3/10/23 Day -> 3

Date 12.9

Numpy -> 11

Agenda -> Element wise Operation
- Matrix Multiplication
- Sorting
- Vectorization
- Broadcasting

```
a = np.arange(12).reshape(-1,4)
b = np.arange(12).reshape(-1,-1)
```
↓ Ambiguity

Sorting -> np.sort(a)
- Return a new array which is sorted & it is Not In-place

(ii) a.sort() -> In place & sort the original array

Element wise Operations

a = [1, 2, 3, 4]

a * 2 -> [2, 4, 6, 8]
- Not In place

Spiral

3/10/23

Date 190

$b = [6, 7, 8, 9, 10]$

$a * b = [1, 2, 3, 4, 5] * [6, 7, 8, 9, 10] = [6, 14, 24, 36, 50]$

Note → Both Array Must have same Shape

i) Array & Single element → ✓

(ii) Arrays have same Shape → ✓

(iii) Array with Diff shape → ✗

⇒ Also we can/this can applicable on 2-D Array

Matrix Multiplication →

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}_{2\times2} \begin{bmatrix} 5 & 6 & 7 \\ 8 & 9 & 0 \end{bmatrix}_{2\times3} = \begin{bmatrix} 1(5)+(2)(8) & (1)(6)+(2)(9) & (1)(7)+(2)(0) \\ (3)(5)+(4)(8) & (3)(6)+(4)(9) & (3)(7)+(4)(0) \end{bmatrix}$$

$m \times n$ equal $n \times x$

$$= \begin{bmatrix} 21 & 24 & 7 \\ 63 & 54 & 21 \end{bmatrix}_{2\times3}$$

first elem of 1st Row * first elem of first col

we Interpret it as

No of column in array 1 = No of Rows in array 2

Spiral

Date 19.1

13/10/23

Different ways to do Matrix Multiplication

① → np.dot(a,b) → np.dot(a,5) → can work
↳ flexible

② → np.matmul(a,b) → a & b must have Matrix

③ → a @ b

ex → a @ 5 → Doesn't work

Why There are 3 ways to do same
↳ Python introduced np.dot first & after that other methods. But Python doesn't want to deprecate there methods.

Extra (att of Context info)

(4,3) ← Image Res
↳ 4x3 pixels

(RGB)
each → pixel → color value
↓
contain code of that color

123	497	626
100	192	303
201	392	256
052	001	122

↑ apply on all cell

My code →

```
if color < 250:
    color + = 50
else:
    color -= 50
```

H/W

Spiral

5/10/25

Date 192

So, element wise operation we are applying

Vectorize | Vectorization

↳ without loops code are also

Array → Function → Return Modified Array

I have to use Loops

Send every value of Array through loop to function

ex> *, +, - → works on element wise because numpy supports.

But want a Function to work on every element without loops

=> np.vectorize( ) ← function that works on single elements

Returns new function which can work with arrays

Spiral

8/10/23

Date 193

Broadcasting →

0	0	0
10	10	10
20	20	20
30	30	30

+

0	1	2
0	1	2
0	1	2
0	1	2

=

0	1	2
10	11	12
20	21	22
30	31	32

Now :-

→ Not Put the 0

0	0	0
10	10	10
20	20	20
30	30	30

+

0	1	2
10	1	2
0	1	2
0	1	2

= → added automatically

These Not compatiable to add → Not same shape

↓

but Python Try to reach the same either horizontally or vertically

↓

Broadcast Array to match shape to perform operation

exp :-

0	0	0
10	10	10
20	20	20
30	30	30

+

0	1	2
0	1	2
0	1	2
0	1	2

→ Broadcasting (added + Stretch shape)

8/10/23

Date 194

& Ans of all 3 matrix plus will be same

tile -> np.tile([0, 10, 20, 30], (3, 1))

repetition row

repetition col

Repeat things

0	10	20	30
0	10	20	30
0	10	20	30

⁂ => One of array should be 1-D to Broadcast

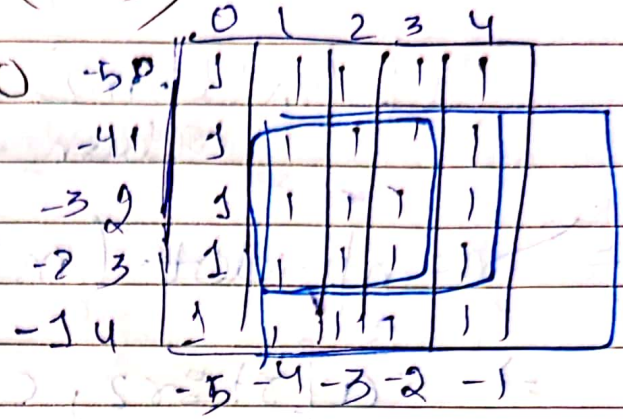
Spiral

3/10/28 Assignment Date 195

(1) -> Matrix element

x = np.ones((5,5))

X[1:-1, 1:-1] = 0 (excluded)



=

1 to 3 in Row
1 to 3 in Col
-> All element become zero

(2) -> Mapping in numpy

arr

2	3	4	5
1	7	3	5
2	8	6	19
11	23	12	19

```
def func(x,y)
    return x*y
vec = np.vectorize(func)
vec(arr, arr1)
```

(A) -> arr 1 = [2, 2, 2, 2] ↓ axis=0 Broadcasting vertically

(B) -> arr 1 = [2, 2, 2, 2] → axis=1 Broadcasting Horizontally

(C) -> arr 1 = [2, 2, 2, 2] → axis=1, ↓ axis=0

Broadcasting at vertical & horizontal level

Spiral

Date 196

3/10/23

(8) -> Specific elements

m1 = mat1.shape[1] -> col of first Matrix

m2 = mat2.shape[0] -> row of second Matrix

If (m1 != m2)

return -1

=> a = np.dot(m1, m2)

=> a[r1:r2, c1:c2]

(9) -> Sort the birds

We will use argsort

a = [1, 3, 7, 5, 2]

0 1 2 3 4

np.argsort(a) -> [0, 4, 1, 3, 2]

element 1 2 3 5 7

↳ Return index in sorted manner

Name = [_ , _ , _ , _ , _]

age = [_ , _ , _ , _ , _]

sort name to age

ind = np.argsort(age)

name[ind]

Spiral

3/10/23

Date 19.7

(5) -> Comparing in Numpy

(a) -> Ram, Astha, Raghavendra

Sort

Astha, Raghavendra, Ram -> Last

Option (1)

(b) - arr1 Ram, Astha, Brahal

arr2 Shyam, Kalyan, Naveen

-> false

option (3) -> [false, false, false]

Additional Problems

(1) -> Reshape me

np.arange(10,22).reshape((3,4))

Option (3) -> A 3×4 matrix filled with 10 to 21 number range

(2) -> Vectorized codes without loop

(A) -> for i in range(len(arr1)):

arr3[i] = arr1[i] * arr2[i]

Loop -> Not Vectorized

Spiral

3/10/23

Date 198

(B) → arr 1 * arr 2

arr 1 = [1, 2, 3, 6, 3, 2]
arr 2 = [4, 2, 1, 3, 3, 2]
arr 3 = np.zeros (len (arr 1))

(C) → for i in range (len (arr 1)):
    if (arr[i] > 0):
        arr [i] = -1
    else:
        arr 1 [i] = 1

(D) → np.where (arr 1 > 0, 1, -1)

Vectorized means which will be applied on each element without using loop

(E) → Dot Dot Dash →

A → [1, 2, 3] [9, 8, 7]
3x3 3x3
↳ 1-D Array → NOT Throw error

(B) → [1 2; 3 4] [1; 2]
2x2 2x1
No of Cols = 2 = No of Rows : 2
No T, Throw error

Spiral

Date 199

3/10/23

C -> [1,2,3] -> 1-D
k = 3
np.dot(arr1, k) -> NOT Throw error

2-D -> [1 2; 3 4] 2×2 or [1 1] 1×2 -> 1-D Array

NOT THROW Error

(1) It is also correct because only (this) is 1-D Array

(4) -> Swappers

print A[0, ::-1]

A -> | 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

=> | 3 | 2 | 1 |
| 6 | 5 | 4 |
| 9 | 8 | 7 |

option (2) -> Reverse the columns of 2-D Array

(5) -> Calculate Age

mask = birds == 'Cranes'

crange_age = age[mask]

mean_age = np.mean(crange_age)

rounding off = np.round(mean_age, 2)

Spiral

3/10/23

Date 20

6 -> Oldest bird

```
max_age_index = np.argmax(bird)
old_bird = birds[max_age_index]
return old_bird
```