Cheat sheet

Python, NumPy and SciPy for signals

Arrays, maths, FFTs, filters and plotting: the Python you need for signals, communications and lab data.

1 Arrays

Import

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

Create

```
np.array([1, 2, 3])
np.zeros(8)   np.ones((2, 3))
```

Ranges

```
np.arange(0, 1, 0.01)
np.linspace(0, 1, 101)
```

linspace includes the end point

Shape

```
a.shape   a.reshape(2, -1)
a.T
```

-1 means "work it out"

2 Indexing

Slices

```
a[2:5]   a[-1]   a[::-1]
a[::2]
```

2D

```
m[0, :]   m[:, 1]   m[1:, :2]
```

By condition

```
a[a > 0]   np.where(a > 0, a, 0)
```

3 Maths on whole arrays

Element by element

```
a + b   a * b   np.sin(a)
np.abs(a)
```

No loops needed

Statistics

```
a.mean()   a.std()   a.max()
a.argmax()
```

Along an axis

```
m.sum(axis=0)
m.mean(axis=1)
```

Matrix product

```
A @ B
```

Linear algebra

```
np.linalg.solve(A, b)
np.linalg.eig(A)
np.linalg.inv(A)
```

Decibels

```
20 * np.log10(np.abs(h))
```

10 log10 for power

4 Signals

Time axis and sine

```
fs = 1000
t = np.arange(0, 1, 1/fs)
x = np.sin(2*np.pi*50*t)
```

Spectrum of a real signal

```
X = np.fft.rfft(x * np.hanning(len(x)))
f = np.fft.rfftfreq(len(x), 1/fs)
```

The window reduces leakage

Convolution

```
y = np.convolve(x, h, mode="same")
```

Add noise

```
rng = np.random.default_rng(0)
x_noisy = x + 0.1 * rng.standard_normal(len(x))
```

Complex signals

```
z = np.exp(1j*2*np.pi*f0*t)
np.abs(z)   np.angle(z)
z.real   z.imag
```

5 Filters (scipy.signal)

Import

```
from scipy import signal
```

IIR Butterworth	`b, a = signal.butter(4, 100, btype="low", fs=fs)` `y = signal.filtfilt(b, a, x)`

filtfilt runs forwards and backwards: no phase delay

FIR	`h = signal.firwin(101, 100, fs=fs)` `y = signal.lfilter(h, 1, x)`
Frequency response	`w, H = signal.freqz(b, a, fs=fs)`
Find peaks	`peaks, _ = signal.find_peaks(x, height=0.5)`

6 Plotting

Line plot	`plt.plot(t, x, label="x")`
	`plt.xlabel("time (s)"); plt.legend(); plt.grid(True)`
	`plt.show()`

Two plots	`fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True)`
Log scale	`plt.semilogy(snr_db, ber)` `plt.semilogx(f, gain_db)`
Save	`plt.savefig("plot.png", dpi=200, bbox_inches="tight")`

7 Lab data

Read CSV	`data = np.loadtxt("scope.csv", delimiter=",", skiprows=1)` `t, v = data[:, 0], data[:, 1]`
Save CSV	`np.savetxt("out.csv", np.column_stack([t, y]), delimiter=",")`
Fit a line	`slope, offset = np.polyfit(x, y, 1)`