Numpy

What is Numpy?

- Numpy stands for numerical python
- It is a python library used for working with arrays.
- Numpy is a perfect tool for scientific computing and performing basic and advanced array operations.

Advantages

- Numpy is much faster than the standard python ways to do computations.
- Numpy can handle the vast amount of data.
- Used for scientific computing and data analysis.

In [1]:

```
import numpy as np
```

In [2]:

```
# creating 1D array

a1 = np.array([1,2,3,4,5])
print(a1)
print(type(a1))
print(a1.ndim)
```

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

In [5]:

```
# create 2D array

a2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(a2)
print(a2.ndim)
print(a2.shape)
```

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

In [6]:

```
# create 3D array

a3 = np.array([[[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15],[16,17,18,19,20]]])
print(a3)
```

```
[[[ 1  2  3  4  5]
  [ 6  7  8  9 10]
  [11 12 13 14 15]
  [16 17 18 19 20]]]
```

In [7]:

```
print(a3.ndim)
```

```
3
```

In [9]:

```
print(a3.size)
print(a3.itemsize)
```

```
20
4
```

In [10]:

```
a1 = np.array((1,2,3,4))
print(a1)
print(a1.ndim)
```

```
[1 2 3 4]
1
```

In [12]:

```
# Creating 2D array using tuple of itmes|elements

a2 = np.array(((1,2,3),(4,5,6)))
print(a2)
print(a2.ndim)
```

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

In [17]:

```
# Creating 3D array using tuple of items

a3 = np.array((((1,2,3),(4,5,6),(7,8,9))))
print(a3)
print(a3.ndim)
```

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

Creating an array using range()

- np.array(range(start,end,step))
- np.array(range(start,end,step)).reshape(rows, columns)

In [19]:

```
arr = np.array(range(1,13))
print(arr)
print(arr.ndim)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12]
1
```

In [21]:

```
arr = np.array(range(1,13)).reshape(4,3)
print(arr)
print(arr.ndim)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
2
```

In [24]:

```
print(arr.reshape(1,4,3))
print("==============")
print(arr.reshape(2,2,3))
```

```
[[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]
  [10 11 12]]]
==============
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]
```

In [25]:

```
arr2 = np.array(range(1,13,2))
```

In [26]:

```
print(arr2)
```

```
[ 1  3  5  7  9 11]
```

In [27]:

```
print(arr2.reshape(2,3))
```

```
[[ 1  3  5]
 [ 7  9 11]]
```

In [28]:

```

print(arr2.reshape(1,3,2))
```

```
[[[ 1  3]
  [ 5  7]
  [ 9 11]]]
```

Creating an array using arange()

Syntax: np.arange(start, end, step)

In [29]:

```
arr = np.arange(8)
print(arr)
```

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

In [30]:

```
print(arr.reshape(4,2))
```

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

In [31]:

```
print(arr.reshape(1,4,2))
```

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

In [32]:

```
arr = np.arange(10,40).reshape(5,6)
print(arr)
```

```
[[10 11 12 13 14 15]
 [16 17 18 19 20 21]
 [22 23 24 25 26 27]
 [28 29 30 31 32 33]
 [34 35 36 37 38 39]]
```

In [33]:

```
arr = np.arange(1,40,2).reshape(5,4)
print(arr)
```

```
[[ 1  3  5  7]
 [ 9 11 13 15]
 [17 19 21 23]
 [25 27 29 31]
 [33 35 37 39]]
```

Creating an array with 0's, 1's and full

- np.zeros(shape,dtype) --> by default it returns flaot values
- np.ones(shape, dtype) --> by default it returns flaot values
- np.full(shape, value)

In [34]:

```
# zero matrix

z = np.zeros((3,3))
print(z)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [36]:

```
print(type(z[0][0]))
```

```
<class 'numpy.float64'>
```

In [37]:

```
z
```

Out[37]:

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

In [38]:

```
z = np.zeros((4,5),dtype=int)
print(z)
```

```
[[0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]]
```

In [39]:

```
print(type(z[0][1]))
```

```
<class 'numpy.int32'>
```

In [40]:

```
o = np.ones((5,5))
print(o)
```

```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

In [41]:

```
o = np.ones((5,5),dtype=int)
print(o)
```

```
[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
```

In [42]:

```
print(o*5)
```

```
[[5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]]
```

In [44]:

```python
# np.full(shape, value)

arr = np.full((3,3),5)
print(arr)
```

```
[[5 5 5]
 [5 5 5]
 [5 5 5]]
```

In [45]:

```python
np.diag(arr)
```

Out[45]:

```
array([5, 5, 5])
```

In [2]:

```python
import numpy as np
f = int(input("Enter 1st value: "))
s = int(input("Enter 2nd value: "))
row = int(input("Enter no.of rows: "))
col = int(input("Enter no.of columns: "))
a = np.arange(f,s).reshape(row,col)
print(a)
```

```
Enter 1st value: 1
Enter 2nd value: 10
Enter no.of rows: 3
Enter no.of columns: 3
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [47]:

```python
print(np.diag(a))
```

```
[1 5 9]
```

Identity matrix using eye() method

Syntax: np.eye(value, dtype)

In [48]:

```python
print(np.eye(4,dtype=int))
```

```
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]
```

Random

In [9]:

```
# Example for random.randint()

r = np.random.randint(10)
print(r)
```

```
7
```

In [11]:

```
r = np.random.randint(10, 40)
print(r)
```

```
28
```

In [15]:

```
r2 = np.random.randint(10, 40, 6)
print(r2)
print(r2.ndim)
```

```
[30 23 17 31 12 28]
1
```

In [18]:

```
r3 = np.random.randint(10,40,20).reshape(5,4)
print(r3)
print(r3.ndim)
```

```
[[27 21 19 17]
 [29 36 25 26]
 [30 13 17 21]
 [21 35 13 36]
 [33 17 18 11]]
2
```

In [19]:

```
r3 = np.random.randint(10,40,20).reshape(1,5,4)
print(r3)
print(r3.ndim)
```

```
[[[21 15 20 15]
  [23 37 13 28]
  [12 33 30 31]
  [36 11 17 10]
  [10 32 15 14]]]
3
```

In [21]:

```
# random.random()

np.random.random((2,3))
```

Out[21]:

```
array([[0.11711987, 0.01705921, 0.14526509],
       [0.7084277 , 0.93135272, 0.98336565]])
```

In [22]:

```
help(np.random.randint)
```

```
...
```

In [23]:

```python
# Built-in method in numpy library

print(dir(np))
```

```
['ALLOW_THREADS', 'AxisError', 'BUFSIZE', 'CLIP', 'ComplexWarning', 'DataSou
rce', 'ERR_CALL', 'ERR_DEFAULT', 'ERR_IGNORE', 'ERR_LOG', 'ERR_PRINT', 'ERR_
RAISE', 'ERR_WARN', 'FLOATING_POINT_SUPPORT', 'FPE_DIVIDEBYZERO', 'FPE_INVAL
ID', 'FPE_OVERFLOW', 'FPE_UNDERFLOW', 'False_', 'Inf', 'Infinity', 'MAXDIM
S', 'MAY_SHARE_BOUNDS', 'MAY_SHARE_EXACT', 'MachAr', 'ModuleDeprecationWarni
ng', 'NAN', 'NINF', 'NZERO', 'NaN', 'PINF', 'PZERO', 'RAISE', 'RankWarning',
'SHIFT_DIVIDEBYZERO', 'SHIFT_INVALID', 'SHIFT_OVERFLOW', 'SHIFT_UNDERFLOW',
'ScalarType', 'Tester', 'TooHardError', 'True_', 'UFUNC_BUFSIZE_DEFAULT', 'U
FUNC_PYVALS_NAME', 'VisibleDeprecationWarning', 'WRAP', '_NoValue', '_UFUNC_
API', '__NUMPY_SETUP__', '__all__', '__builtins__', '__cached__', '__config_
_', '__dir__', '__doc__', '__file__', '__getattr__', '__git_revision__', '__
loader__', '__mkl_version__', '__name__', '__package__', '__path__', '__spec
__', '__version__', '_add_newdoc_ufunc', '_distributor_init', '_globals', '_
mat', '_pytesttester', 'abs', 'absolute', 'absolute_import', 'add', 'add_doc
string', 'add_newdoc', 'add_newdoc_ufunc', 'alen', 'all', 'allclose', 'alltr
ue', 'amax', 'amin', 'angle', 'any', 'append', 'apply_along_axis', 'apply_ov
er_axes', 'arange', 'arccos', 'arccosh', 'arcsin', 'arcsinh', 'arctan', 'arc
tan2', 'arctanh', 'argmax', 'argmin', 'argpartition', 'argsort', 'argwhere',
'around', 'array', 'array2string', 'array_equal', 'array_equiv', 'array_rep
r', 'array_split', 'array_str', 'asanyarray', 'asarray', 'asarray_chkfinit
e', 'ascontiguousarray', 'asfarray', 'asfortranarray', 'asmatrix', 'asscala
r', 'atleast_1d', 'atleast_2d', 'atleast_3d', 'average', 'bartlett', 'base_r
epr', 'binary_repr', 'bincount', 'bitwise_and', 'bitwise_not', 'bitwise_or',
'bitwise_xor', 'blackman', 'block', 'bmat', 'bool', 'bool8', 'bool_', 'broad
cast', 'broadcast_arrays', 'broadcast_to', 'busday_count', 'busday_offset',
'busdaycalendar', 'byte', 'byte_bounds', 'bytes0', 'bytes_', 'c_', 'can_cas
t', 'cast', 'cbrt', 'cdouble', 'ceil', 'cfloat', 'char', 'character', 'chara
rray', 'choose', 'clip', 'clongdouble', 'clongfloat', 'column_stack', 'commo
n_type', 'compare_chararrays', 'compat', 'complex', 'complex128', 'complex6
4', 'complex_', 'complexfloating', 'compress', 'concatenate', 'conj', 'conju
gate', 'convolve', 'copy', 'copysign', 'copyto', 'core', 'corrcoef', 'correl
ate', 'cos', 'cosh', 'count_nonzero', 'cov', 'cross', 'csingle', 'ctypesli
b', 'cumprod', 'cumproduct', 'cumsum', 'datetime64', 'datetime_as_string',
'datetime_data', 'deg2rad', 'degrees', 'delete', 'deprecate', 'deprecate_wit
h_doc', 'diag', 'diag_indices', 'diag_indices_from', 'diagflat', 'diagonal',
'diff', 'digitize', 'disp', 'divide', 'division', 'divmod', 'dot', 'double',
'dsplit', 'dstack', 'dtype', 'e', 'ediff1d', 'einsum', 'einsum_path', 'emat
h', 'empty', 'empty_like', 'equal', 'errstate', 'euler_gamma', 'exp', 'exp
2', 'expand_dims', 'expm1', 'extract', 'eye', 'fabs', 'fastCopyAndTranspos
e', 'fft', 'fill_diagonal', 'find_common_type', 'finfo', 'fix', 'flatiter',
'flatnonzero', 'flexible', 'flip', 'fliplr', 'flipud', 'float', 'float16',
'float32', 'float64', 'float_', 'float_power', 'floating', 'floor', 'floor_d
ivide', 'fmax', 'fmin', 'fmod', 'format_float_positional', 'format_float_sci
entific', 'format_parser', 'frexp', 'frombuffer', 'fromfile', 'fromfunctio
n', 'fromiter', 'frompyfunc', 'fromregex', 'fromstring', 'full', 'full_lik
e', 'fv', 'gcd', 'generic', 'genfromtxt', 'geomspace', 'get_array_wrap', 'ge
t_include', 'get_printoptions', 'getbufsize', 'geterr', 'geterrcall', 'geter
robj', 'gradient', 'greater', 'greater_equal', 'half', 'hamming', 'hanning',
'heaviside', 'histogram', 'histogram2d', 'histogram_bin_edges', 'histogramd
d', 'hsplit', 'hstack', 'hypot', 'i0', 'identity', 'iinfo', 'imag', 'in1d',
'index_exp', 'indices', 'inexact', 'inf', 'info', 'infty', 'inner', 'inser
t', 'int', 'int0', 'int16', 'int32', 'int64', 'int8', 'int_', 'int_asbuffe
r', 'intc', 'integer', 'interp', 'intersect1d', 'intp', 'invert', 'ipmt', 'i
rr', 'is_busday', 'isclose', 'iscomplex', 'iscomplexobj', 'isfinite', 'isfor
tran', 'isin', 'isinf', 'isnan', 'isnat', 'isneginf', 'isposinf', 'isreal',
'isrealobj', 'isscalar', 'issctype', 'issubclass_', 'issubdtype', 'issubscty
pe', 'iterable', 'ix_', 'kaiser', 'kron', 'lcm', 'ldexp', 'left_shift', 'les
s', 'less_equal', 'lexsort', 'lib', 'linalg', 'linspace', 'little_endian',
'load', 'loads', 'loadtxt', 'log', 'log10', 'log1p', 'log2', 'logaddexp', 'l
ogaddexp2', 'logical_and', 'logical_not', 'logical_or', 'logical_xor', 'logs
pace', 'long', 'longcomplex', 'longdouble', 'longfloat', 'longlong', 'lookfo
r', 'ma', 'mafromtxt', 'mask_indices', 'mat', 'math', 'matmul', 'matrix', 'm
atrixlib', 'max', 'maximum', 'maximum_sctype', 'may_share_memory', 'mean',
'median', 'memmap', 'meshgrid', 'mgrid', 'min', 'min_scalar_type', 'minimu
m', 'mintypecode', 'mirr', 'mkl', 'mod', 'modf', 'moveaxis', 'msort', 'multi
ply', 'nan', 'nan_to_num', 'nanargmax', 'nanargmin', 'nancumprod', 'nancumsu
m', 'nanmax', 'nanmean', 'nanmedian', 'nanmin', 'nanpercentile', 'nanprod',
'nanquantile', 'nanstd', 'nansum', 'nanvar', 'nbytes', 'ndarray', 'ndenumera
te', 'ndfromtxt', 'ndim', 'ndindex', 'nditer', 'negative', 'nested_iters',
'newaxis', 'nextafter', 'nonzero', 'not_equal', 'nper', 'npv', 'numarray',
'number', 'obj2sctype', 'object', 'object0', 'object_', 'ogrid', 'oldnumeri
c', 'ones', 'ones_like', 'outer', 'packbits', 'pad', 'partition', 'percentil
e', 'pi', 'piecewise', 'place', 'pmt', 'poly', 'poly1d', 'polyadd', 'polyde
r', 'polydiv', 'polyfit', 'polyint', 'polymul', 'polynomial', 'polysub', 'po
lyval', 'positive', 'power', 'ppmt', 'print_function', 'printoptions', 'pro
d', 'product', 'promote_types', 'ptp', 'put', 'put_along_axis', 'putmask',
'pv', 'quantile', 'r_', 'rad2deg', 'radians', 'random', 'rate', 'ravel', 'ra
vel_multi_index', 'real', 'real_if_close', 'rec', 'recarray', 'recfromcsv',
'recfromtxt', 'reciprocal', 'record', 'remainder', 'repeat', 'require', 'res
hape', 'resize', 'result_type', 'right_shift', 'rint', 'roll', 'rollaxis',
'roots', 'rot90', 'round', 'round_', 'row_stack', 's_', 'safe_eval', 'save',
'savetxt', 'savez', 'savez_compressed', 'sctype2char', 'sctypeDict', 'sctype
NA', 'sctypes', 'searchsorted', 'select', 'set_numeric_ops', 'set_printoptio
ns', 'set_string_function', 'setbufsize', 'setdiff1d', 'seterr', 'seterrcal
l', 'seterrobj', 'setxor1d', 'shape', 'shares_memory', 'short', 'show_confi
g', 'sign', 'signbit', 'signedinteger', 'sin', 'sinc', 'single', 'singlecomp
lex', 'sinh', 'size', 'sometrue', 'sort', 'sort_complex', 'source', 'spacin
g', 'split', 'sqrt', 'square', 'squeeze', 'stack', 'std', 'str', 'str0', 'st
r_', 'string_', 'subtract', 'sum', 'swapaxes', 'sys', 'take', 'take_along_ax
is', 'tan', 'tanh', 'tensordot', 'test', 'testing', 'tile', 'timedelta64',
'trace', 'tracemalloc_domain', 'transpose', 'trapz', 'tri', 'tril', 'tril_in
dices', 'tril_indices_from', 'trim_zeros', 'triu', 'triu_indices', 'triu_ind
ices_from', 'true_divide', 'trunc', 'typeDict', 'typeNA', 'typecodes', 'type
name', 'ubyte', 'ufunc', 'uint', 'uint0', 'uint16', 'uint32', 'uint64', 'uin
t8', 'uintc', 'uintp', 'ulonglong', 'unicode', 'unicode_', 'union1d', 'uniqu
e', 'unpackbits', 'unravel_index', 'unsignedinteger', 'unwrap', 'ushort', 'v
ander', 'var', 'vdot', 'vectorize', 'version', 'void', 'void0', 'vsplit', 'v
stack', 'warnings', 'where', 'who', 'zeros', 'zeros_like']
```

In [24]:

```
print(np.floor(5.4))
```

```
5.0
```

In [25]:

```
print(np.ceil(5.4))
```

```
6.0
```

In [26]:

```
# Arithmetic Operations

a1 = np.arange(11,20).reshape(3,3)
print(a1)
print("=========")
a2 = np.arange(1,10).reshape(3,3)
print(a2)
```

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

In [27]:

```
print(a1+a2)
print('----------')
print(a1-a2)
print('----------')
print(a1*a2)
print('----------')
print(a1/a2)
print('----------')
print(a1%a2)
print('----------')
print(a1**a2)
```

```
[[12 14 16]
 [18 20 22]
 [24 26 28]]
----------
[[10 10 10]
 [10 10 10]
 [10 10 10]]
----------
[[ 11  24  39]
 [ 56  75  96]
 [119 144 171]]
----------
[[11.          6.          4.33333333]
 [ 3.5         3.          2.66666667]
 [ 2.42857143  2.25        2.11111111]]
----------
[[0 0 1]
 [2 0 4]
 [3 2 1]]
----------
[[         11         144        2197]
 [      38416      759375    16777216]
 [  410338673 -1864941312   565150579]]
```

In [ ]:

```

```