PYTHON TUTOR BOT: DISEÑO Y DESARROLLO DE UN TUTOR EN LA ERA DE LA INTELIGENCIA ARTIFICIAL GENERATIVA

PYTHON TUTOR BOT: DESIGN AND DEVELOPMENT OF A TUTOR IN THE AGE OF GENERATIVE ARTIFICIAL INTELLIGENCE

Gerardo Ramírez Medina
Universidad Autónoma Metropolitana, México
merbock07@gmail.com

Reyna Carolina Medina Ramírez
Universidad Autónoma Metropolitana, México
rmedina@izt.uam.mx

Miguel Ángel Ramos Castellanos
Universidad Autónoma Metropolitana, México
angeluam.0927@gmail.com

Recepción: 5/enero/2026 **Aceptación:** 26/marzo/2026

Resumen

El presente documento describe un chatbot educativo del tipo tutor para coadyuvar al proceso de enseñanza-aprendizaje (PEA) de conceptos fundamentales de programación a través de Python Colab. Se toma como caso de estudio la UEA Fundamentos de Programación del plan de estudios de la Licenciatura en Computación en la UAM Iztapalapa. El objetivo principal del chatbot es motivar en el alumnado el aprendizaje autogestivo partiendo de un aprendizaje guiado. Se realizaron 114 recursos correspondientes a 19 temas considerados en el programa oficial de la UEA de estudio, una planeación de clase interactiva en la cual se considera la intervención de Python Tutor Bot en actividades individuales, colaborativas, así como en evaluaciones formativas. En su desarrollo se utilizó una base de conocimiento para el diálogo conversacional e inferencia del tipo de recurso asociado al tema buscado. Los resultados preliminares muestran la viabilidad para aplicarse a otras UEA con ajustes respectivos.

Palabras Clave: Aprendizaje guiado, chatbot educativo, fundamentos de programación, Python Colab, tutor.

Abstract

This document describes an educational tutor-type chatbot to assist in the teaching-learning process (TLP) of fundamental programming concepts through Python Colab. The Fundamentals of Programming (FP) program of the Bachelor's Degree in Computer Science at the Universidad Autónoma Metropolitana, Iztapalapa Campus is used as a case study. The chatbot's main objective is to motivate students to engage in self-directed learning through guided learning. 114 *resources were created corresponding to* 19 *topics included in the official FP program, along with an interactive lesson plan that includes the Python Tutor Bot in individual and collaborative activities, as well as formative assessments. A knowledge base was used in its development for conversational dialogue and inference of the type of resource associated with the topic being studied. Preliminary results suggest that the approach can be effectively applied to other courses with appropriate modifications.*

Keywords: *Educational chatbot, guided learning, programming fundamentals, Python Colab, tutor.*

1. Introducción

La Inteligencia artificial generativa (IAg) cobra importancia en nuestros días dada la vasta cantidad de aplicaciones que coadyuvan a la realización de actividades cotidianas. En particular, el proceso de enseñanza-aprendizaje (PEA) está adquiriendo un nuevo significado con el advenimiento de tecnologías de IAg. En el ámbito educativo, los chatbots han emergido como herramientas innovadoras capaces de ofrecer apoyo personalizado en diversos contextos de aprendizaje. Sin embargo, sigue vigente el problema de como evaluar lo aprendido, lo enseñado, las formas (estrategias), los contenidos (¿Qué se enseña?, ¿Cómo se enseña?, ¿Qué se evalúa?, ¿Para qué se evalúa? y ¿Cómo queremos o podemos evaluar?), así como los medios (TIC, TAC, TICCAD) [Vallejo, 2014; Chio, 2025]. Estas preguntas conllevan a una reflexión, por un lado, sobre el PEA en la era de la Inteligencia artificial generativa y proponer soluciones, ajustes o mejoras en cada uno de los ítems mencionados. Por otro lado, también reflexionar sobre la importancia de un

tutor digital. En la UAM-Iztapalapa, uno de los desafíos que se tienen en cursos introductorios a la programación es la diversidad de conocimientos del alumnado, están los que ya tienen una formación técnica básica, los que recuerdan solo algunas cosas y aquellos que es su primer curso de programación. Estructurar un curso considerando dichos perfiles es complicado pues varía de un trimestre a otro. Así mismo, también está la dificultad de adaptar los contenidos al nivel y preferencias de cada estudiante. Por otro lado, los estudiantes que inician en Python enfrentan dificultades para acceder a recursos adecuados, navegar entre conceptos relacionados y obtener respuesta a dudas cuando las herramientas disponibles no comprenden las consultas imperfectas.

Los chatbots educativos surgen como una solución innovadora, al ofrecer acceso inmediato a la información y personalización del aprendizaje [Kuhail, 2022; Malik, 2022; Pérez, 2020; Medina, 2024]. En este documento se presenta Python Tutor Bot, un chatbot educativo del tipo tutor para coadyuvar a la enseñanza de fundamentos de programación con Python Colab.

La propuesta surge con el objetivo de motivar en el alumnado el aprendizaje autogestivo partiendo de un aprendizaje guiado, proporcionando una experiencia interactiva y flexible acompañando al alumnado desde los conceptos básicos hasta los niveles avanzados, conectando la teoría con ejemplos y la práctica (ejercicios). La propuesta en su diseño integra tecnologías como grafos de conocimiento [Hogan, 2022] y un modelo conversacional estructurado, ambos aplicados en una arquitectura web diseñada para facilitar la interacción. Se toma como caso de estudio la UEA Fundamentos de Programación del plan de estudios de la Licenciatura en Computación en la UAM Iztapalapa [UEA, 2025]. La contribución de este trabajo es un chatbot mínimo funcional con potencial de replicabilidad en otras unidades de enseñanza, coadyuvando a la enseñanza y aprendizaje de la programación [Sánchez, 2023].

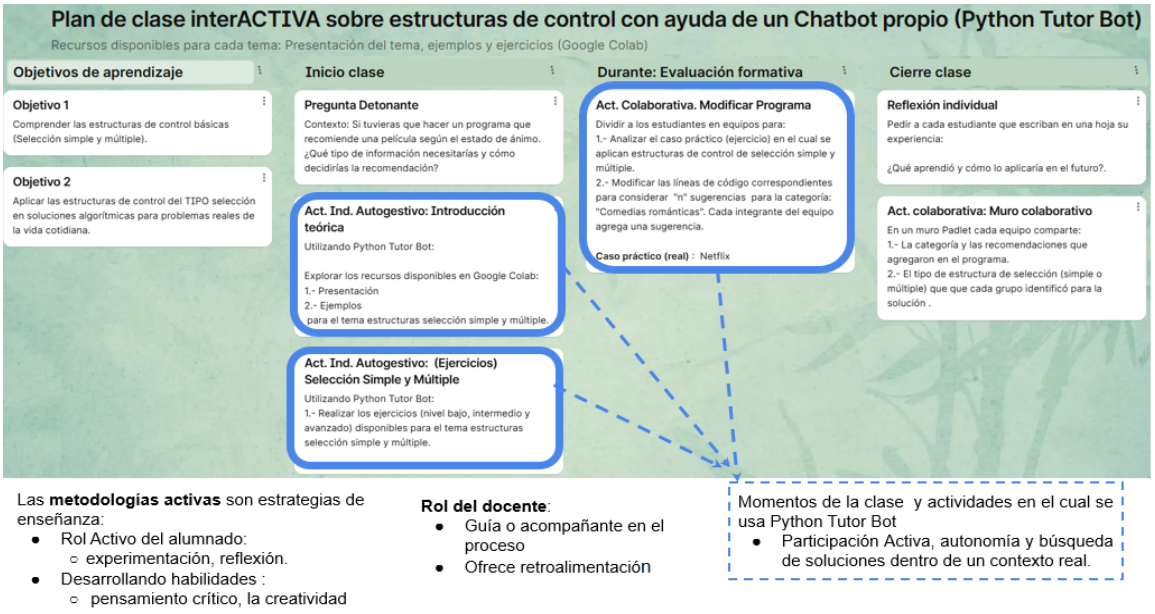
2. Métodos

El desarrollo del chatbot educativo denominado Python Tutor Bot, considera los principios de las metodologías activas [Zapata, 2024]. En particular, el diseño de

una clase interactiva basada en este enfoque y en la cual se indica el uso de Python Tutor Bot en actividades específicas. Las metodologías activas se caracterizan por promover un rol activo del alumnado, fomentando la experimentación y la reflexión. A través de estas estrategias, se busca desarrollar habilidades fundamentales, como el pensamiento crítico y la creatividad. Por su parte, una clase interactiva es un enfoque pedagógico en el que los estudiantes participan activamente, ya sea mediante actividades individuales, colaborativas, empleando tecnología, realizando una reflexión o prácticas kinestésicas. Se trata de un enfoque en el cual el alumnado se relaciona con el contenido del tema, sus compañeros y profesorado para lograr un aprendizaje profundo, significativo y dinámico [Nash, 2018]. La planeación de una clase interactiva se estructura en tres etapas: antes de la clase (inicio), durante la clase (desarrollo) y después de la clase (cierre). En la etapa inicio de la clase, se determinan los objetivos de aprendizaje, se seleccionan los materiales y recursos necesarios, así como se preparan las ayudas visuales que apoyarán al desarrollo de la clase. En la etapa desarrollo de la clase, se ejecutan actividades de diferentes tipos:

- Actividad detonante: Introducción al tema o problema mediante una pregunta, imagen o dinámica que despierte el interés del alumnado.
- Actividad instruccional: Presenta conceptos clave y herramientas necesarias para resolver la actividad detonante.
- Actividad individual: Se realizan ejercicios prácticos que permitan aplicar los conceptos aprendidos.
- Actividad colaborativa: Se apoya en el trabajo en equipo para diseñar y desarrollar partes de la solución de forma conjunta.
- Actividad integradora: Sintetiza y reúne las propuestas desarrolladas por los equipos, así como la presentación de los resultados obtenidos.

Finalmente, en la etapa cierre de la clase, se define una actividad para recuperar las experiencias de aprendizaje. La Figura 1 muestra un ejemplo de clase interactiva sobre el tema estructuras de control del tipo selección simple y múltiple. Se pueden observar tres actividades (2 individuales y 1 colaborativa) en las cuales se indica el uso del chatbot educativo Python Tutor Bot.



Fuente: elaboración propia.

Figura 1 Planeación clase interactiva con intervención de Python Tutor Bot.

La clase inicia con los siguientes objetivos de aprendizaje: comprender las estructuras de control de selección simple y múltiple y aplicar las estructuras de control en soluciones algorítmicas para problemas de la vida cotidiana. Después de fijar los objetivos de aprendizaje, se tiene la actividad detonante que establece tanto el contexto del problema de la vida cotidiana en el cual se aplican estructuras de selección simple y/o múltiple (hacer un programa de recomendaciones de películas según el estado de ánimo), así como la pregunta detonante: ¿Qué tipo de información necesitarías y cómo decidirías la recomendación de películas? La segunda actividad es la instruccional en la que se indica utilizar Python Tutor Bot para recuperar y explorar los recursos disponibles (presentación y ejemplos) sobre el tema estructuras de selección simple y múltiple. La tercera actividad es individual y autogestiva para solicitar al chatbot ejercicios con grados de complejidad, bajo, intermedio y avanzados sobre el tema estructuras de control de selección simple y múltiple. La última intervención del chatbot es a través de la actividad colaborativa “Modificar Programa”. El chatbot muestra a los equipos un programa en código fuente para su análisis y modificación. En este caso, los integrantes del equipo insertan líneas de código correspondientes a las sugerencias de películas para la categoría: "comedias románticas". El empleo del chatbot educativo favorece una

participación activa del alumnado. Finalmente, la clase interactiva termina con dos actividades de cierre: una actividad individual que solicita una reflexión sobre lo aprendido y su aplicación en otras situaciones de la vida cotidiana. La segunda actividad de cierre corresponde a una actividad colaborativa colocando en un muro digital colaborativo tanto las recomendaciones de películas que hizo el equipo, como el tipo de estructura de selección (simple, múltiple) que utilizaron e incorporaron en el programa colaborativo.

El desarrollo del chatbot educativo (Python Tutor Bot) sigue un enfoque tecnológico híbrido, priorizando la accesibilidad multiplataforma. También, se trata de un desarrollo hecho por estudiantes de la licenciatura en computación para estudiantes que desean aprender conceptos fundamentales de programación con el lenguaje de programación Python. Este enfoque alumno-alumno favorece el aprendizaje desde una perspectiva de alumno, así como una empatía con el Bot que se ve reflejada en el tono conversacional que utiliza en las interacciones con el usuario. Así mismo, la aplicación web desarrollada, utiliza Flask como backend junto con HTML/CSS nativo para la interfaz, asegurando compatibilidad con navegadores y dispositivos móviles sin requerir instalación. El uso de NetworkX para el grafo de conocimiento, así como FuzzyWuzzy para el procesamiento de lenguaje generan un sistema eficiente con dependencias mínimas, ideal para entornos con limitaciones de ancho de banda. Esta solución web garantiza una implementación rápida y un mantenimiento simplificado, con potencial para evolucionar hacia una Aplicación Web Progresiva (AWP) que aproveche el caché y las notificaciones push.

Por otro lado, Google Colab es un entorno basado en Jupyter Notebook que opera en la nube y ofrece una plataforma intuitiva, altamente flexible sin requerir configuración inicial. Incluye soporte para Python 3.6+ y acceso gratuito a GPUs/TPUs. Al estar basado en Jupyter, facilita la colaboración en tiempo real (con funciones similares a Google Docs) donde múltiples usuarios pueden editar cuadernos simultáneamente sin necesidad de instalar software local, requiriendo únicamente un navegador web.

Se utiliza Google Colab como un entorno de aprendizaje para facilitar el flujo de trabajo. Adicionalmente, esta plataforma permite: escribir y ejecutar código Python

de manera fluida; crear, cargar y compartir cuadernos interactivos Jupyter; así como importar y guardar archivos directamente desde/hacia Google Drive. Para el diseño del chatbot educativo (Python Tutor Bot) se analizaron los Recursos Digitales de Aprendizaje (RDA) disponibles para el curso de Fundamentos de Programación del tipo presentaciones temáticas, ejemplos (programas) [UEA, 2025].

La Tabla 1 muestra los 19 temas considerados en el proyecto y los 114 tipos de RDA desarrollados para ser gestionados por Python Tutor Bot. Para cada tema se realizaron 6 RDA del tipo: presentación (.pdf), ejemplo (.py), ejercicios de grado de dificultad (BIA): Básico, Intermedio y Avanzado, así como la solución en un solo archivo (programa en .py) de los tres ejercicios (BIA). El chatbot sigue una arquitectura cliente-servidor en capas, combinando patrones de diseño para separar responsabilidades y garantizar escalabilidad [Batra, 2016; Gamma,1994].

Tabla 1 Temas y recursos digitales de aprendizaje (RDA) asociados.

Num.	Tema	Presentación (pdf)	Ejemplo (.py)	Ejercicios BIA (.py)	Solución ejercicios (.py)
1	Identificadores	1	1	3	1
2	Variables	1	1	3	1
3	Tipos de datos simples	1	1	3	1
4	Operadores	1	1	3	1
5	Escritura	1	1	3	1
6	Lectura	1	1	3	1
7	Selección simple (if-else)	1	1	3	1
8	Selección múltiple (switch)	1	1	3	1
9	Iteración (for)	1	1	3	1
10	Iteración (while)	1	1	3	1
11	Iteración (do-while)	1	1	3	1
12	Arreglos (Introducción)	1	1	3	1
13	Arreglos (tipos)	1	1	3	1
14	Arreglos de registros (struct)	1	1	3	1
15	Arreglos & funciones	1	1	3	1
16	String	1	1	3	1
17	String & funciones propias	1	1	3	1
18	Matrices	1	1	3	1
19	Funciones	1	1	3	1

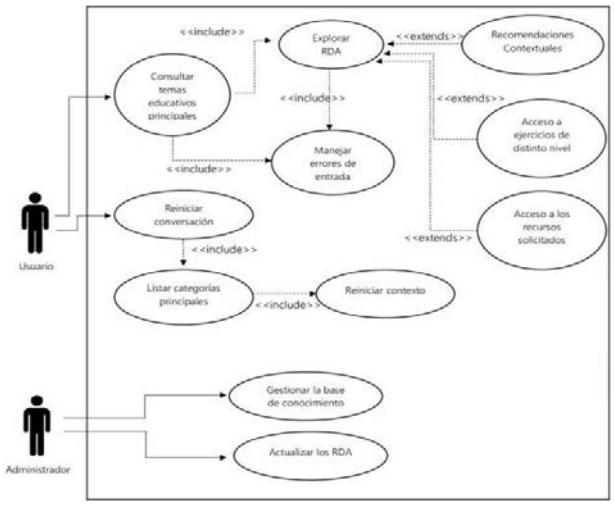
Fuente: elaboración propia.

La Tabla 2 describe las capas que conforman a Python Tutor Bot. Mientras que la Figura 2 muestra los casos de uso.

Tabla 2 Descripción de las capas que conforman a Python Tutor Bot.

Capa	Componentes	Responsabilidad
Frontend	HTML/CSS, JavaScript	Interfaz de usuario: renderiza el chat y envía mensajes al backend
Backend	Flash (Python)	Servidor web: recibe peticiones HTTP y las deriva al núcleo del chatbot
Lógica del chatbot	Chatbot.py, chatbot_wrapper.py	Procesa mensajes, maneja estados y consulta el grafo de conocimiento
Grafo de conocimiento	Knowledge_graph.py, kb.json	Almacena y estructura la información en un grafo (temas-recursos)

Fuente: elaboración propia.



Fuente: elaboración propia.

Figura 2 Python Tutor Bot Casos de Uso.

El archivo *kb.json* es la base de conocimiento estructurada en formato JSON, en el cual se establecen los tipos de nodos: categoría_principal (tema raíz), subtema (subdivisiones de un tema), RDA (materiales de apoyo: presentaciones, ejemplos, ejercicio con niveles de dificultad básico, intermedio y avanzado). Las relaciones se establecen de manera jerárquica entre los temas y subtemas. Cada nodo (categoría o subtema) puede tener recursos asociados.

El archivo *knowledge _graph.py* define la clase KnowledgeGraph, que utiliza la librería NetworkX para construir un grafo dirigido que representa la estructura jerárquica de conocimientos almacenada en *kb.json*. Transforma los datos del JSON en nodos (temas, subtemas, recursos) y aristas (relaciones como contains y has_resource), permitiendo una navegación eficiente entre categorías principales, sus subtemas anidados y los recursos asociados. Su función principal es servir

como base de conocimiento consultable para el chatbot, facilitando la recuperación de información estructurada mediante métodos como get_related_nodes. Finalmente, el archivo *chatbot.py* implementa el núcleo inteligente del chatbot mediante la clase PythonTutorChatbot, que gestiona conversaciones estructuradas usando un sistema de estados finitos (ConversationState) para guiar al usuario a través de temas jerárquicos. Utiliza FuzzyWuzzy para interpretar consultas con errores ortográficos y ofrece respuestas contextuales. Cabe mencionar que Python Tutor Bot utiliza un grafo de conocimiento en lugar de un árbol de decisión dado que se requiere modelar relaciones complejas y multidireccionales, permitiendo con ello conexiones flexibles entre conceptos [Durdymyradov, 2024]. En contraste, un árbol de decisión puede utilizarse para estructuras jerárquicas y predecibles, donde la información se organiza de forma lineal y sin retroalimentación entre nodos. Algunos chatbots comerciales son implementados a través de árboles de decisión. Sin embargo, los grafos son ideales para razonamiento contextual avanzado, mientras que los árboles funcionan mejor en escenarios con reglas claras y estáticas [Durdymyradov, 2024].

El diseño conversacional del chatbot se basa en un enfoque híbrido que combina un flujo guiado por estados con la flexibilidad para interpretar entradas variadas del usuario. Este modelo permite mantener una interacción coherente y controlada, sin perder la capacidad de adaptarse a consultas naturales o imprecisas. El sistema guía al usuario paso a paso según el contexto de la conversación. Uno de los componentes clave del chatbot es la máquina de estados, implementada mediante la clase ConversationState. Esta clase define los distintos momentos en los que puede encontrarse la conversación, como: IDLE (esperando interacción inicial), AWAITING_TOPIC (esperando selección de tema principal), AWAITING_SUBTEMA (esperando subtema), AWAITING_RESOURCE_TYPE (esperando tipo de recurso como teoría o ejercicios), y AWAITING_EXERCISE_LEVEL (esperando nivel de dificultad). Esta estructura facilita el diseño de un flujo conversacional predecible, segmentado y lógico. El procesamiento de inputs se realiza mediante técnicas de fuzzy matching utilizando la biblioteca FuzzyWuzzy, lo que permite corregir errores ortográficos comunes, por

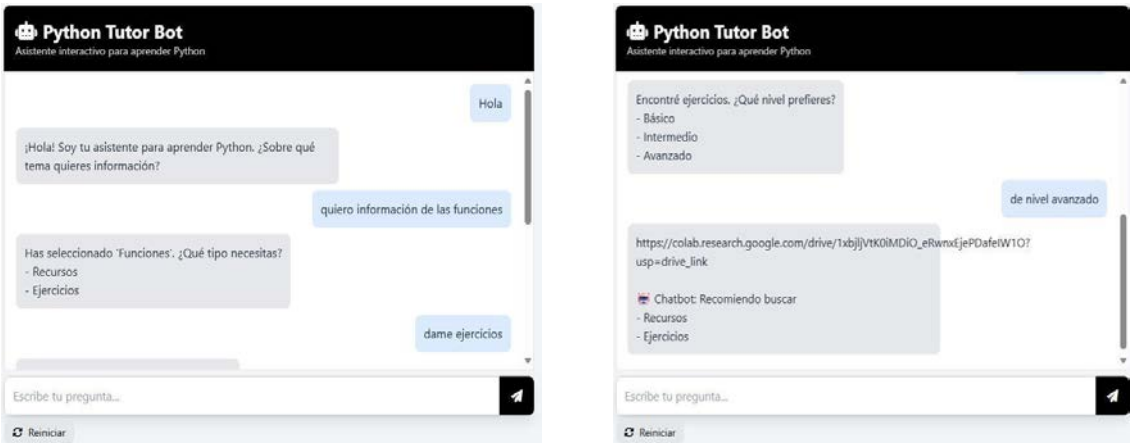
ejemplo, interpretando “funncones” como “funciones” si se supera un umbral mínimo de similitud del 70%. Adicionalmente, se aplica limpieza de texto usando expresiones regulares para eliminar acentos, símbolos y espacios innecesarios, lo cual mejora la precisión del sistema al interpretar las consultas del usuario. En cuanto a los patrones de diseño utilizados, el proyecto implementa varios enfoques clásicos de ingeniería de software [Batra, 2016; Gamma,1994]. El patrón State se emplea para gestionar el ConversationState, asegurando un flujo conversacional estructurado. El patrón Facade se aplica en el archivo *chatbot_wrapper.py*, que simplifica la interfaz entre el núcleo del chatbot y la aplicación web.

Cabe mencionar que la interacción básica con Python Tutor Bot consiste en respuestas predefinidas extraídas del grafo de conocimiento, útiles para quienes buscan contenido educativo estructurado. En su modalidad avanzada, el chatbot permite filtrar ejercicios por nivel de dificultad (básico, intermedio o avanzado) y manejar consultas ambiguas mediante fuzzy matching. Además, gestiona contextos de conversación multi-paso, recordando la temática en curso para ofrecer respuestas coherentes y personalizadas. Python Tutor Bot proporciona acceso directo a presentaciones, ejemplos de código y ejercicios prácticos, todos ellos vinculados a temas específicos. Esta funcionalidad facilita el aprendizaje dirigido y el refuerzo conceptual a través de materiales organizados [Holmes, 2019]. A través de la generación de enlaces a Google Colab, los usuarios pueden acceder a notebooks interactivos. Estos contienen ejercicios que pueden ejecutarse y modificarse directamente en la nube. Lo anterior fomenta el aprendizaje activo mediante la práctica con código real.

3. Resultados

Como resultado se tiene Python Tutor Bot un chatbot mínimo funcional accesible desde computadoras, gracias a una interfaz web adaptable. El sistema permite la navegación fluida por 19 temas y subtemas, así como el acceso rápido a 114 recursos como presentaciones, ejercicios-soluciones y enlaces a RDA en Google Colab. Esta accesibilidad amplía su alcance a diversos contextos, desde aulas presenciales hasta entornos de autoaprendizaje. Los recursos externos

almacenados en Google Drive y Google Colab, están organizados en una base de conocimiento (*kb.json*) con enlaces directos a presentaciones (PDF), ejemplos (notebooks interactivos), ejercicios por nivel (básico/intermedio/avanzado) y programas fuente (*scripts .py*). Al seleccionar un tema, el bot filtra estos recursos y devuelve la URL correspondiente, permitiendo a los usuarios acceder al contenido con la ventaja de ejecutar código directamente en Google Colab para un aprendizaje interactivo. La Figura 3, muestra un ejemplo de Interacción con Python Tutor Bot solicitando un Recurso Digital de Aprendizaje (RDA) del tipo ejercicios nivel avanzado para el tema funciones. También, puede observarse una navegación guiada, siendo el chatbot quien dirige la interacción con el usuario mediante preguntas secuenciales. En la Figura 3a se muestra la retroalimentación visual, en la parte derecha y en azul claro se encuentran los mensajes del usuario. En particular la solicitud de información sobre funciones con el texto “quiero información de las funciones”. Las respuestas del bot están alineadas a la izquierda y en color gris. Comenzando con un saludo y preguntando ¿sobre qué tema quieres información? En la parte inferior izquierda se encuentra el botón Reiniciar que permite restablecer la conversación sin recargar la página. En la Figura 3b se muestra la respuesta al usuario proporcionando el enlace al recurso que satisface la solicitud de información proporcionada y que en este caso son ejercicios de nivel avanzado del tema funciones.



a) Retroalimentación visual. *b) Respuesta con el enlace al RDA disponible.*

Fuente: elaboración propia.

Figura 3 Interacción con Python Tutor Bot solicitando RDA.

El sistema implementa múltiples estrategias para manejar entradas inesperadas o erróneas del usuario, garantizando una experiencia fluida incluso con errores ortográficos o ambiguos (Figura 4), por ejemplo, para la entrada “Lectura de sintasis” el bot la considera como “Lectura de Sintaxis” corrigiendo esto en la respuesta. La Figura 4a, muestra este caso, mientras que la Figura 4b muestra un RDA de tipo ejercicio en Google Colab a través de un Notebook.

a) Ejemplo de corrección de error de sintaxis.

b) Ejemplo de RDA disponible en Google Colab.

Fuente: elaboración propia.

Figura 4 Interacción con Python Tutor Bot solicitando RDA tipo ejercicios.

Por otro lado, la prueba piloto aplicada al prototipo estuvo compuesta por 15 alumnos quienes respondieron una encuesta de satisfacción. Los resultados obtenidos mostraron la viabilidad de Python Tutor Bot para coadyuvar al proceso de enseñanza aprendizaje sobre temas de fundamentos de programación.

Como trabajo a corto plazo, se tiene contemplado evaluar el prototipo con un grupo numeroso de control (estudiantes), que pruebe el chatbot en diferentes escenarios y conteste un instrumento de satisfacción en el cual se evalúe la experiencia de usuario, permitiendo identificar oportunidades de mejora basada en datos concretos. Así mismo, dentro de las mejoras futuras del chatbot educativo incluyen integrar PLN para interpretar sinónimos, variantes gramaticales y frases complejas del usuario, así como implementar autenticación y personalización para guardar preferencias y ofrecer recomendaciones adaptadas. Finalmente, la contribución de este trabajo es un chatbot mínimo funcional con potencial de replicabilidad en otras unidades de enseñanza con ajustes respectivos.

4. Discusión

La Inteligencia Artificial Generativa (IAg) está redefiniendo las funciones de los participantes del proceso de enseñanza aprendizaje, le permite al profesorado reinventarse, adaptándose a un panorama educativo en evolución, centrarse en la participación del alumnado, construir un entorno de bienestar y fomentar el apoyo personalizado para un aprendizaje profundo o significativo [Tin, 2025].

La automatización de tareas (como el empleo de Python Tutor Bot en la planeación de una clase interactiva o en un curso) permiten realizar ajustes en tiempo de ejecución y acordes a las necesidades individuales del alumnado [Holmes, 2019]. Así mismo, con recursos generados con IAg, el profesorado puede asumir los roles de creador o curador de recursos educativos relevantes y efectivos para los estudiantes [Tin, 2025].

Por otro lado, el alumnado debe tomar un rol más activo en su aprendizaje (autonomía y regulación), enfatizar en su participación activa y social a través de actividades colaborativas, participación en foros de discusión, exponer los resultados de sus tareas o proyectos de forma verbal o escrita a sus pares o foros adecuados como congresos o revistas, incentivando así, la gestión emocional al recibir una crítica o retroalimentación objetivas. Sin embargo, el aprendizaje socioemocional sigue siendo por el momento, de carácter humano. El profesorado debe guiar al alumnado no sólo académicamente sino también social y emocionalmente.

Finalmente, el proceso de enseñanza aprendizaje no queda completo sin requerir que también las instituciones educativas y la sociedad adquieran nuevos roles. Las instituciones educativas deben ser espacios de conexión social e innovación, ofreciendo experiencias de aprendizaje híbridas, personalizadas para preparar al alumnado tanto a futuros inciertos, como resaltando el valor que ellos tienen en la sociedad.

La sociedad en su nuevo rol debe impulsar una educación libre de violencia, equitativa y ética, asegurando que la tecnología sirva al desarrollo humano y no al contrario. Valorizando al humano que potencializa sus habilidades a través del uso de la IAg.

5. Conclusiones

El chatbot educativo Python Tutor Bot puede tener un impacto en el aprendizaje de conceptos fundamentales de programación utilizando Python Colab, especialmente entre usuarios principiantes. Al proveer una guía clara y accesible, facilita la comprensión de conceptos clave. Su capacidad para ofrecer recursos personalizados por nivel de dificultad permite a los estudiantes avanzar a su propio ritmo, reduciendo la frustración asociada con el aprendizaje autodidacta. La integración con Google Colab favorece la práctica de lo aprendido, lo cual es esencial para un aprendizaje auténtico o significativo. Actualmente para aprender se pueden utilizar una combinación de herramientas de IA generativa, sin embargo, Python Tutor Bot busca incentivar la humanización del proceso de enseñanza-aprendizaje (PEA), es decir, evitar delegar todo el proceso de aprendizaje a la IAg, lo cual impacta en el alumnado en una pérdida de habilidades críticas y pensamiento independiente debido a la dependencia de las herramientas disponibles sin alguna guía u orientación en su uso y construcción de prompts. Finalmente, en la era de la inteligencia artificial generativa, la figura de un tutor adquiere cada vez más importancia no solo en la academia, sino también en la vida. Un tutor humano-académico como digital deberán comprender el contexto psicotecnopedagógico del aprendizaje, fomentar en el alumnado el pensamiento crítico, la creatividad, inspirar la relación de confianza entre pares y la capacidad de empatía. Desafíos que necesitan ser abordados.

6. Referencias y Bibliografía

[1] Batra R., A Primer on Design Patterns. Leanpub, 2016.

[2] Chio, N., (2025). Uso de una herramienta de inteligencia artificial generativa para la creación de instrumentos de evaluación en el curso de sistemas embebidos. [Universidad Autónoma de Bucaramanga UNAB Universidad Oberta de Catalunya. Maestría en E-Learning]. Repositorio Institucional UNAB. http://hdl.handle.net/20.500.12749/28477.

[3] Durdymyradov, K., Moshkov, M., Ostonov, A., Decision Trees Versus Systems of Decision Rules: A Rough Set Approach, 2024.

[4] Gamma, E., Helm,R., Johnson,R., Vlissides, J., Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

[5] Hogan, A., Knowledge Graphs. Synthesis Lectures on Data, Semantics, and Knowledge (Springer/Morgan & Claypool), 2022.

[6] Holmes, W., Bialik, M., Fadel, C., Artificial Intelligence in Education: Promises and Implications for Teaching and Learning. Center for Curriculum Redesign, 2019.

[7] Kuhail, M. A., Alturki, N., Alramlawi, S., & Alhejori, K., (2022). Interacting with educational chatbots: A systematic review. Education and Information Technologies, 28(1). https://doi.org/10.1007/s10639-022-11177-3.

[8] Malik, S. I., Ashfaque, M. W., Mathew, R., Jabbar, J., Al-Nuaimi, R. S., & Alsideiri, A., (2022). Fostering the Learning Process in a Programming Course with a Chatbot. International Journal of Online Pedagogy and Course Design, 12(1), pp. 1–17. https://doi.org/10.4018/ijopcd.306686.

[9] Medina-Ramírez, R. C., Hernández Cerrito, P. C., Cabrera Jiménez, O. L., Rodríguez de la Colina, E., & Rincón García, E. A., (2024). Chatbots educativos en UAM-I: una familia abierta al tiempo. Contactos, Revista De Educación En Ciencias E Ingeniería, (137), pp. 37-42. https://contactos.izt.uam.mx/index.php/contactos/article/view/442.

[10] Nash, R., The InterActive Classroom: Practical Strategies for Involving Students in the Learning Process, 2018.

[11] Pérez, J. Q., Daradoumis, T., & Puig, J. M. M., (2020). Rediscovering the use of chatbots in education: A systematic literature review. Computer Applications in Engineering Education, 28(6). https://doi.org/10.1002/cae.22326.

[12] Sánchez, M. M., (2023). La inteligencia artificial generativa y la evaluación: ¿Qué pasará con los exámenes? Investigación en Educación Médica. Vol. 12. No 48. https://doi.org/10.22201/fm.20075057e.2023.48.23550.

[13] Tin H. H. K., Myae A. C., Nwe T. T., AI vs. Human Teachers: A Comparative Survey on the Potential for Substitution in Education. Indian Journal Science and Research, 5(3), pp. 85-94, 2025.

[14] UEA Fundamentos de Programación, (2025). Programa de estudios. http://lc.izt.uam.mx/wp-content/uploads/2018/10/2151103-Fundamentos-de-Programacion.pdf.

[15] Vallejo M, Molina J., La evaluación auténtica de los procesos de evaluación. Revista Iberoamericana de Educación, No. 64, pp. 11-25. ISSN 1022-6508, 2014.

[16] Zapata, Lascano W. A., Merino, López, F. J., Moreno Jarrín, E. N., Escobar, Vinueza, V. A., (2024). Metodologías activas para impulsar el proceso enseñanza-aprendizaje. Otros Horizontes otros desafíos. Ciencia Latina. Vol. 8, No. 3, Doi.or/103.37811/cl_rcm.v8i3.