Numerical Computing with Python and Numpy

The "data" in Data Analytics refers to numerical data, e.g., stock prices, sales figures, sensor measurements, sports scores, database tables, etc. The numpy library provides specialized data structures, functions, and other tools for numerical computing in Python.

Let's work through an example to see why & how to use Numpy for working with numerical data.

> Suppose we want to use climate data like the temperature, rainfall, and humidity to determine if a region is well suited for growing apples. A simple approach for doing this would be to formulate the relationship between the annual yield of apples (tons per hectare) and the climatic conditions like the average temperature (in degrees Fahrenheit), rainfall (in millimeters) & average relative humidity (in percentage) as a linear equation.
>
> yield_of_apples = w1 * temperature + w2 * rainfall + w3 * humidity

We're expressing the yield of apples as a weighted sum of the temperature, rainfall, and humidity. This equation is an approximation since the actual relationship may not necessarily be linear, and there may be other factors involved. But a simple linear model like this often works well in practice.

Based on some statical analysis of historical data, we might come up with reasonable values for the weights w1, w2, w3. Here's an example set of values:

w1, w2, w3 = 0.3, 0.2, 0.5

To begin, we can define some variables to record climate data for a region.

There are different ways to solve this example.

```
w1, w2, w3 = 0.3, 0.2, 0.5
```

```
swat_temp = 73
swat_rainfall = 67
swat_humidity = 43
```

```
swat_yield_apples = swat_temp * w1 + swat_rainfall * w2 + swat_humidity * w3
swat_yield_apples
```

```
swat = [73, 67, 43]
murree = [91, 88, 64]
```

```
weights = [w1, w2, w3]
```

```
zip(swat, weights)
```

```
def crop_yield(region, weights):
    result = 0
    for x, w in zip(region, weights):
        result += x * w
    return result
```

```
crop_yield(swat, weights)
```

Going from Python lists to Numpy arrays

```
import numpy as np
```

```
swat = np.array([73, 67, 43])
```

```
swat
```

```
weights = np.array([w1, w2, w3])
```

```
weights
```

```
weights[0]
```

```
np.dot(swat, weights)
```

```
(swat * weights).sum()
```

Perform the following:

1. Import the numpy package under the name np
2. Create a vector with values ranging from 10 to 49
3. Reverse a vector (first element becomes last)
4. Create a 3x3 matrix with values ranging from 0 to 8
5. Find indices of non-zero elements from [1,2,0,0,4,0]
6. Create a 3x3 identity matrix
7. Create a 3x3x3 array with random values
8. Create a 10x10 array with random values and find the minimum and maximum values
9. Create a random vector of size 30 and find the mean value