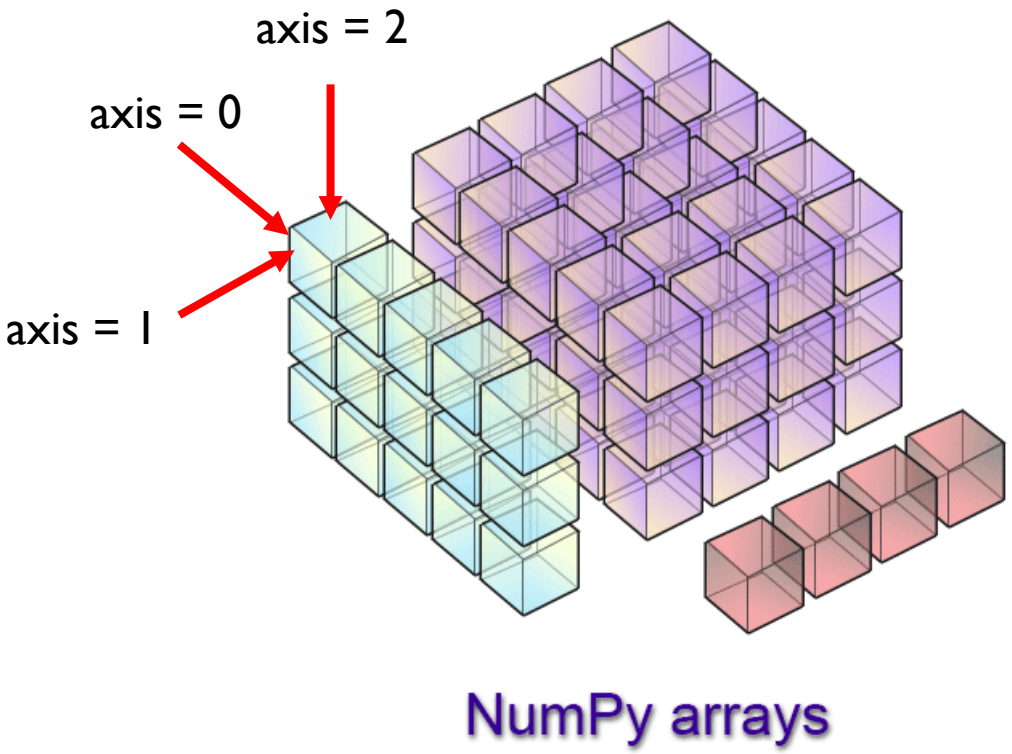
Introduction to ‘numpy’

- NumPy’s main object is the homogeneous multidimensional array.
- NumPy dimensions are called *axes*.



Introduction to ‘numpy’

- Basic operations
  - copyto()
- Changing array shape
  - reshape(a, newshape[, order]) - Gives a new shape to an array without changing its data.
  - ravel(a[, order]) - Return a contiguous flattened array.
  - ndarray.flat - A 1-D iterator over the array.
  - ndarray.flatten([order]) - Return a copy of the array collapsed into one dimension.

Introduction to ‘numpy’

- Transpose-like operations
  - moveaxis(a, source, destination) - Move axes of an array to new positions.
  - rollaxis(a, axis[, start]) - Roll the specified axis backwards, until it lies in a given position.
  - swapaxes(a, axis1, axis2) - Interchange two axes of an array.
  - ndarray.T - The transposed array.
  - transpose(a[, axes]) - Reverse or permute the axes of an array; returns the modified array.

Introduction to ‘numpy’

- Changing number of dimensions
  - atleast_1d(*arys) - Convert inputs to arrays with at least one dimension.
  - atleast_2d(*arys) - View inputs as arrays with at least two dimensions.
  - atleast_3d(*arys) - View inputs as arrays with at least three dimensions.
  - Broadcast - Produce an object that mimics broadcasting.
  - broadcast_to(array, shape[, subok]) - Broadcast an array to a new shape.
  - broadcast_arrays(*args[, subok]) - Broadcast any number of arrays against each other.
  - expand_dims(a, axis) - Expand the shape of an array.
  - squeeze(a[, axis]) - Remove single-dimensional entries from the shape of an array.

Introduction to ‘numpy’

- Changing kind of array
  - asarray(a[, dtype, order]) - Convert the input to an array.
  - asanyarray(a[, dtype, order]) - Convert the input to an ndarray, but pass ndarray subclasses through.
  - asmatrix(data[, dtype]) - Interpret the input as a matrix.
  - asfarray(a[, dtype]) - Return an array converted to a float type.
  - asfortranarray(a[, dtype]) - Return an array (ndim >= 1) laid out in Fortran order in memory.
  - ascontiguousarray(a[, dtype]) - Return a contiguous array (ndim >= 1) in memory (C order).
  - asarray_chkfinite(a[, dtype, order]) - Convert the input to an array, checking for NaNs or Infs.
  - asscalar(a) - Convert an array of size 1 to its scalar equivalent.
  - require(a[, dtype, requirements]) - Return an ndarray of the provided type that satisfies requirements.

Introduction to ‘numpy’

- Joining arrays
  - concatenate([axis, out]) - Join a sequence of arrays along an existing axis.
  - stack(arrays[, axis, out]) - Join a sequence of arrays along a new axis.
  - block(arrays) - Assemble an nd-array from nested lists of blocks.
  - vstack(tup) - Stack arrays in sequence vertically (row wise).
  - hstack(tup) - Stack arrays in sequence horizontally (column wise).
  - dstack(tup) - Stack arrays in sequence depth wise (along third axis).
  - column_stack(tup) - Stack 1-D arrays as columns into a 2-D array.

Introduction to ‘numpy’

- Splitting arrays
  - split(ary, indices_or_sections[, axis]) - Split an array into multiple sub-arrays as views into ary.
  - array_split(ary, indices_or_sections[, axis]) - Split an array into multiple sub-arrays.
  - dsplit(ary, indices_or_sections) - Split array into multiple sub-arrays along the 3rd axis (depth).
  - hsplit(ary, indices_or_sections) - Split an array into multiple sub-arrays horizontally (column-wise).
  - vsplit(ary, indices_or_sections) - Split an array into multiple sub-arrays vertically (row-wise).

Introduction to ‘numpy’

- Tiling arrays
  - tile(A, reps) - Construct an array by repeating A the number of times given by reps.
  - repeat(a, repeats[, axis]) - Repeat elements of an array.

Introduction to ‘numpy’

- Adding and removing elements
  - delete(arr, obj[, axis]) - Return a new array with sub-arrays along an axis deleted.
  - insert(arr, obj, values[, axis]) - Insert values along the given axis before the given indices.
  - append(arr, values[, axis]) - Append values to the end of an array.
  - resize(a, new_shape) - Return a new array with the specified shape.
  - trim_zeros(filt[, trim]) - Trim the leading and/or trailing zeros from a 1-D array or sequence.
  - unique(ar[, return_index, return_inverse, …]) - Find the unique elements of an array.

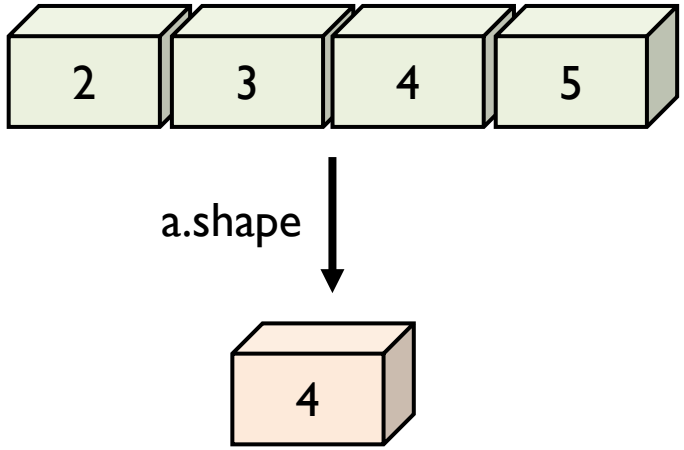
Introduction to ‘numpy’

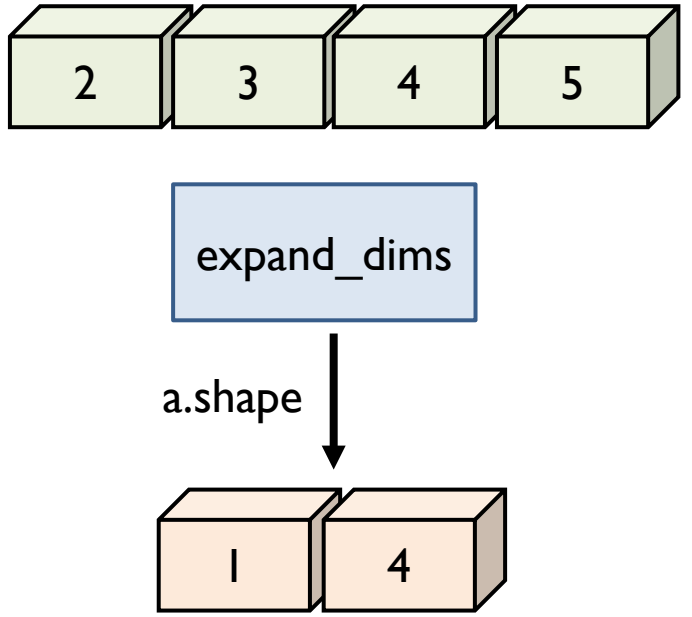
- Rearranging elements
  - flip(m[, axis]) - Reverse the order of elements in an array along the given axis.
  - fliplr(m) – Flip array in the left/right direction.
  - flipud(m) - Flip array in the up/down direction.
  - reshape(a, newshape[, order]) - Gives a new shape to an array without changing its data.
  - roll(a, shift[, axis]) - Roll array elements along a given axis.
  - rot90(m[, k, axes]) - Rotate an array by 90 degrees in the plane specified by axes.

Dimension specification

Dimension specification

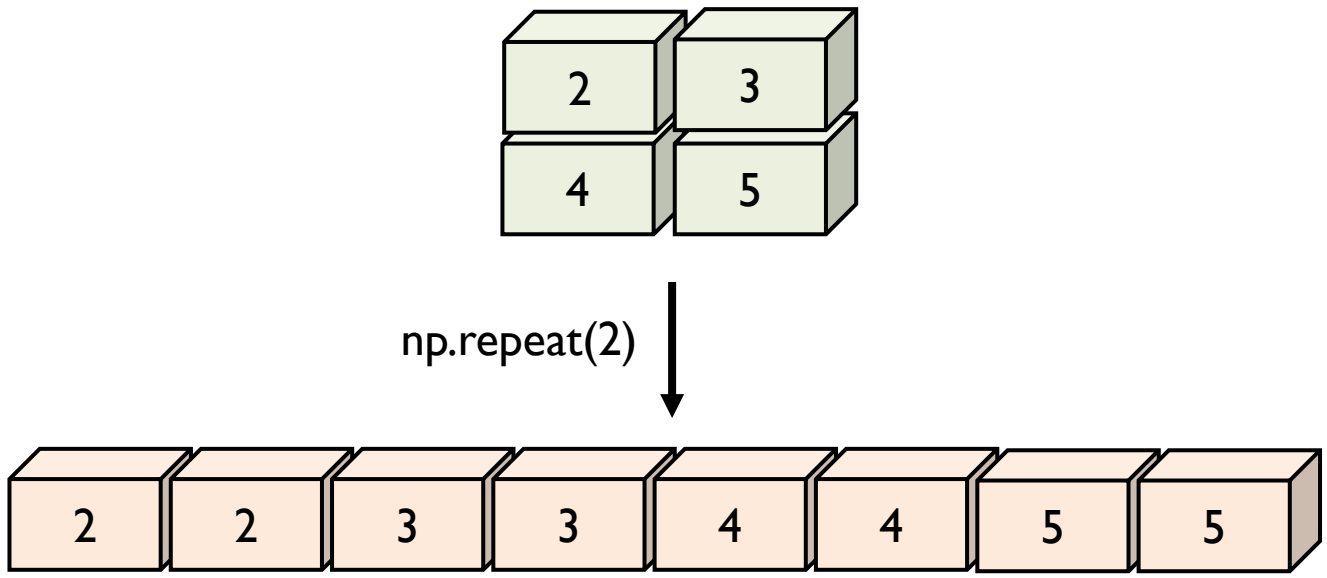
- expand_dims

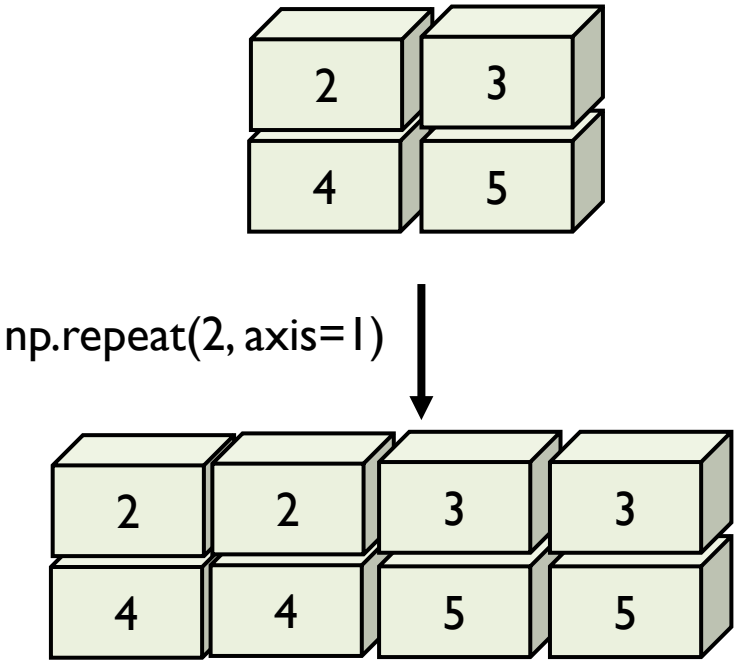




Dimension specification

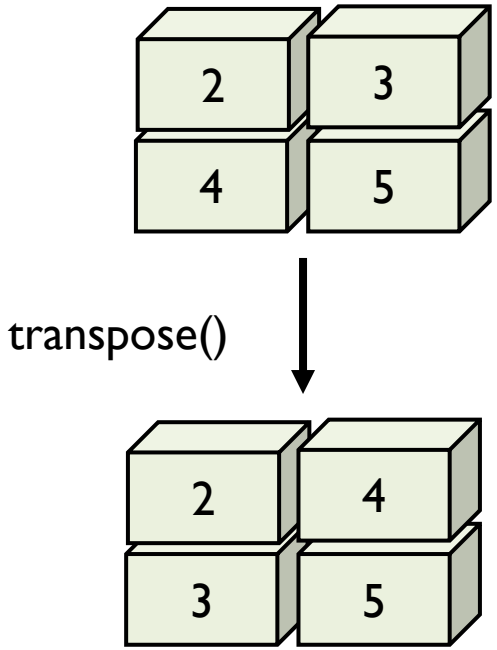
- repeat





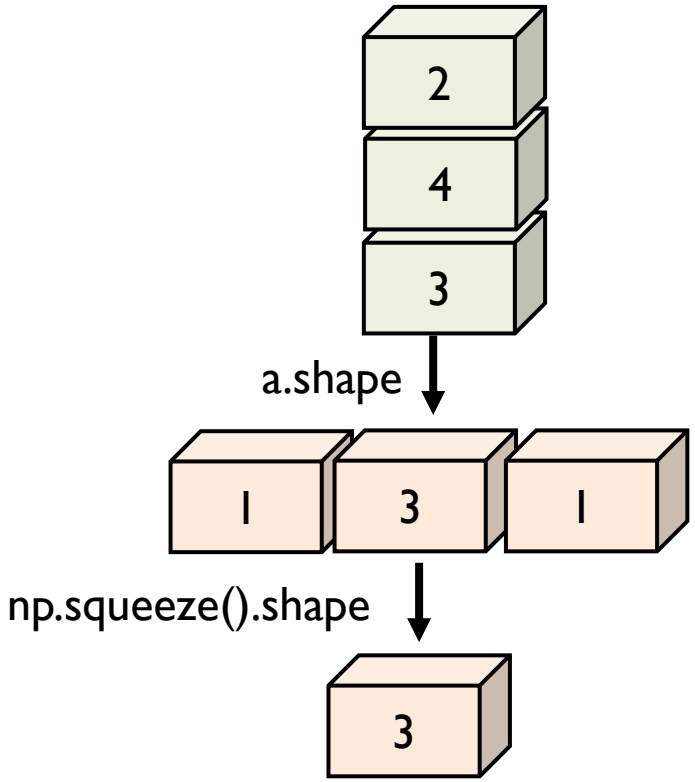
Dimension specification

- transpose()



Dimension specification

- squeeze()



Matrix operations

Matrix operations

- Matrix and vector products
  - dot(a, b[, out]) - Dot product of two arrays.
  - linalg.multi_dot(arrays, *[, out]) - Compute the dot product of two or more arrays in a single function call, while automatically selecting the fastest evaluation order.
  - vdot(a, b) - Return the dot product of two vectors.
  - inner(a, b) - Inner product of two arrays.
  - outer(a, b[, out]) - Compute the outer product of two vectors.
  - matmul(x1, x2, /[, out, casting, order, …]) - Matrix product of two arrays.
  - tensordot(a, b[, axes]) - Compute tensor dot product along specified axes.
  - einsum(subscripts, *operands[, out, dtype, …]) - Evaluates the Einstein summation convention on the operands.
  - einsum_path(subscripts, *operands[, optimize]) - Evaluates the lowest cost contraction order for an einsum expression by considering the creation of intermediate arrays.
  - linalg.matrix_power(a, n) - Raise a square matrix to the (integer) power n.
  - kron(a, b) - Kronecker product of two arrays.

Matrix operations

- Solving equations and inverting matrices
  - linalg.solve(a, b) - Solve a linear matrix equation, or system of linear scalar equations.
  - linalg.tensorsolve(a, b[, axes]) - Solve the tensor equation a x = b for x.
  - linalg.lstsq(a, b[, rcond]) - Return the least-squares solution to a linear matrix equation.
  - linalg.inv(a) - Compute the (multiplicative) inverse of a matrix.
  - linalg.pinv(a[, rcond, hermitian]) - Compute the (Moore-Penrose) pseudo-inverse of a matrix.
  - linalg.tensorinv(a[, ind]) - Compute the ‘inverse’ of an N-dimensional array.