Python Libraries

Top 3 libraries

```
- numpy
- pandas
- matplotlib
```

Data Science

```
- Organizing,processing,analysing the huge amount of data
- Applying data visualization
- extracting the data from the source
- we have 20 python libraries for data science
```

Numpy

```
- Numpy means numerical python
- It is a fundamental package for scienfic computing in python
- numpy the python library
- It is used to create multi dimensional arrays for fast operations
```

How to install numpy

```
- pip install numpy
```

use the following import conversion to access numpy library

```
- import numpy
```

```python
import numpy as np

# 1 dim array
array1d = np.array([1,2,3,4,5])
print(array1d)
print(type(array1d))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

```
# 2 dim array
array2d = np.array([[1,2,3],[6,7,8]])
print(array2d)
print(type(array2d))
```

```
[[1 2 3]
 [6 7 8]]
<class 'numpy.ndarray'>
```

```
# checking numpy version
print(np.__version__)
```

```
1.26.4
```

```
# 0-d array
a = np.array(23)
print(a)
```

```
23
```

```
# conversion of list into an array
li = [4,5,6,'a','p']
print(li,type(li))
ar = np.array(li)
print(ar,type(ar))
```

```
[4, 5, 6, 'a', 'p'] <class 'list'>
['4' '5' '6' 'a' 'p'] <class 'numpy.ndarray'>
```

```
# 3-d array
array3d = np.array([[[1,2,3],[4,5,6]],[[6,7,8],[5,7,3]]])
print(array3d)
print(type(array3d))
```

```
[[[1 2 3]
  [4 5 6]]

 [[6 7 8]
  [5 7 3]]]
<class 'numpy.ndarray'>
```

```
print(array2d)
```

```
[[1 2 3]
 [6 7 8]]
```

```
# builtin functions in numpy
print(array2d.dtype)
print(array2d.shape)
```

```
print(array2d.size)
print(array2d.ndim)
print(array3d.ndim)
```

```
int32
(2, 3)
6
2
3
```

```
print(dir(np),end=' ')
```

```
['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']
```

```
# using range()
a = np.array(range(10))
print(a)
print(a.ndim)
```

```
[0 1 2 3 4 5 6 7 8 9]
1
```

```
b = np.array(range(1,20,3))
print(b)
```

```
[ 1  4  7 10 13 16 19]
```

```
# arange()
a1 = np.arange(1,10)
print(a1)
```

```
[1 2 3 4 5 6 7 8 9]
```

```
b1 = np.arange(1,20,3)
print(b1)
```

```
[ 1  4  7 10 13 16 19]
```

```
# using arange function we can store float values also
c1 = np.arange(1.5,10.5)
print(c1)
```

```
[1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5]
```

```
# reshape()
x = np.arange(20)
print(x.reshape(5,4))
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]]
```

```
# one's method
a = np.ones((3,3))
a
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

```
# one's method
a = np.ones((3,3),dtype=int)
a
```

```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]])
```

```
# zero's method
b = np.zeros((3,3))
b
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
# zero's method
b = np.zeros((3,3),dtype=int)
b
```

```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

```
# full()
a = np.full((3,3),18)
a
```

```
array([[18, 18, 18],
       [18, 18, 18],
       [18, 18, 18]])

# print diagnoal
np.diag(a)

array([18, 18, 18])

# full()
a = np.full((3,3),'age')
a

array([['age', 'age', 'age'],
       ['age', 'age', 'age'],
       ['age', 'age', 'age']], dtype='<U3')

#  random()
r = np.random.randint(1,30)
r

21

s = np.random.randint(1,20,12).reshape(3,4)
s

array([[ 6, 10,  6, 16],
       [ 3,  6, 16,  1],
       [ 7,  9,  1, 15]])

p = np.random.random((6)).reshape(2,3)
p

array([[0.94220961, 0.75360176, 0.90462771],
       [0.39667714, 0.95933138, 0.28486402]])

# randn
f = np.random.randn(2,3)
f

array([[-0.60162204,  0.03316899,  0.91584557],
       [ 1.71801401, -0.46253623,  1.20129047]])

a = np.random.randint(1,30,(10,3))
a

array([[15, 15,  7],
       [20, 16, 19],
       [20,  8, 12],
       [12, 12,  3],
       [ 5, 21, 27],
       [ 8, 17,  9],
```

```
       [16, 19, 23],
       [16,  3, 15],
       [22, 19,  1],
       [21, 11, 23]])
```

```
a[1:5]
```

```
array([[20, 16, 19],
       [20,  8, 12],
       [12, 12,  3],
       [ 5, 21, 27]])
```

```
a[5:]
```

```
array([[ 8, 17,  9],
       [16, 19, 23],
       [16,  3, 15],
       [22, 19,  1],
       [21, 11, 23]])
```

```
a[1::3]
```

```
array([[20, 16, 19],
       [ 5, 21, 27],
       [16,  3, 15]])
```

```
a[::-1]  # reversed matrix
```

```
array([[21, 11, 23],
       [22, 19,  1],
       [16,  3, 15],
       [16, 19, 23],
       [ 8, 17,  9],
       [ 5, 21, 27],
       [12, 12,  3],
       [20,  8, 12],
       [20, 16, 19],
       [15, 15,  7]])
```

```
print(a>5)
```

```
[[ True  True  True]
 [ True  True  True]
 [ True  True  True]
 [ True  True False]
 [False  True  True]
 [ True  True  True]
 [ True  True  True]
 [ True False  True]
 [ True  True False]
 [ True  True  True]]
```

```
print(a<5)
```

```
[[False False False]
 [False False False]
 [False False False]
 [False False  True]
 [False False False]
 [False False False]
 [False False False]
 [False  True False]
 [False False  True]
 [False False False]]
```

```
a[a>5]
```

```
array([15, 15,  7, 20, 16, 19, 20,  8, 12, 12, 12, 21, 27,  8, 17,  9,
16,
       19, 23, 16, 15, 22, 19, 21, 11, 23])
```

```
a[a<15]
```

```
array([ 7,  8, 12, 12, 12,  3,  5,  8,  9,  3,  1, 11])
```

```
#print max and min values in matrix
print(a.max())
print(a.min())
```

```
27
1
```

```
a = np.arange(9).reshape(3,3)
a
```

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

```
b = np.arange(10,19).reshape(3,3)
b
```

```
array([[10, 11, 12],
       [13, 14, 15],
       [16, 17, 18]])
```

```
print(a+b)  # matrix addition
```

```
[[10 12 14]
 [16 18 20]
 [22 24 26]]
```

```
print(a-b) # matrix substraction
```

```
[[-10 -10 -10]
 [-10 -10 -10]
 [-10 -10 -10]]
```

```
print(a%b) # matrix modules
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

```
# matrix multiplication using dot method
# rule :-  1st matrix columns=2nd matrix rows
```

```
a.dot(b)
```

```
array([[ 45,  48,  51],
       [162, 174, 186],
       [279, 300, 321]])
```

```
b.dot(a)
```

```
array([[105, 138, 171],
       [132, 174, 216],
       [159, 210, 261]])
```

```
print(np.mean(a))
print(np.median(a))
```

```
4.0
4.0
```

```
print(a+3)
print(b+5)
```

```
[[ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
[[15 16 17]
 [18 19 20]
 [21 22 23]]
```

```
np.log(2)
```

```
0.6931471805599453
```

```
np.log(10)
```

```
2.302585092994046
```

```
np.log([4,6,7,8,9])
```

```
array([1.38629436, 1.79175947, 1.94591015, 2.07944154, 2.19722458])
```

```
np.exp(4)
```

```
54.598150033144236
```

```
np.exp(3)
```

```
20.085536923187668
```

```
np.exp([3,7,8,1,2,6])
```

```
array([2.00855369e+01, 1.09663316e+03, 2.98095799e+03, 2.71828183e+00,
       7.38905610e+00, 4.03428793e+02])
```

```
np.sin(0)
```

```
0.0
```

```
np.cos(45)
```

```
0.5253219888177297
```

```
np.tan(0)
```

```
0.0
```

References

w3schools.com
javatpoint.com
geeksforgeeks