

Lab Assignment 19

Student Name : Aryan Verma

Student Id: AF0417100

Topic: Numpy stastical functions

KEY POINTS OF NUMPY:

- MEMORY EFFICIENCY: NUMPY ARRAYS CONSUME LESS MEMORY COMPARED TO TRADITIONAL PYTHON LISTS BECAUSE THEY STORE ELEMENTS OF THE SAME TYPE, WHICH ALLOWS FOR MORE EFFICIENT MEMORY USAGE.

PERFORMANCE OPTIMIZATION: OPERATIONS ON NUMPY ARRAYS ARE IMPLEMENTED IN C, WHICH LEADS TO SIGNIFICANT SPEED IMPROVEMENTS FOR MATHEMATICAL COMPUTATIONS OVER PURE PYTHON IMPLEMENTATIONS.

N-DIMENSIONAL SUPPORT: WHILE PYTHON LISTS CAN BE NESTED TO CREATE MULTI-DIMENSIONAL STRUCTURES, NUMPY PROVIDES A NATIVE N-DIMENSIONAL ARRAY OBJECT (NDARRAY), MAKING IT EASIER TO WORK WITH COMPLEX DATA STRUCTURES.

ADVANCED INDEXING AND SLICING: NUMPY ALLOWS FOR MORE ADVANCED INDEXING AND SLICING TECHNIQUES, ENABLING USERS TO ACCESS AND MANIPULATE ARRAY ELEMENTS IN FLEXIBLE AND POWERFUL WAYS.

BUILT-IN FUNCTIONS: THE LIBRARY COMES WITH NUMEROUS BUILT-IN FUNCTIONS THAT CAN OPERATE ON ENTIRE ARRAYS, SUCH AS SORTING,

SEARCHING, AND STATISTICAL OPERATIONS, FURTHER ENHANCING ITS USABILITY.

INTEROPERABILITY: NUMPY CAN EASILY INTERFACE WITH LIBRARIES THAT REQUIRE C OR FORTRAN LIBRARIES, ALLOWING FOR THE INTEGRATION OF LEGACY CODE OR HIGH-PERFORMANCE COMPUTATIONAL ROUTINES.

DATA TYPE SUPPORT: NUMPY PROVIDES EXTENSIVE SUPPORT FOR VARIOUS DATA TYPES, INCLUDING INTEGERS, FLOATS, COMPLEX NUMBERS, AND MORE, ALLOWING FOR GREATER VERSATILITY IN NUMERICAL COMPUTATIONS.

DATA BROADCASTING: THE BROADCASTING RULES IN NUMPY ALLOW FOR AUTOMATIC EXPANSION OF ARRAYS DURING ARITHMETIC OPERATIONS, ELIMINATING THE NEED TO MANUALLY ADJUST SHAPES.

EXTENSIVE DOCUMENTATION AND COMMUNITY SUPPORT: NUMPY HAS COMPREHENSIVE DOCUMENTATION AND A LARGE USER COMMUNITY, MAKING IT EASIER FOR BEGINNERS TO LEARN AND FOR EXPERIENCED USERS TO FIND SOLUTIONS TO COMPLEX PROBLEMS.

ECOSYSTEM INTEGRATION: BEYOND INTEGRATION WITH LIBRARIES LIKE PANDAS AND MATPLOTLIB, NUMPY SERVES AS THE FOUNDATION FOR MANY SCIENTIFIC COMPUTING LIBRARIES IN PYTHON, INCLUDING SCIPY AND TENSORFLOW, ENHANCING ITS ROLE IN THE PYTHON ECOSYSTEM.

SOME FUNCTIONS USED IN NUMPY:

NUMPY.STD(): COMPUTES THE STANDARD DEVIATION OF ARRAY ELEMENTS, PROVIDING INSIGHT INTO THE VARIABILITY OR DISPERSION OF THE DATA.

NUMPY.VAR(): CALCULATES THE VARIANCE OF AN ARRAY, INDICATING HOW MUCH THE VALUES DIFFER FROM THE MEAN ON AVERAGE.

NUMPY.MEDIAN(): DETERMINES THE MEDIAN VALUE OF AN ARRAY, REPRESENTING THE MIDDLE VALUE THAT SEPARATES THE HIGHER HALF FROM THE LOWER HALF OF THE DATA.

NUMPY.PERCENTILE(): COMPUTES THE NTH PERCENTILE OF THE DATA, ALLOWING USERS TO UNDERSTAND THE DISTRIBUTION OF VALUES WITHIN THE DATASET.

NUMPY.MIN(): RETURNS THE MINIMUM VALUE IN AN ARRAY, USEFUL FOR IDENTIFYING THE SMALLEST DATA POINT IN A DATASET.

NUMPY.MAX(): RETURNS THE MAXIMUM VALUE IN AN ARRAY, HELPING TO IDENTIFY THE LARGEST DATA POINT IN A DATASET.

NUMPY.CORRELATE(): CALCULATES THE CORRELATION COEFFICIENT BETWEEN TWO ARRAYS, ASSESSING THE STRENGTH AND DIRECTION OF A LINEAR RELATIONSHIP BETWEEN THEM.

NUMPY.COV(): COMPUTES THE COVARIANCE MATRIX OF TWO OR MORE DATASETS, PROVIDING A MEASURE OF HOW MUCH TWO RANDOM VARIABLES VARY TOGETHER.

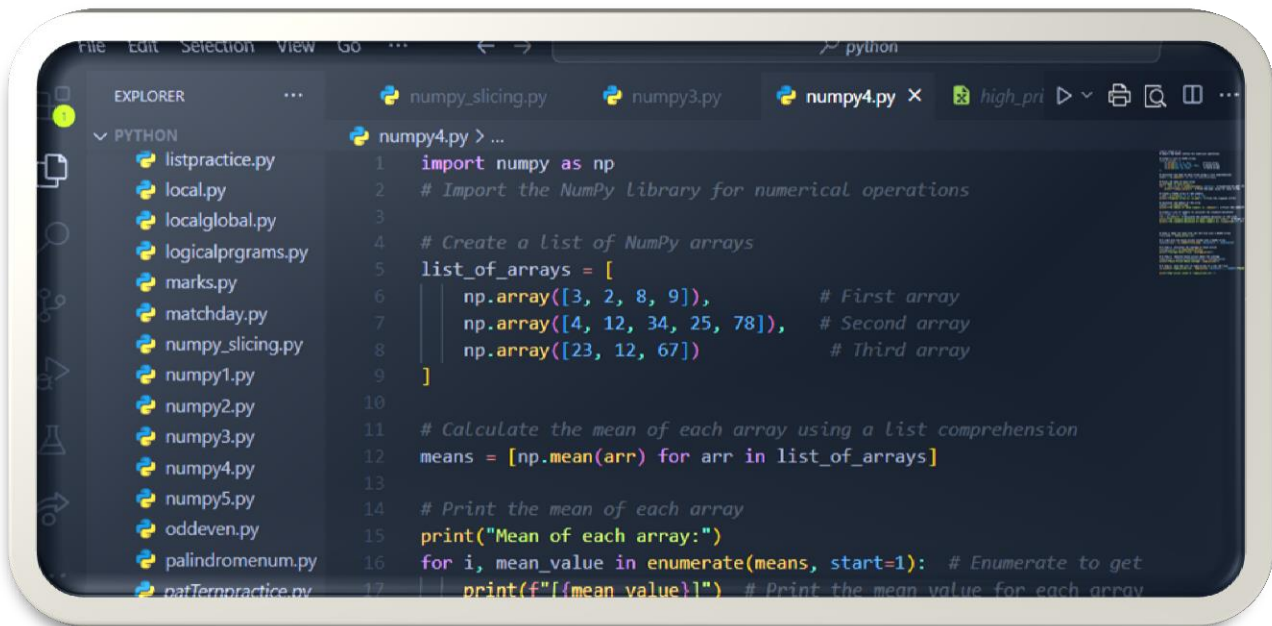
NUMPY.HISTOGRAM(): COMPUTES THE HISTOGRAM OF A DATASET, WHICH VISUALIZES THE FREQUENCY DISTRIBUTION OF A SET OF VALUES.

NUMPY.UNIQUE(): IDENTIFIES UNIQUE ELEMENTS IN AN ARRAY, USEFUL FOR DATA DEDUPLICATION AND UNDERSTANDING THE DIVERSITY OF THE DATASET.

1. How to find the mean of every NumPy array in the given list?

Input: list = [np.array([3, 2, 8, 9]), np.array([4, 12, 34, 25, 78]), np.array([23, 12, 67])]

Ans:

A screenshot of a Python IDE with a dark theme. The Explorer panel on the left shows a file named 'numpy4.py' selected. The main editor window displays the following Python code:

```
1 import numpy as np
2 # Import the NumPy library for numerical operations
3
4 # Create a list of NumPy arrays
5 list_of_arrays = [
6     np.array([3, 2, 8, 9]),           # First array
7     np.array([4, 12, 34, 25, 78]),    # Second array
8     np.array([23, 12, 67])           # Third array
9 ]
10
11 # Calculate the mean of each array using a list comprehension
12 means = [np.mean(arr) for arr in list_of_arrays]
13
14 # Print the mean of each array
15 print("Mean of each array:")
16 for i, mean_value in enumerate(means, start=1): # Enumerate to get
17     print(f"[{mean_value}]") # Print the mean value for each array
```

Output:

```
PS D:\python> python -u "d:\python\numpy4.py"
Mean of each array:
[5.5]
[30.6]
[34.0]
```

2. Compute the median of the flattened NumPy array Input:
`x_odd = np.array([1, 2, 3, 4, 5, 6, 7])`

Ans:

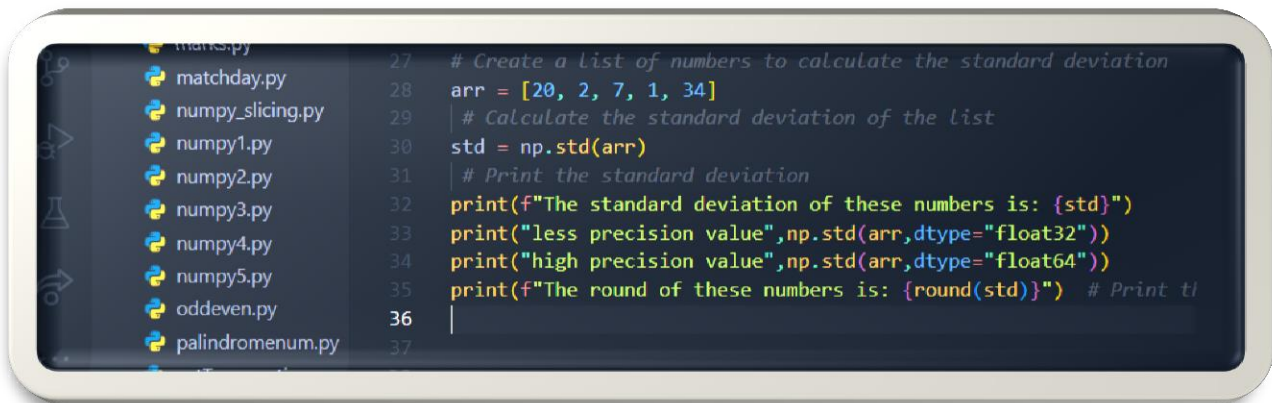
```
19 # Create a NumPy array of odd numbers
20 x_odd = np.array([1, 2, 3, 4, 5, 6, 7])
21 print(f"Original array is: {x_odd}") # Print the original array
22
23 # Calculate the median of the array
24 median = np.median(x_odd)
25 print(f"The median of these numbers is: {median}") # Print the median
26
```

output:

```
Original array is: [1
 2 3 4 5 6 7]
The median of these numbers is: 4.0
```

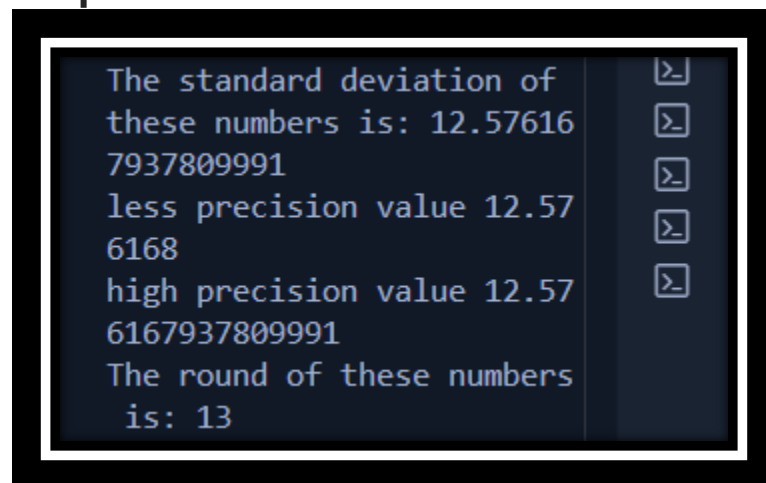
3. Compute the standard deviation of the NumPy array Input:
arr = [20, 2, 7, 1, 34]

Ans:



```
27 # Create a list of numbers to calculate the standard deviation
28 arr = [20, 2, 7, 1, 34]
29 # Calculate the standard deviation of the list
30 std = np.std(arr)
31 # Print the standard deviation
32 print(f"The standard deviation of these numbers is: {std}")
33 print("less precision value", np.std(arr, dtype="float32"))
34 print("high precision value", np.std(arr, dtype="float64"))
35 print(f"The round of these numbers is: {round(std)}") # Print the rounded value
36
37
```

output:

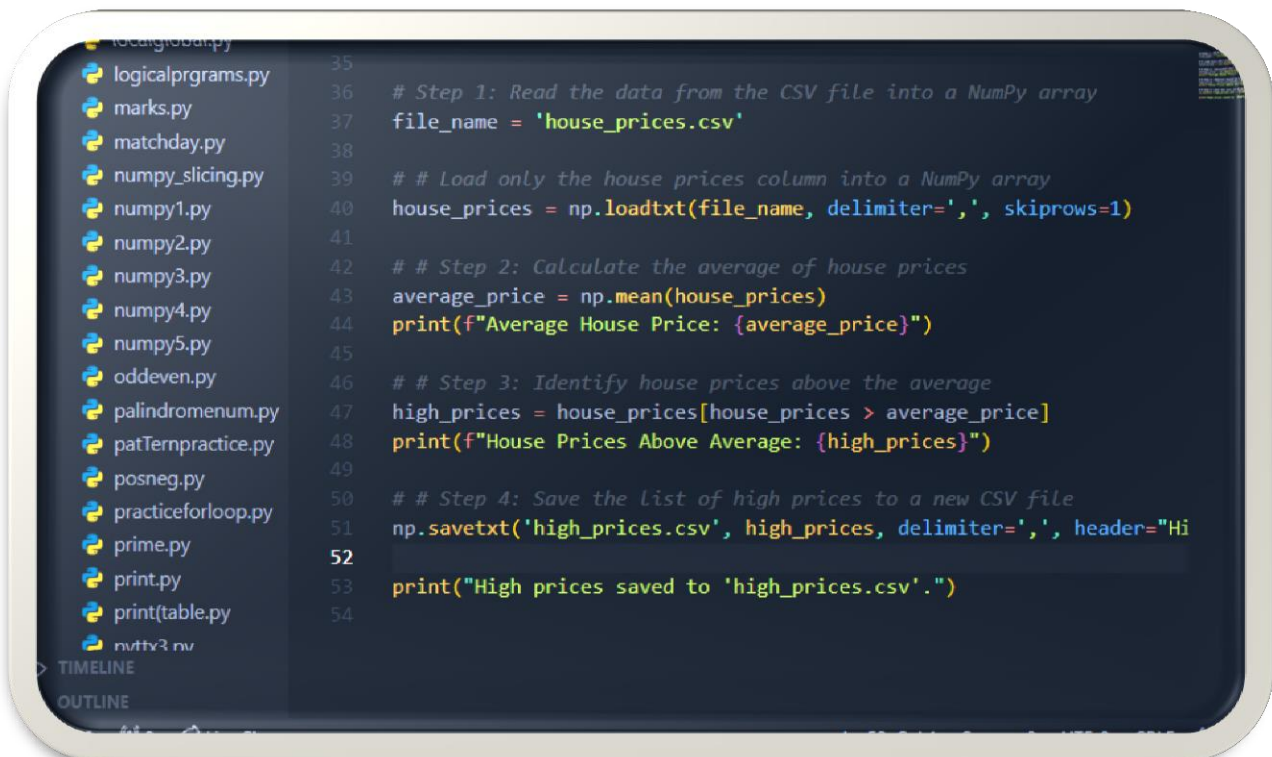


```
The standard deviation of
these numbers is: 12.57616
7937809991
less precision value 12.57
6168
high precision value 12.57
6167937809991
The round of these numbers
is: 13
```


4. Suppose you have a CSV file named 'house_prices.csv' with price information, and you want to perform the following operations:

- 1. Read the data from the CSV file into a NumPy array.
- 2. Calculate the average of house prices.
- 3. Identify house price above the average.
- 4. Save the list of high prices to a new CSV file. Note: Download 'house_prices.csv' file from LMS.

Ans:



```
35
36 # Step 1: Read the data from the CSV file into a NumPy array
37 file_name = 'house_prices.csv'
38
39 # # Load only the house prices column into a NumPy array
40 house_prices = np.loadtxt(file_name, delimiter=',', skiprows=1)
41
42 # # Step 2: Calculate the average of house prices
43 average_price = np.mean(house_prices)
44 print(f"Average House Price: {average_price}")
45
46 # # Step 3: Identify house prices above the average
47 high_prices = house_prices[house_prices > average_price]
48 print(f"House Prices Above Average: {high_prices}")
49
50 # # Step 4: Save the list of high prices to a new CSV file
51 np.savetxt('high_prices.csv', high_prices, delimiter=',', header="Hi
52
53 print("High prices saved to 'high_prices.csv'.")
54
```

output:

```
Average House Price: 50374.32274881517
House Prices Above Average: [ 75000.  52778.  58500. ... 187528. 187529. 187530.]
High prices saved to 'high_prices.csv'.
PS D:\python>
```

3.12.4 64-bit Go Live Prettier