



## Ejercicio Práctico: Escaneo Básico de Puertos con Python

### Descripción

En este ejercicio desarrollarás un script básico en Python que permita escanear los puertos abiertos de un host determinado. Esta herramienta será una versión simplificada de lo que hacen herramientas como Nmap, y te ayudará a comprender cómo funciona la comunicación de red a nivel de puertos y cómo Python puede ser utilizado para realizar tareas de reconocimiento (reconocimiento pasivo).

---

### Objetivo del ejercicio

Crear un script funcional que tome como entrada una dirección IP o nombre de dominio, y realice un escaneo básico sobre un rango definido de puertos (por ejemplo, del 20 al 1024), informando cuáles están abiertos.

---

### Requisitos técnicos

- Python 3.x instalado.
  - Librería estándar `socket` (incluida en Python).
  - Acceso a una terminal o entorno de desarrollo como Visual Studio Code, Jupyter o Spyder.
- 

### Instrucciones

Crea un archivo llamado `port_scanner.py`.

Dentro del archivo, importa el módulo necesario:

```
import socket
```

Define una función llamada `scan_ports()` que reciba dos parámetros: el `host` y el `rango` de puertos a escanear.

Utiliza un bucle `for` para recorrer los puertos desde el 20 al 1024 (puedes usar `range(20, 1025)`).

Dentro del bucle:

- Crea un socket usando `socket.socket()`.
- Usa `connect_ex()` para intentar conectarte al puerto actual del host.
- Si el resultado es `0`, imprime que el puerto está abierto.
- Asegúrate de cerrar cada socket después de usarlo para evitar errores.

Pide al usuario que ingrese un dominio o dirección IP usando `input()`.

Llama a la función con el valor ingresado.

---

### Ejemplo de salida esperada

Si el usuario ingresa `scanme.nmap.org`, la salida podría ser:

Escaneando puertos del host: scanme.nmap.org

Puerto 22: ABIERTO

Puerto 80: ABIERTO

Puerto 443: ABIERTO

...

Escaneo finalizado.

---

### Sugerencias

- Usa **try-except** para capturar errores si el host no responde o si hay problemas de red.
  - Agrega un mensaje al inicio y al final del escaneo para mayor claridad.
  - Si tienes tiempo extra, podrías añadir detección de banners o una interfaz simple de menú con **print()**.
- 

### ✅ Solución: **port\_scanner.py**

```
import socket
```

```
def scan_ports(host, start_port=20, end_port=1024):  
    print(f"\nEscaneando puertos del host: {host}\n")
```

```
    try:
```

```
        for port in range(start_port, end_port + 1):  
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
            s.settimeout(0.5) # Tiempo de espera reducido para acelerar el escaneo  
            result = s.connect_ex((host, port))  
            if result == 0:  
                print(f"Puerto {port}: ABIERTO")  
            s.close()
```

```
    except socket.gaierror:
```

```
        print("❌ Error: Host no encontrado.")
```

```
    except socket.error:
```

```
        print("❌ Error: No se pudo conectar al servidor.")
```

```
  
    print("\n✅ Escaneo finalizado.")
```

```
# Programa principal
```

```
if __name__ == "__main__":
```

```
    print("🔍 Escáner de Puertos Básico con Python")
```

```
    target = input("👉 Ingresa el host o dirección IP a escanear: ")
```

```
    scan_ports(target)
```

---


### 📌 ¿Qué hace este script?

1. Solicita al usuario ingresar un dominio o IP (como **scanme.nmap.org** o **127.0.0.1**).
2. Escanea los puertos del **20 al 1024**.

3. Informa en pantalla los puertos que están abiertos.
  4. Usa `socket.connect_ex()` para intentar conexiones TCP.
  5. Utiliza `settimeout(0.5)` para que no se quede pegado si un puerto no responde.
- 

### Ejemplo de ejecución

 Escáner de Puertos Básico con Python


 Ingresa el host o dirección IP a escanear: `scanme.nmap.org`

Escaneando puertos del host: `scanme.nmap.org`

Puerto 22: ABIERTO

Puerto 80: ABIERTO

Puerto 443: ABIERTO

 Escaneo finalizado.