Informe del Trabajo Práctico Integrador

-

Programación I

Integrantes:

- **Ramiro Gonzalez**
- **Patricio Gonzalez**
- **Matias Azcurra**

Informe teórico – Conceptos aplicados

★Listas

Una **lista** en Python es una estructura de datos que permite almacenar varios elementos en una sola variable.
Las listas son ordenadas, modificables y pueden contener distintos tipos de datos.

En este trabajo, las listas se utilizaron principalmente para guardar todos los países leídos desde el archivo CSV.
Cada país válido se agrega dentro de una lista llamada paises, con la instrucción:

```
paises.append(pais)
```

También se usaron listas para guardar los mensajes de error (errores) y para devolver resultados filtrados o buscados.
Por ejemplo, al buscar por nombre, la función devuelve una nueva lista con los países que coinciden con la búsqueda.

Ejemplo en el programa:

```
paises = []       # Lista de países válidos
errores = []      # Lista de errores detectados
```

★Diccionarios

Un diccionario es una estructura que guarda información en pares de clave y valor, lo que permite acceder a los datos por nombre en lugar de por posición.

En el trabajo, cada país se representa como un diccionario con sus atributos:

```
{
  "nombre": "Argentina",
  "poblacion": 45376763,
  "superficie": 2780400,
  "continente": "América"
}
```

Los diccionarios se usan porque permiten acceder fácilmente a los datos, por ejemplo:

```
p['nombre']
p['poblacion']
```

Ventaja: son ideales para manejar registros con muchos campos (como en una base de datos o CSV).

★Funciones

Una función es un bloque de código reutilizable que realiza una tarea específica.
En este trabajo, se utilizaron funciones para organizar el código y hacerlo más legible.

Ejemplos:

- `cargar_paises_desde_csv(ruta_csv)` → lee y valida los datos.
- `buscar_por_nombre(paises, consulta)` → devuelve países que coinciden con un texto.
- `ordenar_paises(paises, clave, descendente)` → ordena los países según un campo.
- `promedio_poblacion(paises)` → calcula el promedio de población.

Beneficio: las funciones permiten dividir el programa en partes pequeñas y manejables, evitando repetir código.

★Condicionales

Las estructuras condicionales (if, elif, else) permiten ejecutar distintas acciones según se cumplan ciertas condiciones.

Ejemplo en el programa:

```
if not os.path.isfile(ruta_csv):
    errores.append("Archivo no encontrado.")
```

También se usan para validar datos:

```
if not es_entero_valido(poblacion_s):
    errores.append("Población inválida.")
```

Función del condicional: tomar decisiones y evitar errores, ejecutando diferentes caminos según los datos o las elecciones del usuario.

★Ordenamientos

El ordenamiento se utiliza para mostrar los países según un criterio, como el nombre, la población o la superficie.
Python facilita esta tarea con la función sorted().

Ejemplo:

```
paises_ordenados = sorted(paises, key=lambda x: x['poblacion'])
```

También puede ordenarse en forma descendente:

```
paises_ordenados = sorted(paises, key=lambda x: x['nombre'],
reverse=True)
```

Aplicación práctica:
El usuario puede elegir desde el menú cómo quiere ordenar los países (alfabéticamente o por cantidad de habitantes).

★Estadísticas básicas

El programa calcula diferentes estadísticas sobre los países cargados, como:

- País con mayor población (max()).
- País con menor población (min()).
- Promedio de población (sum(...) / len(...)).
- Promedio de superficie.
- Cantidad de países por continente (usando un diccionario contador).

Ejemplo:

```
mayor = max(paises, key=lambda x: x['poblacion'])
promedio = sum(p['poblacion'] for p in paises) / len(paises)
```

Objetivo: obtener información general y comparativa sobre los datos del archivo.

★Archivos CSV

Los archivos CSV (Comma Separated Values) son archivos de texto donde los datos se separan por comas.
Cada línea representa un registro y cada columna un campo.

Ejemplo del archivo paises.csv:

```
nombre,poblacion,superficie,continente
Argentina,45376763,2780400,América
Brasil,214326223,8515767,América
Japón,125360000,377975,Asia
```

El programa abre y lee este archivo con:

```
with open(ruta_csv, 'r', encoding='utf-8') as f:
    lineas = f.read().splitlines()
```

Luego separa cada línea con:

```
partes = linea.split(',')
```

Importancia: el uso de archivos CSV permite trabajar con datos reales y mantenerlos separados del código.

Conclusión general

Cada uno de estos conceptos fue aplicado en conjunto para lograr un programa completo, funcional y modular.
Las listas y diccionarios permitieron organizar los datos; las funciones dieron estructura al código; los condicionales manejaron decisiones y validaciones; y las operaciones de ordenamiento y estadísticas brindaron análisis útiles.
El manejo de archivos CSV integró todos estos conocimientos en un proyecto práctico y realista.

Anexo

bibliografía:

Listas, Diccionarios y Funciones:

- https://docs.python.org/es/3/tutorial/datastructures.html (Contiene Listas y Diccionarios)
- https://docs.python.org/es/3/tutorial/controlflow.html#definiendo-funciones (Funciones)

Condicionales (`if`, `elif`, `else`):

- https://docs.python.org/es/3/tutorial/controlflow.html#sentencia-if

Ordenamientos (`sorted()` y `key=lambda`):

- https://docs.python.org/es/3.13/howto/sorting.html

Estadísticas Básicas (`max()`, `sum()`, `len()`):

- https://docs.python.org/es/3/library/functions.html

Archivos CSV (Módulo `csv`):

- https://docs.python.org/es/3.9/library/csv.html

LINK GITHUB:

https://github.com/Pato-001/Carpeta-Digital