🌐 Python Cheatsheet – Módulo `requests`

> **Tema:** Uso del módulo `requests` para realizar solicitudes HTTP en Python **Curso:** Google IT Automation with Python **Objetivo:** Entender cómo enviar peticiones web, manejar respuestas, errores, autenticación y trabajar con APIs.

🚀 Introducción

El módulo `requests` permite interactuar con servicios web de forma fácil y pythonic. Es ideal para consumir **APIs REST**, enviar datos, descargar archivos o automatizar tareas de red.

Instalación (si no está instalado):

```
pip install requests
```

Importación:

```
import requests
```

🛰️ Solicitudes HTTP Básicas

▶️ GET — Obtener información

```
response = requests.get("https://api.example.com/data")
print(response.status_code)
print(response.text)
```

▶️ POST — Enviar datos

```
payload = {"username": "axel", "password": "1234"}
response = requests.post("https://api.example.com/login", data=payload)
print(response.json())
```

▶️ PUT — Actualizar recursos

```
requests.put("https://api.example.com/item/1", data={"name": "Nuevo nombre"})
```

▶ DELETE — Borrar recursos

```
requests.delete("https://api.example.com/item/1")
```

Cuerpo de la Respuesta (`response`)

◈ Código de estado

```
response.status_code
```

◈ Texto de la respuesta

```
response.text
```

◈ JSON (muy usado en APIs)

```
data = response.json()
```

◈ Encabezados

```
response.headers
```

Enviar Parámetros

◈ En la URL (query parameters)

```
params = {"page": 2, "limit": 20}
response = requests.get("https://api.example.com/items", params=params)
```

◈ Enviar datos JSON (recomendado para APIs REST)

```
response = requests.post(
    "https://api.example.com/create",
    json={"nombre": "Axel", "rol": "Admin"}
)
```

Autenticación

Basic Auth

```
from requests.auth import HTTPBasicAuth

response = requests.get(
    "https://api.example.com/secure",
    auth=HTTPBasicAuth("usuario", "clave")
)
```

Bearer Token (común en APIs modernas)

```
headers = {"Authorization": "Bearer TU_TOKEN"}
response = requests.get("https://api.example.com/me", headers=headers)
```

Subir y Descargar Archivos

Subir archivo

```
files = {"file": open("documento.pdf", "rb")}
response = requests.post("https://api.example.com/upload", files=files)
```

Descargar archivo

```
response = requests.get("https://example.com/image.jpg")

with open("image.jpg", "wb") as f:
    f.write(response.content)
```

Manejo de Errores

`raise_for_status()` – Lanza excepción si hay error

```
try:
    response = requests.get("https://api.example.com/data")
    response.raise_for_status()
except requests.exceptions.RequestException as e:
    print("Error:", e)
```

Timeout

```
requests.get("https://api.example.com", timeout=5)
```

Headers Personalizados

```
headers = {"User-Agent": "AxelBot/1.0"}
requests.get("https://api.example.com", headers=headers)
```

Sesiones (Mantener cookies/headers entre requests)

```
session = requests.Session()
session.headers.update({"User-Agent": "AxelBot"})

r = session.get("https://api.example.com")
```

Ejemplo Completo – Consumir API REST

```
import requests

url = "https://jsonplaceholder.typicode.com/posts"

payload = {
    "title": "Hola Mundo",
    "body": "Probando API con Python",
    "userId": 1
}

response = requests.post(url, json=payload)

if response.status_code == 201:
    print("🔥 Creado correctamente")
    print(response.json())
else:
    print("Error:", response.status_code)
```

Excepciones comunes de `requests`

Excepción	Causa
`ConnectionError`	No se pudo conectar al servidor
`Timeout`	Se excedió el tiempo límite
`HTTPError`	Código HTTP indica error
`TooManyRedirects`	Demasiadas redirecciones
`RequestException`	Excepción base (atrapa todas)