

3.1.1.1. Numpy array operations

18:04

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 and columns.

Sample Test Cases

+

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 print(matrix)
9 print(matrix.ndim)
10 print(matrix.shape)
11 print(matrix.size)
```

Terminal

Test cases

< Prev Reset Submit Next >

CODETANTRA

Home

202401070216@mitaoe.ac.in

Support

Logout

3.2.1. Numpy: Matrix Operations

03:36

🔍

🌙

📄

🔗

⋮

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

+

matrixOp...

Submit

Debugger

```
1 import numpy as np
2
3 # Input matrices
4 print("Enter Matrix A:")
5 matrix_a = np.array([list(map(int, input().split())) for i in
6 range(3)])
7
8 print("Enter Matrix B:")
9 matrix_b = np.array([list(map(int, input().split())) for i in
10 range(3)])
11
12 # Addition
13 print("Addition (A + B):")
14 print(matrix_a + matrix_b)
15
16 # Subtraction
17 print("Subtraction (A - B):")
18 print(matrix_a - matrix_b)
19
20 # Multiplication (element-wise)
21 print("Element-wise Multiplication (A * B):")
22 print(matrix_a * matrix_b)
23
24 # Matrix multiplication (dot product)
```

Terminal

Test cases

< Prev

Reset

Submit

Next >

3.2.2. Numpy: Horizontal and Vertical Stacking of Arrays

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

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

CODETANTRA

Home

202401070216@mitaoe.ac.inSupportLogout

3.2.3. Numpy: Custom Sequence Generation02:53

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

customS...

```
1 import numpy as np
2
3 # Take user input for the start, stop, and step of the sequence
4 start = int(input())
5 stop = int(input())
6 step = int(input())
7
8 # Generate the sequence using np.arange()
9 sequence = np.arange(start, stop, step)
10
11 # Print the generated sequence
12 print(sequence)
```

TerminalTest cases

< PrevResetSubmitNext >

3.2.4. Numpy: Arithmetic and Statistical Operations, Mathematical Operati...08:57

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: $A_i \text{ OR } B_i$).

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.

Output Format:

- For each operation (arithmetic, statistical, and bitwise), print the results in the specified

Sample Test Cases+

different...

1import numpy as np

2

3def array_operations(A, B):

4

5 # Convert A and B to NumPy arrays

6 A = np.array(A)

7 B = np.array(B)

8

9 # Arithmetic Operations

10 sum_result = A+B

11 diff_result =A-B

12 prod_result =A*B

13

14 # Statistical Operations

15 mean_A = np.mean(A)

16 median_A = np.median(A)

17 std_dev_A = np.std(A)

18

19 # Bitwise Operations

20 and_result = A & B

21 or_result = A | B

22 xor_result = A ^ B

23

TerminalTest cases

3.2.5. Numpy: Copying and Viewing Arrays

03:07

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>`

- After modifying the copy:

Sample Test Cases

+

copyAnd...

Submit

```
1 import numpy as np
2
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)
```

Terminal Test cases

Debugger

3.2.6. Numpy: Searching, Sorting, Counting, Broadcasting

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:

1. A single line containing space-separated integers representing `array1`.

Sample Test Cases



arrayOpe...

```
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)
22 print(d)
```

Terminal

Test cases

< Prev Reset Submit Next >

3.2.7. Student Data Analysis and Operations

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.
- **Find average marks of Subject 4 and Subject 3:** Compute the average marks for

Sample Test Cases

Operatio... Submit

5

1. Print all student details

6

print("All student Details:\n",a)

7

8

2. print total students

9

r,c=a.shape

10

print("Total Students:",r)

11

12

3. Print all student Roll numbers

13

print("All Student Roll Nos",a[:,0])

14

15

4. Print subject 1 marks

16

print("Subject 1 Marks",a[:,1])

17

18

5. print minimum marks of Subject 2

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

Terminal Test cases

3.2.7. Student Data Analysis and Operations

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.
- **Find average marks of Subject 1 and Subject 2:** Compute the average marks for

Sample Test Cases

```
65 print("Roll no:",a[:,0])
66 print("Count of subjects in which student got marks >=
90:",np.count_nonzero(a[:,1:]>90,axis=1))
67
68
69 # 20. Print S1 marks in ascending order
70 srt=np.sort(a[:,1])
71 print(srt)
72
73
74 # 21. Print S1 marks >= 50 and <= 90
75 print(a[(a[:,1]>=50)&(a[:,1]<90)])
76
77
78 # 22. Print the index position of marks 79
79 print(a)
80 ip=np.where(a[:,1]==79)
81 print(ip)
82
83
84
```