Task M3 T01

Numerical programming, dataframes and statistical analysis

- Exercise 1

Create a function that, given a one-dimensional Array, gives you a basic statistical summary of the data. If it detects that the array has more than one dimension, it must display an error message.

In [26]:
```
import numpy as np
import pandas as pd

def summary_array(array_number):
    print(array_number.describe())
    return()


def dimension_right(array_number):
    size_array = array_number.ndim
    if size_array == 1:
        print('Array 1D')

    else:
        print('Error!!!')


#array_send=np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
array_send=np.array([1, 2, 3, 4, 5, 6, 7, 8])
df = pd.DataFrame(array_send)
summary_array(df)
dimension_right(array_send)
```

```
             0
count  8.00000
mean   4.50000
std    2.44949
min    1.00000
25%    2.75000
50%    4.50000
75%    6.25000
max    8.00000
Array 1D
```

- Exercise 2

Create a function that generates an NxN square of random numbers between 0 and 100.

In [89]:
```
from numpy import random

def random_N_x_N(N):
    x = random.randint(100, size=(N,N))
    print (x)

random_N_x_N(5)
```

```
[[12 47 87 74 60]
 [24 92 67 16 30]
 [11 97 80 17 62]
 [96 33 24 38 32]
 [ 5 86 65 63 34]]
```

- Exercise 3

Create a function that, given a two-dimensional table (NxM), calculates the totals per row and the totals per column.

In [18]:
```
import pandas as pd
from numpy import random

def generate_array(N,M):
    x= random.randint(100, size=(N,M))
    print(x)
    return(x)

def total_sum_row(result_array):
    print('Totals per row:',np.sum(result_array,axis=1))

def total_sum_column(result_array):
    print('Totals per column:',np.sum(result_array,axis=0))

rows = 2
columns = 5
array_one = generate_array(rows, columns)
total_sum_row(array_one)
total_sum_column(array_one)
```

```
[[35 89 57 43 89]
 [58 18 49 71 56]]
Totals per row: [313 252]
Totals per column: [ 93 107 106 114 145]
```

- Exercise 4

Manually implement a function that calculates the correlation coefficient. Learn about its uses and interpretation.

In [1]:
```
def correlation_coeficient(x,y):
    x_mean=np.mean(x)
    y_mean=np.mean(y)
    print('Mean of x:', x_mean)
    print('Mean of y:', y_mean)

    # Subtract the mean from each value
    xi_xm =x-x_mean
    print ("xi-xm = ",xi_xm)
    yi_ym =y-y_mean
    print ("yi-ym = ",yi_ym)

    # Calculate the numerator of the equation, multiply the last two array
    xi_xm_x_yi_xm = np.prod([xi_xm,yi_ym],axis=0)
    print("(xi-xm)(yi-xm) = ",xi_xm_x_yi_xm)

    # Sum the values of the array
    numerator = np.sum(xi_xm_x_yi_xm)
    print ("Numerator of the coefficient equation : ", numerator)

    # Calculate the denominator of the equation. We take the square of both arrays
    xi_xm_2 = np.power(xi_xm,2)
    print("(xi-xm)^2 = ",xi_xm_2)
    yi_ym_2 = np.power(yi_ym,2)
    print("(yi-ym)^2 = ",yi_ym_2)

    # Calculate the sum of the values of each array
    sum_xi_xm_2 = np.sum(xi_xm_2)
    print ("summation (xi-xm)^2 = ",sum_xi_xm_2)
    sum_yi_ym_2 = np.sum(yi_ym_2)
    print ("summation (yi-ym)^2 = ",sum_yi_ym_2)

    # Multiply both sums
    prod_sum = sum_xi_xm_2*sum_yi_ym_2
    print('Summation',prod_sum)

    # Take the square root of the last value obtained
    denominator = np.sqrt(prod_sum)
    print("Denominator of the coefficient equation: ",denominator)

    # Divide the numerator and denominator to get the coefficient.
    result = numerator / denominator

    return (result)
```

Formula

In [2]:
```
array_one = [156,167,160,150,149]
array_two = [55,64.5,58,60.9,46]

coefficient = correlation_coeficient(array_one,array_two)
```

```
Mean of x: 156.4
Mean of y: 56.879999999999995
xi-xm =  [-0.4 10.6  3.6 -6.4 -7.4]
yi-ym =  [ -1.88   7.62   1.12   4.02 -10.88]
(xi-xm)(yi-xm) =  [  0.752  80.772   4.032 -25.728  80.512]
Numerator of the coefficient equation :  140.34
(xi-xm)^2 =  [  0.16 112.36  12.96  40.96  54.76]
(yi-ym)^2 =  [  3.5344  58.0644   1.2544  16.1604 118.3744]
summation (xi-xm)^2 =  221.2
summation (yi-ym)^2 =  197.38799999999998
Summation 43662.22559999999
Denominator of the coefficient equation:  208.9550803402492
```

In [81]:
```
print ("Correlation coefficient, manual calculation = ", coefficient )
```

```
Correlation coefficient, manual calculation =  0.6716276042270868
```

Calculation of the coefficient with the numpy function

In [84]:
```
coeficiente_corr = np.corrcoef(array_one,array_two)
print("Correlation coefficient, calculation with numpy function = ")
print (coeficiente_corr)
```

```
Correlation coefficient, calculation with numpy function =
[[1.        0.6716276]
 [0.6716276 1.       ]]
```