A function is said periodic with period L, if it exhibits same pattern after interval L along x-axis.

Any periodic function can be approximated as

$$f(x) = \frac{a_0}{2} + a_1 cos kx + b_1 sin kx + a_2 cos 2kx + b_2 sin 2kx + \cdots \ldots \ldots \ldots \ldots$$

$$or, \quad f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\ nkx + \sum_{n=1}^{\infty} b_n \sin\ nkx$$

Where $k = \frac{2\pi}{L}$

This is called FOURIER series expansion of a function

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{+\pi} f(x)dx$$

$$a_n = \frac{1}{\pi} \int_{-\pi}^{+\pi} f(x) \cos nx\, dx$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{+\pi} f(x) \sin nx\, dx$$

a_0, a_n, b_n are Fourier Coefficients.

If f(x) is even, then b_ns are zero

If f(x) ix odd ,then a_ns are zero

A function is odd, if $f(-x) = -f(x)$

A function is even, if $f(-x) = f(x)$

General example: Cos X is even function, Sin x is odd function

For Odd function with period 2L,

$$\int_{-L}^{+L} f(x)dx = \int_{-L}^{0} f(x)dx + \int_{0}^{+L} f(x)dx = -\int_{0}^{+L} f(x)dx + \int_{0}^{+L} f(x)dx = 0$$

For Even function with period 2L,

$$\int_{-L}^{+L} f(x)dx = \int_{-L}^{0} f(x)dx + \int_{0}^{+L} f(x)dx = \int_{0}^{+L} f(x)dx + \int_{0}^{+L} f(x)dx = 2\int_{0}^{L} f(x)dx$$

The coefficient a_0 represents, the area under f(x) in one time period 2π and is scaled by the time period, represents the average value.

Example 1: SQUARE WAVE

$$f(x) = \begin{cases} 0, & -\pi \le x \le 0 \\ 1, & 0 \le x \le \pi \end{cases}$$

f(x) has period 2π

To start with, we can make use of the built-in piecewise continuous functions such as **square, sawtooth** etc from the scipy.signal module.

For integration, **quad** from scipy.integrate module

For mathematical functions, like sine, cosine, **math** from math module.

*Signifies all math functions are imported

For defining function I am using **lambda** from numpy module. You may use **def & return** instead.

Python Code for drawing square wave:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import square
from scipy.integrate import quad
from math import* # import all function from math
```

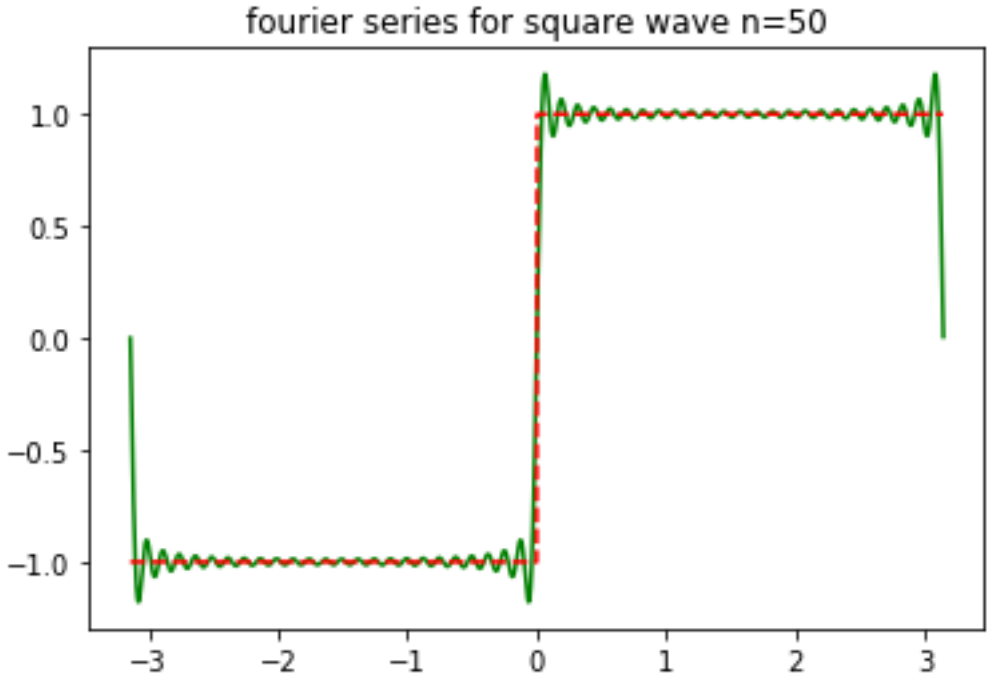
x=np.arange(-np.pi,np.pi,0.001) #x axis has been chosen from –π to +π, value of 1 smallest square along x axis is 0.001

y=square(x) # defining square wave function $y = \begin{matrix} -1,\ for -\pi \le x \le 0 \\ +1,\ for\ 0 \le x \le \pi \end{matrix}$

```
#define fuction
fc=lambda x:square(x)*cos(i*x) # i :dummy index
fs=lambda x:square(x)*sin(i*x)

n=50#max value of I, not taken infinity, better result with high value
An=[] # defining array
Bn=[]
sum=0

for i in range(n):
    an=quad(fc,-np.pi,np.pi)[0]*(1.0/np.pi)
    An.append(an)



for i in range(n):
    bn=quad(fs,-np.pi,np.pi)[0]*(1.0/np.pi)
    Bn.append(bn) # putting value in array Bn

for i in range(n):
    if i==0.0:
        sum=sum+An[i]/2
    else:
        sum=sum+(An[i]*np.cos(i*x)+Bn[i]*np.sin(i*x))
plt.plot(x,sum,'g')
plt.plot(x,y,'r--')
```

plt.title("fourier series for square wave")

plt.show()



Repeat this with n=10 and 100 . see what happens.

Example:2 **SQUARE WAVE**

$$f(x) = \begin{cases} 0, & -L \le x \le 0 \\ 1, & 0 \le x \le L \end{cases}$$

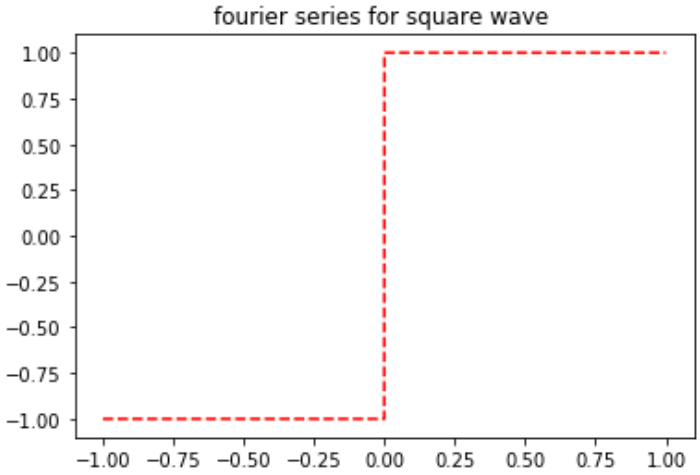
f(x) has period 2L

for L=1

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import square

L=1
x=np.arange(-L,L,0.001)
y=square(x)
plt.plot(x,y,'r--')
```

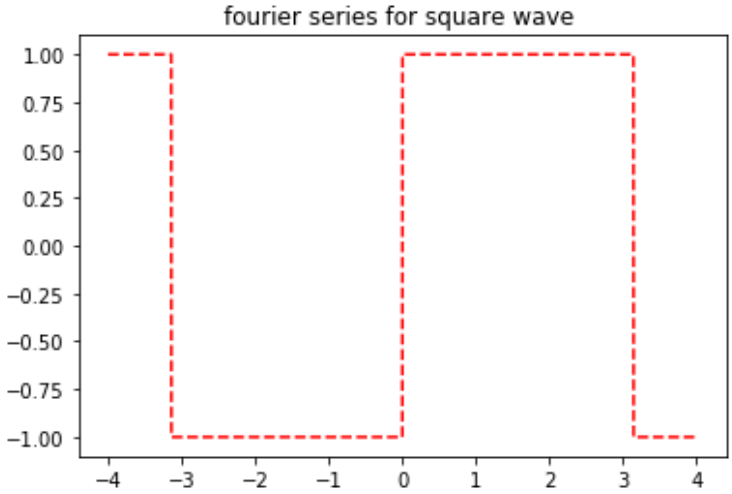
plt.title("fourier series for square wave ")

plt.show()



For L=4, period=8

Change L=4



For, L=0.5, period=1

import numpy as np

import matplotlib.pyplot as plt

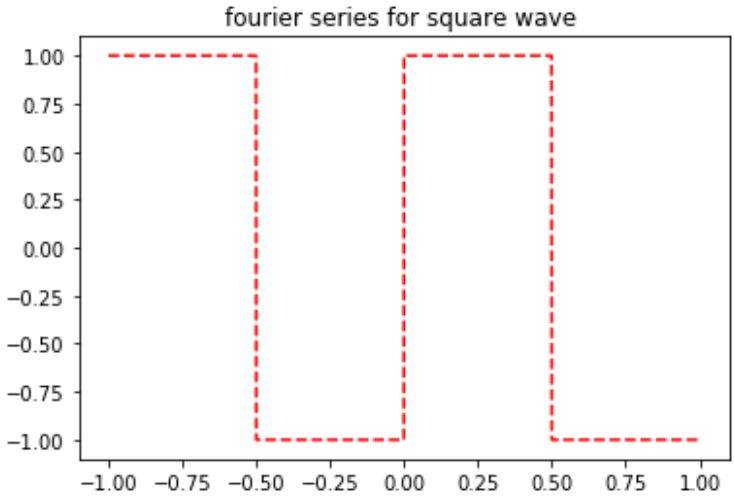
from scipy.signal import square

L=1

x=np.arange(-L,L,0.001)

y=square(2*np.pi*x)

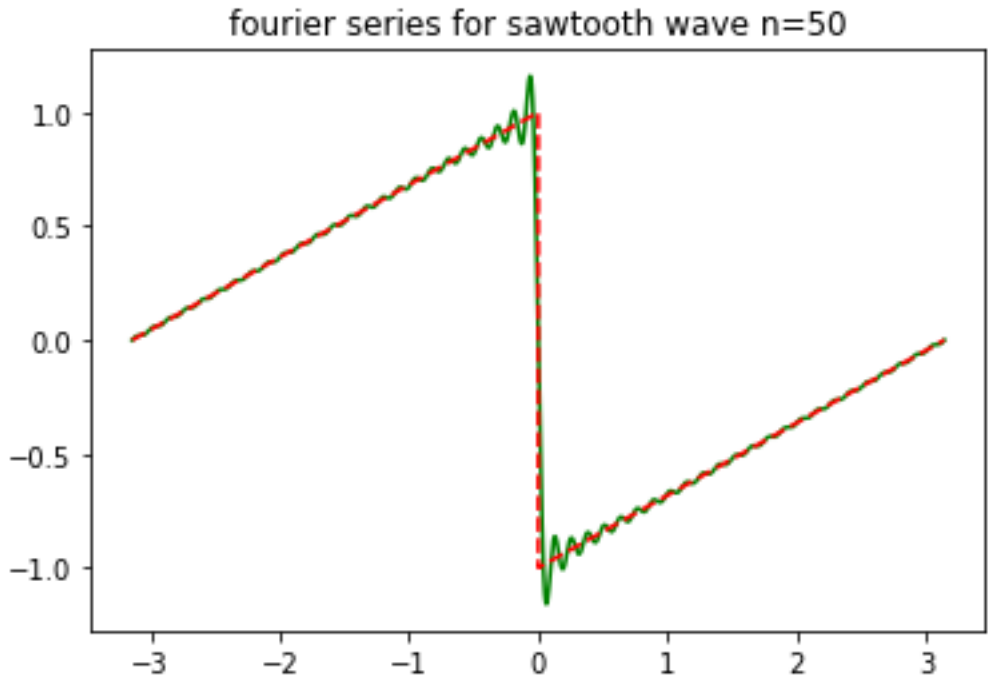
```
plt.plot(x,y,'r--')
plt.title("fourier series for square wave ")
plt.show()
```



For sawtooth wave,

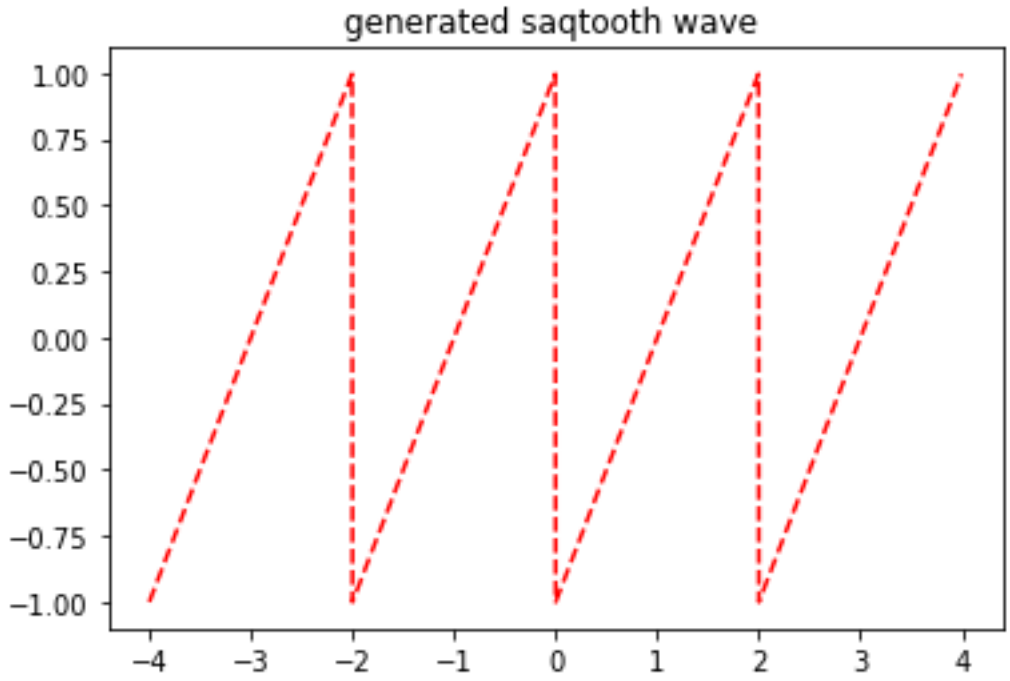
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import sawtooth
from scipy.integrate import quad
from math import* # import all function from math


x=np.arange(-np.pi,np.pi,0.001)
y=sawtooth(x)

#define fuction
fc=lambda x:sawtooth(x)*cos(i*x)
fs=lambda x:sawtooth(x)*sin(i*x)
```

```
n=50
An=[]
Bn=[]
sum=0

for i in range(n):
    an=quad(fc,-np.pi,np.pi)[0]*(1.0/np.pi)
    An.append(an)



for i in range(n):
    bn=quad(fs,-np.pi,np.pi)[0]*(1.0/np.pi)
    Bn.append(bn)

for i in range(n):
    if i==0.0:
        sum=sum+An[i]/2
    else:
        sum=sum+(An[i]*np.cos(i*x)+Bn[i]*np.sin(i*x))

print("sum=", sum)
plt.plot(x,sum,'g')
plt.plot(x,y,'r--')
plt.title("fourier series for sawtooth wave n=50")
plt.show()
```



Sawtooth wave / generated

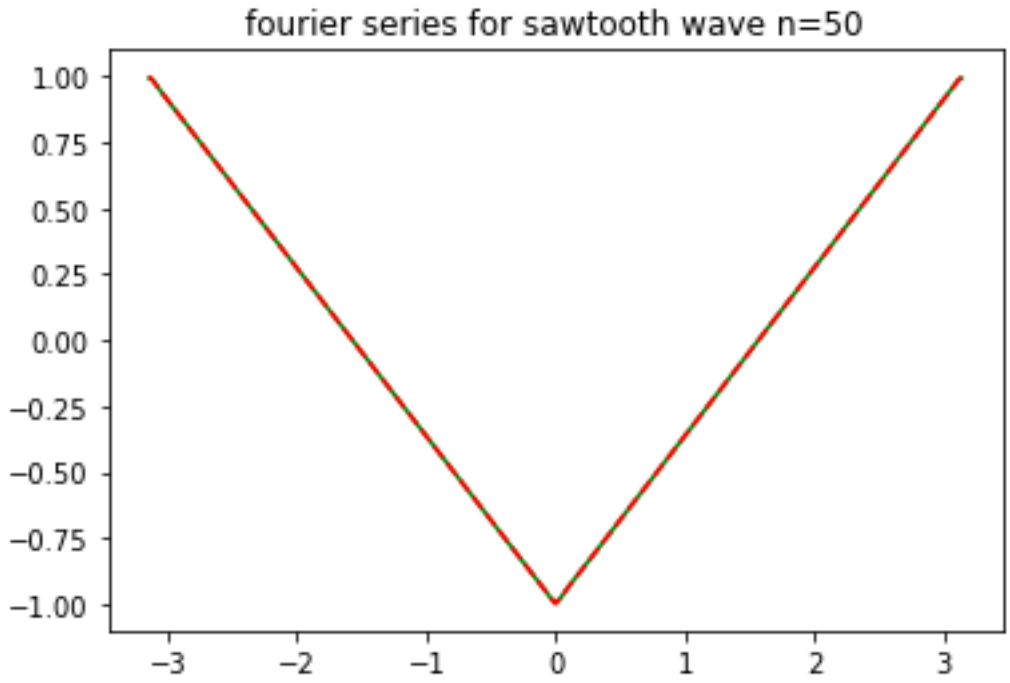
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

L=1
x=np.arange(-L,L,0.001)
xp=4*x
y=lambda xp:xp%(2*L)-L
plt.plot(xp,y(xp),'r--')
plt.title("generated saqtooth wave ")
plt.show()
```



For triangular wave

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import sawtooth
from scipy.integrate import quad
from math import* # import all function from math


x=np.arange(-np.pi,np.pi,0.001)
y=sawtooth(x, width=0.5)#width take care of time of rise =time od fall

#define fuction
fc=lambda x:sawtooth(x, width=0.5)*cos(i*x)
fs=lambda x:sawtooth(x, width=0.5)*sin(i*x)
```

```
n=50
An=[]
Bn=[]
sum=0

for i in range(n):
    an=quad(fc,-np.pi,np.pi)[0]*(1.0/np.pi)
    An.append(an)



for i in range(n):
    bn=quad(fs,-np.pi,np.pi)[0]*(1.0/np.pi)
    Bn.append(bn)

for i in range(n):
    if i==0.0:
        sum=sum+An[i]/2
    else:
        sum=sum+(An[i]*np.cos(i*x)+Bn[i]*np.sin(i*x))

print("sum=", sum)
plt.plot(x,sum,'g')
plt.plot(x,y,'r--')
plt.title("fourier series for sawtooth wave n=50")
plt.show()
```

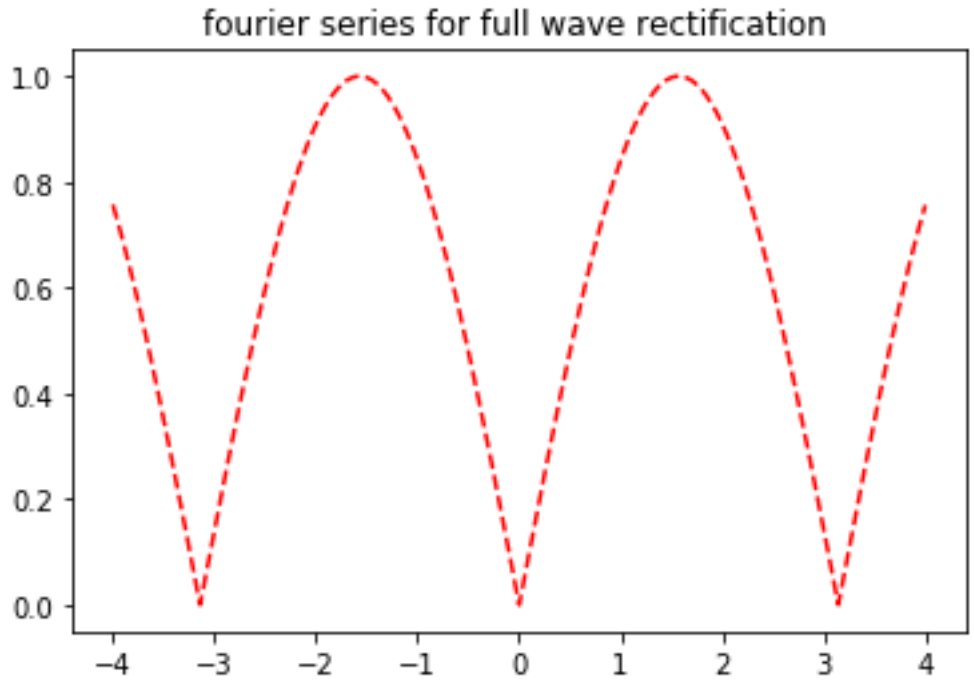


Exercise:

1.Generate triangular wave, square wave and sawtooth wave and do fourier series analysis.

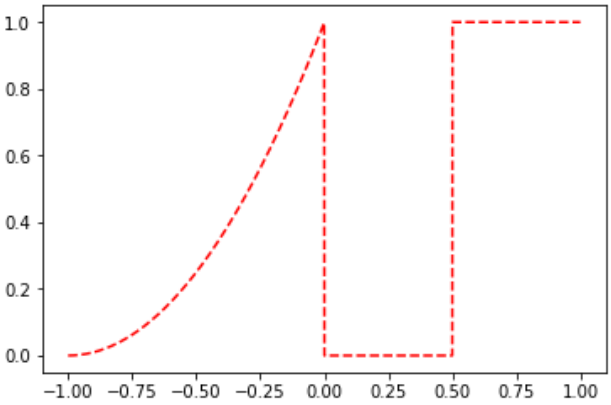
2. Generate full-wave rectification curve, f = abs(np.sin(x)) , do fourier series analysis

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

L=4
x=np.arange(-L,L,0.001)
y=abs(np.sin(x))
plt.plot(x,y,'r--')
plt.title("fourier series for full wave rectification ")
plt.show()
```



3. Generate a function

y=0 for -1$y = \begin{matrix} 0, for -1 \le x \le -0.5 \\ 1, for -0.5 \le x \le 0 \\ x^2, for\ 0 \le x \le 1 \end{matrix}$



Do fourier series analysis.

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from scipy import signal


L=1
x=np.arange(-L,L,0.001)


F=lambda x:np.array([0 if -L<=i<-0.5 else 1 if -0.5<=i<0 else i**2 for i in x])
xp=4*x # you may choose any no, even 1 instead of 4
def y(xp):
    y=xp%(2*L)-L
    return F(y)
plt.plot(xp,y(xp),'r--')
plt.title("generating given function ")
plt.show()
```