características del lenguaje Python y Rust.
- Mostrar los usos de los lenguajes Python y Rust en la creación de aplicaciones.

Introducción

Los lenguajes de programación son herramientas fundamentales para el desarrollo de software, que permite crear soluciones tecnológicas para diversos dominios y problemas. Sin embargo, no todos los lenguajes de programación son iguales, ni tienen las mismas capacidades, ventajas y desventajas. Por ello, es importante conocer las características, fortalezas y debilidades de los diferentes lenguajes disponibles, así como su adecuación para cada tipo de proyecto y contexto.

Se pretende realizar una comparativa entre dos lenguajes de programación emergentes, enfocados en aplicaciones para la industria de infraestructura, que abarca sectores como la energía, el transporte, la construcción, el agua y las telecomunicaciones. Estos lenguajes son Rust y Python, que tienen un creciente interés y uso por parte de desarrolladores y empresas, debido a sus prestaciones y beneficios. El objetivo es analizar las similitudes y diferencias entre estos dos lenguajes, así como sus ventajas y desventajas para el desarrollo de aplicaciones de infraestructura, que requieren de alto rendimiento, escalabilidad, seguridad y facilidad de desarrollo.

Lenguajes de Programación Emergentes

Se entiende por lenguajes de programación emergentes aquellos que han surgido recientemente, o que han experimentado un aumento significativo de popularidad y uso en los últimos años, debido a sus innovaciones, características o aplicaciones. Estos lenguajes suelen responder a las demandas y necesidades de los desarrolladores y de las industrias, ofreciendo soluciones más eficientes, versátiles y robustas que otros lenguajes más establecidos o tradicionales.

Para seleccionar los lenguajes a comparar, se han considerado los siguientes criterios:

- Que hayan sido creados o lanzados en el siglo XXI, o que hayan tenido un crecimiento notable en ese periodo.
- Que tengan un diseño orientado a la concurrencia, la paralelización y la distribución, aprovechando los recursos de hardware disponibles.
- Que ofrezcan garantías de seguridad, fiabilidad y prevención de errores, mediante el uso de mecanismos como la gestión de memoria, el tipado estático, el control de excepciones o el análisis estático.
- Que sean multiparadigma, permitiendo el uso de diferentes estilos de programación, como el funcional, el imperativo, el orientado a objetos o el declarativo.
- Que sean multiplataforma, pudiendo ejecutarse en diferentes sistemas operativos y arquitecturas, sin necesidad de modificaciones o adaptaciones.

- Que tengan una comunidad activa y creciente, que aporte documentación, soporte, librerías y herramientas para el desarrollo y el mantenimiento de los proyectos.

Basándose en estos criterios, se han elegido dos lenguajes de programación emergentes para realizar la comparativa: Rust y Python. A continuación, se describen brevemente estos dos lenguajes, así como sus ventajas y desventajas, y sus casos de uso en aplicaciones de infraestructura.

Comparativa de Lenguajes

Rust

Historia Completa

Rust es un lenguaje de programación de sistemas, creado por Graydon Hoare en 2006, como un proyecto personal, inspirado en otros lenguajes como C++, Haskell, OCaml y Erlang.

En 2009, el proyecto fue adoptado por Mozilla Research, con el fin de crear un motor de navegador web más seguro y eficiente que el existente.

En 2010, se publicó la primera versión alfa de Rust, y en 2011, se anunció oficialmente el lenguaje. Desde entonces, Rust ha evolucionado y mejorado, incorporando nuevas características y funcionalidades, como el sistema de gestión de paquetes Cargo, el compilador autocontenido rustc, o el soporte para macros.

En 2015, se lanzó la primera versión estable de Rust, la 1.0, y desde entonces, se han publicado actualizaciones periódicas cada seis semanas. En 2016, se creó la Fundación Rust, una entidad independiente que se encarga de supervisar y coordinar el desarrollo y el gobierno del lenguaje.

En 2021, se lanzó la versión 1.50 de Rust, que incluye mejoras en el rendimiento, la estabilidad y la compatibilidad.

Descripción General

Rust es un lenguaje de programación de sistemas, que tiene como objetivo ofrecer un alto rendimiento, una gran seguridad y una buena productividad. Está diseñado para ser un lenguaje de bajo nivel, capaz de interactuar directamente con el hardware y el sistema operativo, pero también para ser un lenguaje de alto nivel, que facilita la expresividad, la abstracción y la modularidad.

Rust es un lenguaje compilado, que genera código nativo optimizado para la plataforma objetivo, y que puede integrarse con otros lenguajes como C o C++. Es multiparadigma, que soporta los estilos de programación funcional, imperativo, orientado a objetos y concurrente. Se caracteriza por su enfoque en la seguridad y la fiabilidad, que se basa en dos conceptos clave: la propiedad y el préstamo.

La propiedad es el mecanismo que determina quién es el responsable de gestionar la memoria de cada dato, evitando los problemas de fugas o de uso indebido. El préstamo es el mecanismo que permite el acceso temporal a los datos, sin transferir la propiedad, pero garantizando que no se produzcan conflictos o modificaciones no deseadas. Estos mecanismos son verificados por el

compilador en tiempo de compilación, lo que evita la necesidad de recolectores de basura o de comprobaciones en tiempo de ejecución.

Ventajas

- Ofrece un excelente rendimiento, comparable al de C o C++, pero con menos sobrecarga y complejidad.
- Garantiza una alta seguridad, previniendo los errores más comunes y peligrosos en la programación de sistemas, como los desbordamientos de búfer, los punteros colgantes, las condiciones de carrera o las inyecciones de código.
- Favorece una buena productividad, proporcionando herramientas integradas para el desarrollo, la depuración, el testing, el benchmarking, el análisis, el empaquetado y la distribución de código.
- Tiene una comunidad muy activa y creciente, que contribuye con documentación, tutoriales, ejemplos, librerías, frameworks y soporte técnico para el lenguaje.
- Tiene una gran portabilidad, siendo compatible con múltiples plataformas y sistemas operativos, incluyendo Windows, Linux, macOS, Android, iOS, WebAssembly, etc.

Desventajas

- Curva de aprendizaje elevada, debido a la complejidad de sus conceptos y reglas, que pueden resultar confusos o frustrantes para los principiantes o los programadores acostumbrados a otros lenguajes.
- Compatibilidad limitada con otros lenguajes, especialmente con aquellos que usan recolectores de basura o que manejan la memoria de forma diferente, lo que puede dificultar la interoperabilidad o la integración.
- Madurez relativa, ya que es un lenguaje joven que aún está en desarrollo y evolución, lo que puede implicar cambios, inestabilidades o inconsistencias en el lenguaje o en las librerías.
- Disponibilidad reducida de librerías y frameworks, especialmente para algunos dominios o aplicaciones, en comparación con otros lenguajes más populares o establecidos.
- Escasez de desarrolladores cualificados, ya que es un lenguaje relativamente nuevo y poco conocido, lo que puede dificultar la contratación o el mantenimiento de equipos de desarrollo.

Casos de Uso en Infraestructura

Rust es un lenguaje muy adecuado para el desarrollo de aplicaciones de infraestructura, que requieren de alto rendimiento, seguridad y confiabilidad. Algunos ejemplos de casos de uso son:

- Sistemas embebidos: Puede programar en dispositivos con recursos limitados, como sensores, controladores, actuadores, etc., que forman parte de sistemas de infraestructura como redes eléctricas inteligentes, sistemas de transporte inteligentes, sistemas de riego inteligentes, etc.
- Servicios web: Permite crear servicios web escalables, eficientes y robustos, que proveen de funcionalidades o datos a aplicaciones de infraestructura, como sistemas de

gestión de recursos, sistemas de alerta temprana, sistemas de monitoreo ambiental, etc.

- Computación en la nube: Se usa para desarrollar soluciones de computación en la nube, que aprovechan las ventajas de la distribución, la paralelización y la virtualización, para ofrecer servicios o recursos a aplicaciones de infraestructura, como almacenamiento, procesamiento, análisis, etc.
- Aprendizaje automático: Tiene la capacidad de implementar algoritmos de aprendizaje automático, que permiten extraer conocimiento, patrones o predicciones a partir de grandes volúmenes de datos, generados por aplicaciones de infraestructura, como sistemas de detección de anomalías, sistemas de optimización de rendimiento, sistemas de recomendación, etc.
- Blockchain: Se puede usar para crear sistemas de blockchain, que proporcionan mecanismos de seguridad, transparencia y descentralización, para aplicaciones de infraestructura, como sistemas de identificación, sistemas de trazabilidad, sistemas de gobernabilidad, etc.

Python

Historia Completa

Python es un lenguaje de programación de propósito general, creado por Guido van Rossum en 1991, como un sucesor del lenguaje ABC, que buscaba combinar la simplicidad, la legibilidad y la expresividad.

El nombre del lenguaje se inspiró en el grupo de humor británico Monty Python, y refleja el tono divertido y lúdico del lenguaje y de su comunidad. Desde su creación, Python ha evolucionado y mejorado, incorporando nuevas características y funcionalidades, como el sistema de gestión de paquetes pip, el framework web Django, la librería científica NumPy, o el soporte para programación asíncrona.

En 2008, se lanzó la versión 3.0 de Python, que introdujo cambios significativos e incompatibles con la versión anterior, 2.7, que aún se mantiene y se usa ampliamente. En 2014, se creó la Python Software Foundation, una organización sin fines de lucro que se encarga de apoyar y promover el desarrollo y el uso del lenguaje. En 2020, se lanzó la versión 3.9 de Python, que incluye mejoras en el rendimiento, la sintaxis y la modularidad.

Descripción General

Python es un lenguaje de programación de propósito general, que tiene como objetivo ofrecer una sintaxis simple, clara y consistente, que facilite la escritura y la lectura de código. Python está diseñado para ser un lenguaje de alto nivel, que permite la abstracción y la expresión de conceptos complejos, pero también para ser un lenguaje de bajo nivel, que permite el acceso y la manipulación de recursos del sistema.

Es un lenguaje interpretado, que se ejecuta sobre una máquina virtual, que puede adaptarse a diferentes plataformas y sistemas operativos, y que puede integrarse con otros lenguajes como C o Java. Es un lenguaje multiparadigma, que soporta los estilos de programación funcional,

imperativo, orientado a objetos y declarativo. Se caracteriza por su flexibilidad y su dinamismo, que se basan en dos conceptos clave: el tipado dinámico y la introspección.

El tipado dinámico es el mecanismo que determina el tipo de los datos en tiempo de ejecución, lo que permite el uso de variables genéricas, la sobrecarga de operadores, la herencia múltiple, etc. La introspección es el mecanismo que permite el acceso y la modificación de las propiedades y los comportamientos de los objetos en tiempo de ejecución, lo que permite la creación de metaclasses, la reflexión, la serialización, etc.

Ventajas

- Ofrece una sintaxis sencilla, clara y consistente, que facilita la escritura y la lectura de código, reduciendo el tiempo de desarrollo y el número de errores.
- Gran flexibilidad y dinamismo, que permiten la adaptación y la personalización de los programas, según las necesidades y preferencias de los desarrolladores y los usuarios.
- Amplia disponibilidad de librerías y frameworks, que cubren una gran variedad de dominios y aplicaciones, como el desarrollo web, el análisis de datos, la inteligencia artificial, la visualización, etc.
- Tiene una comunidad muy grande y diversa, que contribuye con documentación, tutoriales, ejemplos, librerías, frameworks y soporte técnico para el lenguaje.
- Gran portabilidad, siendo compatible con múltiples plataformas y sistemas operativos, incluyendo Windows, Linux, macOS, Android, iOS, WebAssembly, etc.

Desventajas

- Tiene un rendimiento inferior al de otros lenguajes, especialmente los compilados, debido a su naturaleza interpretada, su tipado dinámico y su recolector de basura, que generan una mayor sobrecarga y latencia.
- Ofrece una seguridad menor que otros lenguajes, especialmente los de tipado estático, debido a su flexibilidad y dinamismo, que pueden provocar errores inesperados o difíciles de detectar, como los de tipado, los de atributos o los de herencia.
- Tiene una compatibilidad limitada entre sus versiones, especialmente entre la 2.7 y la 3.x, lo que puede ocasionar problemas de migración, integración o mantenimiento de código.
- Posee una concurrencia deficiente, debido al mecanismo conocido como Global Interpreter Lock (GIL), que impide la ejecución simultánea de múltiples hilos de Python, limitando el aprovechamiento de los recursos de hardware disponibles.
- Tiene una sintaxis ambigua, debido a la falta de caracteres o símbolos que delimiten los bloques de código, como las llaves o los puntos y coma, lo que puede generar confusiones o inconsistencias en la identación o el anidamiento.

Casos de Uso en Infraestructura

Python es un lenguaje muy adecuado para el desarrollo de aplicaciones de infraestructura, que requieren de simplicidad, expresividad y versatilidad. Algunos ejemplos de casos de uso son:

- Análisis de datos: Puede realizar análisis de datos, que permiten extraer información, estadísticas o indicadores a partir de grandes volúmenes de datos, generados por aplicaciones de infraestructura, como sistemas de medición, sistemas de gestión, sistemas de simulación, etc.

- Inteligencia artificial: Permite implementar algoritmos de inteligencia artificial, que permiten crear sistemas inteligentes, capaces de aprender, razonar o tomar decisiones, para aplicaciones de infraestructura, como sistemas de diagnóstico, sistemas de predicción, sistemas de optimización, sistemas de control, etc.
- Desarrollo web: Puede crear aplicaciones web, que proveen interfaces gráficas, interactivas y amigables, para aplicaciones de infraestructura, como sistemas de información, sistemas de comunicación, sistemas de colaboración, etc.
- Automatización: Ayuda en la automatización de tareas o procesos, que requieren de una ejecución rápida, precisa y repetitiva, para aplicaciones de infraestructura, como sistemas de prueba, sistemas de configuración, sistemas de despliegue, etc.
- Scripting: Capaz de escribir scripts, que son programas de corta duración, que realizan funciones específicas, para aplicaciones de infraestructura, como sistemas de administración, sistemas de monitoreo, sistemas de reporte, etc.

Comparación Directa

A continuación, se realiza una comparación directa entre los dos lenguajes de programación emergentes seleccionados, Rust y Python, en base a los siguientes criterios:

Rendimiento

Rust tiene un rendimiento superior al de Python, ya que es un lenguaje compilado, de tipado estático y de gestión manual de memoria, que genera código nativo optimizado para la plataforma objetivo, y que evita la sobrecarga de los recolectores de basura o las comprobaciones en tiempo de ejecución.

Python tiene un rendimiento inferior al de Rust, ya que es un lenguaje interpretado, de tipado dinámico y de gestión automática de memoria, que se ejecuta sobre una máquina virtual, y que genera una mayor latencia por el uso de los recolectores de basura o las comprobaciones en tiempo de ejecución.

Escalabilidad

Rust tiene una escalabilidad mayor que la de Python, ya que es un lenguaje que facilita la concurrencia, la paralelización y la distribución, mediante el uso de conceptos como la propiedad, el préstamo, los traits o las macros, que permiten crear programas seguros, modulares y componibles, que pueden aprovechar los recursos de hardware disponibles.

Python tiene una escalabilidad menor que la de Rust, ya que es un lenguaje que dificulta la concurrencia, la paralelización y la distribución, debido al uso de mecanismos como el GIL, las variables globales o la herencia múltiple, que limitan la creación de programas concurrentes, modulares y componibles, que puedan aprovechar los recursos de hardware disponibles.

Facilidad de desarrollo

Python tiene una facilidad de desarrollo mayor que la de Rust, ya que es un lenguaje que ofrece una sintaxis simple, clara y consistente, que facilita la escritura y la lectura de código, reduciendo el tiempo de desarrollo y el número de errores. Además, Python ofrece una amplia disponibilidad de librerías y frameworks, que cubren una gran variedad de dominios y aplicaciones, simplificando el desarrollo de proyectos complejos.

Rust tiene una facilidad de desarrollo menor que la de Python, ya que es un lenguaje que tiene una sintaxis compleja, confusa e inconsistente, que dificulta la escritura y la lectura de código, aumentando el tiempo de desarrollo y el número de errores. Además, Rust tiene una disponibilidad reducida de librerías y frameworks, especialmente para algunos dominios o aplicaciones, complicando el desarrollo de proyectos complejos.

Soporte y comunidad

Python tiene un soporte y una comunidad mayores que los de Rust, ya que es un lenguaje más popular, establecido y conocido, que cuenta con una gran cantidad y diversidad de desarrolladores, usuarios y empresas, que contribuyen con documentación, tutoriales, ejemplos, librerías, frameworks y soporte técnico para el lenguaje.

Rust tiene un soporte y una comunidad menores que los de Python, ya que es un lenguaje menos popular, establecido y conocido, que cuenta con una menor cantidad y diversidad de desarrolladores, usuarios y empresas, que contribuyen con documentación, tutoriales, ejemplos, librerías, frameworks y soporte técnico para el lenguaje.

Seguridad

Rust tiene una seguridad mayor que la de Python, ya que es un lenguaje que garantiza una alta seguridad y fiabilidad, previniendo los errores más comunes y peligrosos en la programación de sistemas, como los desbordamientos de búfer, los punteros colgantes, las condiciones de carrera o las inyecciones de código, mediante el uso de mecanismos como la propiedad, el préstamo, el tipado estático, el control de excepciones o el análisis estático, que son verificados por el compilador en tiempo de compilación.

Python tiene una seguridad menor que la de Rust, ya que es un lenguaje que ofrece una seguridad menor y fiabilidad, permitiendo un mayor control sobre el lenguaje.

Conclusión

En conclusión, se puede decir que Rust y Python son dos lenguajes de programación emergentes, que tienen sus propias ventajas y desventajas, y que se adaptan mejor a diferentes tipos de aplicaciones de infraestructura.

Rust es un lenguaje que destaca por su rendimiento, su seguridad y su escalabilidad, pero que tiene una mayor complejidad, una menor compatibilidad y disponibilidad de librerías y frameworks.

Python es un lenguaje que destaca por su simplicidad, su flexibilidad y su versatilidad, pero que tiene un menor rendimiento, una menor seguridad y concurrencia.

Por lo tanto, se puede recomendar el uso de Rust para aplicaciones de infraestructura que requieren de un alto nivel de eficiencia, fiabilidad y robustez, como sistemas embebidos, servicios web, computación en la nube, aprendizaje automático o blockchain. Por otro lado, se puede recomendar el uso de Python para aplicaciones de infraestructura que requieren de un alto nivel de expresividad, dinamismo y diversidad, como análisis de datos, inteligencia artificial, desarrollo web, automatización o scripting.

Finalmente, se puede prever que ambos lenguajes seguirán evolucionando y mejorando, incorporando nuevas características y funcionalidades, así como ampliando su alcance y su popularidad, en el campo de las aplicaciones de infraestructura, que representa un desafío y una oportunidad para el desarrollo de software.

Bibliografía

Challenger-Pérez, I., Díaz-Ricardo, Y., & Becerra-García, R. A. (2014). El lenguaje de programación Python. *Ciencias Holguín*, *20*(2), 1-13.

Mirjalili, V., & Raschka, S. (2020). *Python machine learning*. Marcombo.

VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data*. " O'Reilly Media, Inc.".

Klabnik, S., & Nichols, C. (2023). *The Rust programming language*. No Starch Press.

Jung, R. (2020). Understanding and evolving the Rust programming language.