1. **Fast Fourier Transform : scipy.fft**

Fourier analysis is a technique for describing a function as a sum of periodic components and recovering the signal.

It is applied to a time-domain signal to assess its frequency domain performance. Fourier transformation is used in signal and noise processing, image processing, audio signal processing, and other fields.

The discrete Fourier transform of a complex or real sequence can be returned using the fft functions.

The main functions of fft are as follows:

* scipy.fftpack.fft() to compute the FFT
* scipy.fftpack.fftfreq() to generate the sampling frequencies
* scipy.fftpack.ifft() computes the inverse FFT, from frequency space to signal space

**Example 1 : Finding Fourier Transform of an array**

*import numpy as np*

*from scipy.fftpack import fft*

*x = np.array([1,2,3,4])*

*y = fft(x)*

*print(y)*

*Output:* [10.-0.j -2.+2.j -2.-0.j -2.-2.j]

**Example 2 : Generating noisy input signal and generate sampling frequencies from signal**

*import numpy as np*

*time\_step = 0.1*

*period = 2.*

*time\_vec = np.arange(0, 5, time\_step)*

*sig = np.sin(2 \* np.pi / period \* time\_vec) + 0.5 \*np.random.randn(time\_vec.size)*

*print(sig.size)*

*from scipy import fftpack*

*sample\_freq = fftpack.fftfreq(sig.size, d = time\_step)*

*sig\_fft = fftpack.fft(sig)*

*print (sig\_fft)*

**Example 3: Finding Inverse of discrete complex or real numbers**

*from scipy.fftpack import ifft*

*x = np.array([1,2,3,4])*

*y = ifft(x)*

*print(y)*

Output: [ 2.5-0.j -0.5-0.5j -0.5-0.j -0.5+0.5j]

**Example 4: Discrete Cosine Transform**

A discrete cosine transform (DCT) is a method of expressing a finite sequence of data points as a sum of cosine functions oscillating at different frequencies. The functions dct and idct in SciPy offer a DCT and a corresponding IDCT, respectively.

*from scipy.fftpack import dct*

*print(dct(np.array([3., 4., 10., 5.])))*

*Output : [44. -8.28771932 -8.48528137 9.55582066]*