SciPy (Scientific Computing) Cheat Sheet

SciPy is a powerful library for scientific computing in Python. It extends NumPy and provides functionalities for optimization, integration, linear algebra, signal processing, and more.

---

1. Importing SciPy

- **import scipy as sp** - Imports the SciPy library.
- **from scipy import linalg, optimize, stats, integrate, interpolate, fft, signal, ndimage, special** - Imports specific SciPy modules.

---

2. Linear Algebra (scipy.linalg)

- **linalg.inv(A)** - Computes the inverse of matrix A.
- **linalg.det(A)** - Computes the determinant of matrix A.
- **linalg.solve(A, b)** - Solves the linear system Ax = b.
- **linalg.eig(A)** - Computes the eigenvalues and eigenvectors of A.
- **linalg.svd(A)** - Computes Singular Value Decomposition (SVD) of A.
- **linalg.norm(A, ord=None)** - Computes matrix or vector norm.
- **linalg.lu(A)** - Computes LU decomposition of A.
- **linalg.qr(A)** - Computes QR decomposition of A.
- **linalg.cholesky(A)** - Computes Cholesky decomposition of A.

---

3. Optimization (scipy.optimize)

- **optimize.minimize(f, x0, method='BFGS')** - Minimizes function f starting at x0 using the BFGS algorithm.
- **optimize.root(f, x0, method='hybr')** - Finds the root of function f starting at x0.
- **optimize.curve_fit(model, xdata, ydata)** - Fits a function to data points.
- **optimize.linprog(c, A_ub, b_ub, bounds)** - Solves linear programming problems.
- **optimize.differential_evolution(f, bounds)** - Performs global optimization using evolutionary algorithms.

---

4. Integration (scipy.integrate)

- **integrate.quad(f, a, b)** - Computes the definite integral of *f* from a to b.
- **integrate.dblquad(f, a, b, g1, g2)** - Computes a double integral over a region.

- **integrate.odeint(func, y0, t)** - Solves ordinary differential equations (ODEs).
- **integrate.simps(y, x)** - Computes numerical integration using Simpson's rule.
- **integrate.trapz(y, x)** - Computes numerical integration using the trapezoidal rule.

---

5. Interpolation (scipy.interpolate)

- **interpolate.interp1d(x, y, kind='cubic')** - Performs 1D interpolation.
- **interpolate.griddata(points, values, xi, method='cubic')** - Performs interpolation on a grid.
- **interpolate.Rbf(x, y, z, function='multiquadric')** - Performs radial basis function interpolation.
- **interpolate.PchipInterpolator(x, y)** - Performs shape-preserving piecewise cubic interpolation.

---

6. Fast Fourier Transform (FFT) (scipy.fft)

- **fft.fft(x)** - Computes the Fast Fourier Transform of x.
- **fft.ifft(x)** - Computes the inverse Fast Fourier Transform.
- **fft.fftfreq(n, d=1.0)** - Returns the frequencies for FFT components.
- **fft.fftshift(x)** - Shifts zero frequency component to the center.

---

7. Statistical Analysis (scipy.stats)

- **stats.describe(data)** - Computes summary statistics (mean, variance, etc.).
- **stats.norm.pdf(x, loc, scale)** - Computes the probability density function (PDF) of a normal distribution.
- **stats.norm.cdf(x, loc, scale)** - Computes the cumulative distribution function (CDF) of a normal distribution.
- **stats.ttest_ind(a, b)** - Computes t-test for independent samples.
- **stats.chisquare(f_obs, f_exp)** - Performs a chi-square test.
- **stats.pearsonr(x, y)** - Computes Pearson correlation coefficient.

---

8. Signal Processing (scipy.signal)

- **signal.butter(order, cutoff, btype='low', analog=False)** - Designs a Butterworth filter.
- **signal.filtfilt(b, a, data)** - Applies a digital filter forward and backward.
- **signal.spectrogram(data, fs)** - Computes a spectrogram.

- **signal.find_peaks(data, height=0.5)** - Finds peaks in a signal.
- **signal.welch(data, fs)** - Computes power spectral density using Welch's method.

---

9. Image Processing (scipy.ndimage)

- **ndimage.gaussian_filter(image, sigma=1)** - Applies a Gaussian filter to an image.
- **ndimage.median_filter(image, size=3)** - Applies a median filter to an image.
- **ndimage.sobel(image, axis=0)** - Applies a Sobel filter for edge detection.
- **ndimage.zoom(image, zoom=2.0)** - Zooms in or out on an image.
- **ndimage.rotate(image, angle=45, reshape=True)** - Rotates an image by a given angle.

---

10. Special Functions (scipy.special)

- **special.gamma(x)** - Computes the Gamma function.
- **special.erf(x)** - Computes the error function.
- **special.jn(n, x)** - Computes the nth-order Bessel function of the first kind.
- **special.beta(a, b)** - Computes the Beta function.
- **special.ellipj(u, m)** - Computes Jacobi elliptic functions.
- **special.zeta(x, q=1.0)** - Computes the Riemann zeta function.

Official documentation: https://docs.scipy.org/doc/scipy/