Solution Review

This lesson gives a detailed solution review of the problem.

WE'LL COVER THE FOLLOWING

Solution

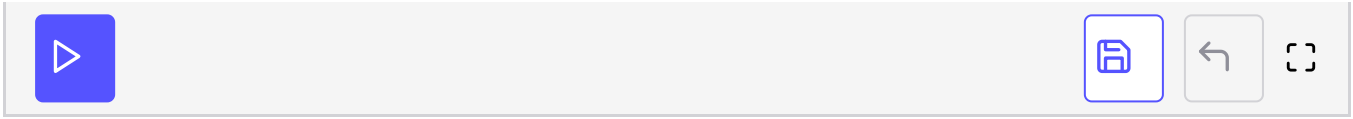
Here's the merged solution to the problem that we discussed in the previous lesson. The solution also takes into account these two factors:

- Negative steps
- Multi-dimensional arrays

```
import numpy as np
import matplotlib.pyplot as plt


def find_index(base, view):
    """
    Given an array that is a `view` of a `base`, find an index such that
    `base[index] is view`
    """

    if not isinstance(view, np.ndarray):
        return "..."

    itemsize = view.itemsize

    # Find the start and end pointer of the arrays using the byte_bound method
    offset_start = (np.byte_bounds(view)[0] - np.byte_bounds(base)[0])//itemsize
    offset_stop = (np.byte_bounds(view)[-1] - np.byte_bounds(base)[-1]-1)//itemsize

    # Calculate the start and stop indices from the offsets
    index_start = np.unravel_index(offset_start, base.shape)
    index_stop = np.unravel_index(base.size+offset_stop, base.shape)

    # Use the strides property to find the No. of bytes to go from one element to the other
    index_step  = np.array(view.strides)//np.array(base.strides)

    index = ""
    for i in range(len(index_step)):
```

```
        start = index_start[i]
        stop = index_stop[i]
        step = index_step[i]


        if stop == start:
            stop, step = None, None
        else:
            if stop == base.shape[i] - 1:
                stop = None
            else:
                stop = stop
            if start == 0:
                start = None
        if step is not None and stop is not None:
            if step < 0:
                start, stop = stop, start - 1
            else:
                start, stop = start, stop + 1

        if start is not None:
            index += str(start)
        if stop is not None:
            index += ":" + str(stop)
        elif step is not None:
            index += ":"
        if step is not None:
            index += ":" + str(step)
        index += ','
    index = index[:-1]

    return index



if __name__ == '__main__':

    base = np.arange(8*8).reshape(8,8)

    # Sub-array
    Z = base[1:-1,1:-1]
    index = find_index(base,Z)
    print(np.allclose(Z, eval("base[%s]" % index)))

    # Every two items
    Z = base[::2,::2]
    index = find_index(base,Z)
    print(np.allclose(Z, eval("base[%s]" % index)))

    # First column
    Z = base[:,0]
    index = find_index(base,Z)
    print(np.allclose(Z, eval("base[%s]" % index)))

    # First row
    Z = base[0,:]
    index = find_index(base,Z)
    print(np.allclose(Z, eval("base[%s]" % index)))

    # Partial reverse
    Z = base[4:1:-1,6:2:-1]
    index = find_index(base,Z)
    print(np.allclose(Z, eval("base[%s]" % index)))
```

```
# # Full reverse
Z = base[::-1,::-1]

index = find_index(base,Z)
print(np.allclose(Z, eval("base[%s]" % index)))

# Random
Z = base[1:5:3,3:1:-1]
index = find_index(base,Z)
print(np.allclose(Z, eval("base[%s]" % index)))
```



Now that we have learned about the anatomy of an array, let's move on to the next chapter "Code Vectorization".