# Data Analysis using Python

## Universal Functions: Fast Element-Wise Array Functions

A universal function, or *ufunc*, is a function that performs element-wise operations on data in ndarrays.

Many ufuncs are simple element-wise transformations, like sqrt or exp:

```
In []:  arr = np.arange(10)
In []:   arr
Out[]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In []:   np.sqrt(arr)   ### Try This
Out[]:
array([ 0. , 1. , 1.4142, 1.7321, 2. , 2.2361, 2.4495,
2.6458, 2.8284, 3. ])

In []: np.exp(arr)   ### Try This
Out[]:
array([ 1. , 2.7183, 7.3891, 20.0855, 54.5982,
148.4132, 403.4288, 1096.6332, 2980.958 , 8103.0839])
```

These are referred to as *unary* ufuncs. Others, such as add or maximum, take two arrays (thus, *binary* ufuncs) and return a single array as the result:

```
In []: x = np.random.randn(8)
In []: y = np.random.randn(8)
In []: x
Out[]:
array([-0.0119, 1.0048, 1.3272, -0.9193, -1.5491, 0.0222, 0.7584,
-0.6605])

In []: y
Out[]:
array([ 0.8626, -0.01 , 0.05 , 0.6702, 0.853 , -0.9559, -0.0235,
-2.3042])

In []: np.maximum(x, y)  ### Try This
Out[]:
Array([0.8626, 1.0048, 1.3272……])
```

Here, numpy.maximum computed the element-wise maximum of the elements in x and y.

```
arr = np.random.randn(7) * 5
In []: arr  ### Try This
```

###modf Return fractional and integral parts of array as a separate array

```
In []: remainder, whole_part = np.modf(arr)
In []: remainder   ### Try This
In []: whole_part   ### Try This
```

**Array-Oriented Programming with Arrays**

Expressing Conditional Logic as Array Operations

```
In [165]: xarr = np.array([1.1, 1.2, 1.3, 1.4, 1.5])
In [166]: yarr = np.array([2.1, 2.2, 2.3, 2.4, 2.5])
In [167]: cond = np.array([True, False, True, True, False])

In [168]: result = [(x if c else y)
             : for x, y, c in zip(xarr, yarr, cond)]

In []: result  ### Try This
Out[]: [1.1000000000000001, 2.2000000000000002, 1.3, 1.3999999999999999, 2.5]



In []: result = np.where(cond, xarr, yarr)
In []: result   ### Try This

Out[]: array([ 1.1, 2.2, 1.3, 1.4, 2.5])

In []: arr = np.random.randn(4, 4)
In []: arr
Out[]:
array([[-0.5031, -0.6223, -0.9212, -0.7262],
[ 0.2229, 0.0513, -1.1577, 0.8167],
[ 0.4336, 1.0107, 1.8249, -0.9975],
[ 0.8506, -0.1316, 0.9124, 0.1882]])

In []: arr > 0
Out[]:
array([[False, False, False, False],
[ True, True, False, True],
[ True, True, True, False],
[ True, False, True, True]], dtype=bool)

In []: np.where(arr > 0, 2, -2)   ### Try This

In []: np.where(arr > 0, 2, arr)  ### set only positive values to 2(Try This)
```

**From:-**
**Ritu Meena**
**Assistant Professor**
**Shivaji College**
**Delhi University**