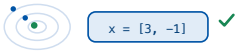
SciPy

A field guide to SciPy, the toolbox built on NumPy that spans optimization, integration, statistics, and signal processing.

Optimize & Fit

Find minima, roots, and best-fit parameters. from scipy.optimize import ...

PURPOSE	COMMAND	RESULT
Minimize a scalar function.	`minimize(f, x0=[0,0], method="BFGS")`	
Solve f(x) = 0 in a bracket.	`root_scalar(f, bracket=[0,2], method="brentq")`	
Solve a nonlinear system.	`root(fun, x0=[0, 0])`	
Fit a model curve to data.	`curve_fit(model, xdata, ydata, p0=[1,1,0])`	
Nonlinear least squares.	`least_squares(resid, x0=[0, 0])`	
Solve a linear program.	`linprog(c, A_ub, b_ub, bounds=...)`	

every solver returns an OptimizeResult; check res.success before trusting res.x

Linear Algebra

Solve systems, decompose matrices, and measure them. from scipy import linalg

PURPOSE	COMMAND	RESULT
Solve Ax = b.	`linalg.solve(A, b)`	
LU decomposition.	`linalg.lu(A)`	
Eigenvalues / vectors.	`linalg.eig(A)`	
Singular value decomp.	`linalg.svd(A)`	
Cholesky factor (SPD).	`linalg.cholesky(A, lower=True)`	
Determinant, inverse, norm.	`linalg.det / .inv / .norm`	

never invert to solve a system: call solve(A, b); decompose for structure

Integrate & ODEs

Numerically integrate functions, samples, and differential equations.

PURPOSE	COMMAND	RESULT
Definite integral of a function.	`quad(f, 0, np.inf)`	
Double integral over a region.	`dblquad(f, 0, 1, 0, 1)`	
Integrate samples (trapezoid).	`trapezoid(y, x)`	
Integrate samples (Simpson).	`simpson(y, x=x)`	
Solve an initial-value ODE.	`solve_ivp(f, [0, 10], y0=[2.0])`	

quad integrates a callable; trapezoid/simpson integrate samples you already have

Interpolate

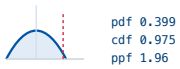
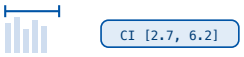
Build a callable that fills in values between your data points.

PURPOSE	COMMAND	RESULT
1-D cubic spline through points.	`CubicSpline(x, y)`	
Generic 1-D interpolation.	`interp1d(x, y, kind="cubic")`	
Shape-preserving (monotone).	`PchipInterpolator(x, y)`	
Smoothing B-spline fit.	`make_splrep(x, y, s=0.5)`	
Scattered 2-D interpolation.	`griddata(points, values, xi)`	
Smooth scattered N-D (RBF).	`RBFInterpolator(points, values)`	

interpolation turns discrete (x, y) samples into a callable you evaluate anywhere

Statistics

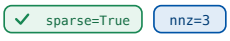
Distributions, summaries, and hypothesis tests via scipy.stats.

PURPOSE	COMMAND	RESULT
Distribution pdf / cdf / ppf.	`norm.pdf(0) .cdf(1.96) .ppf(.975)`	
Freeze a distribution.	`dist =norm(loc=10, scale=2)`	
Two-sample t-test.	`ttest_ind(a, b)`	
Correlation between vectors.	`pearsonr(x, y)`	
Simple linear regression.	`linregress(x, y)`	
Bootstrap a confidence interval.	`bootstrap((data,), np.mean)`	

read tests off the result's .statistic and .pvalue; freeze a dist and reuse it

Sparse Matrices

Store and solve with mostly-zero matrices using the modern array API.

PURPOSE	COMMAND	RESULT
Build from coordinates (COO).	`coo_array((data, (row, col)), shape=(3,3))`	
Convert to CSR for math.	`A = A.tocsr()`	
Sparse identity / diagonals.	`eye_array(3)diags_array(...)`	
Densify to a NumPy array.	`A.toarray()`	
Solve a sparse linear system.	`spsolve(A.tocsc(), b)`	
Check sparsity / nonzeros.	`issparse(A) A.nnz`	

prefer the sparse *array* classes; build in COO, convert to CSR/CSC, toarray() last

Signal & FFT

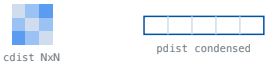
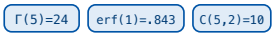
Filter, transform, and find structure in 1-D signals.

PURPOSE	COMMAND	RESULT
Design a Butterworth filter.	`butter(4, 0.2, btype="low")`	
Zero-phase filtering.	`filtfilt(b, a, x)`	
Fast Fourier transform.	`fft.fft(x) fft.fftfreq(n, d)`	
Power spectral density.	`periodogram(x, fs=500)`	
Find peaks in a signal.	`find_peaks(x, height=0.5)`	
Convolve two sequences.	`convolve(a, v)`	

design (butter) -> apply without distorting timing (filtfilt) -> inspect (fft)

Spatial & Special

Nearest-neighbor queries, geometry, and the special functions library.

PURPOSE	COMMAND	RESULT
KD-tree, query neighbors.	`KDTree(pts).query(q, k=1)`	
Pairwise distance matrices.	`distance.cdist(A, B) .pdist(A)`	
Convex hull of points.	`ConvexHull(pts)`	
Delaunay triangulation.	`Delaunay(pts)`	
Special functions.	`special.gamma / .erf / .comb`	
ML helpers (expit, softmax).	`special.expit / .softmax`	

scipy.spatial answers geometry; scipy.special catalogs math + stable ML helpers