🤖 Automating Real-World Tasks with Python – Cheatsheet

Módulo 4: Putting It All Together

🎯 Introducción

En este módulo se integra todo lo aprendido anteriormente:

- Manipulación de imágenes
- Interacción con web services
- Generación automática de reportes

El objetivo es **construir un proyecto completo de automatización**, que combine múltiples fuentes de datos y formatos de salida.

🧩 Flujo típico de un proyecto completo

1. **Obtener datos**
   - Desde APIs, CSVs o bases de datos.
2. **Procesar datos**
   - Limpiar, filtrar, transformar.
3. **Generar contenido visual**
   - Redimensionar o anotar imágenes.
   - Crear gráficos.
4. **Generar reportes automáticos**
   - CSV, Excel, PDF o Word.
5. **Automatización completa**
   - Scripts que ejecuten todo el pipeline sin intervención manual.

🧰 Herramientas integradas

Componente	Librería	Uso
Web / APIs	`requests` / `json`	Obtener y procesar datos de internet
Imágenes	`Pillow`	Redimensionar, recortar, convertir
Reportes	`pandas`, `openpyxl`, `csv`, `reportlab`, `python-docx`	Generar archivos de salida
Automatización	`os`, `pathlib`, `shutil`	Gestión de archivos y carpetas

🔷 Ejemplo conceptual

Automatización de un pipeline completo:

```
import requests
import pandas as pd
from PIL import Image
from reportlab.pdfgen import canvas
from pathlib import Path

# 1 Obtener datos
url = "https://api.ejemplo.com/products"
resp = requests.get(url)
productos = resp.json()

# 2 Procesar datos y generar CSV
df = pd.DataFrame(productos)
df.to_csv("productos.csv", index=False)

# 3 Procesar imágenes de productos
imagenes_folder = Path("imagenes")
for archivo in imagenes_folder.glob("*.jpg"):
    img = Image.open(archivo)
    img = img.resize((128,128))
    img.save(imagenes_folder / f"procesadas/{archivo.name}")

# 4 Generar PDF de resumen
c = canvas.Canvas("reporte_final.pdf")
c.drawString(100, 750, "Reporte de Productos")
y = 730
for index, row in df.iterrows():
    c.drawString(100, y, f"{row['nombre']} - ${row['precio']}")
    y -= 20
c.save()
```

> Con un solo script se obtiene: CSV, imágenes procesadas y PDF listo para distribución.

Buenas prácticas

- Separar **funciones por tarea** (API, imagen, reporte) para mantener código limpio.
- Manejar errores de forma **robusta** en cada etapa.
- Usar **logging** para saber qué pasos se ejecutaron.
- Crear **carpetas de salida** automáticamente para no sobreescribir archivos.
- Integrar todo en **un solo pipeline** con posibilidad de agendarlo (`cron` o Windows Task Scheduler).

Referencias

- Requests Library
- Pillow Documentation
- pandas Documentation

- ReportLab User Guide
- python-docx Documentation