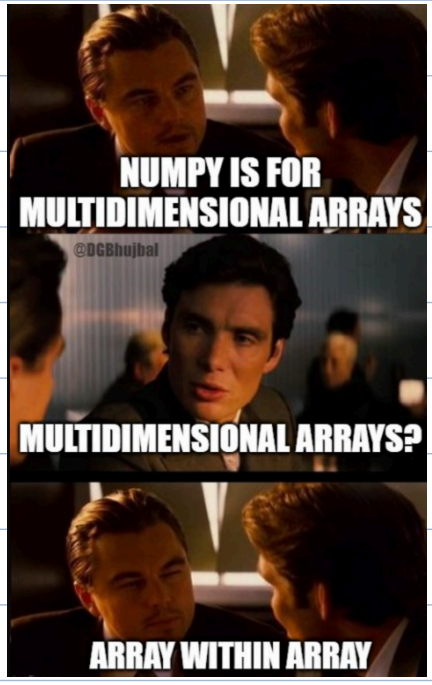
Agenda

- Working with 2D arrays (Matrices)
  - Transpose
  - Indexing
  - Slicing
  - Fancy Indexing (Masking)
- Aggregate Functions
- Logical Operations
  - `np.any()`
  - `np.all()`
  - `np.where()`
- Use Case: Fitness data analysis



How can we convert this to a 2-dimensional array?

- Using `reshape()`

For a 2D array, we will have to specify the following:-

- **First argument** is **no. of rows**
- **Second argument** is **no. of columns**

Code

```
a.reshape(8, -1)
```

Output

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

Notice that Python automatically figured out, that what should be the replacement of `-1` argument, given that the first argument is `8`.

We can also put `-1` as the first argument.

As long as one argument is given, it will calculate the other one.

What if we pass both args as `-1` ?

There is another operation on a multi-dimensional array, known as **Transpose**.

It basically means that the no. of rows is interchanged by no. of cols,and vice-versa.

Code

```
a.T
```

Indexing in 2D arrays

- Similar to Python lists

Slicing in 2D arrays

- Need to **provide two slice ranges**, one for **row** and one for **column**.
- We can also **mix Indexing and Slicing**

Aggregate Functions

Numpy provides various universal functions that cover a wide variety of operations and perform **fast element-wise array operations**.

How would you calculate the sum of elements of an array?

`np.sum()`

- **It sums all the values in a np array**.

What if we want to do the elements row-wise or column-wise?

- By **setting** `axis` **parameter**

What will `np.sum(a, axis=0)` **do?**

- `np.sum(a, axis=0)` adds together values in **different rows**
- `axis = 0` \(\rightarrow\) **Changes will happen along the vertical axis**
- Summation of values happen **in the vertical direction**.
- Rows collapse/merge when we do `axis=0`.

Now, what if we specify `axis=1`?

- `np.sum(a, axis=1)` adds together values in **different columns**
- `axis = 1` \(\rightarrow\) **Changes will happen along the horizontal axis**
- Summation of values happen **in the horizontal direction**.
- Columns collapse/merge when we do `axis=1`.

Logical Operations

What if we want to check whether "any" element of array follows a specific condition?

`np.any()` can become handy here as well

- `any()` returns `True` if **any of the corresponding elements** in the argument arrays follow the **provided condition**.

Imagine you have a shopping list with items you need to buy, but you're not sure if you have enough money to buy everything.

You want to check if there's at least one item on your list that you can afford.

In this case, you can use `np.any`:

What if we want to check whether "all" the elements in our array follow a specific condition?

`np.all()`

Let's consider a scenario where you have a list of chores, and you want to make sure all the chores are done before you can play video games.

You can use `np.all` to check if all the chores are completed.

`np.where()`

- Syntax: `np.where(condition, [x, y])`
- returns an `ndarray` whose elements are chosen from `x` or `y` depending on condition.

Suppose you have a list of product prices, and you want to apply a **10%** discount to all products with prices above **$50**.

You can use `np.where` to adjust the prices.

Use Case: Fitness data analysis

Imagine you are a Data Scientist at Fitbit

You've been given a user data to analyse and find some insights which can be shown on the smart watch.

But why would we want to analyse the user data for desiging the watch?

These insights from the user data can help business make customer oriented decision for the product design.

Let's first look at the data we have gathered.

There are 96 records and each record has 6 features.

These features are:

- Date
- Step Count → check correlation
- Mood
- Calories Burned
- Hours of Sleep → ,,
- Activity Status