Justificación del Tratamiento de Datos Aplicado a la API de REST COUNTRIES

La solución implementada para interactuar con la API REST Countries sigue un enfoque estructurado y modular que garantiza la integridad, eficiencia y utilidad de los datos procesados. A continuación, se justifica cada etapa del tratamiento de datos basado en el código desarrollado y los principios de programación básica en Python:

1. Conexión y Descarga de Datos desde la API

- Justificación Técnica:
    - Se utiliza el módulo requests para enviar solicitudes HTTP a la API (GET https://restcountries.com/v3.1/all), siguiendo el estándar de comunicación REST.
    - La API devuelve datos en formato JSON, que se convierte automáticamente a estructuras nativas de Python (listas y diccionarios), facilitando su manipulación.
    - Ventaja: La API REST Countries ofrece datos estructurados y actualizados, lo que asegura consistencia y confiabilidad en la información (ej.: nombres oficiales, códigos ISO, monedas).
- Ejemplo de Código:

```
def obtener_datos_paises():
    url = "https://restcountries.com/v3.1/all"
    respuesta = requests.get(url, timeout=10)
    return respuesta.json()
```

- Conexión con los Apuntes:
    - Como se menciona en los apuntes, las APIs actúan como "meseros" que conectan clientes con servidores, permitiendo automatizar tareas y acceder a datos globales (ej.: clima, mapas).

2. Estructuración de Datos

- Justificación Técnica:
    - Los datos crudos de la API se transforman en una lista de diccionarios con campos clave (nombre, población, región, idiomas, monedas), simplificando su análisis posterior.

- Ventaja:
    - Reducción de Complejidad: Se eliminan campos irrelevantes y se normalizan valores (ej.: densidad poblacional calculada).
    - Legibilidad: Los campos descriptivos (ej.: "Región", "Área (km²)") facilitan la comprensión del usuario.
- Ejemplo de Código:

```
def estructurar_datos_paises(datos):
    paises_estructurados = []
    for pais in datos:
        paises_estructurados.append({
            "Nombre": pais["name"]["common"],
            "Población": pais.get("population", 0),
            "Área (km²)": pais.get("area", 0),
            "Densidad (hab/km²)": calcular_densidad(poblacion, area),
            "Idiomas": ", ".join(pais.get("languages", {}).values()) or "N/A"
        })
    return paises_estructurados
```

- Conexión con los Apuntes:
    - La Modularización (separar funciones en modulo.py y lógica principal en script.py) refleja el principio de "divide y vencerás" mencionado en los apuntes, reduciendo la complejidad del código.

3. Manejo de Archivos y Limpieza con Expresiones Regulares

- Justificación Técnica:
    - Los datos estructurados se guardan en JSON y Excel para persistencia y análisis posterior.
    - Las expresiones regulares (re.compile(), re.search()) permiten filtrar países por patrones en sus nombres (ej.: "^A" para países que comienzan con "A").
    - Ventaja:
        - Flexibilidad: Permite búsquedas personalizadas sin modificar la API.

        - Eficiencia: Reduce el volumen de datos procesados al enfocarse en subconjuntos específicos.
    - Ejemplo de Código:

```
def filtrar_paises_con_regex(datos, patron_regex):
    regex = re.compile(patron_regex, re.IGNORECASE)
    return [pais for pais in datos if regex.search(pais["Nombre"])]
```

    - Conexión con los Apuntes:
        - El uso de expresiones regulares refleja la aplicación práctica de herramientas avanzadas para manipular texto, como validar nombres de usuario o extraer información estructurada.

4. Análisis Estadístico Básico

    - Justificación Técnica:
        - Se calculan estadísticas descriptivas (media, mediana, varianza) para campos numéricos (ej.: población, área) usando el módulo statistics.
        - Ventaja:
            - Toma de Decisiones: Estas métricas ayudan a identificar patrones (ej.: dispersión de población entre países).
            - Interpretación Contextual: Se relacionan los resultados con datos geográficos reales (ej.: país con mayor/menor población).
    - Ejemplo de Código:

```
def analizar_estadisticas(datos, campo="Población"):
    valores = [pais[campo] for pais in datos if pais.get(campo, 0) > 0]
    return {
        "Media": round(statistics.mean(valores), 2),
        "Mediana": statistics.median(valores),
        "Moda": statistics.mode(valores),
        "Varianza": round(statistics.variance(valores), 2),
        "Desviación Estándar": round(statistics.stdev(valores), 2)
    }
```

    - Conexión con los Apuntes:

- El análisis estadístico cumple con el objetivo de transformar datos en información útil, como se menciona en la Fase III de los apuntes ("manipulación de hojas de cálculo y generación de gráficos").

5. Visualización Gráfica

- Justificación Técnica:
  - Se generan gráficos de barras y líneas con matplotlib para representar datos como la población de los 10 países más poblados.
  - Ventaja:
    - Comunicación Efectiva: Los gráficos facilitan la comprensión de tendencias y comparaciones.
    - Aplicación Práctica: Útil para informes, presentaciones o análisis geográficos.
- Ejemplo de Código:

```
def graficar_datos(datos, tipo_grafico="barras", titulo="Gráfico", eje_x="X", eje_y="Y",
campo_x="Nombre", campo_y="Población"):
    plt.figure(figsize=(12, 6))
    if tipo_grafico == "barras":
        plt.bar(valores_x, valores_y)
    elif tipo_grafico == "lineas":
        plt.plot(valores_x, valores_y, marker='o')
    plt.title(titulo)
    plt.savefig(f"{titulo.lower().replace(' ', '_')}.png")
    plt.show()
```

- Conexión con los Apuntes:
  - La visualización gráfica refleja el enfoque práctico de los apuntes, donde se destaca la importancia de generar "gráficos que comuniquen información".

6. Interpretación de Resultados

- Justificación Técnica:
  - Se interpreta el significado de las estadísticas y gráficos en contexto geográfico y demográfico.
  - Ventaja:

        - Acciones Concretas: Permite tomar decisiones informadas (ej.: planificación urbana, políticas sociales).
        - Transparencia: Relaciona números abstractos con realidades tangibles (ej.: "China tiene la mayor población, lo que explica su alta media global").
- Ejemplo de Interpretación:
    - Media vs. Mediana:
        - Si la media es mucho mayor que la mediana, indica alta asimetría (ej.: pocos países con población extrema influyen en el promedio).
    - Coeficiente de Variación:
        - Un valor > 1 implica alta dispersión entre los países.
- Conexión con los Apuntes:
    - La interpretación refleja el objetivo final del curso: "construir soluciones inteligentes" que vayan más allá del procesamiento de datos.

Conclusión

El tratamiento de datos aplicado a la API REST Countries cumple con los principios de modularidad, eficiencia y utilidad descritos en los apuntes de programación básica en Python:

1. Automatización: Acceso a datos globales mediante APIs, reduciendo trabajo manual.
2. Análisis Estructurado: Uso de herramientas estándar (pandas, matplotlib, statistics) para transformar datos en conocimiento.
3. Aplicabilidad Práctica: Resultados interpretables que respaldan decisiones en áreas como geografía, economía o política.

Este enfoque no solo cumple con los objetivos técnicos del proyecto, sino que también refuerza las habilidades adquiridas en cada fase del curso, desde conceptos básicos hasta aplicaciones avanzadas.