

```
import hashlib
```

```
def simular_sha256(texto):
```

```
    # Convertimos el texto a bytes y aplicamos el algoritmo
```

```
    resultado = hashlib.sha256(texto.encode())
```

```
    # Retornamos el hash en formato hexadecimal (64 caracteres)
```

```
    return resultado.hexdigest()
```

```
def minar_bloque(datos_transaccion, dificultad):
```

```
    """
```

```
    Intenta encontrar un hash que comience con 'n' ceros (dificultad).
```

```
    """
```

```
    nonce = 0
```

```
    objetivo = "0" * dificultad
```

```
    print(f"--- Iniciando minería con dificultad: {dificultad} ---")
```

```
    while True:
```

```
        # Combinamos los datos con el nonce actual
```

```
        contenido = f"{datos_transaccion}{nonce}"
```

```
        hash_intentado = simular_sha256(contenido)
```

```
        # Mostramos los primeros intentos para ver el cambio
```

```
        if nonce < 3:
```

```
            print(f"Nonce: {nonce} -> Hash: {hash_intentado}")
```

```
        elif nonce == 3:
```

```
            print("ciclo terminado.....")
```

```
        # Si el hash empieza con los ceros requeridos, ¡ganamos!
```

```
if hash_intentado.startswith(objetivo):  
    print(f"\n¡BLOQUE MINADO!")  
    print(f"Nonce final: {nonce}")  
    print(f"Hash válido: {hash_intentado}")  
    return hash_intentado
```

```
nonce += 1
```

```
# --- PRUEBA DEL SIMULADOR ---
```

```
mensaje = "Soy el alumno Máximo Diego Gamon Simental del grupo 5 B"
```

```
# 1. Ver un hash simple
```

```
print(f"Hash inicial: '{mensaje}':\n{simular_sha256(mensaje)}\n")
```

```
# 2. Simular el proceso de minería (buscar 4 ceros al principio)
```

```
minar_bloque(mensaje, dificultad=6)
```