Coding Example: How to find if one vector is view of the other?

As a conclusion to this chapter, we'll look at one simple case study to find if one vector is view of the other.

WE'LL COVER THE FOLLOWING

Problem Statement

Given two vectors `Z1` and `Z2`. We would like to know if `Z2` is a view of `Z1` and if yes, what is this view?

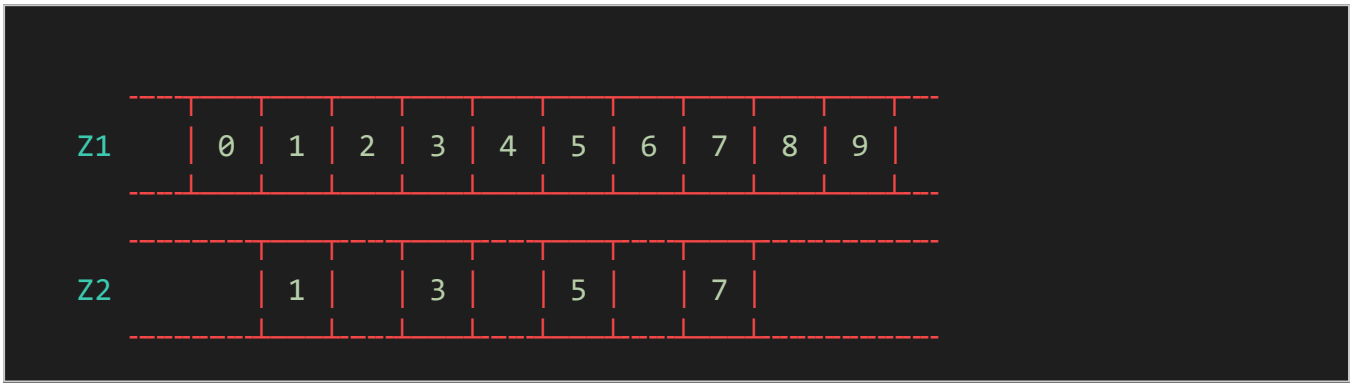
Example

Given below is a running example to give you a better understanding:

```
import numpy as np
Z1 = np.arange(10) #store a numpy array in `Z1` of size 10 containing values from 1-10
Z2 = Z1[1:-1:2] #store values of alternating indices of Z1 in Z2
print(Z1)
print(Z2)
```

Illustration

The below illustration shows what the two vectors `Z1` and `Z2` would look like:



Step 1: Check if view exists?

The `base` method lets you know if the view really exists:

```
print(Z2.base is Z1)#check if Z2 is a base of Z1
```

At this point, we know `Z2` is a view of `Z1`, meaning `Z2` can be expressed as `Z1[start:stop:step]`. The difficulty is to find `start`, `stop` and `step`.

Step 2: Find out the value of `step`

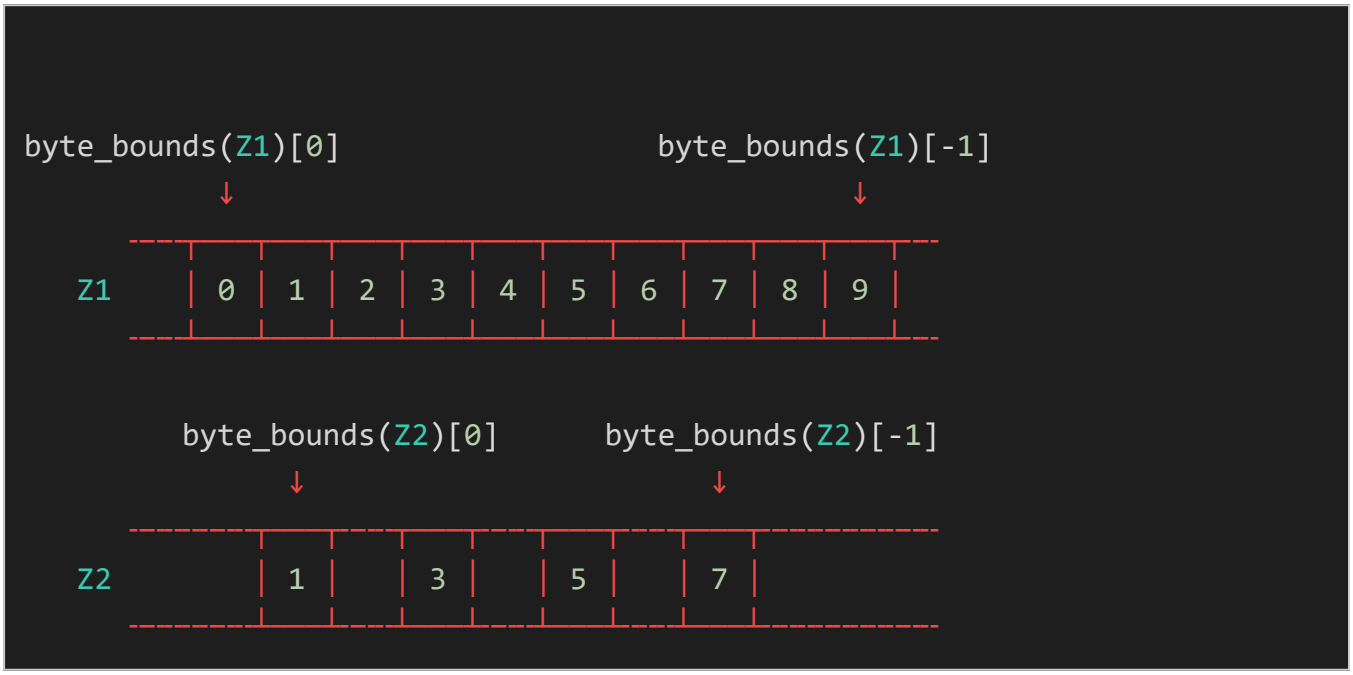
For the `step`, we can use the `strides` property of any array that gives the number of bytes to go from one element to the other in each dimension. In our case, and because both arrays are one-dimensional, we can directly compare the first stride only:

```
step = Z2.strides[0] # Z2.strides[0]
print("step:",step)
```

Step 3: Find `start` and `stop` indices in the array

Next difficulty is to find the `start` and the `stop` indices.

To do this, we can take advantage of the `byte_bounds` method that returns a pointer to the end-points of an array.



Here, `byte_bounds(Z1)[0]` and `byte_bounds(Z2)[0]` returns a pointer to the start point of the array `Z1` and `Z2` respectively and `byte_bounds(Z1)[-1]` and `byte_bounds(Z1)[-1]` returns a pointer to the endpoints of `Z1` and `Z2` respectively.

Coding Challenge: Try it yourself!

Looking at the illustration, try solving this challenge below! You have to find the values of `offset_start` and `offset_stop`. Take the assumption that the code already imports the NumPy library as `np`.

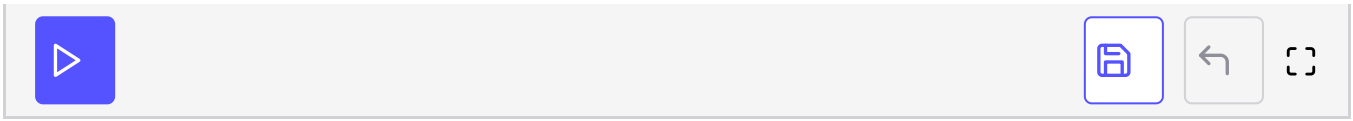
```
def calculate_offsets(Z1, Z2):
  offset_start = offset_stop = 0
  # Write - Your - Code

  return [offset_start, offset_stop]
```

Step 4: Convert the offsets into index values

Converting these offsets into index values is straightforward using the `itemsize` and taking into account that the `offset_stop` is negative (end-bound

of `Z2` is logically smaller than end-bound of `Z1` array). We thus need to add the items size of `Z1` to get the right end index.

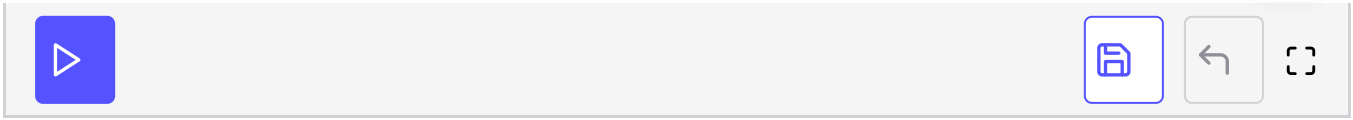
```
start = offset_start # compute the starting index
stop = Z1.size + offset_stop # compute the ending index
print("start:",start)
print("stop:",stop)
```



Step 5: Test your result!

Last, we test our results by using the `allclose` method in NumPy. It returns `True` if the two arrays are equal element-wise by comparing the relative and absolute difference of the two arrays with the already set threshold value, called *tolerance*.

```
print(np.allclose(Z1[start:stop:step], Z2))#returns true if two arrays are equal elemet-wise
```



The next lesson provides a detailed solution to this problem, see you there!