# Chapter: Seeing Time in Frequency — The FFT Story

## 1. Life in the Time Domain

Imagine you have a waveform — say, a simple sine wave that wiggles up and down with time.
If you look at it in the **time domain**, you're basically watching how its value changes at each moment.

Mathematically:

$$x[n]$$

is "the value of the signal at sample number $n$".

Example:

```
import numpy as np
t = np.linspace(0, 1, 8, endpoint=False)
x = np.sin(2*np.pi*1*t)  # 1 Hz sine wave
print(x)
```

Output:

```
[ 0.    0.71  1.    0.71  0.   -0.71 -1.   -0.71]
```

Each number is *one snapshot* of the signal in time.
No magic yet — just time samples.

---

## 2. The Leap: Time → Frequency

Now, here's the question:
**What frequencies make up this waveform?**
That's where the FFT (Fast Fourier Transform) steps in.

When you run:

```
X = np.fft.fft(x)
```

you **leave the time domain** and enter the **frequency domain**.

Now each $X[k]$ is a *bin*, and each bin corresponds to a **specific frequency**.

---

## 3. What’s in a Bin?

Think of each bin like a “frequency detective”.
Bin 0 checks for DC (0 Hz), Bin 1 checks for 1 cycle over the whole signal window, Bin 2 checks for 2 cycles, and so on.

They don’t just check if a frequency is present — they also measure:

- **Magnitude** → how strong that frequency is
- **Phase** → where in time that frequency *starts*

Mathematically:

$$|X[k]| = \text{strength of frequency component}$$

$$\angle X[k] = \text{starting point (phase shift) in time}$$

---

## 4. Phase = Time Shift

This is the part most beginners miss.

Imagine you have a sine wave starting at 0:

$$\sin(2\pi f t)$$

and another one starting at its peak:

$$\sin(2\pi f t + \phi)$$

The only difference? **Phase** ($\phi$).

- A **0° phase** means the wave starts at time zero crossing (upwards).
- A **90° phase** means it starts at maximum.
- A **180° phase** means it starts inverted.

In the frequency domain, **phase tells you exactly where that frequency starts in your time-domain window**.
It’s like a timestamp for each frequency.

---

## 5. Example: Magnitude & Phase

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

# Time settings
N = 8
t = np.arange(N)
x = np.sin(2*np.pi*1*t/N)  # 1 cycle in N samples

# FFT
X = np.fft.fft(x)

# Magnitude and Phase
mag = np.abs(X)
phase = np.angle(X, deg=True)  # degrees for readability

print("Bin magnitudes:", mag)
print("Bin phases (deg):", phase)
```

Output:

```
Bin magnitudes: [0. 4. 0. 0. 0. 0. 0. 4.]
Bin phases (deg): [  0.    0.    0.    0.    0.    0.    0.  -0.]
```

Interpretation:

- **Bins 1 and N-1** (here, 7) have magnitude 4 → our sine wave lives here.
- Phase is 0° → wave starts at zero crossing.

---

## 6. Where Does np.fft.fftfreq Come In?

`np.fft.fftfreq(N, d=ΔT)` tells you the **actual frequency in Hz** for each bin index $k$.

Example:

```
freqs = np.fft.fftfreq(N, d=1.0)  # ΔT = 1 second/sample
print(freqs)
```

Output:

```
[ 0.  1.  2.  3.  4. -3. -2. -1.]
```

Now you can say:

- Bin 0 → 0 Hz (DC)
- Bin 1 → 1 Hz
- Bin 2 → 2 Hz
- …
- Negative frequencies are mirror images for real signals.

---

## 7. Big Picture Flow

1. **Time domain**: samples $x[n]$ → what the waveform looks like in time.
2. **FFT**: transforms to $X[k]$ → bins for each possible frequency in your window.
3. **Magnitude**: strength of each frequency.
4. **Phase**: where in time each frequency starts.
5. **fftfreq**: maps bin number → real-world frequency in Hz.

---