MARCO TEÓRICO – TP INTEGRADOR PROGRAMACION I

Gestión de Datos en Python

Repositorio del Proyecto

La totalidad del desarrollo se encuentra disponible en el repositorio de GitHub ubicado en la siguiente

dirección: https://github.com/Faruu12/TP_Integrador_ProgramacionI-Ultima_instancia

Contenido del Repositorio

El repositorio contiene los siguientes elementos:

Implementación Técnica

• **Código fuente**: Programa desarrollado en lenguaje Python que constituye la solución técnica del trabajo integrador

• **Documentación de ejecución**: Instrucciones detalladas para la correcta ejecución del programa

Documentación Teórica

• **Marco teórico sobre conjuntos y lógica**: Documento en formato PDF que desarrolla los

fundamentos aplicados en el proyecto

Material Audiovisual

En el archivo README del repositorio se encuentra adjunto un video explicativo en el cual nos presentamos formalmente y procedemos a:

• Explicar los aspectos conceptuales y técnicos del proyecto

• Realizar una demostración práctica del funcionamiento del programa desarrollado

Integrantes

• Axel Ponce: 45219484 – Comisión: Ag25-1C-04

Fuentes

1. **Estructuras de Datos.**

son formas de organizar y almacenar datos de manera eficiente. Estos elementos son fundamentales para construir programas robustos y optimizar el rendimiento de tus algoritmos. Tenemos varios tipos de estructuras:

1.1 Listas: Son un tipo de dato que permite almacenar datos de cualquier tipo. Son mutables y dinámicas.

Ejemplo:

paises = []

paises.append({"nombre": "argentina", "poblacion": 46234830, "superficie": 2780400, "continente": "america_del_sur"})

1.2. Diccionarios: Son una estructura de datos que permite almacenar su contenido en forma de llave y valor.

Ejemplo:

pais = {"nombre": "china", "poblacion": 1409000000, "superficie": 9596960, "continente": "asia"}

2. **Funciones:** Nos permiten realizar diferentes operaciones con la entrada, para entregar una determinada salida que dependerá del código que escribamos dentro. Por lo tanto, es totalmente análogo al clásico y=f(x) de las matemáticas. Permiten dividir el programa en partes reutilizables.

Ejemplos: cargar_paises, guardar_paises, agregar_pais, estadisticas.

3. **Condicionales**: sirven para cambiar el flujo de ejecución de un programa, haciendo que ciertos bloques de código se ejecuten si y solo si se dan unas condiciones particulares

Ejemplo:

```
if descendente:
        if a < b:
            paises[j], paises[j + 1] = paises[j + 1], paises[j]
    else:
```

if a > b:

paises[j], paises[j + 1] = paises[j + 1], paises[j]

4. **Bucles:** Sirven para alterar el flujo normal de un programa. Nos permiten repetir una porción de código tantas veces como queramos. Python incluye únicamente dos tipos de bucle: *while* y *for.*

Ejemplo:

```
for p in paises:
   print(p)
```

5. **Archivos CSV**: Ofrece una forma liviana de almacenar e intercambiar datos estructurados. Cada línea contiene una fila, y las comas dividen los campos en ella. El resultado es un formato fácil de leer y generar de forma programática

Ejemplo:

```
nombre,poblacion,superficie,continente
argentina,46234830,2780400,america_del_sur
```

6. **Manejo de errores**

Se valida que los números sean correctos y que no falten datos.

Ejemplo:

```
if not valor.isdigit():
   print("Debe ingresar un número entero.")
```

Fuentes:

- https://tutorial.recursospython.com/bucles/

- https://docs.python.org/es/3/tutorial/index.html

-https://ellibrodepython.com/

-https://www.datasunrise.com/es/centro-de-conocimiento/que-es-un-archivo-csv/

Objetivo

El objetivo de este proyecto es desarrollar un sistema en Python que permita gestionar información de países utilizando estructuras de datos básicas (listas y diccionarios), funciones, condicionales, bucles y lectura/escritura de archivos CSV. Algunas de las funciones que veremos en el archivo son:

- Implementar funciones para agregar, modificar, buscar y filtrar información por parte del usuario.
- Almacenar Datos desde un archivo externo (CSV) y ordenarlos adecuadamente
- Ordenar registros según distintos criterios que pida el usuario
- Aplicar validación y prevenir errores básicos del usuario
- Presentar un menú interactivo que permita utilizar todas las funcionalidades pedidas

Metodología Utilizada

Para el desarrollo del trabajo practico integrador se aplicaron los siguientes pasos y una construcción progresiva sobre los problemas que se iban planteando sobre la marcha.

1. Análisis del problema

Se revisaron los requisitos del trabajo integrador: carga de datos, almacenamiento permanente, búsquedas, filtros, ordenamientos y estadísticas.
A partir de esto, se definió qué datos debía manejar el sistema (nombre, población, superficie y continente) y cómo representarlos dentro del programa.

2. Separación del programa

El programa se dividió en módulos (funciones) para cumplir con el principio de separación de responsabilidades.
Se implementaron funciones específicas para:

- Cargar y guardar el archivo CSV
- Agregar, actualizar y buscar países
- Filtrar según distintos criterios
- Ordenar datos
- Calcular estadísticas
- Validar entradas del usuario
- Gestionar el menú principal

Esto mejora la legibilidad y facilita las pruebas.

3. Implementación de la persistencia de datos

Se trabajó con archivos CSV utilizando técnicas básicas:

- Lectura mediante readlines()
- Separación de valores con .split(",")
- Construcción manual del archivo en la función de guardado

Esta etapa garantiza que los datos se conserven entre ejecuciones

4. Validación y control de errores

Se incorporaron verificaciones como:

- isdigit() para asegurar que los campos numéricos sean correctos
- Normalización de texto para evitar diferencias entre mayúsculas y minúsculas
- Confirmación de existencia de países antes de actualizarlos
- Evitar campos vacíos

Esto ayuda a que el programa sea más robusto y fácil de usar.

7. Pruebas y ajustes

Cada función se probó individualmente con distintos casos:

- Búsquedas exactas y parciales
- Filtros válidos y límites incorrectos
- Rangos de población y superficie
- Ordenamientos ascendente y descendente
- Estadísticas con diferentes listas de países

Con esto se validó la correcta integración entre funciones y la persistencia.

Conclusión

La verdad es que este trabajo integrador fue una experiencia de aprendizaje con muchas subidas y bajadas. Al principio, el objetivo de construir un sistema completo resultaba un poco abrumador. La clave fue adoptar la mentalidad de fragmentar el problema en partes lógicas (carga, funciones, interfaz, etc.) hizo que el proyecto se volviera totalmente mas llevadero.

Conseguimos aplicar todos los conceptos teóricos de la materia: estructuras de datos como listas y diccionarios, validaciones esenciales y el manejo de archivos. Además, el ejercicio de ordenar y estructurar el código en funciones reforzó nuestra comprensión sobre la importancia del modularidad, aunque fue una de las cosas más difícil del proyecto.

Un punto culminante fue la implementación del archivo CSV. Nos permitió trascender la programación temporal y entender cómo un programa se integra con el almacenamiento real de datos.

En resumen, estamos seguros de que el proyecto no solo cumple con las expectativas, sino que también nos brindó una visión mucho más profunda de cómo interactúan las distintas piezas dentro de un software funcional. Es un gran paso de confianza para nuestros próximos desafíos