NumPy Arithmetic and Statistics - Comparison and Boolean Masks

WE'LL COVER THE FOLLOWING

3. Comparisons and Boolean Masks

a. Comparisons

In this world reigned by the social media, the trap of making comparisons is just about everywhere. So, staying true to the culture of making comparisons, let's talk about comparisons in NumPy

NumPy provides comparison operators such as *less than* and *greater than* as element-wise functions. The result of these comparison operators is always an array with a Boolean data type, i.e., we get boolean array as output which contains only True and False values depending on whether the element at that index lives up to the comparison or not. Let's see this in action with some examples.

Run the code in the widget below and observe the outputs of the print statements to understand is going on.

```
import numpy as np

x = np.array([1, 2, 3, 4, 5])
```

```
print(x < 2) # less than
print(x >= 4) # greater than or equal
```

We can also do an element-by-element comparison of two arrays and include compound expressions:

```
x = np.array([1, 2, 3, 4, 5])

# Elements for which multiplying by two is the same as the square of the value
(2 * x) == (x ** 2)
#> array([False,  True, False, False, False], dtype=bool)
```

We can also count entries in the boolean array that we get as outputs. This can help us perform other related operations, like getting the total count of values less than 6, `np.count_nonzero`, or checking if all the values in the array are less than 10, `np.all` and `np.any`:

```
import numpy as np

x = np.arange(10)
print(x)

# How many values less than 6?
print(np.count_nonzero(x < 6))

# Are there any values greater than 8?
print(np.any(x > 8))

# Are all values less than 10?
print(np.all(x < 10))
```

b. Boolean Masks

A more powerful pattern than just obtaining a boolean output array is to use boolean arrays as masks. This means that we are selecting particular subsets of the array that satisfy some given conditions by indexing the boolean array.

We don't just want to know if an index holds a value less than 10, we want to get all the values less than 10 themselves.

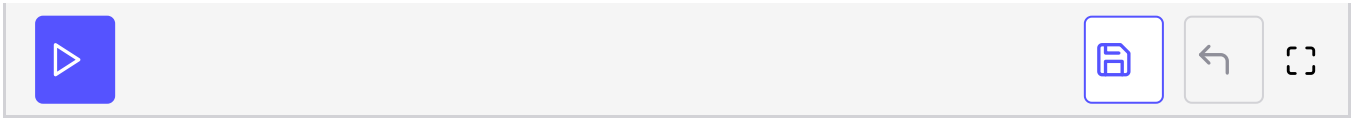
Suppose we have a *3x3* grid with random integers from 0 to 10 and we want an array of all values in the original array that are less than 6. We can achieve this like so:

```
import numpy as np

# Random integers between [0, 10) of shape 3x3
x = np.random.randint(0, 10, (3, 3))
print(x)

# Boolean array
print(x < 6)

# Boolean mask
print(x[x < 6])
```



Why are these operations important?

By combining boolean operations, masking operations, and aggregates, ***we can very quickly answer a lot of useful questions about our dataset***.

Say we are given a population dataset, we can answer questions like:

- What is the minimum age of people in that dataset?
- What is the maximum age?
- How many people in a given country are below the age of 18?
- How many people are above the age of 25 and unemployed?

In general, we can select subsets of the data based on some conditions of interest.

Final Thoughts

Congratulations! We are at the end of our lessons on NumPy 🙌 Of course, we will keep bumping into it in the upcoming lessons as well; especially in the Projects section.

For a deeper dive into all the goodness NumPy has to offer, here is their official documentation.

Last but not the least, before moving on with new concepts, make sure to **test your NumPy knowledge and solidify the concepts learned so far** by completing the exercises in the next lesson.