**Lab report no 5**

****

**Fall 2022**

**Data Analytics Lab**

**Submitted By**

**Names Registration No**

**Muhammad Ali 19pwcse1801**

**Section: A**

**Date: 12,10,22**

**Submitted to: Engr. Faiz Ullah**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

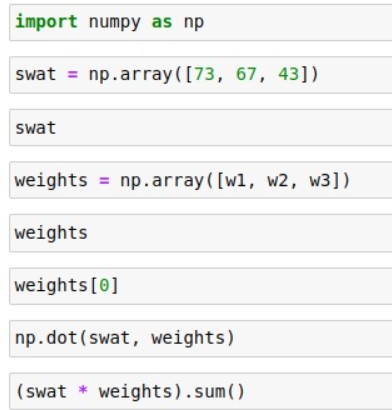
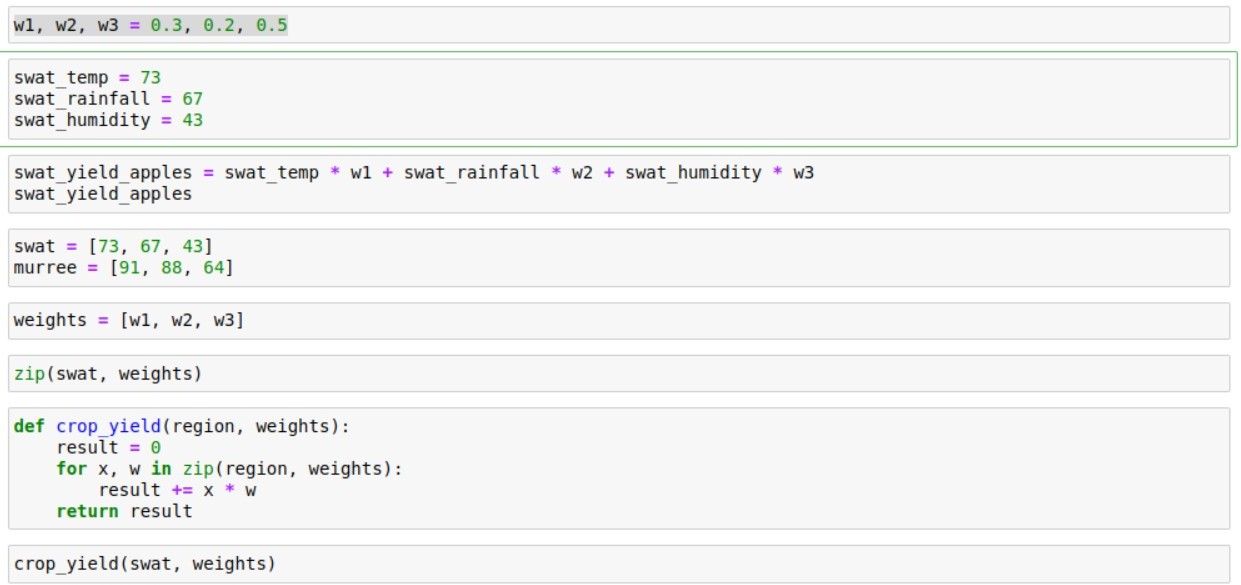
**Numerical Computing with Python and Numpy**

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  s w1, w2, w3 . Here's an example set of values:  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  for the weight  Based on some statical analysis of historical data, we might come up with reasonable values  works well in practice.  linear, and there may be other factors involved. But a simple linear model like this often  humidity. This equation is an approximation since the actual relationship may not necessarily be  We're expressing the yield of apples as a weighted sum of the temperature, rainfall, and | | | | |

Going from Python lists to Numpy arrays



Perform the following:

Import the numpy package under the name np

Create a vector with values ranging from 10 to 49

CODE:

import numpy as np  
v = np.arange(10,49)  
print("Original vector:",v)

**OUTPUT**:

Original vector: [10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48]

Reverse a vector (first element becomes last)

**CODE:**

import numpy as np  
import numpy as np  
x = np.arange(10, 49)  
print("Original array:")  
print(x)  
print("Reverse array:")  
x = x[::-1]  
print(x)

**OUTPUT:**

Original array:

[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

34 35 36 37 38 39 40 41 42 43 44 45 46 47 48]

Reverse array:

[48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25

24 23 22 21 20 19 18 17 16 15 14 13 12 11 10]

**Create a 3x3 matrix with values ranging from 0 to 8**

**CODE:**

import numpy as np  
x = np.arange(0, 9).reshape(3,3)  
print(x)

**OUTPUT:**

[[0 1 2]

[3 4 5]

[6 7 8]]

**Find indices of non-zero elements from [1,2,0,0,4,0]**

**CODE:**

import numpy as np  
li = [1,2,0,0,4,0]  
arr = np.array(li)  
li\_new = list(np.nonzero(arr)[0])  
print(li\_new)

OUTPUT:

[0, 1, 4]

Create a 3x3 identity matrix

CODE:

import numpy as np  
array=np.identity(3)  
print('3x3 matrix:')  
print(array)

OUTPUT:

3x3 matrix:

[[1. 0. 0.]

[0. 1. 0.]

[0. 0. 1.]]

**Create a 3x3x3 array with random values**

**CODE:**

import numpy as np  
x = np.random.random((3,3,3))  
print(x)

OUTPUT:

[[[0.239397 0.70592661 0.16779964]

[0.61538006 0.99244953 0.33006598]

[0.98686364 0.80053492 0.31828398]]

[[0.70054083 0.34139627 0.18742634]

[0.3956967 0.34005877 0.57402976]

[0.0068521 0.13260891 0.67833905]]

[[0.11677414 0.00248551 0.63252639]

[0.98297989 0.59705738 0.41877137]

[0.73964934 0.07535519 0.39161222]]]

**Create a 10x10 array with random values and find the minimum and maximum values**

CODE:

import numpy as np  
x = np.random.random((10,10))  
print("Original Array:")  
print(x)  
xmin, xmax = x.min(), x.max()  
print("Minimum and Maximum Values:")  
print(xmin, xmax)

OUTPUT:

Original Array:

[[0.45471772 0.16372249 0.49516158 0.07404616 0.00194067 0.5540181

0.25698665 0.83819394 0.42228094 0.62116131]

[0.73125664 0.41204534 0.7848792 0.1193337 0.90020367 0.04196206

0.72393924 0.09573403 0.57511383 0.2504495 ]

[0.90849866 0.01978921 0.06474617 0.04607789 0.71181283 0.11312504

0.03476751 0.14576269 0.89036473 0.98524645]

[0.89298734 0.47269423 0.73932716 0.62163781 0.42398385 0.80350249

0.63566599 0.41451035 0.84693476 0.47006796]

[0.39800108 0.9557314 0.96553157 0.22888523 0.52639206 0.48197871

0.57777838 0.60368393 0.73420812 0.55688458]

[0.52572123 0.15817401 0.64632678 0.86029458 0.76056589 0.72413092

0.83278926 0.59013481 0.67000808 0.16448396]

[0.70080223 0.07934383 0.35121364 0.09915054 0.8170931 0.61066889

0.07374335 0.06031893 0.05589868 0.49494716]

[0.87261895 0.77424827 0.53776411 0.70246855 0.44650308 0.04371878

0.6902047 0.31940581 0.28721905 0.83766587]

[0.85655704 0.4278047 0.27468552 0.26172289 0.77627001 0.6680702

0.93644819 0.85729726 0.21194133 0.69850741]

[0.04295163 0.30336031 0.08394411 0.41361745 0.44142514 0.74876409

0.53227264 0.7316164 0.50651469 0.19331186]]

Minimum and Maximum Values:

0.0019406727943312996 0.9852464536033229

**Create a random vector of size 30 and find the mean value**

**CODE:**

import numpy as np  
f = np.random.random((5,6))  
print("The mean value is:",np.mean(f))

**OUTPUT:**

The mean value is: 0.5912525680595817