Numpy

- Numpy is a python pcackge used for computing
- Numpy means: **Number python or Numerical python**
- Numpy able to create multi dimensional arrays
- Array also kind of a list
- Numpy arrays one step ahead better than list
- In maths we can have matrix vectors etc can perform by numpy

- the basic array is **list**
- next level arrays is **numpy**
- next level arrays is **tensors: pytorch and tensorflow**
- list does not have any dimensions
- so for the ML, DL and NLP we will use numpy arrays only not list

In [1]:
```
import numpy as np
```

In [2]:
```
np.__version__
```

Out[2]:
```
'1.26.4'
```

In [ ]:
```
# packages will update
# versions different
# packages has functions
# random has randint === might be
# warning if that can be deprecated in the future
```

In [4]:
```
import sys
sys.version
```

Out[4]:
```
'3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.191
6 64 bit (AMD64)]'
```

numpy array

In [7]:
```
list1=[1,2,3,4]
type(list1)
list1   # scalar
```

Out[7]:
```
[1, 2, 3, 4]
```

In [8]:
```
arr1=np.array([1,2,3,4]) # vector
arr1
```

Out[8]:
```
array([1, 2, 3, 4])
```

In [9]:
```
type(arr1)
```

Out[9]:
```
numpy.ndarray
```

In [ ]:
```
# numpy n dimensional array
```

shape-dimension

In [ ]:
```
# step-1: import the package
#          dir(np)   numpy related functions
# step-2: create the array
# step-3: dir(array) array related functions
```

In [10]:
```
import numpy as np
dir(np)
```

Out[10]:
```
['ALLOW_THREADS',
 'BUFSIZE',
 'CLIP',
 'DataSource',
 'ERR_CALL',
 'ERR_DEFAULT',
 'ERR_IGNORE',
 'ERR_LOG',
 'ERR_PRINT',
 'ERR_RAISE',
 'ERR_WARN',
 'FLOATING_POINT_SUPPORT',
 'FPE_DIVIDEBYZERO',
 'FPE_INVALID',
 'FPE_OVERFLOW',
 'FPE_UNDERFLOW',
 'False_',
 'Inf',
 'Infinity',
 'MAXDIMS',
 'MAY_SHARE_BOUNDS',
 'MAY_SHARE_EXACT',
 'NAN',
 'NINF',
 'NZERO',
 'NaN',
 'PINF',
 'PZERO',
 'RAISE',
 'RankWarning',
 'SHIFT_DIVIDEBYZERO',
 'SHIFT_INVALID',
 'SHIFT_OVERFLOW',
 'SHIFT_UNDERFLOW',
 'ScalarType',
 'True_',
 'UFUNC_BUFSIZE_DEFAULT',
 'UFUNC_PYVALS_NAME',
 'WRAP',
 '_CopyMode',
 '_NoValue',
 '_UFUNC_API',
 '__NUMPY_SETUP__',
 '__all__',
 '__builtins__',
 '__cached__',
 '__config__',
 '__deprecated_attrs__',
 '__dir__',
 '__doc__',
 '__expired_functions__',
 '__file__',
 '__former_attrs__',
 '__future_scalars__',
 '__getattr__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
```

```
'__version__',
'_add_newdoc_ufunc',
'_builtins',
'_distributor_init',
'_financial_names',
'_get_promotion_state',
'_globals',
'_int_extended_msg',
'_mat',
'_no_nep50_warning',
'_pyinstaller_hooks_dir',
'_pytesttester',
'_set_promotion_state',
'_specific_msg',
'_typing',
'_using_numpy2_behavior',
'_utils',
'abs',
'absolute',
'add',
'add_docstring',
'add_newdoc',
'add_newdoc_ufunc',
'all',
'allclose',
'alltrue',
'amax',
'amin',
'angle',
'any',
'append',
'apply_along_axis',
'apply_over_axes',
'arange',
'arccos',
'arccosh',
'arcsin',
'arcsinh',
'arctan',
'arctan2',
'arctanh',
'argmax',
'argmin',
'argpartition',
'argsort',
'argwhere',
'around',
'array',
'array2string',
'array_equal',
'array_equiv',
'array_repr',
'array_split',
'array_str',
'asanyarray',
'asarray',
'asarray_chkfinite',
'ascontiguousarray',
'asfarray',
'asfortranarray',
```

```
'asmatrix',
'atleast_1d',
'atleast_2d',
'atleast_3d',
'average',
'bartlett',
'base_repr',
'binary_repr',
'bincount',
'bitwise_and',
'bitwise_not',
'bitwise_or',
'bitwise_xor',
'blackman',
'block',
'bmat',
'bool_',
'broadcast',
'broadcast_arrays',
'broadcast_shapes',
'broadcast_to',
'busday_count',
'busday_offset',
'busdaycalendar',
'byte',
'byte_bounds',
'bytes_',
'c_',
'can_cast',
'cast',
'cbrt',
'cdouble',
'ceil',
'cfloat',
'char',
'character',
'chararray',
'choose',
'clip',
'clongdouble',
'clongfloat',
'column_stack',
'common_type',
'compare_chararrays',
'compat',
'complex128',
'complex64',
'complex_',
'complexfloating',
'compress',
'concatenate',
'conj',
'conjugate',
'convolve',
'copy',
'copysign',
'copyto',
'corrcoef',
'correlate',
'cos',
```

```
'cosh',
'count_nonzero',
'cov',
'cross',
'csingle',
'ctypeslib',
'cumprod',
'cumproduct',
'cumsum',
'datetime64',
'datetime_as_string',
'datetime_data',
'deg2rad',
'degrees',
'delete',
'deprecate',
'deprecate_with_doc',
'diag',
'diag_indices',
'diag_indices_from',
'diagflat',
'diagonal',
'diff',
'digitize',
'disp',
'divide',
'divmod',
'dot',
'double',
'dsplit',
'dstack',
'dtype',
'dtypes',
'e',
'ediff1d',
'einsum',
'einsum_path',
'emath',
'empty',
'empty_like',
'equal',
'errstate',
'euler_gamma',
'exceptions',
'exp',
'exp2',
'expand_dims',
'expm1',
'extract',
'eye',
'fabs',
'fastCopyAndTranspose',
'fft',
'fill_diagonal',
'find_common_type',
'finfo',
'fix',
'flatiter',
'flatnonzero',
'flexible',
```

```
'flip',
'fliplr',
'flipud',
'float16',
'float32',
'float64',
'float_',
'float_power',
'floating',
'floor',
'floor_divide',
'fmax',
'fmin',
'fmod',
'format_float_positional',
'format_float_scientific',
'format_parser',
'frexp',
'from_dlpack',
'frombuffer',
'fromfile',
'fromfunction',
'fromiter',
'frompyfunc',
'fromregex',
'fromstring',
'full',
'full_like',
'gcd',
'generic',
'genfromtxt',
'geomspace',
'get_array_wrap',
'get_include',
'get_printoptions',
'getbufsize',
'geterr',
'geterrcall',
'geterrobj',
'gradient',
'greater',
'greater_equal',
'half',
'hamming',
'hanning',
'heaviside',
'histogram',
'histogram2d',
'histogram_bin_edges',
'histogramdd',
'hsplit',
'hstack',
'hypot',
'i0',
'identity',
'iinfo',
'imag',
'in1d',
'index_exp',
'indices',
```

```
'inexact',
'inf',
'info',
'infty',
'inner',
'insert',
'int16',
'int32',
'int64',
'int8',
'int_',
'intc',
'integer',
'interp',
'intersect1d',
'intp',
'invert',
'is_busday',
'isclose',
'iscomplex',
'iscomplexobj',
'isfinite',
'isfortran',
'isin',
'isinf',
'isnan',
'isnat',
'isneginf',
'isposinf',
'isreal',
'isrealobj',
'isscalar',
'issctype',
'issubclass_',
'issubdtype',
'issubsctype',
'iterable',
'ix_',
'kaiser',
'kron',
'lcm',
'ldexp',
'left_shift',
'less',
'less_equal',
'lexsort',
'lib',
'linalg',
'linspace',
'little_endian',
'load',
'loadtxt',
'log',
'log10',
'log1p',
'log2',
'logaddexp',
'logaddexp2',
'logical_and',
'logical_not',
```

```
'logical_or',
'logical_xor',
'logspace',
'longcomplex',
'longdouble',
'longfloat',
'longlong',
'lookfor',
'ma',
'mask_indices',
'mat',
'matmul',
'matrix',
'max',
'maximum',
'maximum_sctype',
'may_share_memory',
'mean',
'median',
'memmap',
'meshgrid',
'mgrid',
'min',
'min_scalar_type',
'minimum',
'mintypecode',
'mod',
'modf',
'moveaxis',
'msort',
'multiply',
'nan',
'nan_to_num',
'nanargmax',
'nanargmin',
'nancumprod',
'nancumsum',
'nanmax',
'nanmean',
'nanmedian',
'nanmin',
'nanpercentile',
'nanprod',
'nanquantile',
'nanstd',
'nansum',
'nanvar',
'nbytes',
'ndarray',
'ndenumerate',
'ndim',
'ndindex',
'nditer',
'negative',
'nested_iters',
'newaxis',
'nextafter',
'nonzero',
'not_equal',
'numarray',
```

```
'number',
'obj2sctype',
'object_',
'ogrid',
'oldnumeric',
'ones',
'ones_like',
'outer',
'packbits',
'pad',
'partition',
'percentile',
'pi',
'piecewise',
'place',
'poly',
'poly1d',
'polyadd',
'polyder',
'polydiv',
'polyfit',
'polyint',
'polymul',
'polynomial',
'polysub',
'polyval',
'positive',
'power',
'printoptions',
'prod',
'product',
'promote_types',
'ptp',
'put',
'put_along_axis',
'putmask',
'quantile',
'r_',
'rad2deg',
'radians',
'random',
'ravel',
'ravel_multi_index',
'real',
'real_if_close',
'rec',
'recarray',
'recfromcsv',
'recfromtxt',
'reciprocal',
'record',
'remainder',
'repeat',
'require',
'reshape',
'resize',
'result_type',
'right_shift',
'rint',
'roll',
```

```
'rollaxis',
'roots',
'rot90',
'round',
'round_',
'row_stack',
's_',
'safe_eval',
'save',
'savetxt',
'savez',
'savez_compressed',
'sctype2char',
'sctypeDict',
'sctypes',
'searchsorted',
'select',
'set_numeric_ops',
'set_printoptions',
'set_string_function',
'setbufsize',
'setdiff1d',
'seterr',
'seterrcall',
'seterrobj',
'setxor1d',
'shape',
'shares_memory',
'short',
'show_config',
'show_runtime',
'sign',
'signbit',
'signedinteger',
'sin',
'sinc',
'single',
'singlecomplex',
'sinh',
'size',
'sometrue',
'sort',
'sort_complex',
'source',
'spacing',
'split',
'sqrt',
'square',
'squeeze',
'stack',
'std',
'str_',
'string_',
'subtract',
'sum',
'swapaxes',
'take',
'take_along_axis',
'tan',
'tanh',
```

```
 'tensordot',
 'test',
 'testing',
 'tile',
 'timedelta64',
 'trace',
 'tracemalloc_domain',
 'transpose',
 'trapz',
 'tri',
 'tril',
 'tril_indices',
 'tril_indices_from',
 'trim_zeros',
 'triu',
 'triu_indices',
 'triu_indices_from',
 'true_divide',
 'trunc',
 'typecodes',
 'typename',
 'ubyte',
 'ufunc',
 'uint',
 'uint16',
 'uint32',
 'uint64',
 'uint8',
 'uintc',
 'uintp',
 'ulonglong',
 'unicode_',
 'union1d',
 'unique',
 'unpackbits',
 'unravel_index',
 'unsignedinteger',
 'unwrap',
 'ushort',
 'vander',
 'var',
 'vdot',
 'vectorize',
 'version',
 'void',
 'vsplit',
 'vstack',
 'where',
 'who',
 'zeros',
 'zeros_like']
```

In [11]:
```
np.array([10,20,30,40])
```

Out[11]:
```
array([10, 20, 30, 40])
```

In [12]:
```
list1=[10,20,30,40]
np.max(list1)
```

Out[12]: 40

In [13]:
```
np.mean(list1)
```

Out[13]: 25.0

In [16]:
```
sum(list1)/len(list1)
```

Out[16]: 25.0

In [17]:
```
import numpy as np
arr1=np.array([10,20,30,40])
arr1.shape
```

Out[17]: (4,)

In [18]:
```
arr1.ndim
```

Out[18]: 1

In [21]:
```
l1=[10,20,30,40]
arr1=np.array(l1)
print("dim:",arr1.ndim)
print('shape:',arr1.shape)
```

```
dim: 1
shape: (4,)
```

In [24]:
```
l1=['A','B','C']   # 'A'  ==== num
l2=[10,20,30]     # 10  ===== '10'
arr1=np.array([l1,l2])
arr1
```

Out[24]:
```
array([['A', 'B', 'C'],
       ['10', '20', '30']], dtype='<U11')
```

In [25]:
```
arr1.shape  # its look like matrix
# 2 rows and 3 columns
```

Out[25]: (2, 3)

In [26]:
```
arr1.ndim
```

Out[26]: 2

indexing

In [28]:
```
arr1[0,0]
# or
arr1[0][0]
```

Out[28]: 'A'

Note

- Numpy always gives one data type

- when we provide strings and integers it combinely gives as string data type only

In [29]:
```
arr1[0,0],arr1[0,1],arr1[0,2]
```

Out[29]:
```
('A', 'B', 'C')
```

1D

In [31]:
```
arr1=np.array([10,20,30])
print("arr1:",arr1)
print('shape:',arr1.shape)
print('dim:',arr1.ndim)
```

```
arr1: [10 20 30]
shape: (3,)
dim: 1
```

2D

In [32]:
```
arr2=np.array([ [10,20,30],
                [100,200,300]
              ])
print("arr1:",arr2)
print('shape:',arr2.shape)
print('dim:',arr2.ndim)
```

```
arr1: [[ 10  20  30]
 [100 200 300]]
shape: (2, 3)
dim: 2
```

3D

In [33]:
```
arr3=np.array([[[10,20,30],
                [100,200,300]
               ]])
print("arr1:",arr3)
print('shape:',arr3.shape)
print('dim:',arr3.ndim)
```

```
arr1: [[[ 10  20  30]
  [100 200 300]]]
shape: (1, 2, 3)
dim: 3
```

In [ ]:
```
one layer available
inside one layer how many list are avilable two list
inside each list how many elements
```

4D

In [34]:
```
arr4=np.array([[[[10,20,30],
                 [100,200,300]
                ]]])
print("arr4:",arr4)
print('shape:',arr4.shape)
print('dim:',arr4.ndim)
```

```
arr4: [[[[ 10  20  30]
   [100 200 300]]]]
shape: (1, 1, 2, 3)
dim: 4
```

In [ ]: