Numpy – Part 1

GITAA
Transforming careers

In this lecture

- Numpy
- Creating an array
- Generating arrays using built-in functions
- Advantages of Numpy

Numpy

- Numpy stands for numerical python
- Fundamental package for numerical computations in Python
- Supports N-dimensional array objects that can be used for processing multi-dimensional data
- Supports different data-types

Numpy

- Using Numpy we can perform
  - Mathematical and logical operations on arrays
  - Fourier transforms
  - Linear algebra operations
  - Random number generation

Create an array

- Ordered collection of elements of basic data types of given length
- Syntax: `numpy.array(object)`

```
import numpy as np

In [2]: x=np.array([2,3,4,5])

In [3]: print(type(x))
<class 'numpy.ndarray'>


In [4]: print(x)
[2 3 4 5]
```

Arrays

- Numpy can handle different categorical entities

```
x=np.array([2,3,'n',5])

In [8]: print(x)
['2' '3' 'n' '5']
```

- All elements are coerced to same data type

Generate arrays using linspace( )

- `numpy.linspace()`- returns equally spaced numbers within the given range based on the sample number
- Syntax: `numpy.linspace(start,stop,num,dtype,retstep)`
- start - start of interval range
- stop - end of interval range
- num - number of samples to be generated
- dtype - type of output array
- retstep - return the samples, step value

Generate arrays using linspace( )

- Generate an array ***b*** with `start=1` and `stop=5`

```
b=np.linspace(start=1, stop=5, num=10, endpoint=True, retstep =False)
```

```
In [45]: print(b)
[ 1.          1.44444444  1.88888889  2.33333333  2.77777778  3.22222222
  3.66666667  4.11111111  4.55555556  5.        ]
```

Generate arrays using linspace( )

- Specifying **`retstep=True`** returns samples as well the step value

```
c=np.linspace(start=1, stop=5, num=10, endpoint=True, retstep =True)
```

```
In [47]: print(c)
(array([ 1.        ,  1.44444444,  1.88888889,  2.33333333,  2.77777778,
        3.22222222,  3.66666667,  4.11111111,  4.55555556,  5.        ]),
0.4444444444444442)
```

Generate arrays using arange( )

- `numpy.arange()`- returns equally spaced numbers with in the given range based on step size
- Syntax: `numpy.arange(start,stop,step)`
- start - start of interval range
- stop - end of interval range
- step - step size of interval

Generate arrays using arange( )

- Generate an array with `start=1` and `stop=10` by specifying `step=2`

```
d=np.arange(start=1,stop=10,step= 2)

In [49]: print(d)
[1 3 5 7 9]
```

Generate arrays using ones( )

- `numpy.ones()`- returns an array of given shape and type filled with ones
- Syntax: `numpy.ones(shape,dtype)`
- shape - integer or sequence of integers
- dtype - data type (default: float)

```
In [2]: np.ones((3,4))
Out[2]:
array([[ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.]])
```

Generate arrays using zeros( )

- `numpy.zeros()`- returns an array of given shape and type filled with zeros
- Syntax: `numpy.zeros(shape,dtype)`
- shape - integer or sequence of integers
- dtype - data type (default: float)

```
In [3]: np.zeros((3,4))
Out[3]:
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

Generate arrays using random.rand( )

- `numpy.random.rand()`- returns an array of given shape filled with random values
- Syntax: `numpy.random.rand(shape)`
- shape - integer or sequence of integers

```
In [8]: np.random.rand(5)
Out[8]: array([ 0.25649999,  0.68811372,  0.81316095,  0.76650141,  0.99914849])
```

Generate arrays using random.rand( )

- Generate an array of random values with 5 rows and 2 columns

```
In [6]: np.random.rand(5,2)
Out[6]:
array([[ 0.60537469,  0.38555842],
       [ 0.08722991,  0.55243026],
       [ 0.63857877,  0.22372736],
       [ 0.55870566,  0.26430069],
       [ 0.38618774,  0.7022909 ]])
```

Generate arrays using logspace( )

- `numpy.logspace()`- returns equally spaced numbers based on log scale
- Syntax:
  `numpy.logspace(start,stop,num,endpoint,base,dtype)`
- start - start value of the sequence
- stop - end value of the sequence
- num - number of samples to generate (default : 50)
- endpoint- if true, stop is the last sample
- base - base of the log space (default : 10.0)
- dtype- type of output array

Generate arrays using logspace( )

- Generate an array with 5 samples with base 10.0

```
In [9]: np.logspace(1, 10, num=5, endpoint=True, base=10.0)
Out[9]:
array([  1.00000000e+01,   1.77827941e+03,   3.16227766e+05,
         5.62341325e+07,   1.00000000e+10])
```

Advantages of numpy

- Numpy supports vectorized operations
- Array operations are carried out in C and hence the universal functions in numpy are faster than operations carried out on python lists

Advantages of numpy- speed

- `timeit` module can be used to measure the execution time for snippets of code
- Comparing the processing speed of a list and an array using an addition operation
- Creating a list

```
x=range(1000)
```

```
In [8]: timeit sum(x)
17.7 µs ± 1.16 µs per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

Advantages of numpy- speed

- Creating a numpy array

```
y=np.array(x)
```

```
In [12]: timeit np.sum(y)
5.41 µs ± 48.2 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

- Note that array works faster when compared to lists

Advantages of numpy- storage space

- Comparing the list ***x*** and array **y** from the previous example to find the memory used at the run time
- `getsizeof()`- returns the size of the object in bytes
- Syntax: `sys.getsizeof(object)`
- `itemsize`- returns the size if one element of a numpy array
- Syntax: `numpy.ndarray.itemsize`

Advantages of numpy - storage space

- Size of the list can be found by multiplying the size of an individual element with the number of elements in the list

```
import sys

In [14]: sys.getsizeof(1) * len(x)
Out[14]: 28000
```

Advantages of numpy - storage space

- Size of an array can be found by multiplying the size of an individual element with number of elements in the array

```
In [15]: y.itemsize * y.size
Out[15]: 4000
```

- Note that numpy array uses less bytes for storage than the python list.

Summary

- Create array
- Generate arrays using functions:
  - linspace
  - arange
  - ones
  - zeros
  - random.rand
  - logspace
- Advantages of numpy


operation == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True
selection at the end -add
ob.select= 1
er_ob.select=1
ntext.scene.objects.acti
("Selected" + str(modifier
irror_ob.select = 0
bpy.context.selected_ob
ata.objects[one.name].se
int("please select exact

THANK YOU