# PRACTICE #01

## NumPy

**(*Keyword:* NumPy*)***

## I. Goals

- Students can use **Python** with **NumPy** library to implement basic statistical functions.

## II. Introduction

- **NumPy** is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- **NumPy** has included quite a few useful statistical functions for finding minimum, maximum, percentile standard deviation and variance, etc. from the given elements in the array.

## III. Content

1. Prepare the necessary programming environment.
   - **Python** programming language, minimum recommend version **3.6**
   - IDE / Text Editor: recommend JetBrains PyCharm Community (**PyCharm**) or Microsoft Visual Studio Code (**VS Code**).
   - Recommend using a **virtual environment** for developing:
   https://www.jetbrains.com/help/pycharm/creating-virtual-environment.html
   https://code.visualstudio.com/docs/python/environments
   - Recommend installing libraries by using ***pip****:*
   https://www.w3schools.com/python/python_pip.asp
   https://packaging.python.org/tutorials/installing-packages/

2. Features
   Implement some basic functions by using Python and NumPy:
   - Mean, Median, Max, Min, Range
   - Variance & Standard Deviation
   - Correlation, etc.

## IV. Requirements

1. The directory structure of the compressed submission

   - *doc*: report files include *MSSV_report_p01.doc/docx/latex source* and *MSSV_report_p01.pdf*
   - *source*: contains entire source code, removed temporary files, intermediate compiled files if it has...
   - *bonus*: optional, for plus points, if available

   If the student submits only one Jupyter Notebook file (naming *MSSV_p01*), it must be accompanied by clear descriptions and explanations as in the report files above.

2. Other requirements

   - The report should be presented clearly and intuitively: a self-scoring table (assessment) of the results of the work compared to the corresponding requirements (0-100%), a list of the functions included in the program with proof images, summarize the usage and implementation (for example: through pseudo-code, description of methods, or how to do it, *do not copy the source code into the report*).
   - The source code needs to be commented on the corresponding lines.

## V. Additional guidance

1. Install NumPy with a virtual environment by using pip

```
pip install virtualenv
virtualenv your-env
source your-env/bin/activate
(your-env) pip install numpy
```

Import NumPy

```
import numpy as np
```

2. NumPy basic functions for statistical

- Central Tendency: Mean, Median

X = [1 2 3 4 5]

mean(X) = $\frac{1+2+3+4+5}{5}$ = 3 and median(X) = 3

```
    # Mean
X = [1, 2, 3, 4, 5]
print("Mean X = ", np.mean(X))

X = np.array([[1, 2], [3, 4]])
print("Mean X = ", np.mean(X))
print("Mean X with axis = 0: ", np.mean(X, axis=0))
print("Mean X with axis = 1: ", np.mean(X, axis=1))

a = np.zeros((2, 512 * 512), dtype=np.float32)
a[0, :] = 1.0
a[1, :] = 0.1
```

```
print("a.shape: ", a.shape)
print("mean a = ", np.mean(a))

print("mean a = ", np.mean(a, dtype=np.float64))
# Median
X = np.array([2, 5, 3, 1, 7])
Y = np.array([2, 1, 8, 5, 7, 9])

print("Median X = ", np.median(X))
print("Median Y = ", np.median(Y))

arr = np.array([[7, 4, 2], [3, 9, 5]])
print("median arr (axis = 0) = ", np.median(arr, axis=0))
print("median arr (axis = 1) = ", np.median(arr, axis=1))
```

Handle with NaN - not a number

```
# Mean & Median with NaN
x = np.array([2, np.nan, 5, 9])
print("mean = ", np.mean(x))
print("median = ", np.median(x))
print("mean = ", np.nanmean(x))
print("median = ", np.nanmedian(x))
```

- Variance & Standard Deviation

$$\sigma^2 = \frac{\sum(\chi - \mu)^2}{N}$$

X = [1, 2, 4, 6, 7]

mean(X) = $\frac{1+2+4+6+7}{5}$ = 4

Y = [-3, -2, 0, 2, 3]

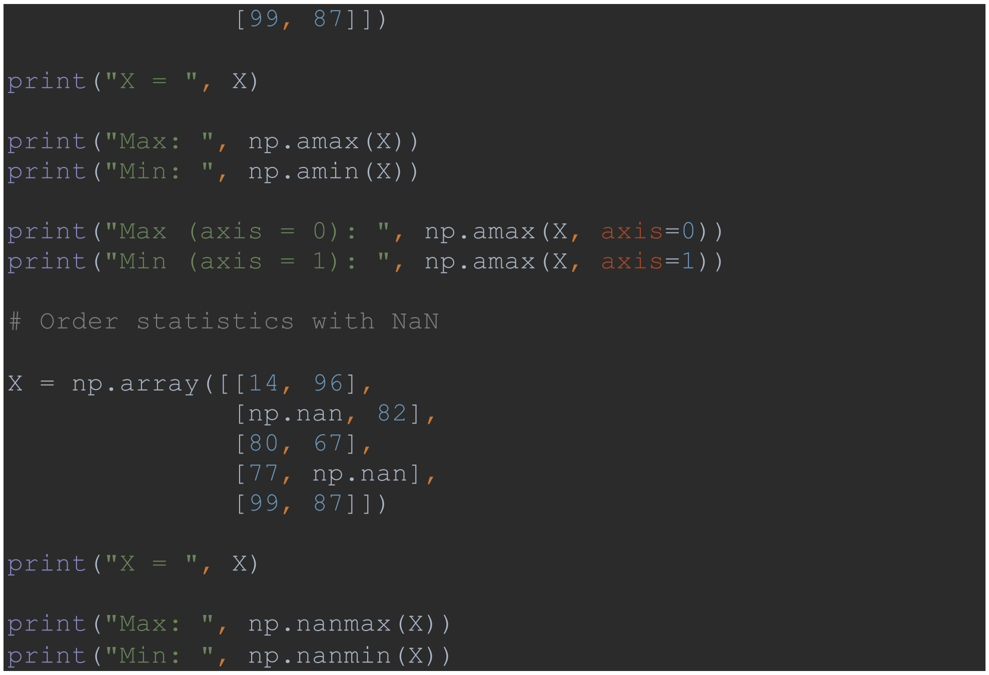
$$\sigma^2 = \frac{\sum(\chi - \mu)^2}{N} = \frac{(-3)^2 + (-2)^2 + 0^2 + 2^2 + 3^2}{5} = 5.2$$

$\sigma = \sqrt{5.2} \approx 2.28$

```
# Variance & Standard Deviation
X = [19, 33, 51, 22, 18, 13, 45, 24, 58, 11, 25, 27, 26, 29]

print("Variance: ", np.var(X))
print("Standard Deviation: ", np.std(X))


# Variance & Standard Deviation with NaN
A = np.array([1, np.nan, 3, 4])
print("var = ", np.var(A))
print("std = ", np.std(A))
print("nan var = ", np.nanvar(A))
print("nan std = ", np.nanstd(A))
```

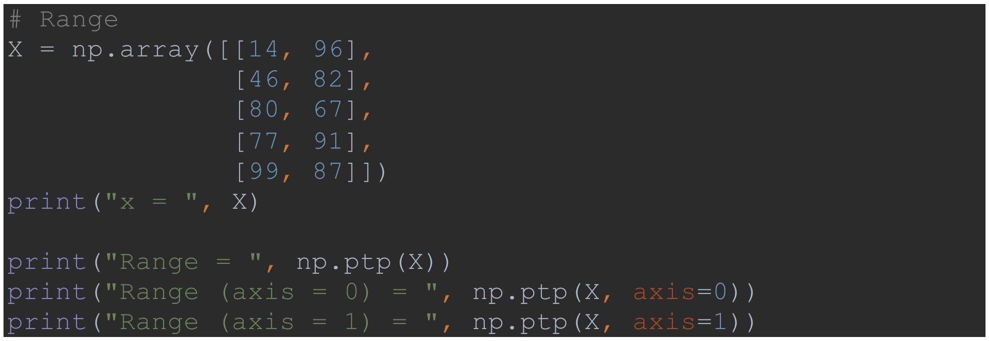
- Order statistics

```
# Order statistics
X = np.array([[14, 96],
              [46, 82],
              [80, 67],
              [77, 91],
```



```
              [99, 87]])

print("X = ", X)

print("Max: ", np.amax(X))
print("Min: ", np.amin(X))

print("Max (axis = 0): ", np.amax(X, axis=0))
print("Min (axis = 1): ", np.amax(X, axis=1))

# Order statistics with NaN

X = np.array([[14, 96],
              [np.nan, 82],
              [80, 67],
              [77, np.nan],
              [99, 87]])

print("X = ", X)

print("Max: ", np.nanmax(X))
print("Min: ", np.nanmin(X))
```

- Range



```
# Range
X = np.array([[14, 96],
              [46, 82],
              [80, 67],
              [77, 91],
              [99, 87]])
print("x = ", X)

print("Range = ", np.ptp(X))
print("Range (axis = 0) = ", np.ptp(X, axis=0))
print("Range (axis = 1) = ", np.ptp(X, axis=1))
```