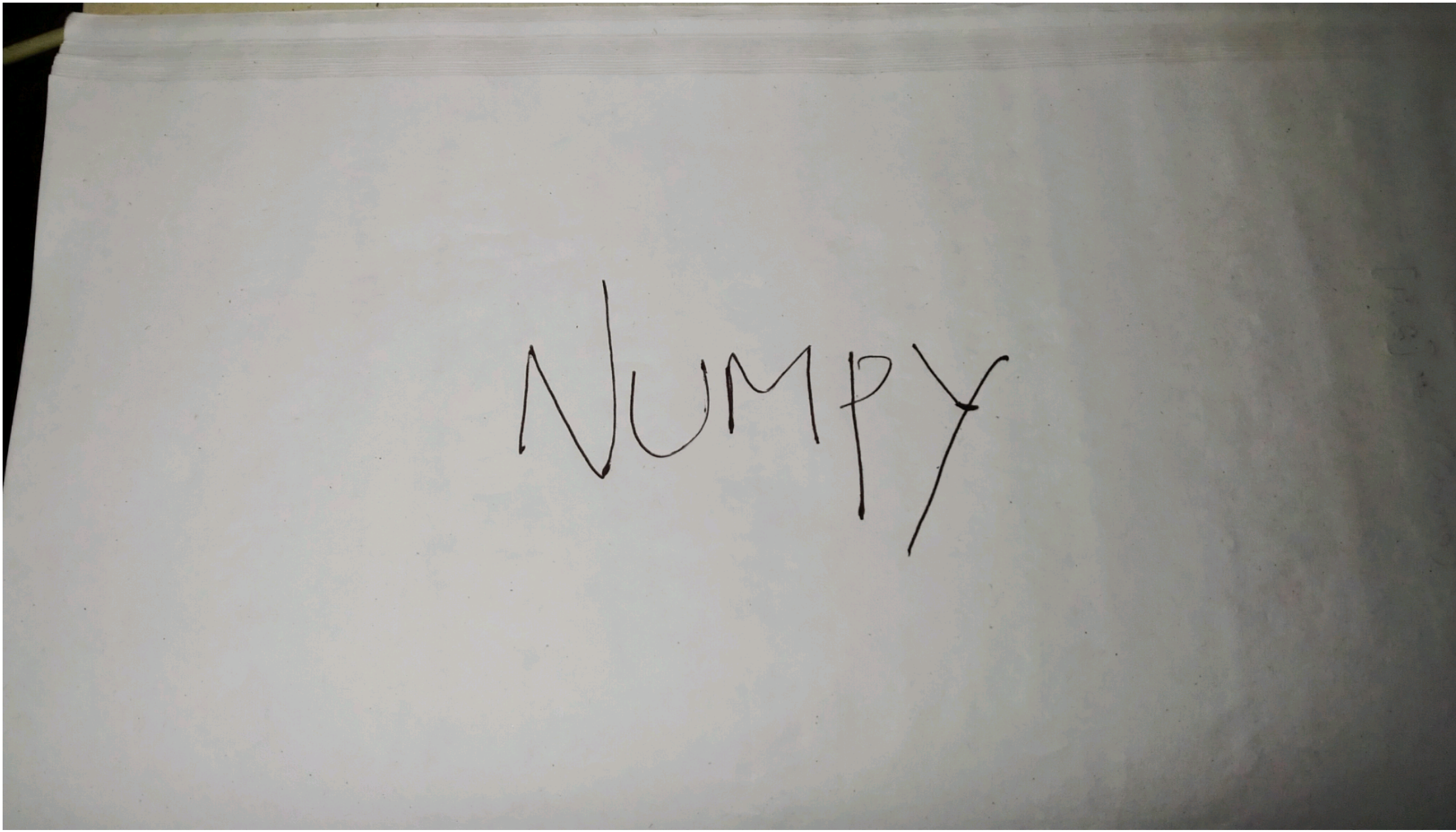
NUMPY

→ Numpy Origin :-

To solve array problem in python.

→ Travis oliphant developed Numpy.

→ written in C lang. Compiled faster (50x)

→ Numpy Arrays :-

Features	Python list	Numpy Array.
Data type	Mixed allowed	Same datatypes only.
Performance	Slower	Much faster (C level)
Memory efficiency	Low	High
Vector operations	Manual loops	Easy + fast.

① Basic array creations :-

```
import numpy as np
arr = np.array([1, 2, 3, 4])   # 1D Array.
arr
```

```
l = [[1,2,3], [4,5,6], [7,8,9]]
                                # 2D Array.
np.array(l)
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 10]])
```

---

② Array generation functions

```
arr = np.arange(1, 11, 2)
arr
```

→ array([1, 3, 5, 7, 9])

```
np.zeros(6)
```

→ array([0., 0., 0., 0., 0., 0.])

np. zeros ((4,8))

→ 4 rows of zero

→ 8 ~~rows~~ columns of zero

---

np. ones (6).

array ([1., 1., 1., 1., 1., 1.])

---

np. ones ((6,6))

→ 6 rows of 1

→ 6 cols of 1

---

np. linspace(0, 1, 100)

→ Space between 0 and 1 divided equally.

---

(3) Random generation function

np. random. rand (10)

np. random. randn (10)

np. random. randint (10, 20, 10)

④ Array Attributes

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

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

arr.shape → (3,3)

arr.size → (9)

arr.dtype → dtype('int64')

---

⑤ Array Method.

arr.min() → np.int64(1)

arr.max() → np.int64(9)

arr.sum() → np.int64(45)

np.sum(arr, axis = 1) → sum of cols.
0 → sum of rows.

arr.mean() → 5.0

arr.std() → 2.5819...

arr.argmax() → 8.

arr.argmin() → 0.

⑥ Reshaping & Resizing.

arr = np.arange(1, 31)

→ a array of 1 to 30.

arr.reshape(6, 5)

→ Reshaping of above array into 6,5 matrix (2D Array).

② Array indexing

① Numpy indexing & slicing of vectors.

```
import numpy as np
arr = np.arange(11, 21)
arr
→ array([11, 12, 13, 14, 15, 16, 17, 18, 19, 20])
arr_slice = arr[3::2]
arr_slice
→ array([14, 16, 18, 20])
arr[4]
→ 15.
```

---

② Numpy indexing & slicing of matrix.

```
arr = np.arange(1,31).reshape(6,5)
arr
→ array([[ 1, 2, 3, 4, 5],
         [6, 7, 8, 9, 10],
         [11, 12, 13, 14, 15],
         [16, 17, 18, 19, 20],
         [21, 22, 23, 24, 25],
         [26, 27, 28, 29, 30]])
```

```
slice = arr[3:, 3:]
slice
→ array([[19, 20],
         [24, 25],
         [29, 30]])

arr[:, 2]
array([3, 8, 13, 18, 23, 28])
```

---

③ Boolean indexing :-

```
arr = np.arange(11, 21)
arr
→ array([11, 12, 13, 14, 15, 16, 17, 18, 19, 20])

bool_index = arr % 2 == 0
bool_index
→ array([False, True, False, True .... ])

arr = arr[bool_index]
arr
→ array([12, 14, 16, 18, 20])
```

Array operations.

① Arithmatic operations :-

```
import numpy as np
a1 = np.array([1, 2, 3, 4, 5])
a2 = np.array([6, 7, 8, 9, 10])

a1 + a2
→ array([7, 9, 11, 13, 15])

a1 - a2
→ array([-5, -5, -5, -5, -5])

a1 * a2
→ array([6, 14, 24, 36, 50])

a1 / a2
→ array([0.16..., 0.28571..., 0.375..., 0.44..., 0.5])

a1 // a2
→ array([0, 0, 0, 0, 0])

a1 ** a2
→ array([1, 128, 6561, 262144, 9765625])

>> arr2 * 2
→ array([[42, 44, 46, 48, 50],
         [52, 54, 56, 58, 60],
         [62, 64, 66, 68, 70],
         [72, 74, 76, 78, 80],
         [82, 84, 86, 88, 90]])
```

② Broadcasting :-

```
l = [10,20,30,40]
arr = np.array(l)

arr + 10
→ array([20,30,40,50])

arr2 = np.arange(1,26).reshape(5,5)

arr2
→ array([[ 1, 2, 3, 4, 5],
         [ 6, 7, 8, 9, 10],
         [11, 12, 13, 14, 15],
         [16, 17, 18, 19, 20],
         [21, 22, 23, 24, 25]])

arr2 = arr2 + 10

arr2
→ array([[21, 22, 23, 24, 25],
         [26, 27, 28, 29, 30],
         [31, 32, 33, 34, 35],
         [36, 37, 38, 39, 40],
         [41, 42, 43, 44, 45]])
```

③ Deep & Shallow Copy :-

```
a = np.arange(1, 21)

a
array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
       16, 17, 18, 19, 20])

slice = a[:5]

slice = slice * 10

slice
array([10, 20, 30, 40, 50])

a[4:6] * 10

array([50, 60])

a
b = a.copy()

b[0] = 99

b
array([99, 2, 3, 4, 5 .... 20])

a
array([1, 2, 3, 4, 5, 6, 7 .... 20])
```

④ Matrix operations :-

```
A = np.array([[1,2], [3,4]])
B = np.array([[5,6], [7,8]])

np.dot((A, B))

→ array([[19, 22],
         [43, 50]])

A.T  # changing row to col

→ array([[1,3],
         [2,4]])
```

---

⑤ Advance Array manipulation :-

→ Stacking Array :-

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

np.vstack((a,b))

→ array([[1, 2, 3, 4],
         [5, 6, 7, 8]])

np.hstack((a,b))

→ array([1, 2, 3, 4, 5, 6, 7, 8])
```

(np.column_stack ((a, b))

→ array ([[1, 5],
[2, 6],
[3, 7],
[4, 8]])

⑥ Splitting Array :

c = np.arange (16). reshape (4, 4)

c

→ array ([[0, 1, 2, 3],
[4, 5, 6, 7],
[8, 9, 10, 11],
[12, 13, 14, 15]])

np. hsplit (c, 4) → # splitting horizontally

np. Vsplit (c, 2) → # splitting vertically.