calc

Provide tools for unit-aware, meteorological calculations.

Dry Thermodynamics

`add_height_to_pressure`(pressure, height)	Calculate the pressure at a certain height above another pressure level.
`add_pressure_to_height`(height, pressure)	Calculate the height at a certain pressure above another height.
`density`(pressure, temperature, mixing_ratio)	Calculate density.
`dry_lapse`(pressure, temperature[, ...])	Calculate the temperature at a level assuming only dry processes.
`dry_static_energy`(height, temperature)	Calculate the dry static energy of parcels.
`geopotential_to_height`(geopotential)	Compute height above sea level from a given geopotential.
`height_to_geopotential`(height)	Compute geopotential for a given height above sea level.
`mean_pressure_weighted`(pressure, *args[, ...])	Calculate pressure-weighted mean of an arbitrary variable through a layer.
`potential_temperature`(pressure, temperature)	Calculate the potential temperature.
`signma_to_pressure`(sigma, pressure, sfc, ...)	Calculate pressure from sigma values.
`static_stability`(pressure, temperature[, ...])	Calculate the static stability within a vertical profile.
`temperature_from_potential_temperature`(...)	Calculate the temperature from a given potential temperature.
`thickness_hydrostatic`(pressure, temperature)	Calculate the thickness of a layer via the hypsometric equation.

Moist Thermodynamics

`dewpoint`(vapor_pressure)	Calculate the ambient dewpoint given the vapor pressure.
`dewpoint_from_relative_humidity`(temperature, ...)	Calculate the ambient dewpoint given air temperature and relative humidity.
`dewpoint_from_specific_humidity`(pressure, ...)	Calculate the dewpoint from specific humidity, temperature, and pressure.
`equivalent_potential_temperature`(pressure, ...)	Calculate equivalent potential temperature.
`mixing_ratio`(partial_press, total_press[, ...])	Calculate the mixing ratio of a gas.
`mixing_ratio_from_relative_humidity`(...)	Calculate the mixing ratio from relative humidity, temperature, and pressure.
`mixing_ratio_from_specific_humidity`(...)	Calculate the mixing ratio from specific humidity.
`moist_lapse`(pressure, temperature[, ...])	Calculate the temperature at a level assuming liquid saturation processes.
`moist_static_energy`(height, temperature, ...)	Calculate the moist static energy of parcels.
`precipitable_water`(pressure, dewpoint, *[, ...])	Calculate precipitable water through the depth of a sounding.
`psychrometric_vapor_pressure_wet`(pressure, ...)	Calculate the vapor pressure with wet bulb and dry bulb temperatures.
`relative_humidity_from_dewpoint`(temperature, ...)	Calculate the relative humidity.
`relative_humidity_from_mixing_ratio`(...)	Calculate the relative humidity from mixing ratio, temperature, and pressure.
`relative_humidity_from_specific_humidity`(...)	Calculate the relative humidity from specific humidity, temperature, and pressure.
`relative_humidity_wet_psychrometric`(...)	Calculate the relative humidity with wet bulb and dry bulb temperatures.
`saturation_equivalent_potential_temperature`(...)	Calculate saturation equivalent potential temperature.
`saturation_mixing_ratio`(total_press, temperature)	Calculate the saturation mixing ratio of water vapor.
`saturation_vapor_pressure`(temperature)	Calculate the saturation water vapor (partial) pressure.
`scale_height`(temperature_bottom, temperature_top)	Calculate the scale height of a layer.
`specific_humidity_from_dewpoint`(pressure, ...)	Calculate the specific humidity from the dewpoint temperature and pressure.
`specific_humidity_from_mixing_ratio`(mixing_ratio)	Calculate the specific humidity from the mixing ratio.
`thickness_hydrostatic_from_relative_humidity`(...)	Calculate the thickness of a layer given pressure, temperature and relative humidity.
`vapor_pressure`(pressure, mixing_ratio)	Calculate water vapor (partial) pressure.
`vertical_velocity`(omega, pressure, temperature)	Calculate w from omega assuming hydrostatic conditions.
`vertical_velocity_pressure`(w, pressure, ...)	Calculate omega from w assuming hydrostatic conditions.
`virtual_potential_temperature`(pressure, ...)	Calculate virtual potential temperature.
`virtual_temperature`(temperature, mixing_ratio)	Calculate virtual temperature.
`wet_bulb_temperature`(pressure, temperature, ...)	Calculate the wet-bulb temperature using Normand's rule.

Soundings

`bulk_shear`(pressure, u, v[, height, bottom, ...])	Calculate bulk shear through a layer.
`bunkers_storm_motion`(pressure, u, v, height)	Calculate the Bunkers right-mover and left-mover storm motions and sfc-6km mean flow.
`cape_cin`(pressure, temperature, dewpoint, ...)	Calculate CAPE and CIN.
`critical_angle`(pressure, u, v, height, ...)	Calculate the critical angle.
`el`(pressure, temperature, dewpoint[, ...])	Calculate the equilibrium level.
`lcl`(pressure, temperature, dewpoint[, ...])	Calculate the lifted condensation level (LCL) from the starting point.
`lfc`(pressure, temperature, dewpoint[, ...])	Calculate the level of free convection (LFC).
`lifted_index`(pressure, temperature, ...)	Calculate Lifted Index from the pressure temperature and parcel profile.
`mixed_layer`(pressure, *args[, height, ...])	Mix variable(s) over a layer, yielding a mass-weighted average.
`mixed_layer_cape_cin`(pressure, temperature, ...)	Calculate mixed-layer CAPE and CIN.
`mixed_parcel`(pressure, temperature, dewpoint)	Calculate the properties of a parcel mixed from a layer.
`most_unstable_cape_cin`(pressure, ...)	Calculate most unstable CAPE/CIN.
`most_unstable_parcel`(pressure, temperature, ...)	Determine the most unstable parcel in a layer.
`parcel_profile`(pressure, temperature, dewpoint)	Calculate the profile a parcel takes through the atmosphere.
`parcel_profile_with_lcl`(pressure, ...)	Calculate the profile a parcel takes through the atmosphere.
`parcel_profile_with_lcl_as_dataset`(pressure, ...)	Calculate the profile a parcel takes through the atmosphere, returning a Dataset.
`showalter_index`(pressure, temperature, dewpoint)	Calculate Showalter Index.
`significant_tornado`(sbcape, ...)	Calculate the significant tornado parameter (fixed layer).
`storm_relative_helicity`(height, u, v, depth, *)	Calculate storm relative helicity.
`supercell_composite`(mucape, ...)	Calculate the supercell composite parameter.
`surface_based_cape_cin`(pressure, ...)	Calculate surface-based CAPE and CIN.

Dynamic/Kinematic

`absolute_momentum`(u, v[, index])	Calculate cross-sectional absolute momentum (also called pseudoangular momentum).
`absolute_vorticity`(u, v[, dx, dy, latitude, ...])	Calculate the absolute vorticity of the horizontal wind.
`advection`(scalar[, u, v, w, dx, dy, dz, ...])	Calculate the advection of a scalar field by the wind.
`ageostrophic_wind`(height, u, v[, dx, dy, ...])	Calculate the ageostrophic wind given from the height or geopotential.
`coriolis_parameter`(latitude)	Calculate the coriolis parameter at each point.
`divergence`(u, v, *[, dx, dy, x_dim, y_dim])	Calculate the horizontal divergence of a vector.
`exner_function`(pressure[, reference_pressure])	Calculate the Exner function.
`frontogenesis`(potential_temperature, u, v[, ...])	Calculate the 2D kinematic frontogenesis of a temperature field.
`geostrophic_wind`(height[, dx, dy, latitude, ...])	Calculate the geostrophic wind given from the height or geopotential.
`inertial_advective_wind`(u, v, u_geostrophic, ...)	Calculate the inertial advective wind.
`kinematic_flux`(vel, b[, perturbation, axis])	Calculate the kinematic flux from two time series.
`montgomery_streamfunction`(height, temperature)	Compute the Montgomery Streamfunction on isentropic surfaces.
`potential_vorticity_baroclinic`(...[, dx, ...])	Calculate the baroclinic potential vorticity.
`potential_vorticity_barotropic`(height, u, v)	Calculate the barotropic (Rossby) potential vorticity.
`q_vector`(u, v, temperature, pressure[, dx, ...])	Calculate Q-vector at a given pressure level using the u, v winds and temperature.
`shearing_deformation`(u, v[, dx, dy, x_dim, ...])	Calculate the shearing deformation of the horizontal wind.
`stretching_deformation`(u, v[, dx, dy, ...])	Calculate the stretching deformation of the horizontal wind.
`total_deformation`(u, v[, dx, dy, x_dim, y_dim])	Calculate the horizontal total deformation of the horizontal wind.
`vorticity`(u, v, *[, dx, dy, x_dim, y_dim])	Calculate the vertical vorticity of the horizontal wind.
`wind_components`(speed, wind_direction)	Calculate the U, V wind vector components from the speed and direction.
`wind_direction`(u, v[, convention])	Compute the wind direction from u and v-components.
`wind_speed`(u, v)	Compute the wind speed from u and v-components.

Boundary Layer/Turbulence

`brunt_vaisala_frequency`(height, ...[, ...])	Calculate the Brunt-Vaisala frequency.
`brunt_vaisala_frequency_squared`(height, ...)	Calculate the square of the Brunt-Vaisala frequency.
`brunt_vaisala_period`(height, ...[, vertical_dim])	Calculate the Brunt-Vaisala period.
`friction_velocity`(u, w[, v, perturbation, axis])	Compute the friction velocity from the time series of velocity components.
`gradient_richardson_number`(height, ...[, ...])	Calculate the gradient (or flux) Richardson number.
`tke`(u, v, w[, perturbation, axis])	Compute turbulence kinetic energy.

Mathematical Functions

`cross_section_components`(data_x, data_y[, index])	Obtain the tangential and normal components of a cross-section of a vector field.
`first_derivative`(f, axis, x, delta])	Calculate the first derivative of a grid of values.
`gradient`(f[, axes, coordinates, deltas])	Calculate the gradient of a grid of values.
`laplacian`(f[, axes, coordinates, deltas])	Calculate the laplacian of a grid of values.
`lat_lon_grid_deltas`(longitude, latitude[, ...])	Calculate the actual delta between grid points that are in latitude/longitude format.
`normal_component`(data_x, data_y[, index])	Obtain the normal component of a cross-section of a vector field.
`second_derivative`(f, axis, x, delta])	Calculate the second derivative of a grid of values.
`tangential_component`(data_x, data_y[, index])	Obtain the tangential component of a cross-section of a vector field.
`unit_vectors_from_cross_section`(cross[, index])	Calculate the unit tangent and unit normal vectors from a cross-section.

Apparent Temperature

`apparent_temperature`(temperature, ...[, ...])	Calculate the current apparent temperature.
`heat_index`(temperature, relative_humidity[, ...])	Calculate the Heat Index from the current temperature and relative humidity.
`windchill`(temperature, speed[, ...])	Calculate the Wind Chill Temperature Index (WCTI).

Standard Atmosphere

`altimeter_to_sea_level_pressure`(...)	Convert the altimeter setting to sea-level pressure.
`altimeter_to_station_pressure`(...)	Convert the altimeter measurement to station pressure.
`height_to_pressure_std`(height)	Convert height data to pressures using the U.S.
`pressure_to_height_std`(pressure)	Convert pressure data to height using the U.S.

Smoothing

`smooth_gaussian`(scalar_grid, n)	Filter with normal distribution of weights.
`smooth_window`(scalar_grid, window[, passes, ...])	Filter with an arbitrary window smoother.
`smooth_rectangular`(scalar_grid, size[, passes])	Filter with a rectangular window smoother.
`smooth_circular`(scalar_grid, radius[, passes])	Filter with a circular window smoother.
`smooth_n_point`(scalar_grid[, n, passes])	Filter with an n-point smoother.

Other

`angle_to_direction`(input_angle[, full, level])	Convert the meteorological angle to directional text.
`azimuth_range_to_lat_lon`(azimuths, ranges, ...)	Convert azimuth and range locations in a polar coordinate system to lat/lon coordinates.
`find_bounding_indices`(arr, values, axis[, ...])	Find the indices surrounding the values within an array along axis.
`find_intersections`(x, a, b[, direction, log_x])	Calculate the best estimate of intersection.
`get_layer`(pressure, *args[, height, bottom, ...])	Return an atmospheric layer from upper air data with the requested bottom and depth.
`get_layer_heights`(height, depth, *args[, ...])	Return an atmospheric layer from upper air data with the requested bottom and depth.
`get_perturbation`(ts[, axis])	Compute the perturbation from the mean of a time series.
`isentropic_interpolation`(levels, pressure, ...)	Interpolate data in isobaric coordinates to isentropic coordinates.
`isentropic_interpolation_as_dataset`(levels, ...)	Interpolate xarray data in isobaric coords to isentropic coords, returning a Dataset.
`nearest_intersection_idx`(a, b)	Determine the index of the point just before two lines with common x values.
`parse_angle`(input_dir)	Calculate the meteorological angle from directional text.
`reduce_point_density`(points, radius[, priority])	Return a mask to reduce the density of points in irregularly-spaced data.
`resample_nn_1d`(a, centers)	Return one-dimensional nearest-neighbor indexes based on user-specified centers.

Last updated on Aug 09, 2021 at 17:24:01.
Created using Sphinx 4.1.2.