3.1.1. Numpy array operations

16:05

Write a python program to demonstrate the usage of ndim, shape and size for a Numpy Array. The program should create a NumPy array using the entered elements and display it. Assume all input elements are valid numeric values.

Input Format:

- User inputs the number of rows and columns with space separated values.
- User inputs elements of the array row-wise followed line by line, separated by spaces.

Output Format:

- The created NumPy array based on the input dimensions and elements.
- Dimensions (ndim): Number of dimensions of the array.
- Shape: Tuple representing the shape of the array (number of rows, number of columns).
- Size: Total number of elements in the array.

Note: Use reshape() function to reshape the input array with the specified number of rows

Sample Test Cases +

Explorer

numpyarr...

Submit

```
1  import numpy as np
2  rows,cols= list(map(int,input().split()))
3  matrix=[]
4  for i in range(rows):
5      row=list(map(int,input().split()))
6      matrix.append(row)
7  matrix=np.array(matrix).reshape(rows,cols)
8
9  print(matrix)
10 print(matrix.ndim)
11 print(matrix.shape)
12 print(matrix.size)
```

Debugger

Terminal Test cases

3.2.1. Numpy: Matrix Operations

04:04

The given code takes two 3×3 matrices, matrix_a, and matrix_b, as input from the user and converts them into NumPy arrays.

Task:

You are required to compute and display the results of the following matrix operations:

1. **Addition** (matrix_a + matrix_b)
2. **Subtraction** (matrix_a − matrix_b)
3. **Element-wise Multiplication** (matrix_a * matrix_b)
4. **Matrix Multiplication** (matrix_a · matrix_b)
5. **Transpose of Matrix A**

Input Format:

- The user will input 3 rows for matrix_a, each containing 3 integers separated by spaces.
- Similarly, the user will input 3 rows for matrix_b, each containing 3 integers separated by spaces.

Sample Test Cases +

Explorer

matrixOp...

Submit

```python
import numpy as np

# Input matrices
print("Enter Matrix A:")
matrix_a = np.array([list(map(int, input().split())) for i in range(3)])

print("Enter Matrix B:")
matrix_b = np.array([list(map(int, input().split())) for i in range(3)])


# Addition
print("Addition (A + B):")
print(matrix_a + matrix_b)
# Subtraction
print("Subtraction (A - B):")
print(matrix_a - matrix_b)
# Multiplication (element-wise)
print("Element-wise Multiplication (A * B):")
print(matrix_a * matrix_b)
```

Terminal Test cases

Debugger

3.2.1. Numpy: Matrix Operations

04:04

The given code takes two 3×3 matrices, **matrix_a**, and **matrix_b**, as input from the user and converts them into NumPy arrays.

Task:

You are required to compute and display the results of the following matrix operations:

1. **Addition** (matrix_a + matrix_b)
2. **Subtraction** (matrix_a − matrix_b)
3. **Element-wise Multiplication** (matrix_a * matrix_b)
4. **Matrix Multiplication** (matrix_a · matrix_b)
5. **Transpose of Matrix A**

Input Format:

- The user will input 3 rows for **matrix_a**, each containing 3 integers separated by spaces.
- Similarly, the user will input 3 rows for **matrix_b**, each containing 3 integers separated by spaces.

Sample Test Cases +

Explorer

matrixOp...

Submit

```
6
7   print("Enter Matrix B:")
8   matrix_b = np.array([list(map(int, input().split())) for i in
    range(3)])
9
10
11  # Addition
12  print("Addition (A + B):")
13  print(matrix_a + matrix_b)
14  # Subtraction
15  print("Subtraction (A - B):")
16  print(matrix_a - matrix_b)
17  # Multiplication (element-wise)
18  print("Element-wise Multiplication (A * B):")
19  print(matrix_a * matrix_b)
20  # Matrix multiplication (dot product)
21  print("A dot B:")
22  print(np.dot(matrix_a,matrix_b))
23  # Transpose
24  print("Transpose of A:")
25  print(matrix_a.T)
```

Debugger

Terminal Test cases

3.2.2. Numpy: Horizontal and Vertical Stacking of Arrays

05:19

You are given two arrays `arr1` and `arr2`. You need to perform horizontal and vertical stacking operations on them using NumPy.

- **Horizontal Stacking**: Stack the two matrices horizontally (side by side).
- **Vertical Stacking**: Stack the two matrices vertically (one below the other).

Input Format:

- The program should first prompt the user to input two 3x3 arrays.
- Each array consists of 3 rows, and each row contains 3 space-separated integers.
- The user will input the two arrays row by row.

Output Format:

- The program should display the result of the Horizontal Stack (side-by-side stacking) of the two arrays.
- The program should then display the result of the Vertical Stack (one below the other) of the two arrays.

Sample Test Cases +

Explorer

stacking.py

Submit

```
1  import numpy as np
2
3  # Input matrices
4  print("Enter Array1:")
5  arr1 = np.array([list(map(int, input().split())) for i in
   range(3)])
6
7  print("Enter Array2:")
8  arr2 = np.array([list(map(int, input().split())) for i in
   range(3)])
9
10 # Perform horizontal stacking (hstack)
11 horizontal_stack = np.hstack((arr1,arr2))
12
13
14 # Perform vertical stacking (vstack)
15 vertical_stack = np.vstack((arr1,arr2))
16 print("Horizontal Stack:")
17 print(horizontal_stack)
18 print("Vertical Stack:")
19 print(vertical_stack)
```

Debugger

Terminal Test cases

3.2.3. Numpy: Custom Sequence Generation

08:31

Write a Python program that takes the following inputs from the user:

- Start value: The starting point of the sequence.
- Stop value: The sequence should end before this value.
- Step value: The increment between each number in the sequence.

The program should then generate a sequence using `numpy` based on these inputs and print the generated sequence.

Input Format:

- The user will input three integer values: start, stop, and step, each on a new line.

Output Format:

- The program should print the generated sequence based on the input values.

Sample Test Cases +

Explorer customS... Submit Debugger

```python
import numpy as np

# Take user input for the start, stop, and step of the sequence
start = int(input())
stop = int(input())
step = int(input())

# Generate the sequence using np.arange()
sequence = np.arange(start, stop, step)
# Print the generated sequence
print(sequence)
```

Terminal Test cases

3.2.4. Numpy: Arithmetic and Statistical Operations, Mathematical Op... 04:21

You are given two arrays `A` and `B`. Your task is to complete the function `array_operations`, which will convert these lists into NumPy arrays and perform the following operations:

1. Arithmetic Operations:
- Compute the element-wise sum, difference, and product of the two arrays.

2. Statistical Operations:
- Calculate the mean, median, and standard deviation of array `A`.

3. Bitwise Operations:
- Perform bitwise `AND`, bitwise `OR`, and bitwise `XOR` on the arrays (ex: A_i `OR` B_i).

Input Format:
- The first line contains space-separated integers representing the elements of array `A`.
- The second line contains space-separated integers representing the elements of array `B`.

Sample Test Cases +

Explorer different... Submit Debugger

```python
import numpy as np

def array_operations(A, B):

    # Convert A and B to NumPy arrays
    A = np.array(A)
    B = np.array(B)
    # Arithmetic Operations
    sum_result = A + B
    diff_result = A - B
    prod_result = A * B

    # Statistical Operations
    mean_A = np.mean(A)
    median_A = np.median(A)
    std_dev_A = np.std(A)

    # Bitwise Operations
    and_result = A & B
    or_result = A | B
    xor_result = A ^ B
```

Terminal Test cases

3.2.4. Numpy: Arithmetic and Statistical Operations, Mathematical Op... 04:21

You are given two arrays `A` and `B`. Your task is to complete the function `array_operations`, which will convert these lists into NumPy arrays and perform the following operations:

1. Arithmetic Operations:
- Compute the element-wise sum, difference, and product of the two arrays.

2. Statistical Operations:
- Calculate the mean, median, and standard deviation of array `A`.

3. Bitwise Operations:
- Perform bitwise `AND`, bitwise `OR`, and bitwise `XOR` on the arrays (ex: A_i `OR` B_i).

Input Format:
- The first line contains space-separated integers representing the elements of array `A`.
- The second line contains space-separated integers representing the elements of array `B`.

Sample Test Cases +

Explorer different... Submit Debugger

```
19      and_result = A & B
20      or_result = A | B
21      xor_result = A ^ B
22
23      # Output results with one space between each element
24      print("Element-wise Sum:", ' '.join(map(str, sum_result)))
25      print("Element-wise Difference:", ' '.join(map(str,
        diff_result)))
26      print("Element-wise Product:", ' '.join(map(str,
        prod_result)))
27
28      print(f"Mean of A: {mean_A}")
29      print(f"Median of A: {median_A}")
30      print(f"Standard Deviation of A: {std_dev_A}")
31
32      print("Bitwise AND:", ' '.join(map(str, and_result)))
33      print("Bitwise OR:", ' '.join(map(str, or_result)))
34      print("Bitwise XOR:", ' '.join(map(str, xor_result)))
35
36  A = list(map(int, input().split()))  # Elements of array A
37  B = list(map(int, input().split()))  # Elements of array B
38  array_operations(A, B)
```

Terminal Test cases

3.2.5. Numpy: Copying and Viewing Arrays

04:45

The given code takes a list of integers as input and converts it into a NumPy array. Your task is to complete the code by:

- Creating a view of the `original_array` and assigning it to `view_array`.
- Creating a copy of the `original_array` and assigning it to `copy_array`.

After completing these steps, observe how modifying the view affects the `original_array`, while modifying the copy does not.

Input Format:

- A single line of space-separated integers.

Output Format:

- After modifying the view:

```
Original array after modifying view: <original_array>
View array: <view_array>
```

Sample Test Cases +

Explorer

copyAnd...

Submit

```python
import numpy as np

inputlist = list(map(int,input().split(" ")))

# Original array
original_array = np.array(inputlist)

# Create a view
view_array =original_array.view()

# Create a copy
copy_array =original_array.copy()

# Modify the view
view_array[0] = 99
print("Original array after modifying view:", original_array)
print("View array:", view_array)

# Modify the copy
copy_array[1] = 88
print("Original array after modifying copy:", original_array)
```

Terminal Test cases

Debugger

3.2.5. Numpy: Copying and Viewing Arrays

04:45

The given code takes a list of integers as input and converts it into a NumPy array. Your task is to complete the code by:

- Creating a view of the `original_array` and assigning it to `view_array`.
- Creating a copy of the `original_array` and assigning it to `copy_array`.

After completing these steps, observe how modifying the view affects the `original_array`, while modifying the copy does not.

Input Format:

- A single line of space-separated integers.

Output Format:

- After modifying the view:

```
Original array after modifying view: <original_array>
View array: <view_array>
```

Sample Test Cases +

Explorer

copyAnd...

Submit

Debugger

```
3   inputlist = list(map(int,input().split(" ")))
4
5   # Original array
6   original_array = np.array(inputlist)
7
8   # Create a view
9   view_array =original_array.view()
10
11  # Create a copy
12  copy_array =original_array.copy()
13
14  # Modify the view
15  view_array[0] = 99
16  print("Original array after modifying view:", original_array)
17  print("View array:", view_array)
18
19  # Modify the copy
20  copy_array[1] = 88
21  print("Original array after modifying copy:", original_array)
22  print("Copy array:", copy_array)
23
```

Terminal Test cases

3.2.6. Numpy: Searching, Sorting, Counting, Broadcasting

07:58

The given code in the editor takes a single array, `array1`, as space-separated integers as input from the user.
Additionally, it takes the following inputs:

- `search_value`: The value to search for in the array.
- `count_value`: The value to count its occurrences in the array.
- `broadcast_value`: The value to add for broadcasting across the array.

You need to complete the code to perform the following operations:
1. Searching: Find the indices where `search_value` appears in `array1` and print these indices.
2. Counting: Count how many times `count_value` appears in `array1` and print the count.
3. Broadcasting: Add `broadcast_value` to each element of `array1` using broadcasting, and print the resulting array.
4. Sorting: Sort `array1` in ascending order and print the sorted array.

Input Format:

Sample Test Cases +

Explorer

arrayOpe...

Submit

```
1  import numpy as np
2
3  # Input array from the user
4  array1 = np.array(list(map(int, input().split())))
5
6  # Searching
7  search_value = int(input("Value to search: "))
8  count_value = int(input("Value to count: "))
9  broadcast_value = int(input("Value to add: "))
10
11 # Find indices where value matches in array1
12 a=np.where(array1==search_value)[0]
13 print(a)
14 # Count occurrences in array1
15 b=np.count_nonzero(array1==count_value)
16 print(b)
17 # Broadcasting addition
18 c=array1+broadcast_value
19 print(c)
20 # Sort the first array
21 d=np.sort(array1)
```

Terminal Test cases

Debugger

3.2.6. Numpy: Searching, Sorting, Counting, Broadcasting

07:58

The given code in the editor takes a single array, `array1`, as space-separated integers as input from the user.
Additionally, it takes the following inputs:

- `search_value`: The value to search for in the array.
- `count_value`: The value to count its occurrences in the array.
- `broadcast_value`: The value to add for broadcasting across the array.

You need to complete the code to perform the following operations:
1. Searching: Find the indices where `search_value` appears in `array1` and print these indices.
2. Counting: Count how many times `count_value` appears in `array1` and print the count.
3. Broadcasting: Add `broadcast_value` to each element of `array1` using broadcasting, and print the resulting array.
4. Sorting: Sort `array1` in ascending order and print the sorted array.

Input Format:

Sample Test Cases +

Explorer

arrayOpe...

Submit

Debugger

```
2
3   # Input array from the user
4   array1 = np.array(list(map(int, input().split())))
5
6   # Searching
7   search_value = int(input("Value to search: "))
8   count_value = int(input("Value to count: "))
9   broadcast_value = int(input("Value to add: "))
10
11  # Find indices where value matches in array1
12  a=np.where(array1==search_value)[0]
13  print(a)
14  # Count occurrences in array1
15  b=np.count_nonzero(array1==count_value)
16  print(b)
17  # Broadcasting addition
18  c=array1+broadcast_value
19  print(c)
20  # Sort the first array
21  d=np.sort(array1)
22  print(d)
```

Terminal Test cases

3.2.7. Student Data Analysis and Operations

28:25

Write a Python program that takes the file name of a CSV file containing student details, including roll numbers and their marks in three subjects as input, reads the data, and performs the following operations:

- **Print all student details**: Display the complete details of all students, including roll numbers and marks for all subjects.
- **Find total students**: Determine the total number of students in the dataset.
- **Print all student roll numbers**: Extract and print the roll numbers of all students.
- **Print Subject 1 marks**: Extract and print the marks of all students in Subject 1.
- **Find minimum marks in Subject 2**: Identify the lowest marks in Subject 2.
- **Find maximum marks in Subject 3**: Identify the highest marks in Subject 3.
- **Print all subject marks**: Display the marks of all students for each subject.
- **Find total marks of students**: Compute the total marks for each student across all subjects.
- **Find the average marks of each student**: Compute the average marks for each student.
- **Find average marks of each subject**: Compute the average marks for all students in each subject.

Sample Test Cases +

Explorer

Operatio...

Submit

Debugger

```python
import numpy as np

a = np.loadtxt("Sample.csv", delimiter=',', skiprows=1)

# 1. Print all student details
print("All student Details:\n",a)

# 2. print total students
r,c=a.shape
print("Total Students:",r)

# 3. Print all student Roll numbers
print("All Student Roll Nos",a[:,0])

# 4. Print subject 1 marks
print("Subject 1 Marks", a[:,1])

# 5. print minimum marks of Subject 2
print("Min marks in Subject 2",np.min(a[:,2]))

# 6. print maximum marks of Subject 3
```

Terminal Test cases

3.2.7. Student Data Analysis and Operations

28:25

Write a Python program that takes the file name of a CSV file containing student details, including roll numbers and their marks in three subjects as input, reads the data, and performs the following operations:

- **Print all student details**: Display the complete details of all students, including roll numbers and marks for all subjects.
- **Find total students**: Determine the total number of students in the dataset.
- **Print all student roll numbers**: Extract and print the roll numbers of all students.
- **Print Subject 1 marks**: Extract and print the marks of all students in Subject 1.
- **Find minimum marks in Subject 2**: Identify the lowest marks in Subject 2.
- **Find maximum marks in Subject 3**: Identify the highest marks in Subject 3.
- **Print all subject marks**: Display the marks of all students for each subject.
- **Find total marks of students**: Compute the total marks for each student across all subjects.
- **Find the average marks of each student**: Compute the average marks for each student.
- **Find average marks of each subject**: Compute the average marks for all students in each subject.

Sample Test Cases +

Explorer

Operatio...

Submit

Debugger

```
19  print("Min marks in Subject 2",np.min(a[:,2]))
20
21  #6. print maximum marks of Subject 3
22  print("Max marks in Subject 3",np.max(a[:,3]))
23
24  #7. Print All subject marks
25  print("All subject marks:", a[:,1:])
26
27  #8. print Total marks of students
28  print("Total Marks",np.sum(a[:,1:],axis=1))
29
30  #9. print average marks of each student
31  avg=np.mean(a[:,1:],axis=1)
32  print(np.round(avg,1))
33  #10. print average marks of each subject
34  print("Average Marks of each subject",np.mean(a[:,1:],axis=0))
35
36  #11. print average marks of S1 and S2
37  print("Average Marks of S1 and S2",np.mean(a[:,1:3],axis=0))
38
39  #12. print average marks of S1 and S3
40  print("Average Marks of S1 and S3",np.mean(a[:,[1,3]],axis=0))
```

Terminal Test cases

3.2.7. Student Data Analysis and Operations

28:25

Write a Python program that takes the file name of a CSV file containing student details, including roll numbers and their marks in three subjects as input, reads the data, and performs the following operations:

- **Print all student details**: Display the complete details of all students, including roll numbers and marks for all subjects.
- **Find total students**: Determine the total number of students in the dataset.
- **Print all student roll numbers**: Extract and print the roll numbers of all students.
- **Print Subject 1 marks**: Extract and print the marks of all students in Subject 1.
- **Find minimum marks in Subject 2**: Identify the lowest marks in Subject 2.
- **Find maximum marks in Subject 3**: Identify the highest marks in Subject 3.
- **Print all subject marks**: Display the marks of all students for each subject.
- **Find total marks of students**: Compute the total marks for each student across all subjects.
- **Find the average marks of each student**: Compute the average marks for each student.
- **Find average marks of each subject**: Compute the average marks for all students in each subject.

Sample Test Cases +

Explorer

Operatio...

Submit

Debugger

```
42  # 13. print Roll number who got maximum marks in Subject 3
43  i=np.argmax(a[:,3])
44  print("Roll no who got maximum marks in Subject 3",a[i,0])
45
46  # 14. print Roll number who got minimum marks in Subject 2
47  mn=np.argmin(a[:,2])
48  print("Roll no who got minimum marks in Subject 2",a[mn,0])
49
50  # 15. print Roll number who got 24 marks in Subject 2
51  whr=np.where(a[:,2]==24)
52  print("Roll no who got 24 marks in Subject 2",a[whr,0])
53
54  # 16. print count of students who got marks in Subject 1 < 40
55  ct=np.count_nonzero(a[:,1]<40)
56  print("Count of students who got marks in Subject 1 < 40",ct)
57
58  # 17. print count of students who got marks in Subject 2 > 90
59  ct1=np.count_nonzero(a[:,2]>90)
60  print("Count of students who got marks in Subject 2 > 90:",ct1)
61
62  # 18. print count of students in each subject who got marks >=
```

Terminal Test cases

3.2.7. Student Data Analysis and Operations

28:25

Write a Python program that takes the file name of a CSV file containing student details, including roll numbers and their marks in three subjects as input, reads the data, and performs the following operations:

- **Print all student details**: Display the complete details of all students, including roll numbers and marks for all subjects.
- **Find total students**: Determine the total number of students in the dataset.
- **Print all student roll numbers**: Extract and print the roll numbers of all students.
- **Print Subject 1 marks**: Extract and print the marks of all students in Subject 1.
- **Find minimum marks in Subject 2**: Identify the lowest marks in Subject 2.
- **Find maximum marks in Subject 3**: Identify the highest marks in Subject 3.
- **Print all subject marks**: Display the marks of all students for each subject.
- **Find total marks of students**: Compute the total marks for each student across all subjects.
- **Find the average marks of each student**: Compute the average marks for each student.
- **Find average marks of each subject**: Compute the average marks for all students in each subject.

Sample Test Cases +

Explorer | Operatio... | Submit | Debugger

```
61
62  # 18. print count of students in each subject who got marks >=
    90
63  print("Count of students in each subject who got marks >=
    90:",np.count_nonzero(a[:,1:]>=90,axis=0))
64
65  # 19. print count of subjects in which each student got marks
    >= 90
66  print("Roll no:",a[:,0])
67  print("Count of subjects in which student got marks >=
    90:",np.count_nonzero(a[:,1:]>=90,axis=1))
68
69  # 20. Print S1 marks in ascending order
70  srt=np.sort(a[:,1])
71  print(srt)
72
73  # 21. Print S1 marks >= 50 and <= 90
74  print(a[(a[:,1]>=50)&(a[:,1]<90)])
75  print(a)
76  ip=np.where(a[:,1]==79)
77  print(ip)
```

Terminal | Test cases