#_ important NumPy Operations [+100]

Basics & Creation:

- `np.array(list)`: Create an array.
- `np.zeros(shape)`: Create an array filled with zeros.
- `np.ones(shape)`: Create an array filled with ones.
- `np.empty(shape)`: Create an uninitialized array.
- `np.arange(start, stop, step)`: Create an array with a range of numbers.
- `np.linspace(start, stop, num)`: Create an array with evenly spaced numbers.

Array Manipulations:

- `np.concatenate((a1, a2, ...), axis)`: Join a sequence of arrays along an existing axis.
- `np.vstack((a, b))`: Stack arrays vertically.
- `np.hstack((a, b))`: Stack arrays horizontally.
- `np.split(ary, indices)`: Split an array into multiple sub-arrays.
- `np.flip(m, axis)`: Reverse the order of elements in an array along the given axis.
- `np.roll(a, shift)`: Roll array elements along a specified axis.
- `np.rot90(m, k=1)`: Rotate an array by 90 degrees in the plane specified by axes.

Input **and Output**:

- `np.save(file, arr)`: Save an array to a binary file in NumPy `.npy` format.
- `np.load(file)`: Load arrays from `.npy` file.
- `np.savetxt(fname, X)`: Save an array to a text file.
- `np.loadtxt(fname)`: Load data from a text file.

Attributes:

- `arr.shape`: Shape of the array.
- `arr.ndim`: Number of array dimensions.

By: Waleed Mousa

- `arr.size`: Number of elements in the array.
- `arr.dtype`: Data type of the array.

Reshaping & Rearranging:

- `arr.reshape(shape)`: Reshape an array.
- `arr.ravel()`: Flatten the array.
- `arr.transpose()`: Transpose the array.
- `arr.swapaxes(axis1, axis2)`: Swap axes of an array.

Indexing & Slicing:

- `arr[index]`: Get element at a specific index.
- `arr[start:stop:step]`: Slice an array.
- `arr[condition]`: Boolean indexing.

Math **Operations**:

- `np.add(arr1, arr2)`: Add arrays element-wise.
- `np.subtract(arr1, arr2)`: Subtract arrays element-wise.
- `np.multiply(arr1, arr2)`: Multiply arrays element-wise.
- `np.divide(arr1, arr2)`: Divide arrays element-wise.

Linear Algebra:

- `np.dot(arr1, arr2)`: Dot product of two arrays.
- `np.linalg.inv(matrix)`: Inverse of a matrix.
- `np.linalg.det(matrix)`: Determinant of a matrix.
- `np.linalg.eig(matrix)`: Eigenvalues and eigenvectors of a matrix.

Aggregations:

- `np.sum(arr)`: Sum of array elements.
- `np.min(arr)`: Minimum value in array.
- `np.max(arr)`: Maximum value in array.
- `np.mean(arr)`: Mean of array elements.
- `np.median(arr)`: Median of array elements.
- `np.std(arr)`: Standard deviation.

Comparison & Logic:

- `np.equal(arr1, arr2)`: Element-wise comparison.

By: Waleed Mousa

- `np.array_equal(arr1, arr2)`: Array-wise comparison.
- `np.logical_and(arr1, arr2)`: Element-wise logical 'and'.
- `np.logical_or(arr1, arr2)`: Element-wise logical 'or'.

Strides and Broadcasting:

- `np.broadcast_arrays(*args)`: Broadcast any number of arrays against each other.
- `np.broadcast_to(array, shape)`: Broadcast an array to a new shape.
- `np.expand_dims(a, axis)`: Expand the shape of an array.
- `np.squeeze(a, axis)`: Remove single-dimensional entries from the shape of an array.

Window functions for FFTs:

- `np.hanning(M)`: Return the Hanning window.
- `np.hamming(M)`: Return the Hamming window.
- `np.blackman(M)`: Return the Blackman window.
- `np.bartlett(M)`: Return the Bartlett window.

Date Functions:

- `np.datetime64()`: Create a datetime64 object from a string.
- `np.is_busday(dates)`: Is a business day (weekday).
- `np.busday_offset(dates, offsets)`: Apply offsets to business days.
- `np.busday_count(begindates, enddates)`: Counts the number of valid days between begindates and enddates.

Financial Functions:

- `np.fv(rate, nper, pmt, pv)`: Compute the future value.
- `np.pmt(rate, nper, pv, fv)`: Compute the payment against loan principal plus interest.
- `np.ipmt(rate, nper, per, pv, fv)`: Compute the interest portion of a payment.
- `np.ppmt(rate, nper, per, pv, fv)`: Compute the payment against loan principal.

By: Waleed Mousa

- `np.nper(rate, pmt, pv, fv)`: Compute the number of periodic payments.
- `np.irr(values)`: Return the Internal Rate of Return (IRR).
- `np.npv(rate, values)`: Returns the NPV (Net Present Value) of a cash flow series.

Masked arrays:

- `np.ma.masked_array(data, mask)`: Masked array.
- `np.ma.masked_where(condition, a)`: Mask an array where a condition is met.
- `np.ma.masked_greater(x, value)`: Mask an array where greater than a given value.
- `np.ma.masked_less(x, value)`: Mask an array where less than a given value.
- `np.ma.masked_inside(x, v1, v2)`: Mask an array inside a given interval.

Binary operations:

- `np.bitwise_and(x1, x2)`: Compute the bit-wise AND of two arrays element-wise.
- `np.bitwise_or(x1, x2)`: Compute the bit-wise OR of two arrays element-wise.
- `np.bitwise_xor(x1, x2)`: Compute the bit-wise XOR of two arrays element-wise.
- `np.left_shift(x1, x2)`: Shift the bits of an integer to the left.
- `np.right_shift(x1, x2)`: Shift the bits of an integer to the right.

Complex Numbers:

- `np.real(complex_arr)`: Real part of complex array.
- `np.imag(complex_arr)`: Imaginary part of complex array.

Polynomials:

- `np.poly1d(coefficients)`: Create a polynomial function.
- `np.roots(polynomial)`: Find the roots of a polynomial.

By: Waleed Mousa

Trigonometry:

- `np.sin(arr)`: Sine.
- `np.cos(arr)`: Cosine.
- `np.tan(arr)`: Tangent.

Exponents & Logarithms:

- `np.exp(arr)`: Exponential.
- `np.log(arr)`: Natural logarithm.
- `np.log10(arr)`: Base-10 logarithm.

Rounding:

- `np.around(arr, decimals)`: Round to desired precision.
- `np.floor(arr)`: Round down.
- `np.ceil(arr)`: Round up.

Special Functions:

- `np.sinh(arr)`: Hyperbolic sine.
- `np.cosh(arr)`: Hyperbolic cosine.
- `np.tanh(arr)`: Hyperbolic tangent.

Set Operations:

- `np.unique(arr)`: Unique elements of an array.
- `np.intersect1d(arr1, arr2)`: Common values between two arrays.
- `np.union1d(arr1, arr2)`: Union of two arrays.

Stacking & Splitting:

- `np.hstack(tup)`: Stack arrays horizontally.
- `np.vstack(tup)`: Stack arrays vertically.
- `np.split(arr, indices)`: Split an array.

Random Numbers:

- `np.random.rand(shape)`: Random numbers between 0 and 1.
- `np.random.randn(shape)`: Random samples from a standard normal distribution.
- `np.random.randint(low, high, size)`: Random integers.

By: Waleed Mousa

Others:

- `np.full(shape, fill_value)`: Create an array filled with a specific value.
- `np.tile(arr, reps)`: Construct an array by repeating another array.
- `np.repeat(arr, repeats)`: Repeat elements of an array.
- `np.clip(arr, min, max)`: Clip values outside a given interval.
- `np.where(condition, x, y)`: Return elements based on condition.
- `np.pad(arr, pad_width)`: Pad an array.
- `np.searchsorted(arr, values)`: Find indices where elements should be inserted.
- `np.sort(arr)`: Sort an array.
- `np.argsort(arr)`: Indices that would sort an array.
- `np.bincount(arr)`: Count occurrences of values.
- `np.extract(condition, arr)`: Extract elements based on a condition.
- `np.convolve(a, v)`: Returns the discrete, linear convolution of two sequences.
- `np.correlate(a, v)`: Discrete, linear correlation of two sequences.
- `np.histogram(arr, bins)`: Compute histogram of data.
- `np.eye(N)`: Return a 2-D array with ones on the diagonal and zeros elsewhere.
- `np.diag(arr)`: Extract or construct a diagonal array.
- `np.outer(a, b)`: Compute the outer product of two vectors.
- `np.inner(a, b)`: Compute the inner product of two arrays.
- `np.kron(a, b)`: Compute the Kronecker product of two arrays.
- `np.cross(a, b)`: Return the cross product of two arrays.
- `np.gradient(f)`: Return the gradient of an N-dimensional array.
- `np.meshgrid(*xi)`: Return coordinate matrices from coordinate vectors.
- `np.percentile(a, q)`: Compute the q-th percentile of the data along the specified axis.
- `np.quantile(a, q)`: Compute the q-th quantile of the data along the specified axis.
- `np.cov(m)`: Estimate a covariance matrix.
- `np.corrcoef(x)`: Return Pearson product-moment correlation coefficients.

By: Waleed Mousa

- `np.diff(a)`: Calculate the n-th discrete difference along the given axis.
- `np.ediff1d(ary)`: The differences between consecutive elements of an array.
- `np.i0(x)`: Modified Bessel function of the first kind, order 0.
- `np.sinc(x)`: Return the normalized sinc function.
- `np.fft.fft(a)`: Compute the one-dimensional discrete Fourier Transform.
- `np.fft.ifft(a)`: Compute the one-dimensional inverse discrete Fourier Transform.
- `np.trapz(y)`: Integrate along the given axis using the composite trapezoidal rule.
- `np.linalg.norm(x)`: Matrix or vector norm.
- `np.linalg.cholesky(a)`: Cholesky decomposition.
- `np.linalg.qr(a)`: Compute the qr factorization of a matrix.
- `np.linalg.svd(a)`: Singular Value Decomposition.
- `np.linalg.eigh(a)`: Return the eigenvalues and eigenvectors of a complex Hermitian (conjugate symmetric) or a real symmetric matrix.
- `np.linalg.lstsq(a, b)`: Return the least-squares solution to a linear matrix equation.
- `np.linalg.matrix_rank(M)`: Return matrix rank of array using SVD method.
- `np.linalg.solve(a, b)`: Solve a linear matrix equation, or system of linear scalar equations.
- `np.linalg.tensorsolve(a, b)`: Solve the tensor equation `a x = b` for x.
- `np.linalg.tensorinv(a)`: Compute the 'inverse' of an N-dimensional array.
- `np.apply_along_axis(func1d, axis, arr)`: Apply a function to 1-D slices along the given axis.
- `np.vectorize(pyfunc)`: Return a vectorized version of the provided function.
- `np.fromfunction(function, shape)`: Construct an array by executing a function over each coordinate.

By: Waleed Mousa

- `np.fromiter(iterable, dtype)`: Create a new 1-dimensional array from an iterable object.
- `np.isclose(a, b)`: Returns a boolean array where two arrays are element-wise equal within a tolerance.
- `np.allclose(a, b)`: Returns `True` if two arrays are element-wise equal within a tolerance.
- `np.fix(x)`: Round to nearest integer towards zero.
- `np.iscomplex(x)`: Returns a bool array, where `True` if input element is complex.
- `np.isreal(x)`: Returns a bool array, where `True` if input element is real.
- `np.nan_to_num(x)`: Replace NaN with zero and infinity with large finite numbers.