```
"""
Registration : xxxx
Description  : Square Wave, Sawtooth Wave, Triangular Wave, Gaussian Pulse Fourier Series
Author       : AKB
"""

import numpy as np
from scipy.signal import square, sawtooth, gausspulse
from scipy.integrate import simps, quad
import matplotlib.pyplot as plt

L         = 1   # periodicity
harmonics = 20; # total number of term in series
x = np.linspace(-L,L,100) # x-grid

#### Square Wave  ####
# Compute Fourier coefficients using simps
a0 = 1.0/L*simps(square(x),x)
an = lambda n: 1.0/L*simps(square(x)*np.cos(n*np.pi*x/L),x)
bn = lambda n: 1.0/L*simps(square(x)*np.sin(n*np.pi*x/L),x)

# Compute Fourier coefficients using quad
a0 = 1.0/L*quad(lambda x: square(x),-L,L)[0]
an = lambda n: 1.0/L*quad(lambda x: square(x)*np.cos(n*np.pi*x/L),-L,L)[0]
bn = lambda n: 1.0/L*quad(lambda x: square(x)*np.sin(n*np.pi*x/L),-L,L)[0]

# Compute Fourier series
summsq = a0/2 + sum([an(k)*np.cos(k*np.pi*x/L) + bn(k)*np.sin(k*np.pi*x/L) for k
                  in range(1,harmonics)])

#### Sawtooth Wave  ####
# Compute Fourier coefficients using simps
a0 = 1.0/L*simps(sawtooth(x),x)
an = lambda n: 1.0/L*simps(sawtooth(x)*np.cos(n*np.pi*x/L),x)
bn = lambda n: 1.0/L*simps(sawtooth(x)*np.sin(n*np.pi*x/L),x)

# Compute Fourier coefficients using quad
a0 = 1.0/L*quad(lambda x: sawtooth(x),-L,L)[0]
an = lambda n: 1.0/L*quad(lambda x: sawtooth(x)*np.cos(n*np.pi*x/L),-L,L)[0]
bn = lambda n: 1.0/L*quad(lambda x: sawtooth(x)*np.sin(n*np.pi*x/L),-L,L)[0]

# Compute Fourier series
summst = a0/2 + sum([an(k)*np.cos(k*np.pi*x/L) + bn(k)*np.sin(k*np.pi*x/L) for k
                  in range(1,harmonics)])

#### Triangular Wave  ####
# Compute Fourier coefficients using simps
a0 = 1.0/L*simps(sawtooth(x,width=0.5),x)
an = lambda n: 1.0/L*simps(sawtooth(x,width=0.5)*np.cos(n*np.pi*x/L),x)
bn = lambda n: 1.0/L*simps(sawtooth(x,width=0.5)*np.sin(n*np.pi*x/L),x)

# Compute Fourier coefficients using quad
a0 = 1.0/L*quad(lambda x: sawtooth(x,width=0.5),-L,L)[0]
an = lambda n: 1.0/L*quad(lambda x: sawtooth(x,width=0.5)*np.cos(n*np.pi*x/L),-L,L)[0]
bn = lambda n: 1.0/L*quad(lambda x: sawtooth(x,width=0.5)*np.sin(n*np.pi*x/L),-L,L)[0]

# Compute Fourier series
summtr = a0/2 + sum([an(k)*np.cos(k*np.pi*x/L) + bn(k)*np.sin(k*np.pi*x/L) for k
                  in range(1,harmonics)])

#### Gauss Pulse  ####
# Compute Fourier coefficients using simps
a0 = 1.0/L*simps(gausspulse(x),x)
an = lambda n: 1.0/L*simps(gausspulse(x)*np.cos(n*np.pi*x/L),x)
bn = lambda n: 1.0/L*simps(gausspulse(x)*np.sin(n*np.pi*x/L),x)
```

```
# Compute Fourier coefficients using quad
a0 = 1.0/L*quad(lambda x: gausspulse(x),-L,L)[0]
an = lambda n: 1.0/L*quad(lambda x: gausspulse(x)*np.cos(n*np.pi*x/L),-L,L)[0]
bn = lambda n: 1.0/L*quad(lambda x: gausspulse(x)*np.sin(n*np.pi*x/L),-L,L)[0]


# Compute Fourier series
summgp = a0/2 + sum([an(k)*np.cos(k*np.pi*x/L) + bn(k)*np.sin(k*np.pi*x/L) for k
                  in range(1,harmonics)])


# Plot
plt.figure(1)
plt.subplot(221)
plt.plot(x, square(x), 'r-', label = 'Signal (Scipy)');
plt.plot(x, summsq,    'o-',  label = 'Computed');
plt.legend(loc='best', prop={'size':16})
plt.title("Square Wave", size=16)
plt.xlabel('x', size=16)
plt.xticks(size=14)
plt.ylabel('f(x)', size=20)
plt.yticks(size=14)
plt.grid()
plt.tight_layout()

plt.subplot(222)
plt.plot(x, sawtooth(x), 'r-', label = 'Signal (Scipy)');
plt.plot(x, summst,      'o-',  label = 'Computed');
plt.legend(loc='best', prop={'size':16})
plt.title("Sawtooth Wave", size=16)
plt.xlabel('x', size=16)
plt.xticks(size=14)
plt.ylabel('f(x)', size=20)
plt.yticks(size=14)
plt.grid()
plt.tight_layout()

plt.subplot(223)
plt.plot(x, sawtooth(x,width=0.5), 'r-', label = 'Signal (Scipy)');
plt.plot(x, summtr,      'o-',  label = 'Computed');
plt.legend(loc='best', prop={'size':16})
plt.title("Triangular Wave", size=16)
plt.xlabel('x', size=16)
plt.xticks(size=14)
plt.ylabel('f(x)', size=20)
plt.yticks(size=14)
plt.grid()
plt.tight_layout()

plt.subplot(224)
plt.plot(x, gausspulse(x), 'r-', label = 'Signal (Scipy)');
plt.plot(x, summgp,      'o-',  label = 'Computed');
plt.legend(loc='best', prop={'size':16})
plt.title("Gaussian Pulse", size=16);
plt.xlabel('x', size=16)
plt.xticks(size=14)
plt.ylabel('f(x)', size=20)
plt.yticks(size=14)
plt.grid()
plt.tight_layout()
plt.show()
```