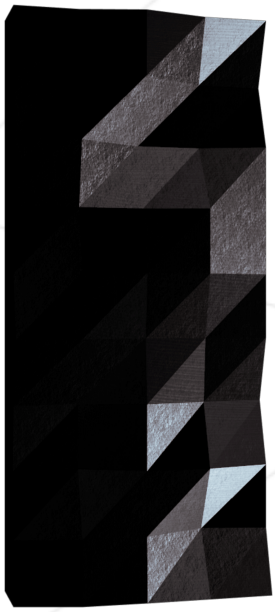

Cultivando Código

Fundamentos de Python a través de datos de huertos comunitarios

Resumen: Descubre la sintaxis y la semántica fundamentales de Python analizando datos de huertos comunitarios. Aprende expresiones, variables, funciones y flujo de control al mismo tiempo que contribuyes a iniciativas comunitarias sostenibles.

Versión: 1.0

Índice general

Capítulo I

Prólogo

En los huertos comunitarios de todo el mundo, los datos cuentan historias de crecimiento, colaboración e impacto.

Desde el seguimiento de las cosechas que alimentan a familias locales hasta la supervisión del consumo de agua, cada medición importa. Cada dato representa la dedicación de alguien: las horas de voluntariado durante un fin de semana, el primer tomate de una niña o niño o la persona mayor del vecindario que comparte décadas de sabiduría hortícola.

Python, al igual que estos huertos, crece a partir de semillas simples hasta convertirse en algo poderoso y enriquecedor. Nombrado en honor al programa Monty Python's Flying Circus, nos recuerda que aprender debe ser algo alegre, accesible e inclusivo. Hoy, Python ayuda al sector científico a comprender el cambio climático, permite que las comunidades optimicen el uso compartido de recursos y empodera a cualquiera para transformar datos en bruto en información significativa.

En este proyecto, descubrirás las bases fundamentales de Python mientras analizas datos reales de huertos comunitarios. Aprenderás que programar, al igual que cultivar, consiste en fomentar el crecimiento, tanto en el código como en las comunidades a las que servimos.

Capítulo II

Instrucciones sobre la IA

Contexto

Durante tu proceso de aprendizaje, la IA puede ayudarte con muchas tareas diferentes. Tómate el tiempo necesario para explorar las diversas capacidades de las herramientas de IA y cómo pueden apoyarte con tu trabajo. Sin embargo, siempre debes abordarlas con precaución y evaluar de forma crítica los resultados. Ya sea código, documentación, ideas o explicaciones técnicas, nunca podrás saber con total certeza si tu pregunta está bien formulada o si el contenido generado es el adecuado. Las personas que te rodean son tu recurso más valioso para ayudarte a evitar errores y puntos ciegos.

Mensaje principal:

- Utiliza la IA para reducir las tareas repetitivas o tediosas.
- Desarrolla habilidades de prompting, ya sea para programacion o para otros temas, que beneficiarán tu futura carrera.
- Aprende cómo funcionan los sistemas de IA para anticipar de forma eficiente y evitar los riesgos comunes, sesgos y problemas éticos.
- Sigue trabajando con tus compañeros para desarrollar tanto habilidades técnicas como habilidades transversales.
- Utiliza únicamente contenido generado por IA que entiendas completamente y del cual puedas responsabilizarte.

Reglas para estudiantes:

- Debes tomarte el tiempo necesario para explorar las herramientas de IA y comprender cómo funcionan, para poder utilizarlas de manera ética y reducir los sesgos potenciales.
- Debes reflexionar sobre tu problema antes de dar instrucciones a la IA. Esto te ayuda a escribir preguntas, instrucciones o conjuntos de datos más claros, detalladas y relevantes utilizando un vocabulario preciso.

- Debes desarrollar el hábito de revisar, cuestionar y probar sistemáticamente cualquier contenido generado por la IA.
- Debes buscar siempre la revisión de otras personas, no te limites a confiar en tu propia validación.

Resultados de esta etapa:

- Desarrollar habilidades de prompting tanto generales como de ámbito especifico.
- Aumentar tu productividad con un uso eficaz de las herramientas de IA.
- Seguir fortaleciendo el pensamiento computacional, la resolución de problemas, la adaptabilidad y la colaboración.

Comentarios y ejemplos:

- Ten en cuenta que la IA puede no tener la respuesta correcta porque esa respuesta no esté ni siquiera en Internet. Además, si te da soluciones incorrectas, intenta no insistir y busca ayuda entre las personas que te rodean. Vas a ahorrarte tiempo y vas a sumar en compresión.
- Vas a enfretarte con frecuencia a situaciones (como exámenes o evaluaciones) donde debes demostrar una comprensión real. Prepárate, sigue construyendo tanto tus habilidades técnicas como transversales.
- Explicar tu razonamiento y debatir con otras personas suele revelar lagunas en tu comprensión de un concepto. Prioriza el aprendizaje entre pares.
- Lo normal es que la herramienta de IA que utilices no conozca tu contexto específico (a menos que se lo indiques), así que te dará respuestas genéricas. Si buscas información más adecuada y más precisa en relación a tu entorno cercano, confía en el resto de estudiantes.
- Donde la IA tiende a generar la respuesta más probable, el resto de estudiantes puede proporcionar perspectivas alternativas y matices valiosos. Confía en la comunidad de 42 como un punto de control de calidad.

✓ Buenas prácticas:

> Le pregunto a la IA: "¿Cómo pruebo una función de ordenación?"Me da algunas ideas. Las pruebo y reviso los resultados con otra persona. Refinamos el enfoque de manera conjunta.

✗ Bad practice:

> Le pido a la IA que escriba una función completa, la copio y la pego en mi proyecto. Durante la evaluación entre pares, no puedo explicar qué hace ni por qué. Pierdo credibilidad. Suspendo mi proyecto.

✓ Good practice:

Utilizo la IA para ayudarme a diseñar un parser. Luego, reviso la lógica con otra persona. Encontramos dos errores y lo reescribimos juntos: mejor, más limpio y comprendiendo al 100

✗ Bad practice:

Dejo que Copilot genere mi código para una parte clave de mi proyecto. Compila, pero no puedo explicar cómo maneja los pipes. Durante la evaluación, no puedo justificarlo y suspendo mi proyecto.

Capítulo III

Introducción

¡Te damos la bienvenida a Cultivando Código!

En este proyecto descubrirás los conceptos básicos de Python a través de escenarios de huertos comunitarios. Trabajarás con ejercicios prácticos que introducen conceptos fundamentales de programación en un contexto atractivo y realista.

Cada ejercicio se construye en base al anterior, ayudándote a desarrollar habilidades esenciales de programación mientras realizas tareas significativas.

IMPORTANTE: En cada ejercicio, debes escribir SOLO una función (no un programa). Cada archivo debe contener únicamente la función solicitada, que gestionará directamente la entrada y salida de información. No escribas bloques if __name__ == "__main__": ni llames a la función directamente en tus archivos.

HERRAMIENTA DE AYUDA: Para ayudarte a probar tus ejercicios, encontrarás adjunto un archivo main.py. Copia este archivo a tu directorio de trabajo y ejecuta python3 main.py para probar fácilmente tus funciones. Esta herramienta las importará y probará automáticamente.

Capítulo IV

Instrucciones Generales

- Tus funciones deben estar escritas en Python 3.10+
- Tu código debe respetar los estándares de linter flake8
- Cada ejercicio debe estar en su propio archivo
- Cada archivo debe contener solo la función solicitada
- Los nombres de las funciones deben coincidir exactamente con lo solicitado
- No necesitas validar la entrada de datos ni manejar casos de error a menos que se indique explícitamente
- Para números negativos o entradas no válidas, el comportamiento es indefinido (no necesitas manejar estos casos)
- Las anotaciones de tipo son opcionales pero recomendadas con fines de aprendizaje (se introducirán en el Ejercicio 7)

Nota específica de Cultivando Código: Dado que este es un proyecto introductorio enfocado en la sintaxis básica de Python, solo necesitas entregar los archivos de cada función (por ejemplo, ft_hello_garden.py, ft_plot_area.py, etc.). Las herramientas avanzadas de gestión de proyectos se introducirán en proyectos posteriores.

Capítulo V

Ejercicio 0: Hola Huerto

	Ejercicio: 0
ft_hello_garden	
Directorio de entrega: *ex0/*	
Archivos a entregar: `ft_hello_garden.py`	
Funciones autorizadas: `print()`	

¡Te damos la bienvenida a tu primera función en Python! Escribe una función llamada `ft_hello_garden` que muestre un mensaje de bienvenida para el huerto comunitario.

```
>>> ft_hello_garden()
Hello, Garden Community!
```

Escribe únicamente la función ft_hello_garden(), no un programa principal. La función debe manejar directamente la entrada y salida de datos.

Este es tu primer paso en la programación con Python. Observa cómo las funciones pueden mostrar mensajes.

Capítulo VI

Ejercicio 1: Área del Huerto

	Ejercicio: 1
ft_plot_area	
Directorio de entrega: *ex1/*	
Archivos a entregar: `ft_plot_area.py`	
Funciones autorizadas: `input(), int(), print()`	

Un huerto comunitario necesita conocer el área de una parcela rectangular. Escribe una función llamada `ft_plot_area` que solicite la longitud y el ancho del terreno, y que luego calcule y muestre el área.

```
>>> ft_plot_area()
Enter length: 5
Enter width: 3
Plot area: 15
```

Escribe únicamente la función ft_plot_area(), no un programa principal. La función debe manejar directamente la entrada y salida de datos.

Observa cómo puedes almacenar números y realizar cálculos con ellos.

Capítulo VII

Ejercicio 2: Cosecha Total

	Ejercicio: 2
ft_harvest_total	
Directorio de entrega: *ex2/*	
Archivos a entregar: `ft_harvest_total.py`	
Funciones autorizadas: `input()`, `int()`, `print()`	

Una persona jardinera recolectó vegetales durante 3 días diferentes. Escribe una función llamada `ft_harvest_total` que solicite el peso de cada cosecha y calcule el total.

```
>>> ft_harvest_total()
Day 1 harvest: 5
Day 2 harvest: 8
Day 3 harvest: 3
Total harvest: 16
```

Escribe únicamente la función ft_harvest_total(), no un programa principal. La función debe gestionar directamente la entrada y salida de datos.

Observa cómo puedes sumar números y almacenar resultados.

Capítulo VIII

Ejercicio 3: Comprobación de la Edad de la Planta

	Ejercicio: 3
ft_plant_age	
Directorio de entrega: *ex*3/	
Archivos a entregar: `ft_plant_age.py`	
Funciones autorizadas: `input()`, `int()`, `print()`	

Escribe una función llamada `ft_plant_age` que solicite la edad de una planta en días y diga si está lista (tiene más de 60 días) o no para cosechar.

```
>>> ft_plant_age()
Enter plant age in days: 75
Plant is ready to harvest!
>>> ft_plant_age()
Enter plant age in days: 45
Plant needs more time to grow.
```

Escribe únicamente la función ft_plant_age(), no un programa principal. La función debe gestionar directamente la entrada y salida de datos.

Los programas pueden tomar decisiones según los valores que reciben.

Capítulo IX

Ejercicio 4: Recordatorio de Riego

	Ejercicio: 4
ft_water_reminder	
Directorio de entrega: *ex4/*	
Archivos a entregar: `ft_water_reminder.py`	
Funciones autorizadas: `input()`, `int()`, `print()`	

Escribe una función llamada `ft_water_reminder` que solicite el número de días desde el último riego. Si han pasado más de 2 días, muestra "Water the plants!", de lo contrario muestra "Plants are fine".

```
>>> ft_water_reminder()
Days since last watering: 4
Water the plants!
>>> ft_water_reminder()
Days since last watering: 1
Plants are fine
```

Escribe únicamente la función ft_water_reminder(), no un programa principal. La función debe gestionar directamente la entrada y salida de datos.

Observa cómo los programas pueden dar respuestas diferentes dependiendo de las condiciones.

Capítulo X

Ejercicio 5: Contar hasta Cosechar

	Ejercicio: 5
ft_count_harvest	
Directorio de entrega: *ex5/*	
Archivos a entregar: `ft_count_harvest_iterative.py`, `ft_count_harvest_recursive.py`	
Funciones autorizadas: `input()`, `int()`, `print()`, `range()`	

Escribe dos funciones llamadas `ft_count_harvest_iterative` y `ft_count_harvest_recursive`. Ambas deben contar desde 1 hasta un número dado, imprimiendo cada día hasta el momento de la cosecha.

```
>>> ft_count_harvest_iterative()
Days until harvest: 5
Day 1
Day 2
Day 3
Day 4
Day 5
Harvest time!

>>> ft_count_harvest_recursive()
Days until harvest: 5
Day 1
Day 2
Day 3
Day 4
Day 5
Harvest time!
```

Escribe únicamente las funciones `ft_count_harvest_iterative()` y `ft_count_harvest_recursive()`, no un programa principal. Las funciones deben gestionar directamente la entrada y salida de datos.

A veces es necesario repetir acciones. Puedes hacerlo mediante bucles (iteración) o llamando a la misma función nuevamente (recursión). ¿Qué sucede cuando repites una acción varias veces?

Capítulo XI

Ejercicio 6: Resumen del Huerto

	Ejercicio: 6
ft_garden_summary	
Directorio de entrega: *ex*6/	
Archivos a entregar: `ft_garden_summary.py`	
Funciones autorizadas: `input(), print()`	

Escribe una función llamada `ft_garden_summary` que solicite el nombre del huerto y su número de plantas, y que luego muestre un resumen sencillo con un mensaje de estado fijo.

```
>>> ft_garden_summary()
Enter garden name: Community Garden
Enter number of plants: 25
Garden: Community Garden
Plants: 25
Status: Growing well!
```

Escribe únicamente la función ft_garden_summary(), no un programa principal. La función debe gestionar directamente la entrada y salida de datos.

Observa cómo puedes combinar diferentes tipos de información para crear algo útil. Ten en cuenta que "Status: Growing well!."es un mensaje invariable que debe mostrarse exactamente como aparece.

Capítulo XII

Ejercicio 7: Inventario de Semillas con Anotaciones de Tipo

	Ejercicio: 7
ft_seed_inventory	
Directorio de entrega: *ex*7/	
Archivos a entregar: `ft_seed_inventory.py`	
Funciones autorizadas: `print()`	

La persona coordinadora del huerto necesita llevar un registro del inventario de semillas con tipos de datos precisos. Escribe una función llamada `ft_seed_inventory` que gestione los paquetes de semillas, mostrando información sobre los diferentes tipos de semillas y sus cantidades.

```
>>> ft_seed_inventory("tomato", 15, "packets")
Tomato seeds: 15 packets available
>>> ft_seed_inventory("carrot", 8, "grams")
Carrot seeds: 8 grams total
>>> ft_seed_inventory("lettuce", 12, "area")
Lettuce seeds: covers 12 square meters
```

Escribe únicamente la función solicitada, no un programa principal. Esta debe gestionar directamente la entrada y salida de datos.

La declaración de la función debe ser: def ft_seed_inventory(seed_type: str, quantity: int, unit: str) ->None:

Para letras mayúsculas, puedes utilizar métodos disponibles en los objetos de string.

Admite estas unidades: "packets" (paquetes disponibles), "grams" (gramos totales), "area" (metros cuadrados abarcados). Para cualquier otra unidad, muestra "Unknown unit type". Escribe únicamente la función, no un programa principal.

Capítulo XIII

Recursos de Ayuda

Para apoyarte en tu proceso de aprendizaje, te proporcionamos un archivo `main.py` que te per
Como acabas de empezar con Python y aún no sabes cómo escribir funciones principales, esta herramienta se encargará de:

- Importar y ejecutar tus funciones automáticamente.
- Mostrar mensajes de error claros si algo falla.
- Permitirte probar ejercicios individuales o todos a la vez.
- Ayudarte a entender cómo funcionan las importaciones en Python.

Para utilizar este recurso, simplemente ejecuta `python3 main.py` en tu terminal y elige qué ejercicio probar. Asegúrate de que tus archivos (como `ft_plot_area.py`) estén en la misma carpeta que `main.py`.

El archivo de ayuda está diseñado para ser legible y educativo. Puedes revisar el código para entender cómo funciona, pero concéntrate primero en desarrollar tus propias soluciones.

Este archivo es solo un recurso de apoyo para el aprendizaje. Las soluciones que entregues deben ser trabajo propio y reflejar tu comprensión de los conceptos.

Capítulo XIV

Entrega y Evaluación

Entrega tu trabajo en tu repositorio `Git` como de costumbre. Solo se evaluará el contenido que esté dentro de tu repositorio durante la defensa. No dudes en comprobar que los nombres de tus archivos sean los correctos.

Para apoyarte en tu proceso de aprendizaje, te proporcionamos un archivo main.py que te permitirá probar tus ejercicios de forma sencilla.

Durante la evaluación, se te puede pedir que expliques tu código, sigas el flujo ejecución o modifiques tus soluciones. Asegúrate de comprender cada línea que escribes.

Debes entregar únicamente los archivos solicitados por el enunciado de este proyecto. Concéntrate en escribir un código limpio, legible y que demuestre tu comprensión de los fundamentos de Python.