NumPy Universal Functions (ufuncs)

✅ **What are ufuncs?**

- **Ufunc** stands for **Universal Function**.
- A **ufunc** performs **fast, element-wise operations** on NumPy arrays.
- They are highly optimized **vectorized operations**, so you don't need explicit Python loops.
- Example: np.add, np.subtract, np.multiply, np.divide are all ufuncs.

✅ **Why use ufuncs?**

- Faster than Python loops (uses compiled C code under the hood).
- Supports **broadcasting**: arrays of different shapes work together.
- Handles **multi-dimensional data** efficiently.
- Reduces your code to clean, single-line operations.

✅ **Most Common ufuncs**

Here's a table of popular ufuncs and what they do:

Function	Purpose	Example
np.add(x, y)	Addition	np.add([1,2], [3,4]) → [4,6]
np.subtract(x, y)	Subtraction	np.subtract([5,6], [1,2]) → [4,4]
np.multiply(x, y)	Multiplication	np.multiply([2,3], [4,5]) → [8,15]
np.divide(x, y)	Division	np.divide([8,9], [2,3]) → [4,3]
np.power(x, y)	Exponentiation	np.power([2,3], 2) → [4,9]
np.exp(x)	Exponential	np.exp([0,1]) → [1, 2.718]
np.sqrt(x)	Square root	np.sqrt([4,9]) → [2,3]
np.log(x)	Natural log	np.log([1, np.e]) → [0,1]
np.sin(x)	Sine	np.sin([0, np.pi/2]) → [0,1]
np.maximum(x, y)	Max element-wise	np.maximum([1,2], [2,1]) → [2,2]

✅ **Broadcasting with ufuncs**

One of the most powerful parts of ufuncs is **broadcasting**.

Example:

python

CopyEdit

```
import numpy as np

a = np.array([1, 2, 3])

b = 2

result = np.add(a, b)  # [3, 4, 5]
```

Here, b is a scalar — NumPy broadcasts it to [2,2,2].

✅ **Vectorization vs. Loops**

Python loop:

python

CopyEdit

```
result = []

for x, y in zip([1, 2, 3], [4, 5, 6]):

   result.append(x + y)
```

Same with ufunc:

python

CopyEdit

```
np.add([1, 2, 3], [4, 5, 6])
```

Much faster and cleaner!

✅ **Advanced: out and where**

Most ufuncs accept:

- out → save result in an existing array (memory-efficient).
- where → apply conditionally.

Example:

python

CopyEdit

```
import numpy as np
```

```
x = np.array([1, 2, 3])
y = np.array([4, 5, 6])
out = np.empty(3)

np.add(x, y, out=out, where=[True, False, True])
print(out)  # [5, any random value, 9]
```

✅ **Creating your own ufunc**

For custom element-wise ops:

python

CopyEdit

```
vectorized_func = np.frompyfunc(lambda x: x + 2, 1, 1)
result = vectorized_func([1, 2, 3])  # [3, 4, 5]
```

🎉 **Key Takeaways**

- Ufuncs are **super fast**, **element-wise**, and **broadcast-ready**.
- They work on **scalars, vectors, matrices**.
- Always prefer ufuncs over explicit loops in NumPy!