Universidad Internacional de Valencia

04 GIAR – Fundamentos de Programación

Actividad 01 Portafolio. Proyecto Básico Python y C

Docente: Francisco Gómez Arnal

Alumno: Jaume Perelló Bosch - Fecha: 12 de diciembre de 2025

Este documento resume el desarrollo de dos aplicaciones prácticas diseñadas para resolver problemas mediante lógica algorítmica bajo el paradigma de programación

procedimental. El primer proyecto utiliza Python por su capacidad de manejo de archivos y estructuras de datos para generar rutinas de entrenamiento. El segundo utiliza C para gestionar memoria y lógica aritmética en una simulación de un juego de azar contra la máquina.

1. Proyecto Python: Generador de Rutinas de Entrenamiento

El objetivo de este script es automatizar la creación de planes de entrenamiento personalizados, gestionando la persistencia de datos (lectura y escritura de archivos) y la aleatoriedad controlada.

1. La Base de Datos: Creación del fichero *ejercicios.csv*

Para que el programa sea dinámico, se separaron los datos de la lógica. Se creó un archivo de texto plano con formato CSV (Comma Separated Values) llamado *ejercicios.csv*.

Estructura: Cada línea contiene dos valores: Nombre del Ejercicio, Grupo Muscular.

Codificación: Se utilizó UTF-8 para asegurar la correcta lectura de caracteres especiales (tildes, ñ) comunes en nombres como "Bíceps" o "Cuádriceps".

Función: Actúa como la base de conocimiento del programa. Si se desea añadir nuevos ejercicios, basta con editar este archivo sin tocar el código fuente.

2. Arquitectura Principal del Script

El código se estructura de forma modular mediante funciones especializadas, siguiendo el flujo: Entrada (Leer CSV) -> Procesamiento (Algoritmo) -> Salida (Archivo TXT).

A. Gestión de Archivos y Estructuras de Datos

La función *cargar_ejercicios_csv* lee el archivo y transforma las líneas de texto en un diccionario, clasificando los ejercicios por grupo muscular. Incluye control de errores (try-except) para evitar fallos si el archivo no existe.

La función *guardar_rutina_txt(nombre_rutina, contenido),* se encarga de escribir la cadena de texto generada en un archivo físico .txt. Garantiza que la carpeta de destino exista (creándola si es necesario) para evitar errores de ruta.

La función *listar_y_cargar_rutinas(), p*ermite la recuperación de información. Lee el sistema de archivos del sistema operativo *(os.listdir)* para mostrar las rutinas guardadas anteriormente y permite al usuario leer su contenido sin salir del programa.

B. Interacción y Validación

La función *pedir_entero_rango(mensaje, min_val, max_val), a*ctúa como filtro de seguridad (validación). Utiliza un bucle infinito (while True) que atrapa errores de

tipo (si el usuario escribe letras) y de lógica (si el número está fuera de rango), impidiendo que el programa se rompa por errores humanos.

C. Algoritmo de Generación de Rutinas

La función *generar_rutina* es el núcleo lógico. Recibe los días de entrenamiento y el tiempo disponible para distribuir la carga de trabajo:

Distribución Semanal: Utiliza el operador módulo (%) para repartir equitativamente los grupos musculares entre los días disponibles (3, 4 o 5 días).

Cálculo de Volumen: Estima cuántos ejercicios caben en la sesión (tiempo // 10 min).

Selección Aleatoria sin Repetición: Utiliza *random.sample* para elegir ejercicios del diccionario. Se implementó una lógica de seguridad *(min())* para evitar errores si el programa solicita más ejercicios de los que existen en el CSV.

D. Control del Flujo

La función *main(),* llamada *menu_principal()* controla el ciclo de vida de la aplicación mediante un bucle principal, inicializa la carga de datos y conecta las decisiones del usuario con las funciones lógicas y de almacenamiento.

3. Colaboración y Asistencia Técnica con IA

Para el desarrollo de estos proyectos se contó con el soporte de varios asistentes de Inteligencia Artificial, principalmente Gemini y con algún soporte de Copilot integrado en VStudio, los cuáles actuaron como consultor técnico para la resolución de bloqueos lógicos y la comprensión profunda de librerías estándar.

En el proyecto de Python, mi solicitud de ayuda se centró fundamentalmente en la implementación correcta de la persistencia de datos y la gestión del sistema de archivos, áreas críticas para la funcionalidad del script:

Módulo *csv:* Busqué ayuda sobre cómo utilizar la librería *csv* para leer bases de datos externas. La IA me explicó el flujo de apertura de archivos *(with open)*, la iteración segura por filas y cómo transformar esos datos crudos en estructuras manipulables *(diccionarios)*.

Módulo *os*: Me respaldé para trabajar con el módulo *os*, con el objetivo de hacer el programa robusto y portable. Se clarificó el uso de funciones como *os.path.join* y *os.path.exists* para gestionar rutas y crear directorios automáticamente, independientemente del sistema operativo donde se ejecute.

Por mi parte, intentando recibir la menor ayuda posible, realicé el bloque principal *def menu_principal(),* donde se concentra el código para la interacción con el usuario.

Para la implementación de los dos módulos mencionados arriba, me basé en la ayuda de Gemini. Le pasé el bloque principal *main()* que había creado como base de partida, y le indiqué con un prompt de que se trataba el proyecto y las funciones que quería implementar. En ese prompt le indique también que me crease una lista de ejercicios con los requisitos del enunciado de la actividad, con la obligatoriedad de que fuese en dos columnas, ejercicio-musculo, para así facilitar la implementación en el diccionario de Python. A partir de ahí, que me ayudase a implementar todas las funciones necesarias *os.,* explicándome paso a paso lo que iba haciendo. El prompt usado fue de esta estructura:

"Estoy trabajando en un proyecto de programación basado en el lenguaje de Python, donde se simula la automatización en la creación de planes de entrenamiento personalizados, donde se le hacen varias preguntas al usuario cómo cuántos días va a entrenar a la semana (3d-5d) y tiempo (45min-90min). A partir de ahora vas a tomar el rol de programador experto en el lenguaje de programación de Python. Necesito que me ayudes a introducir a mi proyecto, varias funciones para implementar la lectura de un archivo .csv que simula una base de datos de varios ejercicios de gym con sus respectivos grupos musculares. En el siguiente script que te dejo a continuación está la estructura de las preguntas al usuario, y de si quiere guardar o no la rutina creada... Tomate el tiempo necesario para analizar el bloque que te he dejado a continuación e implementa lo que te he pedido. Si realizas tu trabajo satisfactoriamente serás recompensado en una parte de lo que genere la venta de este proyecto."

A partir de lo generado le fui indicando que me explicase lo que iba implementando...

Con la ayuda de Copilot integrado en VStudio le pregunté por errores que me iba dando a la hora de buscar el directorio... Pero principalmente la ayuda fue con Gemini.

2. Proyecto C: Juego de Dados

Este proyecto consiste en un juego de estrategia y azar por turnos, donde el usuario compite contra una IA básica. El objetivo es reducir los 100 puntos de vida del oponente a cero.

1. Arquitectura Principal del Script

La función *main(), c*ontrola el bucle principal de la partida (mientras ambos tengan vida), gestiona los puntos de vida globales *(vida_jugador, vida_maquina)* y alterna las llamadas a los turnos de cada contendiente. También inicializa la semilla de aleatoriedad *(srand)* para el lanzamiento al azar del dado.

La función *tirar_dado(),* abstrae la generación de *(srand)*. Utiliza aritmética modular *(rand() % 6 + 1)* para ajustar el rango de salida de 0-5 del pc a 1-6, simulando un dado físico real.

La función *evento_especial_6(int puntos_actuales), s*imula probabilidades binarias (moneda) usando *rand() % 2* para decidir entre operaciones opuestas (multiplicación vs división) y calcula el factor de impacto (x2 o x3), modificando directamente el acumulado del turno.

La función *turno_jugador(int puntos_oponente) g*estiona la interactividad y el riesgo. Contiene un bucle local que permite al usuario decidir si seguir acumulando o plantarse. Maneja la condición de derrota súbita (sacar un 1) y utiliza la información del rival para avisar al jugador si ya tiene puntos suficientes para ganar (victoria matemática).

La función *turno_maquina(int puntos_oponente) e*s la Inteligencia Artificial (IA). Implementa un algoritmo de toma de decisiones basado en umbrales: sigue arriesgando hasta acumular 20 puntos (estrategia conservadora) o hasta superar la vida restante del jugador (estrategia ganadora).

La función *limpiar_buffer()* es vital para la sincronización de entrada/salida *(stdin)*. Elimina los caracteres residuales (como el 'Enter') que deja *scanf* en la memoria. Sin ella, el juego se saltaría las pausas y las confirmaciones del usuario, perdiendo el control del flujo.

2. Colaboración y Asistencia Técnica con IA

La estructura del prompt usado fue algo parecido:

“Pivotamos al lenguaje de programación C, donde se simula un juego de azar y estrategia de dados por turnos contra la máquina ("Duelo C"), donde el objetivo es reducir la vida del oponente de 100 a 0.

A partir de ahora vas a tomar el rol de programador experto en el lenguaje de programación C. Necesito que me ayudes a introducir a mi proyecto varias funciones para implementar la lógica matemática del dado especial (cuando sale un 6, se activa multiplicadores o divisores aleatorios) y la gestión segura de la entrada de datos para evitar errores al preguntar al usuario si quiere seguir tirando.

En el siguiente script que te dejo a continuación está la estructura del main y las variables de vida iniciales... Tómate el tiempo necesario para analizar el bloque que te he dejado a continuación e implementa las funciones de lógica de juego y gestión de memoria que te he pedido. Si realizas tu trabajo satisfactoriamente serás recompensado en una parte de lo que genere la venta de este proyecto."

Con Copilot, me ayudé a la hora de arreglar varios errores o mejoras que quería realizar al ir simulando el juego. Uno de los errores que me encontré fue el error residual del 'Enter' en el buffer, que a veces se lanzaba el dado aleatoriamente. Ese error me ayudó a encontrarlo la IA implementada en el IDE.