«Talento Tech»

Iniciación a la

Programación con Python

Clase 16

Buenos Aires Ciudad

Clase N° 16 | Presentación del TFI y cierre del curso

Temario:

- Presentación de los proyectos realizados.
- Retroalimentación y discusión.
- Repaso de los objetivos del curso.
- Orientación sobre próximos pasos y recursos adicionales.

Objetivos de la Clase

Llegamos a la última clase del curso, un espacio dedicado a **cerrar el ciclo de aprendizaje**, reflexionar sobre lo que hemos aprendido y compartir los proyectos que cada estudiante ha desarrollado.

El objetivo principal de esta clase es que los participantes que lo deseen puedan **presentar su Trabajo Final Integrador**, demostrando cómo aplicaron los conceptos y herramientas vistas a lo largo del curso. A través de esta presentación, se fomentará la comunicación técnica, el pensamiento crítico y la capacidad de explicar decisiones de diseño y desarrollo. También se generará un espacio de **retroalimentación constructiva**, donde tanto los docentes como los compañeros podrán aportar comentarios y sugerencias sobre los proyectos. Esto permitirá identificar oportunidades de mejora y reconocer logros individuales y grupales.

Además de la presentación de proyectos, se realizará un **repaso general del curso**, destacando los conceptos fundamentales adquiridos y cómo estos se pueden seguir utilizando y profundizando en futuros proyectos. Se brindarán recomendaciones y recursos adicionales para seguir explorando el mundo de la programación con Python.

Finalmente, se abrirá un espacio de **cierre y despedida**, en el que se celebrarán los logros alcanzados, se reconocerá el esfuerzo de cada estudiante y se alentará a continuar aprendiendo y creciendo en el ámbito del desarrollo de software.

Fundamentos de la programación con Python.

El curso comenzó con la introducción a los conceptos esenciales de la programación en Python, sentando las bases para comprender cómo funcionan los programas y cómo estructurar el código de manera clara y eficiente. Se trabajó con variables y tipos de datos, entendiendo la diferencia entre valores numéricos, cadenas de texto y estructuras más complejas. También se exploraron los operadores matemáticos y lógicos, fundamentales para la manipulación de datos y la toma de decisiones dentro de un programa.

Uno de los primeros desafíos fue aprender a interactuar con el usuario mediante la entrada y salida de datos. Se trabajó con la función **input()** para recibir información y con **print()** para mostrar resultados en pantalla, incorporando desde el inicio la importancia de escribir código comprensible y funcional.

A medida que se avanzaba, se introdujeron las estructuras de control, que permitieron tomar decisiones dentro del programa con **if**, **elif** y **else**, y ejecutar tareas repetitivas con **while** y **for**. Estas herramientas resultaron clave para estructurar programas dinámicos, capaces de responder a diferentes situaciones y automatizar procesos sin necesidad de escribir instrucciones repetitivas.

La combinación de estos elementos fue el primer paso hacia la construcción de programas más complejos. Al dominar estos fundamentos, cada estudiante adquirió la capacidad de resolver problemas básicos utilizando Python, preparando el camino para abordar estructuras de datos más avanzadas y la organización del código en proyectos reales.

Estructuras de datos y manipulación

A medida que se profundizó en la programación con Python, se introdujeron las estructuras de datos, herramientas fundamentales para organizar y manipular la información de manera eficiente. Se comenzó con las listas, que permiten almacenar múltiples valores en una sola variable y acceder a ellos mediante índices. Se exploraron métodos como **append()**, **remove()** y **sort()**, comprendiendo cómo modificar, ordenar y gestionar los elementos dentro de una lista.

Luego se incorporaron las tuplas, similares a las listas pero con la diferencia clave de que son inmutables. Este concepto ayudó a entender cuándo es conveniente utilizar estructuras de datos que no se pueden modificar, asegurando la integridad de la información en ciertos contextos.

Un paso fundamental fue la introducción de los diccionarios, estructuras que permiten almacenar datos en pares clave-valor. Se comprendió cómo agregar, modificar y acceder a los elementos de un diccionario, reconociendo su utilidad para organizar información de manera más estructurada y facilitar su acceso sin necesidad de recorrer grandes volúmenes de datos.

Finalmente, para reforzar estos conceptos, se trabajó en la iteración sobre estas estructuras mediante los bucles **for** y **while**. Se experimentó con la manipulación de listas y diccionarios dentro de ciclos, optimizando la manera en que se procesan grandes cantidades de datos en un programa. Este aprendizaje resultó clave para desarrollar sistemas más complejos, como los que se aplicarían más adelante en la gestión de bases de datos.

Modularización y funciones

A medida que los programas fueron creciendo en complejidad, se hizo evidente la necesidad de organizar mejor el código para hacerlo más claro, reutilizable y fácil de mantener. En este contexto, se introdujeron las funciones definidas por el usuario, permitiendo encapsular bloques de código en unidades independientes que pueden ser llamadas en distintos momentos sin necesidad de reescribirlas.

Se aprendió a definir funciones con y sin parámetros, comprendiendo cómo enviar información a una función y recibir un resultado a partir de su ejecución. También se exploró el uso de return, que permite que una función devuelva un valor específico para ser utilizado en otras partes del programa. Se enfatizó la importancia de la modularización como una estrategia clave para desarrollar software estructurado y eficiente.

Otro concepto fundamental fue el alcance de las variables dentro y fuera de las funciones. Se explicó la diferencia entre variables locales y globales, destacando las mejores prácticas para evitar conflictos y mejorar la legibilidad del código. Además, se trabajó en la documentación de funciones mediante **docstrings**, asegurando que cada fragmento de código estuviera correctamente explicado y pudiera ser comprendido tanto por su autor como por otras personas que lo utilizaran en el futuro.

Finalmente, se vio la importancia de dividir un programa en módulos, permitiendo importar funciones desde otros archivos y organizando el código en unidades más manejables. Esta práctica no solo simplifica el desarrollo, sino que también es un estándar en la industria para mantener proyectos escalables y bien estructurados.

Persistencia de datos y manejo de errores.

Guardar y recuperar información de manera permanente es una necesidad fundamental en cualquier sistema informático. Para abordar este desafío, se introdujo el concepto de persistencia de datos y se exploraron las distintas formas en que Python permite almacenar información más allá de la ejecución del programa.

Primero, se trabajó con archivos de texto, aprendiendo a leer y escribir datos en archivos mediante el uso de **open()**, **read()**, **write() y close()**. Se explicó cómo abrir archivos en diferentes modos, como lectura (**"r"**), escritura (**"w"**) y adición (**"a"**), permitiendo a los programas conservar información entre distintas ejecuciones. Se enfatizó la importancia de cerrar los archivos correctamente y se mostró el uso de **with open()**, una alternativa más segura para gestionar archivos sin riesgo de errores por omisión de cierre.

El manejo de excepciones fue otro tema clave dentro de esta unidad. Se introdujo la estructura **try-except** como una herramienta fundamental para prevenir fallos inesperados en la ejecución de un programa. Se trabajó con distintos tipos de errores, como **FileNotFoundError**, **ValueError** y **ZeroDivisionError**, comprendiendo cómo capturar y manejar cada uno para mejorar la estabilidad del código. Además, se exploró el uso de **finally**, que garantiza la ejecución de ciertos bloques de código, como el cierre de archivos o la desconexión de bases de datos, independientemente de si ocurrió o no un error.

A medida que el curso avanzó, se presentaron bases de datos como una solución más eficiente y estructurada para almacenar grandes volúmenes de información. Se realizó una introducción a **SQLite**, un sistema de gestión de bases de datos ligero y de fácil integración con Python. Se aprendió a crear bases de datos y tablas, definir estructuras de datos y realizar operaciones básicas con SQL, como **SELECT**, **INSERT**, **UPDATE** y **DELETE**.

Para reforzar la seguridad y prevenir ataques, se abordaron las **inyecciones SQL** y se explicó cómo usar parámetros en consultas para evitar vulnerabilidades. También se trabajó con transacciones en bases de datos, asegurando que las operaciones se ejecuten de forma segura mediante el uso de **BEGIN TRANSACTION**, **COMMIT** y **ROLLBACK**, lo que garantiza la consistencia de los datos ante posibles errores o interrupciones inesperadas.

Desarrollo de aplicaciones con bases de datos

El uso de bases de datos permitió consolidar los conocimientos adquiridos previamente, dando lugar a la construcción de aplicaciones más robustas y funcionales. Se trabajó con **SQLite**, un sistema de bases de datos liviano que se integra directamente con Python sin necesidad de configuraciones adicionales.

El primer paso fue aprender a establecer una conexión con la base de datos y definir una estructura de almacenamiento mediante la creación de tablas. Se exploró el uso de la sentencia **CREATE TABLE**, permitiendo definir los distintos tipos de datos y restricciones para cada campo, como claves primarias (**PRIMARY KEY**), valores únicos (**UNIQUE**) y restricciones de no nulo **(NOT NULL**).

Una vez establecida la estructura, se avanzó en la implementación de las **operaciones CRUD** (Create, Read, Update y Delete), fundamentales para la manipulación de datos en cualquier aplicación. Se desarrollaron funciones que permitieron insertar registros en la base de datos (**INSERT**), consultar información almacenada (**SELECT**), actualizar registros (**UPDATE**) y eliminarlos (**DELETE**). Se puso especial énfasis en el uso de **parámetros en consultas SQL**, evitando la construcción manual de cadenas y minimizando riesgos de inyección de código malicioso.

A medida que el curso avanzó, se integraron herramientas adicionales, como **Colorama**, para mejorar la interfaz del usuario en la terminal y facilitar la visualización de mensajes de error o confirmación en distintos colores. Esto contribuyó a una experiencia más intuitiva y organizada dentro del sistema.

Con estos conocimientos, los alumnos pudieron aplicar bases de datos en sus proyectos, estructurando la información de manera eficiente y asegurando la integridad de los datos almacenados. Este aprendizaje resultó fundamental para la preparación del **Trabajo Final Integrador**, en el que cada estudiante desarrolló su propia aplicación completa.

Trabajo Final Integrador: Aplicación de los conocimientos

El cierre del curso estuvo marcado por el desarrollo del **Trabajo Final Integrador (TFI)**, un proyecto diseñado para aplicar de manera práctica todos los conceptos adquiridos a lo largo de las clases. Este desafío permitió consolidar conocimientos sobre estructuras de datos, control de flujo, manipulación de archivos y bases de datos, asegurando que cada estudiante pudiera desarrollar una aplicación funcional basada en Python.

El proyecto consistió en la creación de un sistema de gestión que permitiera a los usuarios **registrar, consultar, modificar y eliminar datos** de manera eficiente. Para ello, se estructuró un **CRUD completo** utilizando bases de datos SQLite, con validaciones estrictas y un flujo de trabajo optimizado para asegurar la integridad de los datos almacenados. Se puso especial énfasis en el uso de **transacciones**, previniendo inconsistencias en la base de datos y asegurando que cada operación se ejecutara de manera correcta o fuera revertida en caso de error.

Uno de los aspectos fundamentales del TFI fue la implementación de **validaciones en la entrada de datos**, asegurando que los valores ingresados fueran correctos y consistentes. Se utilizó **try-except** para manejar excepciones y se integró **Colorama** para proporcionar una interfaz clara e intuitiva en la terminal, diferenciando mensajes de error, advertencia y confirmación con colores específicos.

A lo largo del desarrollo del proyecto, se incentivó la aplicación de **buenas prácticas de programación**, promoviendo la modularización del código mediante funciones reutilizables. Además, se destacó la importancia de mantener una estructura clara y bien documentada, asegurando que el código fuera fácil de leer y mantener.

La última etapa del TFI consistió en la presentación del proyecto, brindando a cada estudiante la oportunidad de evaluar su trabajo y compartir su experiencia de desarrollo. Se

ofreció una instancia de **retroalimentación**, permitiendo recibir sugerencias y comentarios que pudieran contribuir a mejorar futuras implementaciones. Esta instancia también sirvió para reforzar la confianza en las habilidades adquiridas, demostrando la capacidad de cada alumno para abordar proyectos de programación completos.

El Trabajo Final Integrador no solo fue un ejercicio de evaluación, sino también un primer paso hacia la aplicación real de Python en la resolución de problemas cotidianos. Con este proyecto, los alumnos adquirieron una base sólida para seguir explorando el desarrollo de software y continuar su aprendizaje en áreas más avanzadas.

Conclusión y cierre del curso

A lo largo de este curso, los estudiantes han recorrido un camino progresivo en el aprendizaje de Python, comenzando con los fundamentos de la programación hasta llegar a la integración de bases de datos y el desarrollo de aplicaciones completas. Cada clase fue diseñada para introducir nuevos conceptos de manera gradual, asegurando que cada tema fuera comprendido y aplicado en ejercicios prácticos, consolidando así un aprendizaje significativo.

El curso no solo se enfocó en el lenguaje Python como herramienta de desarrollo, sino también en la lógica de programación, el diseño estructurado y las buenas prácticas que garantizan la escritura de código claro, eficiente y mantenible. Desde el uso de estructuras condicionales y bucles hasta la implementación de funciones y módulos, cada elemento aprendido se convirtió en una pieza clave para el desarrollo de programas más robustos y organizados.

La incorporación de bases de datos y la persistencia de datos permitió a los estudiantes dar un salto en la funcionalidad de sus aplicaciones, pasando de programas efímeros que operan solo en memoria a sistemas capaces de almacenar información de manera permanente. El uso de SQLite como base de datos integrada en Python representó una solución accesible y poderosa para gestionar información de forma estructurada, y su combinación con el control de errores y el manejo de excepciones brindó a los estudiantes herramientas para escribir código más seguro y confiable.

El **Trabajo Final Integrador** marcó el punto culminante del curso, permitiendo a cada estudiante demostrar su capacidad para aplicar lo aprendido en un proyecto real. Más allá de la implementación técnica, el proceso de desarrollo les permitió enfrentar desafíos, tomar decisiones de diseño y experimentar el ciclo completo de construcción de una aplicación. La presentación del proyecto y la retroalimentación recibida fueron parte fundamental de la experiencia, reforzando la importancia de la documentación, la organización del código y la comunicación efectiva de los resultados.

Este curso no solo ha sido una introducción a Python, sino también una invitación a continuar explorando el mundo de la programación. Cada herramienta aprendida puede ser la base para seguir avanzando en áreas como el desarrollo web, la automatización, la inteligencia artificial o la ciencia de datos. El aprendizaje no se detiene aquí, y con una base

sólida, cada estudiante tiene la posibilidad de seguir perfeccionando sus habilidades y aplicarlas en nuevos desafíos.

Con esta última clase, cerramos una etapa, pero dejamos abierta la puerta a nuevas oportunidades de aprendizaje y crecimiento en la programación. ¡Felicitaciones a cada estudiante por haber llegado hasta aquí y por todo el esfuerzo invertido en su desarrollo como programadores y programadoras!

Materiales y Recursos Adicionales:

Enlace:
TalentoTech: Construí tu futuro en tecnología y potenciá tu carrera con nuestros cursos

---

Preguntas para Reflexionar:

1. **¿Cómo ha cambiado tu comprensión de la programación desde el inicio del curso hasta ahora?**
   Pensá en lo que sabías al comenzar y en las habilidades que desarrollaste durante cada clase.
2. **Si tuvieras que explicarle a alguien que no conoce Python, ¿cómo describirías su estructura y principales características?**
   Reflexioná sobre qué hace a Python un lenguaje accesible y poderoso para desarrollar software.
3. **¿Cuál de los temas aprendidos te resultó más desafiante y cómo lograste superarlo?**
   Recordá algún concepto o práctica que al principio te pareció difícil, pero que luego comprendiste con la práctica.
4. **¿Cómo creés que el uso de funciones y módulos mejoró la calidad de tu código?**
   Evaluá el impacto de dividir tu código en partes reutilizables y organizadas.
5. **¿Por qué la persistencia de datos es una característica clave en el desarrollo de aplicaciones?**
   Pensá en la importancia de guardar información más allá de la ejecución del programa y cómo lo implementaste con archivos y bases de datos.
6. **¿Qué aprendiste sobre la seguridad en la programación, especialmente con bases de datos y consultas SQL?**
   Reflexioná sobre los riesgos que enfrentan las bases de datos y cómo prevenir inyecciones SQL en tus aplicaciones.
7. **Si tuvieras que mejorar o ampliar el Trabajo Final Integrador, ¿qué funcionalidades agregarías?**
   Imaginá qué cambios o mejoras podrías hacer para optimizar tu proyecto.

8. **¿Cómo podrías aplicar los conocimientos adquiridos en este curso en un proyecto o problema del mundo real?**
   Relacioná lo aprendido con situaciones concretas que podrías enfrentar en un entorno profesional o académico.
9. **¿Qué concepto o herramienta del curso te gustaría seguir profundizando?**
   Identificá qué área te resultó más interesante y cómo podrías seguir aprendiendo sobre ella.
10. **¿Qué consejo le darías a alguien que está por empezar este curso?**
    Pensá en los desafíos y aprendizajes que tuviste y qué recomendación podría ayudar a futuros estudiantes.

Próximos Pasos:

El curso ha llegado a su fin, pero el camino en la programación y el desarrollo de software apenas comienza. A lo largo de estas clases, aprendiste los fundamentos de Python, bases de datos, manipulación de archivos, estructuras de datos y muchos otros conceptos que te permitirán abordar proyectos cada vez más desafiantes. Sin embargo, la programación es un campo en constante evolución, y seguir capacitándote es fundamental para consolidar lo aprendido y expandir tus habilidades.

Si querés continuar tu formación en el ámbito de la tecnología, existen múltiples rutas de especialización disponibles que te permitirán profundizar en distintas áreas según tus intereses y objetivos profesionales.

Si te interesa el **desarrollo web**, podés convertirte en **Desarrollador Full Stack Node o Java**, completando un programa de cuatro módulos que comienza con la **Programación Inicial con Python** que acabas de finalizar, y termina con el desarrollo en **React y Back-End con Node.js o Java**. Esta formación te permitirá crear aplicaciones completas, tanto en la parte visual como en la lógica del servidor, preparándote para trabajar en proyectos integrales de software.

Para quienes prefieran el mundo de los **videojuegos**, existe un trayecto de especialización en **Unity 2D y 3D**, que comienza con una sólida base en **Programación Inicial con Python** que ya tenes, y luego se adentra en el desarrollo de videojuegos utilizando el motor Unity. Esta capacitación te brindará las herramientas necesarias para diseñar y programar experiencias interactivas.

Si tu interés está en el análisis y procesamiento de datos, hay opciones como **Analista de Datos o Data Scientist**. Ambas especializaciones parten tambien de **Programación Inicial con Python**, y luego avanzan en el uso de herramientas de **Business Intelligence, Data Analytics y Machine Learning**. Estas formaciones son ideales para quienes buscan aplicar la programación en la toma de decisiones basadas en datos, un campo con creciente demanda en múltiples industrias.

Por otro lado, si tu objetivo es mejorar la calidad del software, el área de **Testing QA y Tester Engineer** ofrece la posibilidad de especializarte en la detección y automatización de

pruebas para garantizar el funcionamiento óptimo de las aplicaciones. Este camino incluye formación en **Testing QA manual y automatizado**, complementado con conocimientos en Python para desarrollar herramientas que faciliten la validación de software.

Finalmente, si querés enfocarte en el diseño y construcción de interfaces web, la especialización en **Desarrollo Front-End** te permitirá dominar **JavaScript, React y otras tecnologías del lado del cliente** para crear sitios web dinámicos y atractivos.

Cada uno de estos caminos representa una oportunidad para seguir creciendo y consolidando tus habilidades en programación. Independientemente de la especialización que elijas, lo más importante es seguir practicando, construyendo proyectos propios y aplicando lo aprendido en situaciones reales. La tecnología avanza rápidamente, y quienes se capacitan y se mantienen actualizados tienen la posibilidad de acceder a nuevas oportunidades y desafíos profesionales.

Este curso fue solo el primer paso en tu recorrido. Ahora, la decisión está en tus manos: ¿qué camino vas a seguir? 🚀


Buenos Aires
aprende
Agencia de Habilidades para el Futuro
Buenos
Aires
Ciudad