e-8

March 11, 2025

[5]:
```
import numpy as np
from PIL import Image
from IPython.display import display
```

[6]:
```
"""
universal functions :
sometime we should usd from some methods and functions like add for arrays
but python implemeting this with python language is not good and running time␣
 ↪is very slow
to this reason, numpy used from universal finctions(ufunc) and run this␣
 ↪functions
with C language beacuse it is not slow and it is very fast
"""
```

[6]:
```
'\nuniversal functions :\nsometime we should usd from some methods and functions
like add for arrays\nbut python implemeting this with python language is not
good and running time is very slow\nto this reason, numpy used from universal
finctions(ufunc) and run this functions\nwith C language beacuse it is not slow
and it is very fast\n'
```

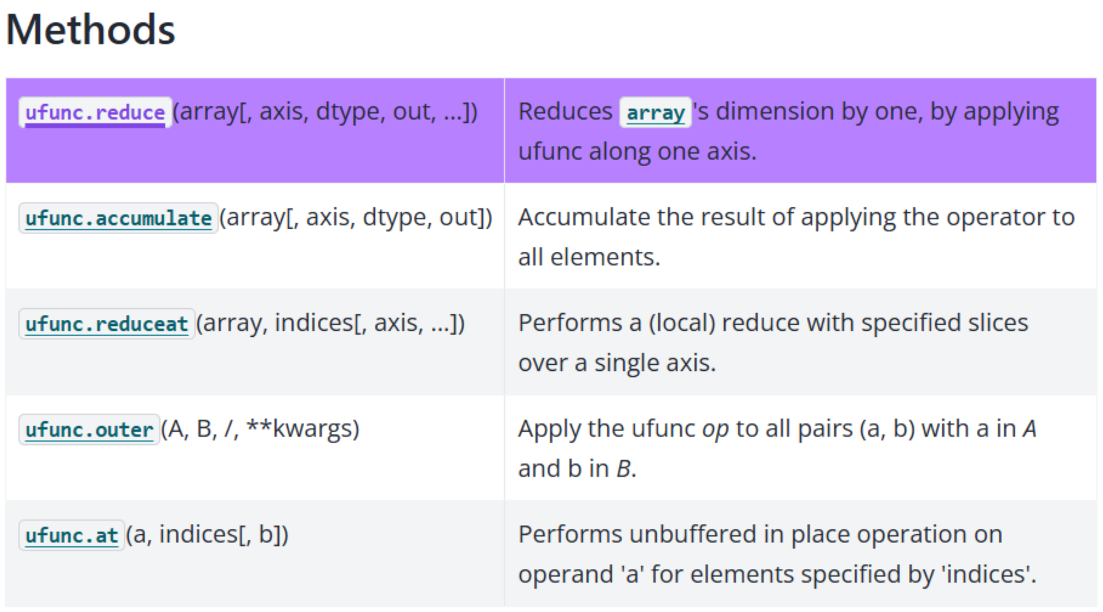
[20]:
```
a = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
])

b = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
])

np.add(a, b)
```

[20]:
```
array([[ 2,  4,  6],
       [ 8, 10, 12],
       [14, 16, 18]])
```

```
[21]: np.sum(a)
```

```
[21]: np.int64(45)
```

```
[22]: display(Image.open('ufunc_methods.png'))
```



Methods

`ufunc.reduce`(array[, axis, dtype, out, ...])	Reduces `array`'s dimension by one, by applying ufunc along one axis.
`ufunc.accumulate`(array[, axis, dtype, out])	Accumulate the result of applying the operator to all elements.
`ufunc.reduceat`(array, indices[, axis, ...])	Performs a (local) reduce with specified slices over a single axis.
`ufunc.outer`(A, B, /, **kwargs)	Apply the ufunc *op* to all pairs (a, b) with a in *A* and b in *B*.
`ufunc.at`(a, indices[, b])	Performs unbuffered in place operation on operand 'a' for elements specified by 'indices'.

```
[23]: # ufunc methods shoud used on ufuncs

      np.add.reduce(a)
```

```
[23]: array([12, 15, 18])
```

```
[24]: np.add.accumulate(a)
```

```
[24]: array([[ 1,  2,  3],
             [ 5,  7,  9],
             [12, 15, 18]])
```

```
[26]: c = np.array([
          [1, 2, 3],
      ])

      d = np.array([
          [4, 5, 6],
      ])

      np.add.outer(c, d)
```

```
[26]: array([[[[5, 6, 7]],

               [[6, 7, 8]],

               [[7, 8, 9]]]])
```