rotate(*axis: Vertex*, *angle:float*) → UCS
Returns a new rotated UCS, with the same origin as the source UCS. The rotation vector is located in the origin and has *WCS* coordinates e.g. (0, 0, 1) is the WCS z-axis as rotation vector.

Parameters

- **axis** – arbitrary rotation axis as vector in *WCS*
- **angle** – rotation angle in radians

rotate_local_x(*angle:float*) → UCS
Returns a new rotated UCS, rotation axis is the local x-axis.

Parameters **angle** – rotation angle in radians

rotate_local_y(*angle:float*) → UCS
Returns a new rotated UCS, rotation axis is the local y-axis.

Parameters **angle** – rotation angle in radians

rotate_local_z(*angle:float*) → UCS
Returns a new rotated UCS, rotation axis is the local z-axis.

Parameters **angle** – rotation angle in radians

shift(*delta: Vertex*) → UCS
Shifts current UCS by *delta* vector and returns *self*.

Parameters **delta** – shifting vector

moveto(*location: Vertex*) → UCS
Place current UCS at new origin *location* and returns *self*.

Parameters **location** – new origin in WCS

static from_x_axis_and_point_in_xy(*origin: Vertex*, *axis: Vertex*, *point: Vertex*) → UCS
Returns an new `UCS` defined by the origin, the x-axis vector and an arbitrary point in the xy-plane.

Parameters

- **origin** – UCS origin as (x, y, z) tuple in *WCS*
- **axis** – x-axis vector as (x, y, z) tuple in *WCS*
- **point** – arbitrary point unlike the origin in the xy-plane as (x, y, z) tuple in *WCS*

static from_x_axis_and_point_in_xz(*origin: Vertex*, *axis: Vertex*, *point: Vertex*) → UCS
Returns an new `UCS` defined by the origin, the x-axis vector and an arbitrary point in the xz-plane.

Parameters

- **origin** – UCS origin as (x, y, z) tuple in *WCS*
- **axis** – x-axis vector as (x, y, z) tuple in *WCS*
- **point** – arbitrary point unlike the origin in the xz-plane as (x, y, z) tuple in *WCS*

static from_y_axis_and_point_in_xy(*origin: Vertex*, *axis: Vertex*, *point: Vertex*) → UCS
Returns an new `UCS` defined by the origin, the y-axis vector and an arbitrary point in the xy-plane.

Parameters

- **origin** – UCS origin as (x, y, z) tuple in *WCS*
- **axis** – y-axis vector as (x, y, z) tuple in *WCS*
- **point** – arbitrary point unlike the origin in the xy-plane as (x, y, z) tuple in *WCS*

static from_y_axis_and_point_in_yz (*origin: Vertex*, *axis: Vertex*, *point: Vertex*) → UCS
Returns an new `UCS` defined by the origin, the y-axis vector and an arbitrary point in the yz-plane.

Parameters

- **origin** – UCS origin as (x, y, z) tuple in *WCS*
- **axis** – y-axis vector as (x, y, z) tuple in *WCS*
- **point** – arbitrary point unlike the origin in the yz-plane as (x, y, z) tuple in *WCS*

static from_z_axis_and_point_in_xz (*origin: Vertex*, *axis: Vertex*, *point: Vertex*) → UCS
Returns an new `UCS` defined by the origin, the z-axis vector and an arbitrary point in the xz-plane.

Parameters

- **origin** – UCS origin as (x, y, z) tuple in *WCS*
- **axis** – z-axis vector as (x, y, z) tuple in *WCS*
- **point** – arbitrary point unlike the origin in the xz-plane as (x, y, z) tuple in *WCS*

static from_z_axis_and_point_in_yz (*origin: Vertex*, *axis: Vertex*, *point: Vertex*) → UCS
Returns an new `UCS` defined by the origin, the z-axis vector and an arbitrary point in the yz-plane.

Parameters

- **origin** – UCS origin as (x, y, z) tuple in *WCS*
- **axis** – z-axis vector as (x, y, z) tuple in *WCS*
- **point** – arbitrary point unlike the origin in the yz-plane as (x, y, z) tuple in *WCS*

render_axis (*layout: BaseLayout*, *length: float = 1*, *colors: Tuple[int, int, int] = (1, 3, 5)*)
Render axis as 3D lines into a *layout*.

Matrix44

class `ezdxf.math.`**Matrix44** (**args*)
This is a pure Python implementation for 4x4 transformation matrices, to avoid dependency to big numerical packages like `numpy`, before binary wheels, installation of these packages wasn't always easy on Windows.

The utility functions for constructing transformations and transforming vectors and points assumes that vectors are stored as row vectors, meaning when multiplied, transformations are applied left to right (e.g. vAB transforms v by A then by B).

Matrix44 initialization:

- `Matrix44()` returns the identity matrix.
- `Matrix44(values)` values is an iterable with the 16 components of the matrix.
- `Matrix44(row1, row2, row3, row4)` four rows, each row with four values.

__repr__ () → str
Returns the representation string of the matrix: `Matrix44((col0, col1, col2, col3), (...), (...), (...))`

get_row (*row: int*) → Tuple[float, ...]
Get row as list of of four float values.

Parameters **row** – row index [0 .. 3]

set_row (*row: int, values: Sequence[float]*) → None
Sets the values in a row.

Parameters

- **`row`** – row index [0 .. 3]
- **`values`** – iterable of four row values

`get_col`(*col: int*) → Tuple[float, ...]
Returns a column as a tuple of four floats.

Parameters **`col`** – column index [0 .. 3]

`set_col`(*col: int, values: Sequence[float]*)
Sets the values in a column.

Parameters

- **`col`** – column index [0 .. 3]
- **`values`** – iterable of four column values

`copy`() → Matrix44
Returns a copy of same type.

`__copy__`() → Matrix44
Returns a copy of same type.

`classmethod scale`(*sx: float*, *sy: float = None*, *sz: float = None*) → Matrix44
Returns a scaling transformation matrix. If *sy* is `None`, *sy* = *sx*, and if *sz* is `None` *sz* = *sx*.

`classmethod translate`(*dx: float*, *dy: float*, *dz: float*) → Matrix44
Returns a translation matrix for translation vector (dx, dy, dz).

`classmethod x_rotate`(*angle: float*) → Matrix44
Returns a rotation matrix about the x-axis.

Parameters **`angle`** – rotation angle in radians

`classmethod y_rotate`(*angle: float*) → Matrix44
Returns a rotation matrix about the y-axis.

Parameters **`angle`** – rotation angle in radians

`classmethod z_rotate`(*angle: float*) → Matrix44
Returns a rotation matrix about the z-axis.

Parameters **`angle`** – rotation angle in radians

`classmethod axis_rotate`(*axis: Vertex*, *angle: float*) → Matrix44
Returns a rotation matrix about an arbitrary *axis*.

Parameters

- **`axis`** – rotation axis as `(x, y, z)` tuple or *`Vec3`* object
- **`angle`** – rotation angle in radians

`classmethod xyz_rotate`(*angle_x: float*, *angle_y: float*, *angle_z: float*) → Matrix44
Returns a rotation matrix for rotation about each axis.

Parameters

- **`angle_x`** – rotation angle about x-axis in radians
- **`angle_y`** – rotation angle about y-axis in radians
- **`angle_z`** – rotation angle about z-axis in radians