week 5

February 12, 2025

1 Week 5

1.1 Ungraded Exercise TA Solution

1.1.1 TA: Abhisek Sarkar

1.1.2 as20ms091@iiserkol.ac.in

1. Create a 1-D NumPy array containing 9 elements, taken as inputs from the user. Now, use a single statement and negative slicing to print the array elements in reverse order.

[1]:

```
import numpy as np

arr = np.array([int(input(f"Enter element {i+1}: ")) for i in range(9)])
print(arr[::-1])
```

```
[9 8 7 6 5 4 3 2 1]
```

2. Create a 1-D NumPy array containing 9 elements, taken as inputs from the user. Now, use array slicing to accomplish the following:

a. Print the last 3 elements from the array
b. Print the first 3 elements from the array
c. Print the middle 3 elements from the array
d. Print the 5th-last element to 2nd-last element (included) using negative slicing
e. Replace every second element starting from index 1 of the array with 0, and print the updated array E.g. If the array is [1 2 3 4 5 6 7 8 9], the desired outcomes are as follows:

```
Last 3 elements from the array [7 8 9]
First 3 elements from the array [1 2 3]
Middle 3 elements from the array [4 5 6]
5th-last element to 2nd-last element (included): [5 6 7 8]
Updated array: [1 0 3 0 5 0 7 0 9]
```

[2]:

```
import numpy as np

arr = np.array([int(input(f"Enter element {i+1}: ")) for i in range(9)])

print("Last 3 elements from the array", arr[-3:])
print("First 3 elements from the array", arr[:3])
print("Middle 3 elements from the array", arr[3:6])
print("5th-last element to 2nd-last element (included):", arr[-5:-1])

arr[1::2] = 0
print("Updated array:", arr)
```

```
Last 3 elements from the array [7 8 9]
First 3 elements from the array [1 2 3]
Middle 3 elements from the array [4 5 6]
5th-last element to 2nd-last element (included): [5 6 7 8]
Updated array: [1 0 3 0 5 0 7 0 9]
```

3. Create a 2-D NumPy array or matrix of dimension . Now, use array slicing to accomplish the following:

a. Print the last 2 columns of the matrix
b. Print the first 2 rows of the matrix
c. Replace its elements in the central matrix by the maximum value present there, and print the updated matrix. [Note that the maximum element value in an NumPy array can be determined by using numpy.max() method] E.g. If the matrix is ,

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

the desired outcomes are as follows: Last 2 columns of the matrix:

```
[[3  4]
 [7  8]
 [11 12]
 [15 16]]
```

First 2 rows of the matrix:

```
[[1 2 3 4]
 [5 6 7 8]]
```

Updated matrix:

```
[[1  2  3  4]
 [5 11 11  8]
 [9 11 11 12]
 [13 14 15 16]]
```

[3]:

```
import numpy as np

matrix = np.array([[int(input(f"Enter element ({i+1},{j+1}): ")) for j in␣
 ↪range(4)] for i in range(4)])

print("Last 2 columns of the matrix:\n", matrix[:, -2:])
print("First 2 rows of the matrix:\n", matrix[:2, :])
```

```
central_max = np.max(matrix[1:3, 1:3])
matrix[1:3, 1:3] = central_max

print("Updated matrix:\n", matrix)
```

```
Last 2 columns of the matrix:
 [[ 3  4]
 [ 7  8]
 [11 12]
 [15 16]]
First 2 rows of the matrix:
 [[1 2 3 4]
 [5 6 7 8]]
Updated matrix:
 [[ 1  2  3  4]
 [ 5 11 11  8]
 [ 9 11 11 12]
 [13 14 15 16]]
```

4. Create two 2-D NumPy arrays or matrices, namely A and B, of dimension each. Use array slicing to replace every second element starting from index 0 along the rows and the columns of matrix A with the corresponding element in matrix B. Now, print the updated matrix A.

E.g. If the matrix A is

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

and the matrix B is

$$\begin{bmatrix} -1 & -2 & -3 & -4 \\ -5 & -6 & -7 & -8 \\ -9 & -10 & -11 & -12 \\ -13 & -14 & -15 & -16 \end{bmatrix}$$

, the desired outcome would be as follows: Updated matrix A:

```
[[-1  2  -3  4]
 [ 5  6   7  8]
 [-9 10 -11 12]
 [13 14  15 16]]
```

[4]:

```
import numpy as np

A = np.array([[int(input(f"Enter element A({i+1},{j+1}): ")) for j in range(4)]␣
 ↪for i in range(4)])
B = np.array([[int(input(f"Enter element B({i+1},{j+1}): ")) for j in range(4)]␣
 ↪for i in range(4)])
```

```
A[::2, ::2] = B[::2, ::2]

print("Updated matrix A:\n", A)
```

```
Updated matrix A:
 [[ -1   2  -3   4]
 [  5   6   7   8]
 [ -9  10 -11  12]
 [ 13  14  15  16]]
```

5. Given any 2-D NumPy array of dimension 4 x 4, create a 3-D array B of dimension 4 x 2 x 2, composed of the 2 x 2 matrix components taken from the top-left, top-right, bottom-left and bottom-right corners of A. Use the concept of array slicing in order to achieve this. E.g. If the matrix A is,
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$
the desired outcome would be as follows:

```
Array B:
[[[1 2]
  [5 6]]]

[[[3 4]
  [7 8]]]

[[[9 10]
  [13 14]]]

[[[11 12]
  [15 16]]]]
```

[5]:

```
import numpy as np

A = np.array([[int(input(f"Enter element A({i+1},{j+1}): ")) for j in range(4)] for i in range(4)])

B = np.array([
    A[:2, :2],  # Top-left
    A[:2, 2:],  # Top-right
    A[2:, :2],  # Bottom-left
    A[2:, 2:]   # Bottom-right
])

print("Array B:")
for block in B:
    print(block, "\n")
```

```
Array B:
[[1 2]
 [5 6]]

[[3 4]
 [7 8]]

[[ 9 10]
 [13 14]]

[[11 12]
 [15 16]]
```