Student ID: 21127690

Name: Ngô Nguyễn Thanh Thanh

Report: Analysis of Basic Statistical Functions Using NumPy

I. Evaluation summary:

Function	Requirement Met(%)	Notes
Mean	100%	
Median	100%	
Variance	100%	
Standard Deviation	100%	
Order Statistics(min, max,range)	100%	
Correlation	100%	

II. List of funtion:

Including image proof below

1. Mean:

```
# Mean
X = [1, 2, 3, 4, 5]
print("Mean:", np.mean(X))  # Tính trung bình của mảng X

X = np.array([[1, 2], [3, 4]])
print("Mean X:", np.mean(X))  # Tính trung bình của mảng X
print("Mean X with axis = 0: ", np.mean(X, axis=0))  # Trung bình theo trục 0
print("Mean X with axis = 1: ", np.mean(X, axis=1))  # Trung bình theo trục 1
```

```
#MEAN
Mean: 3.0
Mean X: 2.5
Mean X with axis = 0:  [2. 3.]
Mean X with axis = 1:  [1.5 3.5]
Creating array 'a' with zeros
a.shape:  (2, 262144)
mean a =  0.54999924
mean a =  0.5500000007450581
```

2. Median

```python
# Median
X = np.array([2, 5, 3, 1, 7])
Y = np.array([2, 1, 8, 5, 7, 9])

print("Median X:", np.median(X))  # Tính trung vị của mảng X
print("Median Y:", np.median(Y))  # Tính trung vị của mảng Y

arr = np.array([[7, 4, 2], [3, 9, 5]])
print("median arr (axis = 0) =", np.median(arr, axis=0))  # Tính trung vị theo trục 0
print("median arr (axis = 1) =", np.median(arr, axis=1))  # Tính trung vị theo trục 1

# Mean & Median with NaN
X = np.array([2, np.nan, 5, 9])
print("Mean X:", np.mean(X))  # Tính trung bình của mảng X với xử lý NaN
print("Median X:", np.median(X))  # Tính trung vị của mảng X với xử lý NaN
print("Mean = ", np.nanmean(X))  # Tính trung bình của mảng X với xử lý NaN
print("Median =", np.nanmedian(X))  # Tính trung vị của mảng X với xử lý NaN
```

```
#MEDIAN
Median X: 3.0
Median Y: 6.0
median arr (axis = 0) = [5.  6.5 3.5]
median arr (axis = 1) = [4. 5.]
#MEAN & MEDIAN with NaN
Mean X: nan
Median X: nan
Mean =  5.333333333333333
Median = 5.0
```

3. Variance

```python
# Variance & standard deviation
X = [19, 33, 51, 22, 18, 13, 45, 24, 58, 11]
print("Variance X:", np.var(X))  # Tính phương sai của mảng X
print("Standard deviation X:", np.std(X))  # Tính độ lệch chuẩn của mảng X

# Variance & standard deviation with NaN
A = np.array([1, np.nan, 3, 4])
print("var = ", np.var(A))  # Tính phương sai của mảng A với xử lý NaN
print("std = ", np.std(A))  # Tính độ lệch chuẩn của mảng A với xử lý NaN
print("nan var =", np.nanvar(A))  # Tính phương sai của mảng A với xử lý NaN
print("nan std = ", np.nanstd(A))  # Tính độ lệch chuẩn của mảng A với xử lý NaN
```

```
#VARIANCE & STANDARD DEVIATION
Variance X: 247.04000000000002
Standard deviation X: 15.717506163510802
#VARIANCE & STANDARD DEVIATION with NaN
var =  nan
std =  nan
nan var = 1.5555555555555554
nan std =  1.247219128924647
```

4. Standard Deviation

```
# Variance & standard deviation
X = [19, 33, 51, 22, 18, 13, 45, 24, 58, 11]
print("Variance X:", np.var(X))  # Tính phương sai của mảng X
print("Standard deviation X:", np.std(X))  # Tính độ lệch chuẩn của mảng X

# Variance & standard deviation with NaN
A = np.array([1, np.nan, 3, 4])
print("var = ", np.var(A))  # Tính phương sai của mảng A với xử lý NaN
print("std = ", np.std(A))  # Tính độ lệch chuẩn của mảng A với xử lý NaN
print("nan var =", np.nanvar(A))  # Tính phương sai của mảng A với xử lý NaN
print("nan std = ", np.nanstd(A))  # Tính độ lệch chuẩn của mảng A với xử lý NaN
```

```
#VARIANCE & STANDARD DEVIATION
Variance X: 247.04000000000002
Standard deviation X: 15.717506163510802
#VARIANCE & STANDARD DEVIATION with NaN
var =  nan
std =  nan
nan var = 1.5555555555555554
nan std =  1.247219128924647
```

5. Min

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

print("min X = ", np.min(X))  # Tìm giá trị nhỏ nhất trong mảng X
print("max X = ", np.max(X))  # Tìm giá trị lớn nhất trong mảng X

print("Max: (axis =0) = ", np.max(X, axis=0))  # Tìm giá trị lớn nhất theo trục 0
print("Min: (axis = 1) = ", np.min(X, axis=1))  # Tìm giá trị nhỏ nhất theo trục 1

# Order statistics with NaN
X = np.array([[14, 96],
              [np.nan, 82],
              [77, np.nan],
              [99, 87]])
print("X = ", X)

print("Max: ", np.nanmax(X))  # Tìm giá trị lớn nhất với xử lý NaN
print("Min: ", np.nanmin(X))  # Tìm giá trị nhỏ nhất với xử lý NaN
```

```
#ORDER STATISTICS
X=  [[14 96]
 [46 82]
 [80 67]
 [77 91]
 [99 87]]
min X =  14
max X =  99
Max: (axis =0) =  [99 96]
Min: (axis = 1) =  [14 46 67 77 87]
#ORDER STATISTICS with NaN
X =  [[14. 96.]
 [nan 82.]
 [77. nan]
 [99. 87.]]
Max:  99.0
Min:  14.0
```

6.Max

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

print("min X = ", np.min(X))  # Tìm giá trị nhỏ nhất trong mảng X
print("max X = ", np.max(X))  # Tìm giá trị lớn nhất trong mảng X

print("Max: (axis =0) = ", np.max(X, axis=0))  # Tìm giá trị lớn nhất theo trục 0
print("Min: (axis = 1) = ", np.min(X, axis=1))  # Tìm giá trị nhỏ nhất theo trục 1

# Order statistics with NaN
X = np.array([[14, 96],
              [np.nan, 82],
              [77, np.nan],
              [99, 87]])
print("X = ", X)

print("Max: ", np.nanmax(X))  # Tìm giá trị lớn nhất với xử lý NaN
print("Min: ", np.nanmin(X))  # Tìm giá trị nhỏ nhất với xử lý NaN
```

```
#ORDER STATISTICS
X=  [[14 96]
 [46 82]
 [80 67]
 [77 91]
 [99 87]]
min X =  14
max X =  99
Max: (axis =0) =  [99 96]
Min: (axis = 1) =  [14 46 67 77 87]
#ORDER STATISTICS with NaN
X =  [[14. 96.]
 [nan 82.]
 [77. nan]
 [99. 87.]]
Max:  99.0
Min:  14.0
```

7. Range

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

print("Range X = ", np.ptp(X))  # Tính phạm vi của mảng X
print("Range X (axis = 0) = ", np.ptp(X, axis=0))  # Tính phạm vi theo trục 0
print("Range X (axis = 1) = ", np.ptp(X, axis=1))  # Tính phạm vi theo trục 1
```

```
#RANGE
x =  [[14 96]
 [46 82]
 [80 67]
 [77 91]
 [99 87]]
Range X =  85
Range X (axis = 0) =  [85 29]
Range X (axis = 1) =  [82 36 13 14 12]
```

8. Correlation

```
# Correlation
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([5, 4, 3, 2, 1])

correlation_matrix = np.corrcoef(arr1, arr2)  # Tính ma trận tương quan
correlation = correlation_matrix[0, 1]  # Lấy giá trị hệ số tương quan

print("Correlation:", correlation)  # In ra hệ số tương quan
```

```
#CORRELATION
Correlation: -0.9999999999999999
```

Notes:

- All of the functions mentioned above have two versions: one for handling regular numbers and another for handling non-numeric values.
- In summary, NumPy's functions automatically handle non-numeric values by excluding them from calculations where appropriate. This behavior ensures that the statistical analysis is performed accurately without being affected by NaN values in the input data.

III. Function Summaries and Implementation

Desciption methods of Numpy library and examples:

Because the task requires utilizing NumPy to compute basic statistical values, therefore the code only calls NumPy library functions such as np.median, np.mean, etc. The explanation below elucidates how the statistical values are computed behind the scenes by the functions in the NumPy library.

- **Mean:** Calculates the arithmetic mean of a dataset. It is implemented using the formula sum of elements divided by the total number of elements.
- **Median:** Finds the middle value of a dataset. It is implemented by sorting the dataset and finding the middle value or the average of the two middle values.
  EX:
  - Central Tendency: Mean, Median

  $X = [1\ 2\ 3\ 4\ 5]$

  $\text{mean}(X) = \frac{1+2+3+4+5}{5} = 3$ and $\text{median}(X) = 3$
- **Variance:** Measures the spread of data points around the mean. It is calculated by finding the average of the squared differences from the mean.
- **Standard Deviation**: Represents the average deviation of data points from the mean. It is the square root of the variance.

  EX:

- Variance & Standard Deviation

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

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

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

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

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

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

- **Order Statistics**: Includes functions like min and max, which find the minimum and maximum values in a dataset, respectively.
- **Range:** Indicates the spread or difference between the maximum and minimum values in a dataset. It provides insight into the variability of the data.

  EX:

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

  Range of X = 99 – 14 = 85 (in the scope of full array)
  Range X (axis = 0) = [85 29] (in the first colum the range is 99 -14= 85, in the second colum the range is 96 – 67 = 29)
  Range X (axis = 1) = [82 36 13 14 12] (similar to above but with each row)
  96 – 14 = 82, 82 – 46 = 36, 80 – 67 = 13, 91 – 77 = 14, 99 – 87 = 12)
- **Correlation:** Measures the strength and direction of the linear relationship between two variables. It is calculated using covariance and standard deviation.

  EX:

```
# Correlation
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([5, 4, 3, 2, 1])
```

1. **Covariance (cov):**
   $$\text{cov}(X,Y) = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$
2. **Standard Deviation (std):**
   $$\text{std}(X) = \sqrt{\frac{\sum(X_i - \bar{X})^2}{n-1}}$$
3. **Correlation (corr):**
   $$\text{corr}(X,Y) = \frac{\text{cov}(X,Y)}{\text{std}(X) \times \text{std}(Y)}$$

  After substituting the values above into the formula, we obtain the result: -0.99999

V. The results:

Full image proof:

```
[Running] python -u "c:\Users\hp\OneDrive\Documents\thanh thanh\nam3\hocki2\Thống kê & TGMT\lab\lab1\lab1.py"
#MEAN
Mean: 3.0
Mean X: 2.5
Mean X with axis = 0:  [2. 3.]
Mean X with axis = 1:  [1.5 3.5]
Creating array 'a' with zeros
a.shape:  (2, 262144)
mean a =  0.54999924
mean a =  0.5500000007450581
#MEDIAN
Median X: 3.0
Median Y: 6.0
median arr (axis = 0) = [5.  6.5 3.5]
median arr (axis = 1) = [4. 5.]
#MEAN & MEDIAN with NaN
Mean X: nan
Median X: nan
Mean =  5.333333333333333
Median = 5.0
#VARIANCE & STANDARD DEVIATION
Variance X: 247.04000000000002
Standard deviation X: 15.717506163510802
#VARIANCE & STANDARD DEVIATION with NaN
var =  nan
std =  nan
nan var = 1.5555555555555554
nan std =  1.247219128924647
```

```
#ORDER STATISTICS
X=  [[14 96]
 [46 82]
 [80 67]
 [77 91]
 [99 87]]
min X =  14
max X =  99
Max: (axis =0) =  [99 96]
Min: (axis = 1) =  [14 46 67 77 87]
#ORDER STATISTICS with NaN
X =  [[14. 96.]
 [nan 82.]
 [77. nan]
 [99. 87.]]
Max:  99.0
Min:  14.0
#RANGE
x =  [[14 96]
 [46 82]
 [80 67]
 [77 91]
 [99 87]]
Range X =  85
Range X (axis = 0) =  [85 29]
Range X (axis = 1) =  [82 36 13 14 12]
#CORRELATION
Correlation: -0.9999999999999999

[Done] exited with code=0 in 0.561 seconds
```