Ejemplo en Python aplicando Firma Digital y Verificación

```
# ---------------------------
# Firma digital y verificación con RSA-PSS + SHA-256
# ---------------------------

# Importamos primitivas criptográficas necesarias:
from cryptography.hazmat.primitives import hashes          # Proporciona funciones hash (SHA-256, etc.)
from cryptography.hazmat.primitives.asymmetric import padding, rsa  # RSA y padding PSS para firmas
from cryptography.exceptions import InvalidSignature        # Excepción cuando la verificación falla
import base64                                               # Para mostrar/transportar la firma en Base64
(legible)

# 1) GENERAR PAR DE CLAVES RSA
#    - generate_private_key(public_exponent, key_size)
#    - public_exponent=65537 es estándar (seguro y eficiente)
#    - key_size=2048 bits es un mínimo razonable hoy en día
priv = rsa.generate_private_key(public_exponent=65537, key_size=2048)  # Clave PRIVADA (mantener secreta)
pub  = priv.public_key()                                               # Clave PÚBLICA (se puede compartir)

# 2) MENSAJE A FIRMAR
#    - Debe estar en bytes (b"...")
mensaje = b'Orden 123: Pagar $10,000 a Proveedor'
```

```
# 3) CREAR LA FIRMA DIGITAL
#    - priv.sign(data, padding, algorithm)
#    - Usamos PSS (Probabilistic Signature Scheme) con MGF1(SHA-256) y salt de 32 bytes
#    - Hash principal: SHA-256
firma = priv.sign(
    mensaje,
    padding.PSS(
        mgf=padding.MGF1(hashes.SHA256()),  # MGF1 con SHA-256 para generar máscara
        salt_length=32                      # Longitud del "salt" en PSS (aleatoriedad por firma)
    ),
    hashes.SHA256()                         # Hash del mensaje para la firma
)

# 4) MOSTRAR LA FIRMA EN BASE64 (opcional, para ver/transportar como texto)
firma_b64 = base64.b64encode(firma).decode('utf-8')  # Conversión bytes -> Base64 -> str
print("Mensaje claro:", mensaje.decode('utf-8'))
print("Firma (Base64):", firma_b64)

# 5) VERIFICAR LA FIRMA
#    - pub.verify(signature, data, padding, algorithm)
```

```
#    - Si la firma es válida, no ocurre excepción
#    - Si es inválida, lanza InvalidSignature
try:
    pub.verify(
        firma,                  # La firma generada con la privada
        mensaje,                # El MISMO mensaje original
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),  # Debe coincidir exactamente con lo usado al firmar
            salt_length=32
        ),
        hashes.SHA256()         # Mismo hash usado en la firma
    )
    print("✓ Verificación EXITOSA: la firma corresponde al mensaje y a la clave pública.")
except InvalidSignature:
    print("✗ Verificación FALLIDA: la firma NO corresponde (mensaje alterado o clave incorrecta).")

# ---------------------------
# (Opcional) Prueba de fallo:
#    - Si descomentas lo siguiente, se modifica el mensaje y la verificación debe fallar.
# ---------------------------
# try:
```

```
#     mensaje_tocado = b'Orden 123: Pagar $10,001 a Proveedor'  # Cambiamos un caracter/monto
#     pub.verify(
#         firma, mensaje_tocado,
#         padding.PSS(mgf=padding.MGF1(hashes.SHA256()), salt_length=32),
#         hashes.SHA256()
#     )
#     print("(!) Sorprendente: verifico, pero no deberia. (Esto NO debería pasar)")
# except InvalidSignature:
#     print("Como era de esperar: al cambiar el mensaje, la verificación FALLA.")
```