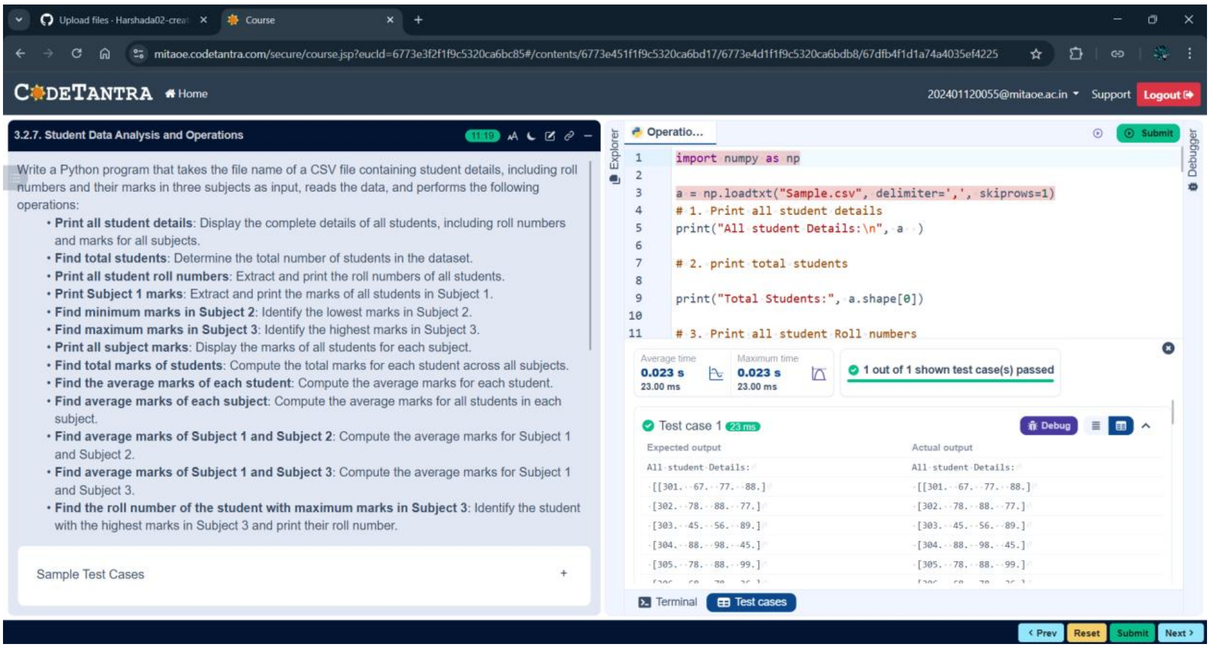
PRACTICAL 3

ALL PRACTICALS:


Upload files · Harshada02-creat
Course
mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/67dfb4f1d1a74a4035ef4225
CODETANTRA
Home
202401120055@mitaoe.ac.in
Support
Logout
11:19
Explorer
Operatio...
Submit
Debugger
Sample Test Cases
Terminal
Test cases
Debug
‹ 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 1 and Subject 2**: Compute the average marks for Subject 1 and Subject 2.
- **Find average marks of Subject 1 and Subject 3**: Compute the average marks for Subject 1 and Subject 3.
- **Find the roll number of the student with maximum marks in Subject 3**: Identify the student with the highest marks in Subject 3 and print their roll number.

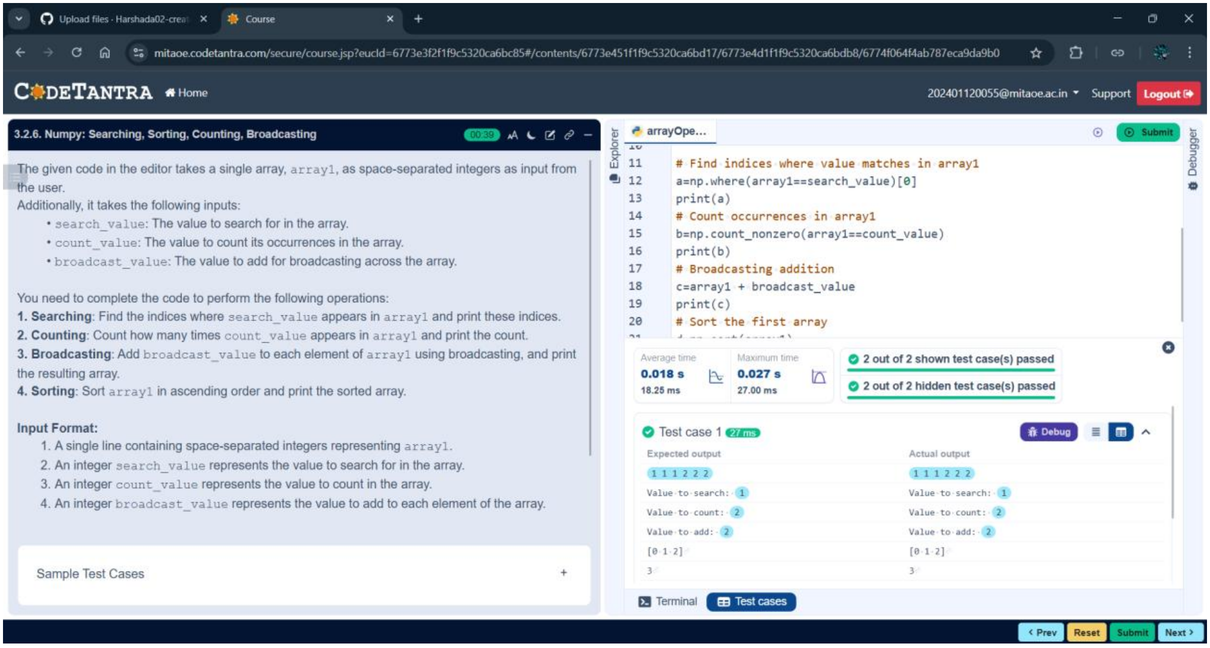
```python
import numpy as np

a = np.loadtxt("Sample.csv", delimiter=',', skiprows=1)
# 1. Print all student details
print("All student Details:\n", a  )

# 2. print total students

print("Total Students:", a.shape[0])

# 3. Print all student Roll numbers
```

Average time 0.023 s 23.00 ms | Maximum time 0.023 s 23.00 ms

1 out of 1 shown test case(s) passed

Test case 1 23 ms

Expected output	Actual output
All student Details:	All student Details:
[[301. 67. 77. 88.]	[[301. 67. 77. 88.]
[302. 78. 88. 77.]	[302. 78. 88. 77.]
[303. 45. 56. 89.]	[303. 45. 56. 89.]
[304. 88. 98. 45.]	[304. 88. 98. 45.]
[305. 78. 88. 99.]	[305. 78. 88. 99.]


Upload files · Harshada02-creat
Course
mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/6774f064f4ab787eca9da9b0
CODETANTRA
Home
202401120055@mitaoe.ac.in
Support
Logout
00:39
Explorer
arrayOpe...
Submit
Debugger
Sample Test Cases
Terminal
Test cases
Debug
‹ Prev
Reset
Submit
Next ›

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`.
2. An integer `search_value` represents the value to search for in the array.
3. An integer `count_value` represents the value to count in the array.
4. An integer `broadcast_value` represents the value to add to each element of the array.

```python
# Find indices where value matches in array1
a=np.where(array1==search_value)[0]
print(a)
# Count occurrences in array1
b=np.count_nonzero(array1==count_value)
print(b)
# Broadcasting addition
c=array1 + broadcast_value
print(c)
# Sort the first array
```

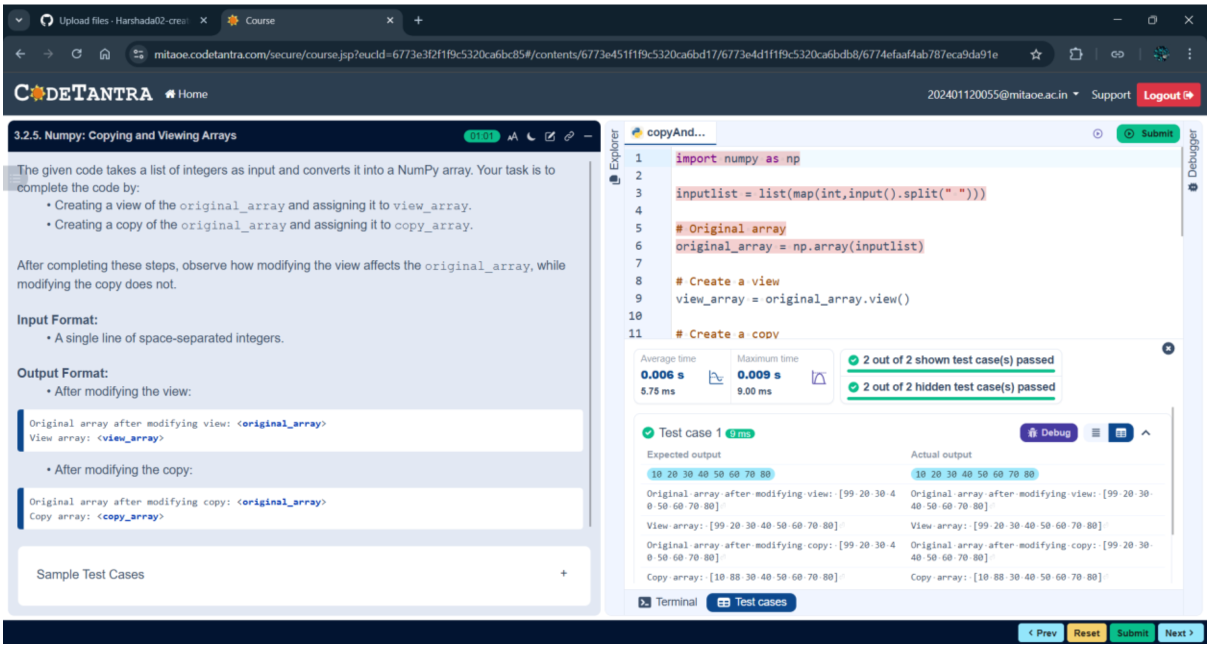
Average time 0.018 s 18.25 ms | Maximum time 0.027 s 27.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 27 ms

Expected output	Actual output
1 1 1 2 2 2	1 1 1 2 2 2
Value to search: 1	Value to search: 1
Value to count: 2	Value to count: 2
Value to add: 2	Value to add: 2
[0 1 2]	[0 1 2]
3	3


Upload files · Harshada02-creat
Course
mitaoe.codetantra.com/secure/course.jsp?eucld=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/6774efaaf4ab787eca9da91e
CODETANTRA
Home
202401120055@mitaoe.ac.in
Support
Logout
01:01
Explorer
copyAnd...
Submit
Debugger
Debug
Terminal
Test cases
‹ Prev
Reset
Submit
Next ›

3.2.5. Numpy: Copying and Viewing Arrays

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:

```
Original array after modifying copy: <original_array>
Copy array: <copy_array>
```

Sample Test Cases +

```
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
```

Average time	Maximum time
0.006 s 5.75 ms	0.009 s 9.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

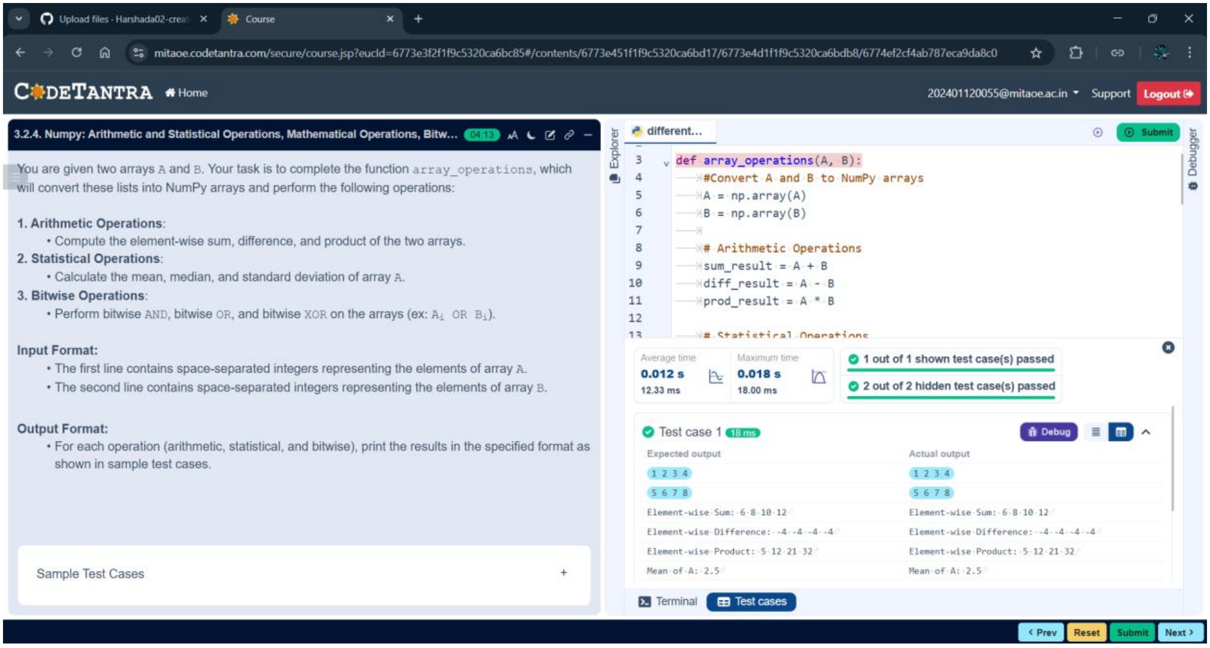
Test case 1 9 ms

Expected output

```
10 20 30 40 50 60 70 80
Original array after modifying view: [99 20 30 4
0 50 60 70 80]
View array: [99 20 30 40 50 60 70 80]
Original array after modifying copy: [99 20 30 4
0 50 60 70 80]
Copy array: [10 88 30 40 50 60 70 80]
```

Actual output

```
10 20 30 40 50 60 70 80
Original array after modifying view: [99 20 30
40 50 60 70 80]
View array: [99 20 30 40 50 60 70 80]
Original array after modifying copy: [99 20 30
40 50 60 70 80]
Copy array: [10 88 30 40 50 60 70 80]
```


Upload files · Harshada02-creat
Course
mitaoe.codetantra.com/secure/course.jsp?eucld=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/6774ef2cf4ab787eca9da8c0
CODETANTRA
Home
202401120055@mitaoe.ac.in
Support
Logout
04:13
Explorer
different...
Submit
Debugger
Debug
Terminal
Test cases
‹ Prev
Reset
Submit
Next ›

3.2.4. Numpy: Arithmetic and Statistical Operations, Mathematical Operations, Bitw...

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`.

Output Format:

- For each operation (arithmetic, statistical, and bitwise), print the results in the specified format as shown in sample test cases.

Sample Test Cases +

```
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
```

Average time	Maximum time
0.012 s 12.33 ms	0.018 s 18.00 ms

1 out of 1 shown test case(s) passed

2 out of 2 hidden test case(s) passed

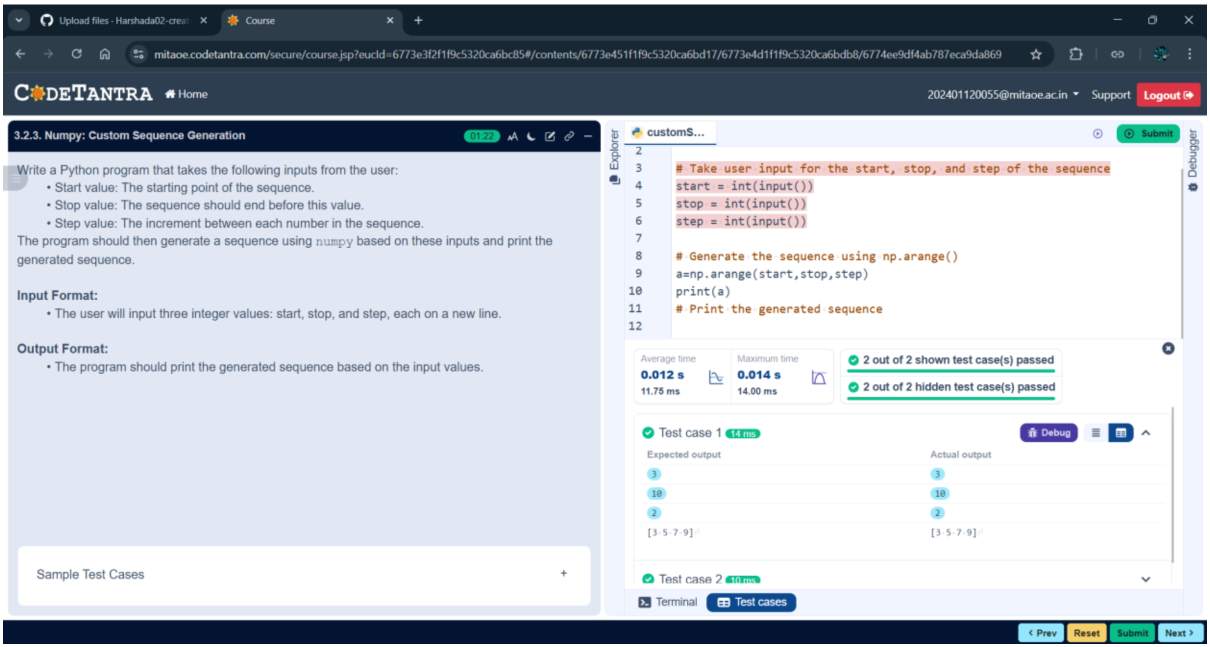
Test case 1 18 ms

Expected output

```
1 2 3 4
5 6 7 8
Element-wise Sum: 6 8 10 12
Element-wise Difference: -4 -4 -4 -4
Element-wise Product: 5 12 21 32
Mean of A: 2.5
```

Actual output

```
1 2 3 4
5 6 7 8
Element-wise Sum: 6 8 10 12
Element-wise Difference: -4 -4 -4 -4
Element-wise Product: 5 12 21 32
Mean of A: 2.5
```


Upload files · Harshada02-creat
Course
mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/6774ee9df4ab787eca9da869
CODETANTRA
Home
202401120055@mitaoe.ac.in
Support
Logout
01:22
Explorer
Debugger
customS...
Submit
Terminal
Test cases
Debug
‹ Prev
Reset
Submit
Next ›

3.2.3. Numpy: Custom Sequence Generation

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 +

```python

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

```

Average time	Maximum time
0.012 s 11.75 ms	0.014 s 14.00 ms

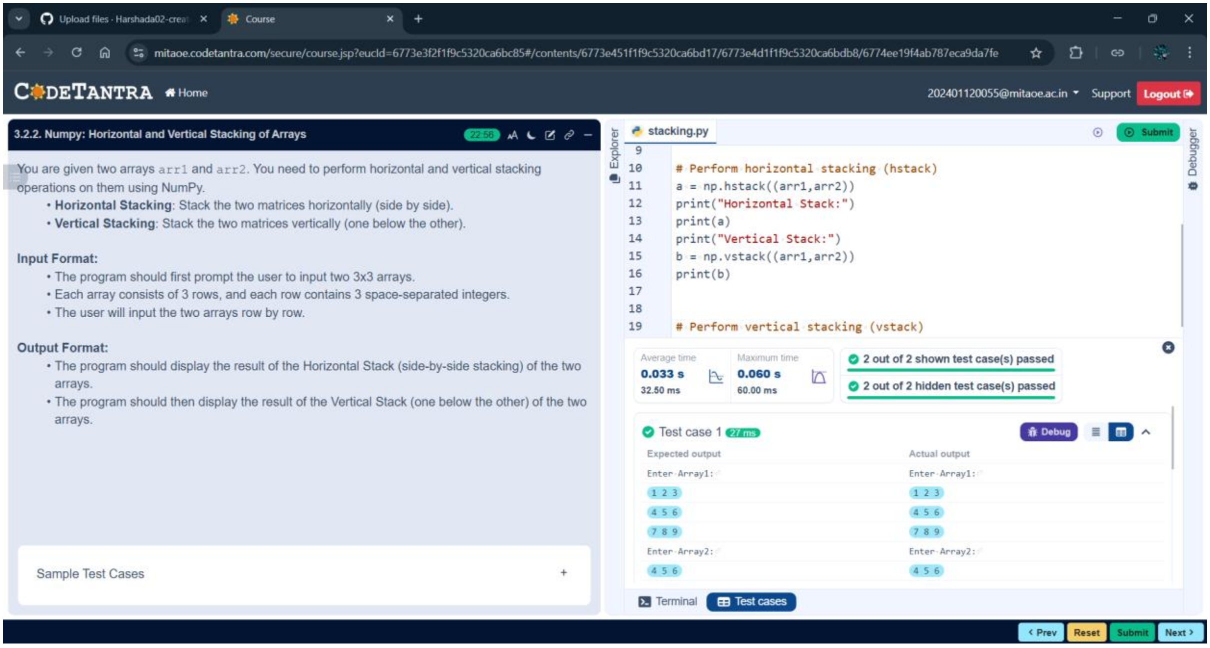
✓ 2 out of 2 shown test case(s) passed

✓ 2 out of 2 hidden test case(s) passed

✓ Test case 1 14 ms

Expected output	Actual output
3	3
10	10
2	2
[3 5 7 9]	[3 5 7 9]

✓ Test case 2 10 ms


Upload files · Harshada02-creat
Course
mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/6774ee19f4ab787eca9da7fe
CODETANTRA
Home
202401120055@mitaoe.ac.in
Support
Logout
22:56
Explorer
Debugger
stacking.py
Submit
Terminal
Test cases
Debug
‹ 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 +

```python

# Perform horizontal stacking (hstack)
a = np.hstack((arr1,arr2))
print("Horizontal Stack:")
print(a)
print("Vertical Stack:")
b = np.vstack((arr1,arr2))
print(b)


# Perform vertical stacking (vstack)
```

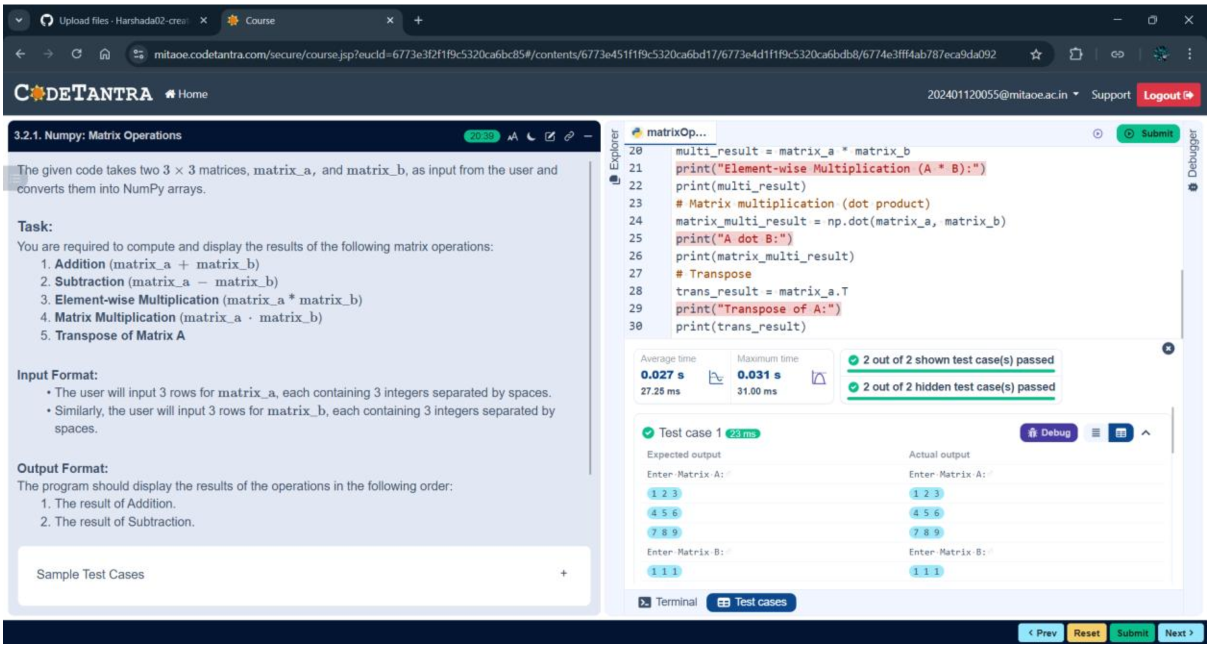
Average time	Maximum time
0.033 s 32.50 ms	0.060 s 60.00 ms

✓ 2 out of 2 shown test case(s) passed

✓ 2 out of 2 hidden test case(s) passed

✓ Test case 1 27 ms

Expected output	Actual output
Enter Array1:	Enter Array1:
1 2 3	1 2 3
4 5 6	4 5 6
7 8 9	7 8 9
Enter Array2:	Enter Array2:
4 5 6	4 5 6



mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/6774e3fff4ab787eca9da092

3.2.1. Numpy: Matrix Operations

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.

Output Format:

The program should display the results of the operations in the following order:

1. The result of Addition.
2. The result of Subtraction.

Sample Test Cases +

```python
multi_result = matrix_a * matrix_b
print("Element-wise Multiplication (A * B):")
print(multi_result)
# Matrix multiplication (dot product)
matrix_multi_result = np.dot(matrix_a, matrix_b)
print("A dot B:")
print(matrix_multi_result)
# Transpose
trans_result = matrix_a.T
print("Transpose of A:")
print(trans_result)
```

Average time 0.027 s 27.25 ms

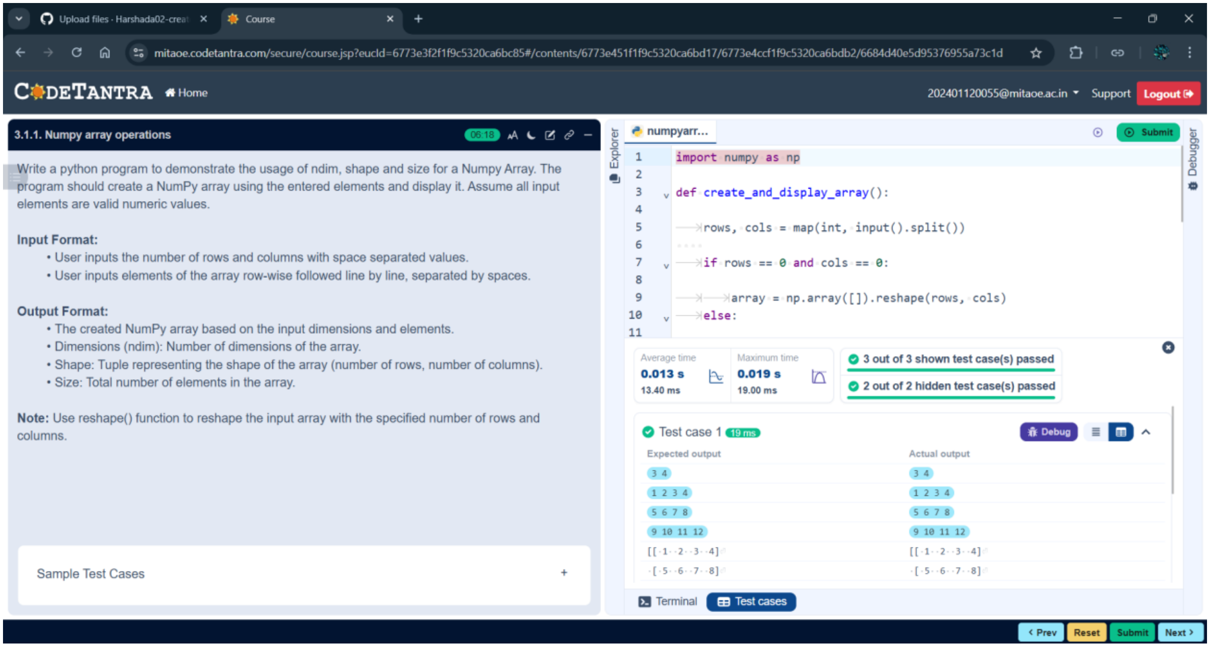
Maximum time 0.031 s 31.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 23 ms

Expected output	Actual output
Enter Matrix A:	Enter Matrix A:
1 2 3	1 2 3
4 5 6	4 5 6
7 8 9	7 8 9
Enter Matrix B:	Enter Matrix B:
1 1 1	1 1 1



mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4ccf1f9c5320ca6bdb2/6684d40e5d95376955a73c1d

3.1.1. Numpy array operations

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 +

```python
import numpy as np

def create_and_display_array():

    rows, cols = map(int, input().split())
    
    if rows == 0 and cols == 0:

        array = np.array([]).reshape(rows, cols)
    else:

```

Average time 0.013 s 13.40 ms

Maximum time 0.019 s 19.00 ms

3 out of 3 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 19 ms

Expected output	Actual output
3 4	3 4
1 2 3 4	1 2 3 4
5 6 7 8	5 6 7 8
9 10 11 12	9 10 11 12
[[1 2 3 4]	[[1 2 3 4]
[5 6 7 8]	[5 6 7 8]