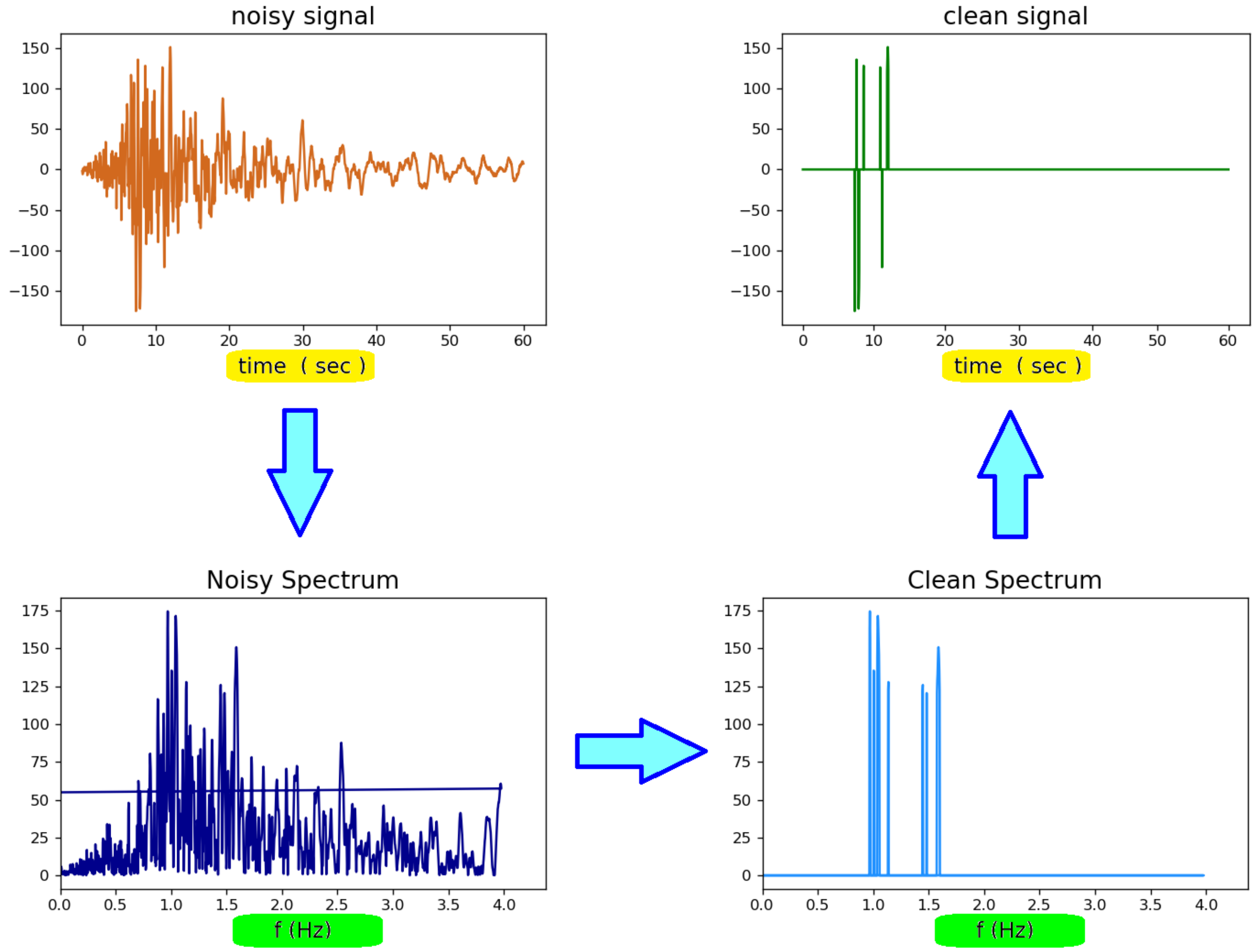
A Python Code To

- Perform FFT on Signals



In [1]:

```python
# (auto)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.fftpack import fft, fftfreq, ifft
```

Reading the Noisy Signal

In [2]:

```python
# (input) signal & dt

noisy_signal = pd.read_csv(
    'https://raw.githubusercontent.com/AmirPeimon/repo1/main/Hollister_2.csv',header=0)
dt = 0.02   # (second)
```

Generating the Time Domain, t (seconds)

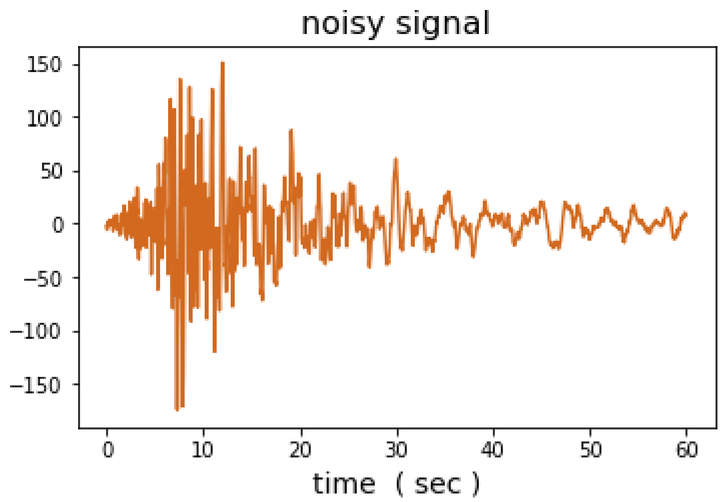
In [3]:

```python
# (auto) Time Domain
t0 = 0
t1 = len( noisy_signal )*dt
t  = np.arange(t0,t1,dt)
n  = len(t)
```

Visualizing Noisy Signal in its Time Domain

In [15]:

```python
# (auto) Plot  noisy_signal : time
fig, ax = plt.subplots( figsize=(5.7,3.5) )
ax.plot( t, noisy_signal, color='chocolate', linewidth=1.5 )
plt.title('noisy signal',fontsize=16)
plt.xlabel("time  ( sec )", fontsize=14)
plt.savefig('Noisy_Signal.png', dpi=120)
plt.show()
```



Performing FFT:

- Generaying Frequency Domain, f (Hz)
- Generating Noisy Spectrum

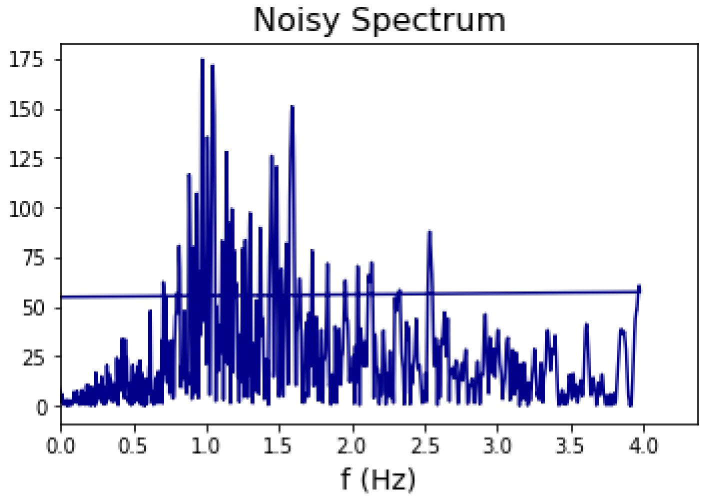
In [5]:

```
# (auto) noisy Spectrum
f_Hz  = fftfreq(n,dt) /(2*np.pi)
noisy_spec     = fft( noisy_signal )    # ( r + i )
noisy_spec_val = np.abs( noisy_spec )   # ( r^2 + i^2 )^.5
```

Visualizing Noisy Spectrum in its Frequency Domain

In [14]:

```
# (auto) plot  noisy_spec : f_Hz
fig,ax = plt.subplots( figsize=(5.7,3.5) )
ax.plot(  f_Hz, noisy_spec_val, color='darkblue', linewidth=1.5 )
plt.title("Noisy Spectrum",fontsize=16)
plt.xlabel('f (Hz)',fontsize=14)
plt.xlim(0,)
plt.savefig('Noisy_Spectrum.png', dpi=120)
plt.show()
```



Determining Threshhold, thr:

- everything less will be treated as noise

In [7]:

```
# (input) threshhold
thr = 120
```

Generating Clean Spectrum

- (By Removing the Noise from Noisy Spectrum)

In [8]:

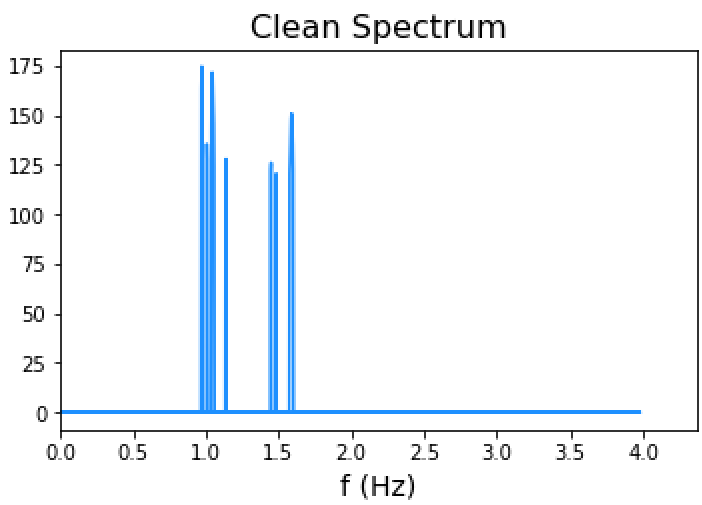
```
# (auto) removing the noise
indices        = noisy_spec_val > thr   # filters out those value under 50
clean_spec     = indices * noisy_spec   # Noise Frequency will be set to 0
clean_spec_val = np.abs( clean_spec )
```

Visualizing Clean Spectrum in its Frequency Domain

- main frequencies get revealed

In [13]:

```
# (auto) plot  clean_spec : f_Hz
fig,ax = plt.subplots( figsize=(5.7,3.5) )
ax.plot(  f_Hz, clean_spec_val, color='dodgerblue', linewidth=1.5 )
plt.title("Clean Spectrum",fontsize=16)
plt.xlabel('f (Hz)',fontsize=14)
plt.xlim(0,)
plt.savefig('Clean_Spectrum.png', dpi=120)
plt.show()
```



Generating Clean Signal

In [10]:

```
# (auto) inverse back to time domain data
clean_signal = ifft( clean_spec )
```

Visualizing Clean Signal in its Time Domain

In [12]:

```
# (auto) Plot  clean_signal : time
fig, ax = plt.subplots( figsize=(5.5,3.5) )
ax.plot( t, np.real(clean_signal), color='green', linewidth=1.5 )
plt.title('clean signal',fontsize=16)
plt.xlabel("time  ( sec )", fontsize=14)
plt.savefig('Clean_Signal.png', dpi=120)
plt.show()
```