**Tarea de Investigación: Los orígenes de Python y su evolución**

**Objetivo:**

Investigar los orígenes del lenguaje de programación Python, conocer a su creador, el contexto en que surgió el proyecto, la importancia del grupo de desarrollo que lo impulsó, y comprender el papel fundamental que tienen las versiones del lenguaje. La tarea también busca fomentar el uso responsable de la información, destacando la importancia de citar adecuadamente las fuentes consultadas.

**Instrucciones:**

¿Quién fue el creador de Python? Investiga el nombre, formación y trayectoria profesional del creador de Python.

Guido van Rossum (31 de enero de 1956. Haarlem, Países Bajos) es Licenciado en Matemáticas y Ciencias de la Computación por la Universidad de Amsterdam (Países Bajos).

|  |  |  |
| --- | --- | --- |
| 1982 - 1995 | Centrum Wiskunde and Informatike (Países Bajos) | Desarrolló Python como proyecto personal en las vacaciones de Navidad de 1989.  Python 0.9.0 en 1991 primera versión. |
| 1995 - 1998 | Instituto Nacional de Estándares y Tecnología (Estados Unidos) | Implementación de tecnologías web. |
| 1998 - 2000 | CNRI – (Estados Unidos) | Desarrolló herramientas para mejorar la programación en internet. |
| 2000 | BeOpen | Dirigió un equipo de desarrolladores de Python. |
| 2005 - 2012 | Google | Colaboró en proyectos como Google App Engine y creó el sistema de revisión de código Mondrian. |
| 2013 - 2019 | Dropbox | Trabajó en optimizar infraestructura y mejorar la seguridad. |
| 2020 - actualidad | Microsoft | Contribuye al desarrollo de Python para optimizar su rendimiento. |

|  |
| --- |
| "Guido van Rossum: El creador de Python"  Publicación: ComputerHoy.com (medio especializado en tecnología).  Artículo: "Guido van Rossum, el padre de Python: quién es y qué hace hoy"  Contenido: Detalla su biografía, motivaciones para crear Python y su impacto actual.  Fecha de consulta: 5 de junio de 2025.  "Python nació como un proyecto para mantenerme ocupado durante las vacaciones... Quería un lenguaje que fuera fácil de usar y divertido".  — Guido van Rossum, en entrevista para ComputerHoy (2020). |

¿Qué lo motivó a desarrollar un nuevo lenguaje de programación?

|  |
| --- |
| En sus palabras:  *"Quería un lenguaje que fuera fácil de usar, intuitivo y poderoso [...] ABC me enseñó que no debía sacrificar la elegancia por la simplicidad"* (ComputerHoy, 2020). |

¿Cómo y cuándo surgió el proyecto Python? Describe el contexto histórico y tecnológico en el que nació Python.

|  |
| --- |
| Diciembre de 1989: Guido van Rossum inició el desarrollo durante las vacaciones de Navidad en el Centrum Wiskunde & Informatica (CWI) de Ámsterdam (Países Bajos).  El contexto tecnológico de la época estaba dominado por lenguajes complejos: C/C++  Perl se usaba para scripting, pero su sintaxis era críptica ("lenguaje de línea de comandos glorificado", según Van Rossum).  La programación estaba dominada por UNIX y mainframes.  Python tomó elementos de ABC e implemento el paradigma de la programación orientada a objetos desde su versión inicial.  Van Rossum lo resume:  "Quería un lenguaje que redujera el tiempo de desarrollo y fuera divertido de usar... Python debía ser inteligible como pseudocódigo" (ComputerHoy, 2020). |

¿En qué proyecto trabajaba su creador antes de desarrollar Python?

|  |
| --- |
| Antes de crear Python, Guido van Rossum trabajaba en el proyecto ABC en el Centrum Wiskunde & Informatica (CWI) de Ámsterdam (Países Bajos) durante la década de 1980.  Van Rossum reflexionó:  *"ABC tenía ideas brillantes [...] pero era demasiado rígido. Quería llevar esa elegancia a un lenguaje que pudiera usarse en el mundo real"* (ComputerHoy, 2020). |

¿Cuál era la necesidad que buscaba cubrir?

|  |
| --- |
| Guido van Rossum con Python buscaba unificar la productividad y la potencia, simplificar la programación y crear un lenguaje puente.  *"Quería escribir un sistema de archivos para Amoeba en un lenguaje que ahorrara semanas de desarrollo, pero sin perder flexibilidad"* (Van Rossum, 1996). |

¿Qué rol tuvo el grupo de desarrollo en la evolución de Python? ¿Cómo fue creciendo la comunidad en torno a Python?

|  |
| --- |
| El núcleo original evitó que Python se fragmentara (como Perl), manteniendo "una única forma obvia" de hacer las cosas (Zen de Python).  Legado  De proyecto personal a lenguaje global:  Menos de 10 colaboradores en 1991 → +1,000 contribuyentes a CPython en 2025.  Rechazaron características complejas que violaran la filosofía de simplicidad (ej: no añadir goto).  "Python domina el 30% del mercado de lenguajes gracias a decisiones tempranas sobre extensibilidad y legibilidad" (TIOBE, 2024). |

|  |
| --- |
| Periodo Hito  1991-1995 Primera lista de correo (python-list)  1995-2005 Nacen PyCon (2003) y PSF (2001) Comunidad organizada;  2005-2015 Boom de data science (NumPy, pandas) y web (Django, Flask)  2015-presente Explosión en IA (TensorFlow, 2015) y educación (bootcamps)  Python Software Foundation. (2023). History of Python. https://www.python.org/about/history/ |

¿Qué es la Python Software Foundation (PSF) y por qué es relevante?

|  |
| --- |
| La Python Software Foundation (PSF) es una organización sin fines de lucro creada en marzo de 2001 para:  Gestionar la propiedad intelectual de Python (marca, licencias, derechos de autor).  Promover el desarrollo y adopción del lenguaje.  Apoyar a la comunidad global de usuarios y contribuyentes. |

|  |  |  |
| --- | --- | --- |
| **Área de impacto** | **Funciones concretas** | **Relevancia** |
| Sustento legal | Custodia la marca "Python" y la licencia OSI-approved (Python Software Foundation License). | Evita fragmentación (ej: conflictos como los de Java/OpenJDK). |
| Financiamiento | Recauda fondos (donaciones corporativas: Google, Microsoft, Meta) para:  - Becas para conferencias (PyCon Diversity). | Mantiene infraestructura crítica: servidores de PyPI, bots de integración.  Financia proyectos comunitarios ($1.5M en 2023). |
| Desarrollo técnico | Organiza equipos de trabajo para CPython, seguridad y PEPs. | Asegura la evolución técnica coherente (ej: transición Python 2 → 3). |
| Comunidad y educación | Coordina PyCon Global (50+ eventos anuales) y programas como:  - Python Core Mentorship (formación de nuevos contribuyentes). | Impulsa inclusividad: PyLadies, PyCon África.  Difusión en escuelas y países en desarrollo. |
| Ecosistema Open-Source | Opera PyPI (Python Package Index), con +500,000 paquetes (2025). | Facilita la innovación: librerías como NumPy, TensorFlow dependen de PyPI. |

¿Qué son las versiones de Python y por qué son importantes? Diferencia entre versiones mayores (por ejemplo, Python 2 vs. Python 3) y menores (por ejemplo, 3.10 vs. 3.11).

|  |
| --- |
| Las versiones de Python son actualizaciones del lenguaje que introducen mejoras, correcciones o cambios. Su gestión sigue el esquema mayor.menor.micro (ej: 3.11.4). |

|  |
| --- |
| ¿Por qué son importantes?  Seguridad: Parches para vulnerabilidades.  Rendimiento: Optimizaciones de velocidad (ej: Python 3.11 es un 60% más rápido que 3.10).  Nuevas características: Sintaxis más expresiva y funcionalidades.  Compatibilidad: Librerías modernas requieren versiones recientes.  Soporte oficial: Versiones antiguas pierden mantenimiento (ej: Python 2 ya no recibe actualizaciones). |

|  |
| --- |
| Las versiones de Python son actualizaciones del lenguaje que introducen mejoras técnicas, correcciones de seguridad o nuevas funcionalidades. Son cruciales porque garantizan que Python siga siendo eficiente, seguro y competitivo: cada versión soluciona vulnerabilidades, optimiza el rendimiento (hasta un 60% más rápido en algunos casos) y añade herramientas para facilitar la programación. Además, mantenerte actualizado asegura compatibilidad con las bibliotecas modernas y evita riesgos técnicos, ya que las versiones antiguas pierden soporte oficial con el tiempo.  Cuando hablamos de versiones mayores, nos referimos a cambios profundos que pueden romper la compatibilidad con código anterior. El salto de Python 2 a Python 3 es el ejemplo más emblemático: no solo modificó funciones básicas como print (que pasó de ser una declaración a una función), sino que cambió el manejo de texto (adoptando Unicode por defecto), ajustó operaciones matemáticas (como la división de enteros) e introdujo una sintaxis más limpia. Esta transición fue tan significativa que requirió años de adaptación y herramientas especiales para migrar proyectos.  En cambio, las versiones menores (como 3.10 a 3.11) mantienen compatibilidad hacia atrás mientras añaden mejoras incrementales. Por ejemplo, Python 3.11 mejoró drásticamente el rendimiento con optimizaciones internas, mientras que 3.10 introdujo el "pattern matching" (una forma más intuitiva de manejar estructuras de datos). Estas actualizaciones suelen enfocarse en refinamientos: mensajes de error más claros, nuevas sintaxis para tipado estático o funciones experimentales que, sin alterar el código existente, abren posibilidades creativas. Seguir estas actualizaciones es menos traumático, pero igualmente vital para aprovechar las ventajas del lenguaje sin quedarse atrás.  Genbeta. (2022, 25 de octubre). Python 3.11 ya está aquí: estas son todas sus novedades y cómo instalarlo.  Genbeta. (2022). *Python 3.11 ya está aquí: estas son todas sus novedades y cómo instalarlo.* https://www.genbeta.com/desarrollo/python-3-11-ya-esta-aqui-estas-son-todas-sus-novedades-como-instalarlo |

¿Por qué es importante conocer la versión de Python que se está usando? ´

|  |
| --- |
| Es fundamental saber la versión de Python que estás utilizando por varias razones. Primero, las diferentes versiones de Python, como Python 2 y Python 3, tienen diferencias significativas en su sintaxis y funcionalidades. Un código escrito para Python 2, por ejemplo, podría no funcionar correctamente en Python 3, y viceversa. Conocer la versión te permite asegurarte de que el código que estás ejecutando es compatible con el entorno de Python que tienes instalado. Segundo, las bibliotecas y los paquetes de Python, que son esenciales para muchas tareas, pueden tener diferentes niveles de compatibilidad con distintas versiones de Python. Algunos paquetes podrían no estar disponibles para ciertas versiones, o podrían requerir versiones específicas para funcionar correctamente. Por lo tanto, saber la versión de Python te ayuda a gestionar las dependencias de tus proyectos y a evitar errores relacionados con la compatibilidad. Finalmente, estar al tanto de la versión te permite aprovechar las nuevas características y mejoras que se introducen en cada actualización de Python, optimizando el rendimiento y la eficiencia de tu código.  Referencia:  Van Rossum, G., & Drake Jr, F. L. (2009). *The Python Language Reference. Network Theory Ltd.* |

¿Qué impacto tuvo la transición de Python 2 a Python 3?

|  |
| --- |
| La transición de Python 2 a Python 3 representó un cambio crucial pero desafiante para la comunidad de desarrollo. Inicialmente, generó resistencia debido a la incompatibilidad intencionada entre versiones, lo que obligó a reescribir grandes cantidades de código y ralentizó la adopción durante años. Muchos proyectos enfrentaron dificultades técnicas, especialmente con el manejo de cadenas de texto, ya que Python 3 adoptó Unicode como estándar predeterminado, corrigiendo problemas históricos de codificación pero requiriendo ajustes significativos. La sintaxis también evolucionó, con cambios como la conversión de print de sentencia a función (print()). Aunque esta ruptura generó fricción y prolongó la coexistencia de ambas versiones hasta 2020 (cuando Python 2 alcanzó su fin de soporte), a largo plazo fortaleció el ecosistema: eliminó redundancias, mejoró la seguridad, optimizó el diseño del lenguaje (como la función range() reemplazando xrange()) y sentó las bases para funcionalidades modernas. Pese al dolor inicial, esta migración demostró el compromiso de Python con la innovación, consolidando su relevancia futura.  Referencia:  Python España. (2020). *El fin de Python 2. Documentación de la comunidad*. Recuperado el 5 de junio de 2025, de https://www.es.python.org/fin-de-python-2 |

Importancia de citar fuentes: Toda la información que uses debe provenir de fuentes confiables (páginas oficiales, documentación técnica, artículos académicos o fuentes periodísticas reconocidas).