

# Aggregation

Use aggregation techniques to combine NumPy data and arrays.

## Chapter Goals:

- Learn how to aggregate data in NumPy
- Write code to obtain sums and concatenations of NumPy arrays

### A. Summation

In the chapter on Math

(<https://www.educative.io/collection/page/6083138522447872/5629499534213120/5681717746597888>), we calculated the sum of individual values between multiple arrays. To sum the values within a single array, we use the `np.sum`

(<https://docs.scipy.org/doc/numpy/reference/generated/numpy.sum.html>) function.

The function takes in a NumPy array as its required argument, and uses the `axis` keyword argument in the same way as described in previous chapters

(<https://www.educative.io/collection/page/6083138522447872/5629499534213120/5661458385862656/>). If the `axis` keyword argument is not specified, `np.sum` returns the overall sum of the array.

The code below shows how to use `np.sum`.

```
1 arr = np.array([[0, 72, 3],
2                 [1, 3, -60],
3                 [-3, -2, 4]])
4 print(np.sum(arr))
5 print(repr(np.sum(arr, axis=0)))
6 print(repr(np.sum(arr, axis=1)))
```





Output



```
18
array([ -2,  73, -53])
array([ 75, -56,  -1])
```

In addition to regular sums, NumPy can perform cumulative sums using

`np.cumsum`

(<https://docs.scipy.org/doc/numpy/reference/generated/numpy.cumsum.html>) . Like `np.sum`, `np.cumsum` also takes in a NumPy array as a required argument and uses the `axis` argument. If the `axis` keyword argument is not specified, `np.cumsum` will return the cumulative sums for the flattened array.

The code below shows how to use `np.cumsum` . For a 2-D NumPy array, setting `axis=0` returns an array with cumulative sums across each column, while `axis=1` returns the array with cumulative sums across each row. Not setting `axis` returns a cumulative sum across all the values of the flattened array.

```
1 arr = np.array([[0, 72, 3],
2                 [1, 3, -60],
3                 [-3, -2, 4]])
4 print(repr(np.cumsum(arr)))
5 print(repr(np.cumsum(arr, axis=0)))
6 print(repr(np.cumsum(arr, axis=1)))
```

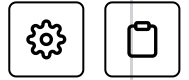


Output

0.497s

```
array([ 0, 72, 75, 76, 79, 19, 16, 14, 18])
array([[ 0,  72,   3],
       [ 1,  75, -57],
       [-2,  73, -53]])
```

```
array([[ 0, 72, 75],
       [ 1,  4, -56],
       [-3, -5, -1]])
```



## B. Concatenation

An important part of aggregation is combining multiple datasets. In NumPy, this equates to combining multiple arrays into one. The function we use to do this is `np.concatenate`

(<https://docs.scipy.org/doc/numpy/reference/generated/numpy.concatenate.html>).

Like the summation functions, `np.concatenate` uses the `axis` keyword argument. However, the default value for `axis` is 0 (i.e. dimension 0). Furthermore, the required argument for `np.concatenate` is a list of arrays, which the function combines into a single array.

The code below shows how to use `np.concatenate`, which aggregates arrays by joining them along a specific dimension. For 2-D arrays, not setting the `axis` argument (defaults to `axis=0`) concatenates the arrays vertically. When we set `axis=1`, the arrays are concatenated horizontally.

```
1 arr1 = np.array([[0, 72, 3],
2                  [1, 3, -60],
3                  [-3, -2, 4]])
4 arr2 = np.array([[-15, 6, 1],
5                  [8, 9, -4],
6                  [5, -21, 18]])
7 print(repr(np.concatenate([arr1, arr2])))
8 print(repr(np.concatenate([arr1, arr2], axis=1)))
9 print(repr(np.concatenate([arr2, arr1], axis=1)))
```



Output

0.434s

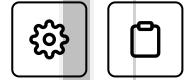
```
array([[ 0, 72,  3],
       [ 1,  3, -60],
       [-3, -2,  4],
```



```

    [-15,  6,  1],
    [  8,  9, -4],
    [  5, -21, 18]])
array([[ 0, 72,  3, -15,  6,  1],
       [ 1,  3, -60,  8,  9, -4],
       [-3, -2,  4,  5, -21, 18]])

```



## Time to Code!

Each coding exercise in this chapter will be to complete a small function that takes in 2-D NumPy matrices as input. The first function to complete is `get_sums`, which returns the overall sum and column sums of `data`.

**Set `total_sum` equal to `np.sum` applied to `data`.**

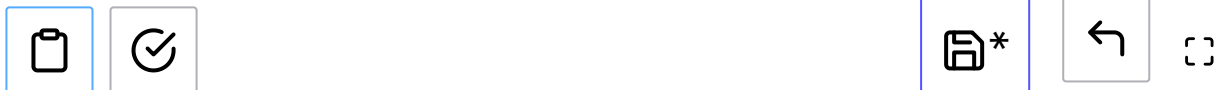
**Set `col_sum` equal to `np.sum` applied to `data`, with `axis` set to `0`.**

**Return a tuple of `total_sum` and `col_sum`, in that order.**

```

1 def get_sums(data):
2     total_sum = np.sum(data)
3     col_sum = np.sum(data, axis=0)
4     return total_sum, col_sum

```



Show Results

Show Console

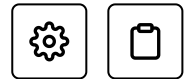


2 of 2 Tests Passed

| Result | Input | Expected Output            | Actual Output              | Reason   |
|--------|-------|----------------------------|----------------------------|--|
| ✓      |       | 63                         | 63                         | Correct value for <code>total_sum</code> , good job! |
| ✓      |       | array([20, 40, -1, -3, 7]) | array([20, 40, -1, -3, 7]) | Correct value for <code>col_sum</code> , good job!   |

0.387s

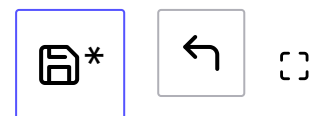
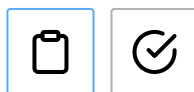
The next function to complete is `get_cumsum`, which returns the cumulative sums for each row of `data`.



**Set `row_cumsum` equal to `np.cumsum` applied to `data` with `axis` set to 1.**

**Then return `row_cumsum`.**

```
1 def get_cumsum(data):
2     row_cumsum = np.cumsum(data, axis=1)
3     return row_cumsum
4
5
6
```



Show Results

Show Console



1 of 1 Tests Passed

| Result | Input | Expected Output                               | Actual Output                                 | Reason  |
|--------|-------|---|---|---|
| ✓      |       | array([[19, 39,<br>34, 34, 35], [ 9,<br>...]) | array([[19, 39,<br>34, 34, 35], [ 9,<br>...]) | Correct value for<br>row_cumsum, good<br>job! |

0.367s

The final function, `concat_arrays`, takes in two 2-D NumPy arrays as input. It returns the column-wise and row-wise concatenations of the input arrays.

**Set `col_concat` equal to `np.concatenate` applied to a list of `data1`, `data2`, in that order.**

**Set `row_concat` equal to `np.concatenate` applied to a list of `data1`, `data2`, in that order. The `axis` keyword argument should be set to 1.**

**Return a tuple containing `col_concat` and `row_concat`, in that order.**

```
1 def concat_arrays(data1, data2):
```



```
2 col_concat = np.concatenate([data1, data2])
3 row_concat = np.concatenate([data1, data2], axis=1)
4 return col_concat, row_concat
5
6
7
```



Show Results

Show Console



2 of 2 Tests Passed

| Result | Input | Expected Output                        | Actual Output                          | Reason  |
|--------|-------|--|--|---|
| ✓      |       | array([[ 19, 20,<br>-5, 0, 1], ...     | array([[ 19, 20,<br>-5, 0, 1], ...     | Correct value for<br>col_concat, good<br>job! |
| ✓      |       | array([[ 19, 20,<br>-5, 0, 1, 1, - ... | array([[ 19, 20,<br>-5, 0, 1, 1, - ... | Correct value for<br>row_concat, good<br>job! |

0.520s

← Back

Next →

Statistics

Saving Data

☒ Mark as Completed

10% completed, meet the [criteria](#) and claim your course certificate!

Buy Certificate



Report an  
Issue



Ask a Question

([https://discuss.educative.io/tag/aggregation\\_\\_data-manipulation-with-numpy\\_\\_machine-learning-for-software-engineers](https://discuss.educative.io/tag/aggregation__data-manipulation-with-numpy__machine-learning-for-software-engineers))

