```
#Installs & Imports

## Installs


## Imports

import numpy as np
from scipy.io import wavfile

# Functions

def GetEnergy(sample):
  return int(np.sqrt(np.var(sample)))

def printInfo(FS, H):
  length = H.shape[0] / FS
  print(f"number of channels = {H.shape[1]}") if len(H.shape) == 2 else print("number of channels = 1")
  print(f"frequency = {FS} Hz")
  print(f"duration = {length}s")

def getLeftChannel(SIGNAL):
  return SIGNAL[:,0]

# Processing

n_of_chars = 38

FS_H, H = wavfile.read('./Church_Schellingwoude.wav')

print("\n\nInfo of H\n" + n_of_chars * "-")

printInfo(FS_H, H)

H_l = getLeftChannel(H)
```

```
H_L_energy = GetEnergy(H_l)
print(f'Energy of H_L: {H_L_energy}')

wavfile.write('./impulse_response_mono.wav', FS_H, H_l)

FS_X, X = wavfile.read("./ave_maria.wav")

print(n_of_chars * "_" + "\n\nInfo of X\n" + n_of_chars * "-")
printInfo(FS_X, X)

X_energy = GetEnergy(X)
print(f'Energy of X: {X_energy}')

input = (H_L_energy/X_energy) * X

print(n_of_chars * "_" + "\n\nConvolution Started\n" + "...")

output = np.convolve(input, H_l, mode='full')
print('Convolution Finished\n')

output_energy = GetEnergy(output)
print(f'Energy of Output: {output_energy}')

balanced_output = ( H_L_energy / output_energy ) * output

normalized_output = (balanced_output / np.max(np.abs(balanced_output)) * 32767).astype(np.int16)

wavfile.write('./output.wav', FS_H, normalized_output)
```