```
pip install requests beautifulsoup4
# ------------------------------------------------
import requests
from bs4 import BeautifulSoup
import time
import smtplib # Módulo estándar para enviar correos, solo para referencia

# --- CONFIGURACIÓN ---
URL_PRODUCTO = "URL_DEL_PRODUCTO_AQUI" # Reemplaza con la URL real
PRECIO_OBJETIVO = 80.00  # Reemplaza con el precio máximo que quieres pagar
PRECIO_ANTERIOR = 99999.00 # Inicializa con un precio muy alto
CSS_SELECTOR_PRECIO = "span.price-tag-fraction" # EJEMPLO: Ajusta esto al
selector real del sitio web
# --------------------

def obtener_precio_actual(url, selector):
    """
    Realiza una solicitud HTTP, analiza el HTML y extrae el precio del producto.
    """
    print(f"[{time.strftime('%H:%M:%S')}] 🔍 Buscando precio en {url}...")

    # 1. Realizar la solicitud HTTP
    try:
        headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}
        respuesta = requests.get(url, headers=headers, timeout=15)
        respuesta.raise_for_status() # Lanza error si hay un 4xx/5xx

    except requests.exceptions.RequestException as e:
        print(f"❌ Error al acceder a la URL: {e}")
        return None

    # 2. Analizar el HTML
    soup = BeautifulSoup(respuesta.text, 'html.parser')

    # 3. Encontrar el elemento que contiene el precio (Ajusta esto según el
sitio)
    precio_element = soup.select_one(selector)

    if not precio_element:
        print(f"❌ No se encontró el elemento con el selector: {selector}")
        return None

    # 4. Extraer el texto y limpiarlo
    precio_texto = precio_element.text.strip()

    # Intenta limpiar el texto para obtener solo números (quitar $, €, puntos,
comas, etc.)
    # Esto es el paso más difícil y debe adaptarse a cada sitio.
    try:
        # Quitamos caracteres no numéricos y reemplazamos la coma por punto para
el float
        precio_limpio = "".join(filter(str.isdigit or str in ('.', ','),
precio_texto))
```

```
        precio_limpio = precio_limpio.replace(',', '.')

        # Intentamos convertir a float (p. ej., si el precio es "100.99")
        precio_float = float(precio_limpio)
        return precio_float

    except ValueError:
        print(f"⚠ No se pudo convertir el texto '{precio_texto}' a número.
Verifique el selector.")
        return None

def enviar_alerta(precio_actual, url):
    """
    Función de notificación (simulación por terminal).
    Aquí iría la lógica para enviar email, mensaje de Telegram, etc.
    """
    print("\n============================================================")
    print("🔔 ¡ALERTA DE PRECIO! ¡PRECIO ENCONTRADO O DISMINUIDO! 🔔")
    print(f"  Nuevo Precio: **{precio_actual:.2f}**")
    print(f"  URL del producto: {url}")
    print(f"  Precio Objetivo: {PRECIO_OBJETIVO:.2f}")
    print("============================================================")

    # Ejemplo de cómo sería un envío de email (requiere configuración)
    # Aquí puedes usar el módulo 'email' y 'smtplib'
    # subject = f"ALERTA: Producto bajó a ${precio_actual}"
    # send_email(subject, body)

def rastrear_precio():
    """Bucle principal de rastreo."""
    global PRECIO_ANTERIOR # Para modificar la variable global

    while True:
        precio_actual = obtener_precio_actual(URL_PRODUCTO, CSS_SELECTOR_PRECIO)

        if precio_actual is not None:

            print(f"Precio actual encontrado: {precio_actual:.2f} | Precio
anterior: {PRECIO_ANTERIOR:.2f}")

            # Condición de alerta 1: El precio bajó
            if precio_actual < PRECIO_ANTERIOR:
                enviar_alerta(precio_actual, URL_PRODUCTO)

            # Condición de alerta 2: El precio alcanzó o superó nuestro objetivo
(compra)
            if precio_actual <= PRECIO_OBJETIVO:
                print("🎯 ¡EL PRECIO HA ALCANZADO EL OBJETIVO DE COMPRA!")
                # Aquí podrías romper el bucle si quieres detener el rastreo
                # break

            PRECIO_ANTERIOR = precio_actual # Actualizamos el precio anterior

        else:
```

```
            print("Saltando verificación, el precio no pudo ser extraído.")

        # Esperar un tiempo antes de la siguiente verificación (ej: 1 hora =
3600 segundos)
        tiempo_espera = 60 * 60 # Esperar 1 hora
        print(f"Esperando {tiempo_espera // 60} minutos para la próxima
verificación...\n")
        time.sleep(tiempo_espera)


if __name__ == "__main__":

    if URL_PRODUCTO == "URL_DEL_PRODUCTO_AQUI":
        print("🚨 **ADVERTENCIA**: Por favor, edita el archivo
`rastreador_precios.py` y reemplaza `URL_DEL_PRODUCTO_AQUI` y
`CSS_SELECTOR_PRECIO` con valores reales para que el script funcione.")
    else:
        try:
            rastrear_precio()
        except KeyboardInterrupt:
            print("\n🛑 Rastreo detenido por el usuario.")
```