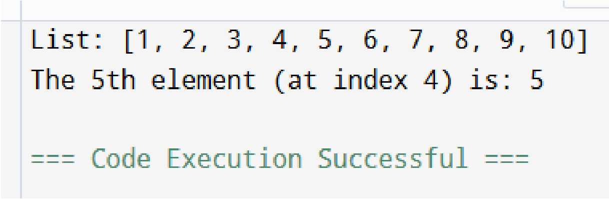
Vaishnavi Ashok Patil
C07
SPP-II

Experiment No.:- 11

List:-

1. **Create a Python list of the first 10 natural numbers. Print the 5th element using indexing.**

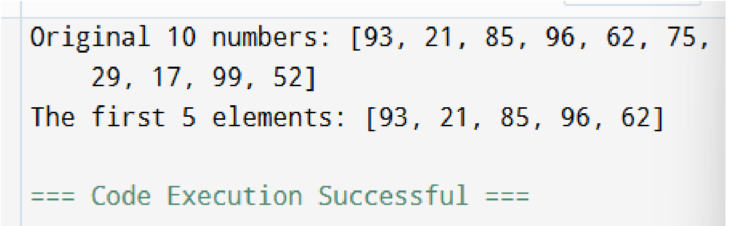
```
def list():

natural_numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(f"List: {natural_numbers}")

fifth_element = natural_numbers[4]

print(f"The 5th element (at index 4) is: {fifth_element}")

list()
```



```
List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The 5th element (at index 4) is: 5

=== Code Execution Successful ===
```

2. **Make a list of 10 random integers. Slice the list to print only the first 5 elements.**

```
import random

def slice():

random_numbers = [random.randint(1, 100) for _ in range(10)]

print(f"Original 10 numbers: {random_numbers}")

first_five = random_numbers[:5]

print(f"The first 5 elements: {first_five}")

slice()
```

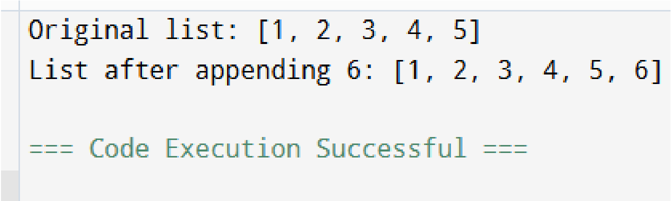


```
Original 10 numbers: [93, 21, 85, 96, 62, 75,
    29, 17, 99, 52]
The first 5 elements: [93, 21, 85, 96, 62]

=== Code Execution Successful ===
```

3. **Create a list [1, 2, 3, 4, 5]. Append the number 6 and print the list.**

```
def append():

my_list = [1, 2, 3, 4, 5]

print(f"Original list: {my_list}")

my_list.append(6)
```

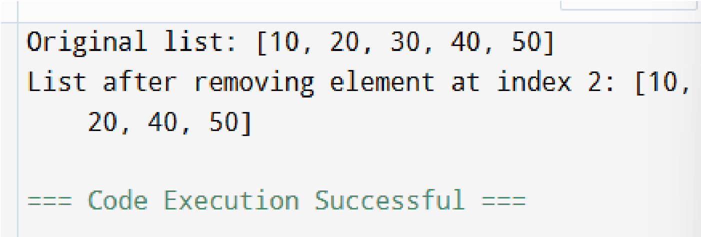
```
print(f"List after appending 6: {my_list}")

append()
```



```
Original list: [1, 2, 3, 4, 5]
List after appending 6: [1, 2, 3, 4, 5, 6]

=== Code Execution Successful ===
```

4. **Create a list [10, 20, 30, 40, 50]. Remove the element at index 2.**

```
def remove():

my_list = [10, 20, 30, 40, 50]

print(f"Original list: {my_list}")

del my_list[2]

print(f"List after removing element at index 2: {my_list}")

if __name__ == "__main__":

remove()
```

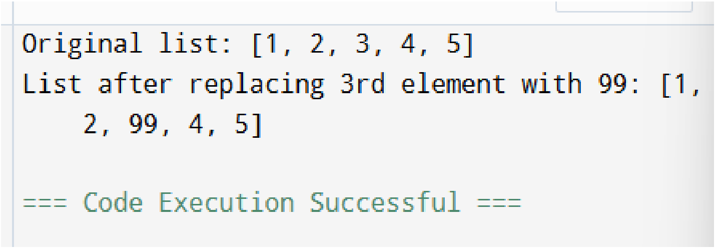


```
Original list: [10, 20, 30, 40, 50]
List after removing element at index 2: [10,
    20, 40, 50]

=== Code Execution Successful ===
```

5. **Create a list [1,2,3,4,5]. Replace the 3rd element with 99.**

```
def replace():

my_list = [1, 2, 3, 4, 5]

print(f"Original list: {my_list}")

my_list[2] = 99

print(f"List after replacing 3rd element with 99: {my_list}")

if __name__ == "__main__":

replace()
```

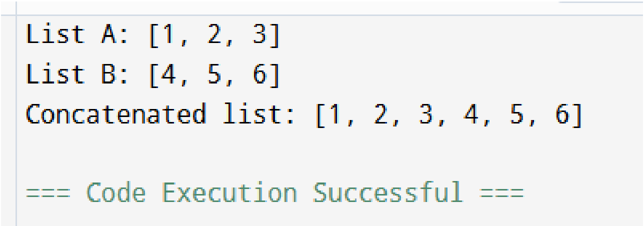
Vaishnavi Ashok Patil
C07
SPP-II



```
Original list: [1, 2, 3, 4, 5]
List after replacing 3rd element with 99: [1,
    2, 99, 4, 5]

=== Code Execution Successful ===
```

6. **Create two lists [1,2,3] and [4,5,6]. Concatenate them.**

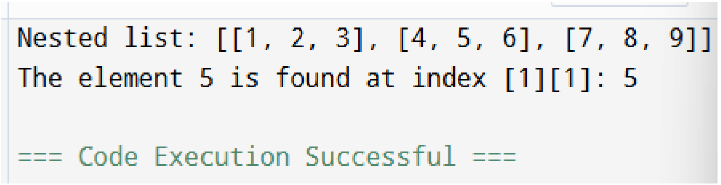
```
def concatenation():

list_a = [1, 2, 3]

list_b = [4, 5, 6]

print(f"List A: {list_a}")

print(f"List B: {list_b}")

combined_list = list_a + list_b

print(f"Concatenated list: {combined_list}")

if __name__ == "__main__":

concatenation()
```



```
List A: [1, 2, 3]
List B: [4, 5, 6]
Concatenated list: [1, 2, 3, 4, 5, 6]

=== Code Execution Successful ===
```

7. **Create a nested list [[1,2,3], [4,5,6], [7,8,9]]. Print the element 5.**

```
def nested():

nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

print(f"Nested list: {nested_list}")

element_five = nested_list[1][1]

print(f"The element 5 is found at index [1][1]: {element_five}")

if __name__ == "__main__":

nested()
```

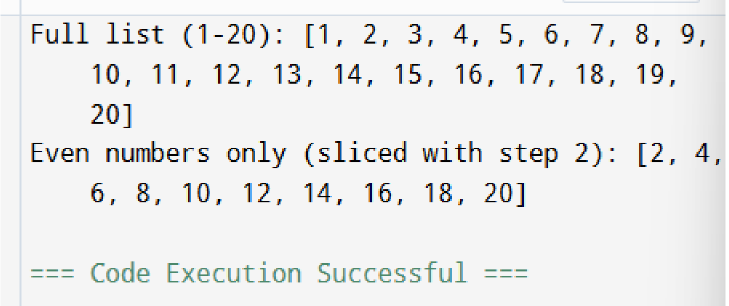


```
Nested list: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
The element 5 is found at index [1][1]: 5

=== Code Execution Successful ===
```

Vaishnavi Ashok Patil
C07
SPP-II

8. **Create a list of numbers from 1 to 20. Slice and print only the even numbers.**

```
def step_slicing():

numbers_20 = list(range(1, 21))

print(f"Full list (1-20): {numbers_20}")

even_numbers = numbers_20[1::2]

print(f"Even numbers only (sliced with step 2): {even_numbers}")

if __name__ == "__main__":

step_slicing()
```

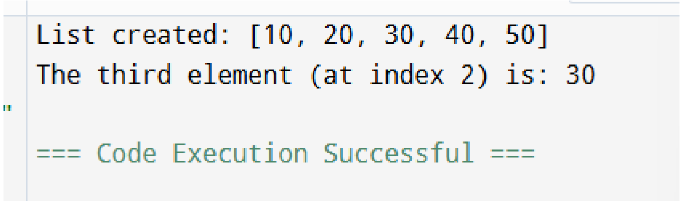


```
Full list (1-20): [1, 2, 3, 4, 5, 6, 7, 8, 9,
    10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
    20]
Even numbers only (sliced with step 2): [2, 4,
    6, 8, 10, 12, 14, 16, 18, 20]

=== Code Execution Successful ===
```

- **Array:-**

1. **Create an integer array [10,20,30,40,50]. Print the third element.**

```
def indexing():

integer_list = [10, 20, 30, 40, 50]

print(f"List created: {integer_list}")

third_element = integer_list[2]

print(f"The third element (at index 2) is: {third_element}")

if __name__ == "__main__":

indexing()
```

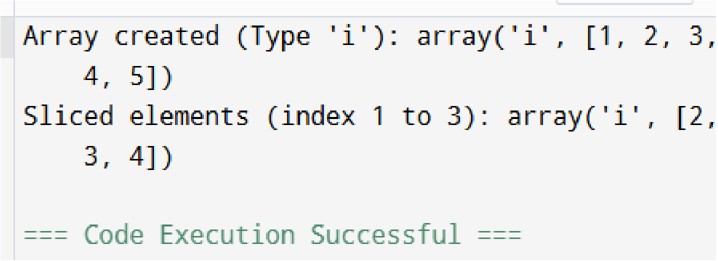


```
List created: [10, 20, 30, 40, 50]
The third element (at index 2) is: 30

=== Code Execution Successful ===
```

2. **Create an array of type 'i' (integers). Insert numbers from 1 to 5. Slice and print elements at index 1–3.**

```
import array

def run_exercise_2():

my_array = array.array('i', [1, 2, 3, 4, 5])

print(f"Array created (Type 'i'): {my_array}")
```

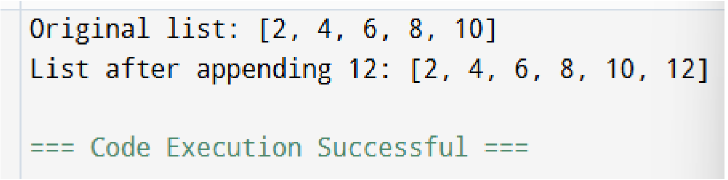
```
sliced_elements = my_array[1:4]

print(f"Sliced elements (index 1 to 3): {sliced_elements}")

if __name__ == "__main__":

run_exercise_2()
```



```
Array created (Type 'i'): array('i', [1, 2, 3,
    4, 5])
Sliced elements (index 1 to 3): array('i', [2,
    3, 4])

=== Code Execution Successful ===
```

3. **Create an integer array [2,4,6,8,10]. Append 12 to the array.**

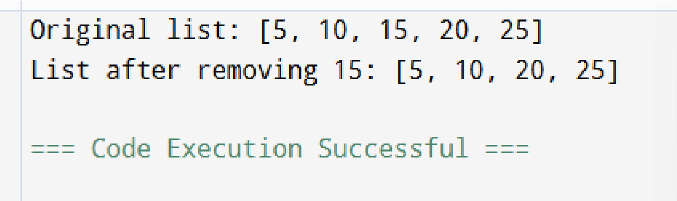
```
def append():

integer_list = [2, 4, 6, 8, 10]

print(f"Original list: {integer_list}")

integer_list.append(12)

print(f"List after appending 12: {integer_list}")

if __name__ == "__main__":

append()
```



```
Original list: [2, 4, 6, 8, 10]
List after appending 12: [2, 4, 6, 8, 10, 12]

=== Code Execution Successful ===
```

4. **Create an array [5,10,15,20,25]. Remove the element 15.**

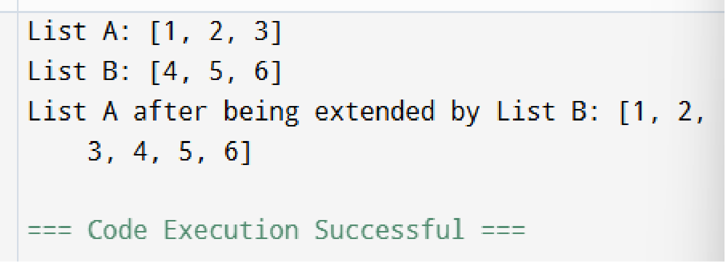
```
def remove():

my_list = [5, 10, 15, 20, 25]

print(f"Original list: {my_list}")

my_list.remove(15)

print(f"List after removing 15: {my_list}")

if __name__ == "__main__":

remove()
```



```
Original list: [5, 10, 15, 20, 25]
List after removing 15: [5, 10, 20, 25]

=== Code Execution Successful ===
```

5. **Create two arrays [1,2,3] and [4,5,6]. Extend the first array with the second**

```
def extend():
list_a = [1, 2, 3]
list_b = [4, 5, 6]
print(f"List A: {list_a}")
print(f"List B: {list_b}")
list_a.extend(list_b)
print(f"List A after being extended by List B: {list_a}")
if __name__ == "__main__":
extend()
```

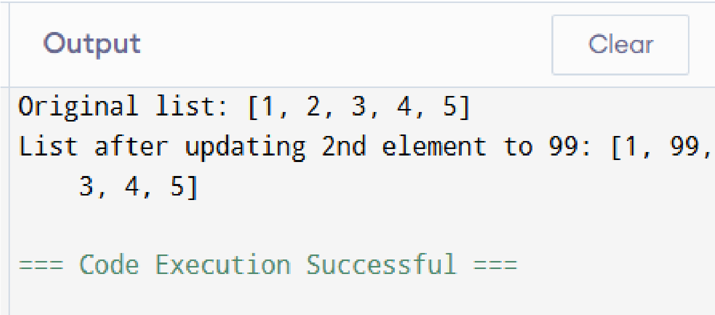


```
List A: [1, 2, 3]
List B: [4, 5, 6]
List A after being extended by List B: [1, 2,
    3, 4, 5, 6]

=== Code Execution Successful ===
```

6. **Create an array [1,2,3,4,5]. Update the 2nd element to 99.**

```
def update():
my_list = [1, 2, 3, 4, 5]
print(f"Original list: {my_list}")
my_list[1] = 99
print(f"List after updating 2nd element to 99: {my_list}")
if __name__ == "__main__":
update()
```

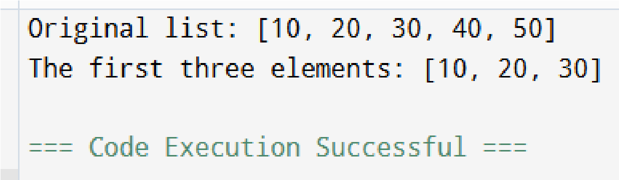
Vaishnavi Ashok Patil
C07
SPP-II



```
Original list: [1, 2, 3, 4, 5]
List after updating 2nd element to 99: [1, 99,
    3, 4, 5]

=== Code Execution Successful ===
```

7. **Create an array [10,20,30,40,50]. Use slicing to print the first three elements.**

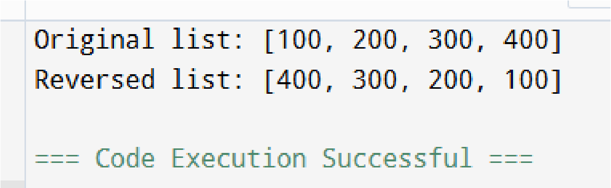
```
def run_exercise_7():

my_list = [10, 20, 30, 40, 50]

print(f"Original list: {my_list}")

first_three = my_list[:3]

print(f"The first three elements: {first_three}")

if __name__ == "__main__":

run_exercise_7()
```



```
Original list: [10, 20, 30, 40, 50]
The first three elements: [10, 20, 30]

=== Code Execution Successful ===
```

8. **Create an array [100,200,300,400]. Reverse the array using slicing.**

```
def reverse():

my_list = [100, 200, 300, 400]

print(f"Original list: {my_list}")

reversed_list = my_list[::-1]

print(f"Reversed list: {reversed_list}")

if __name__ == "__main__":

reverse()
```



```
Original list: [100, 200, 300, 400]
Reversed list: [400, 300, 200, 100]

=== Code Execution Successful ===
```

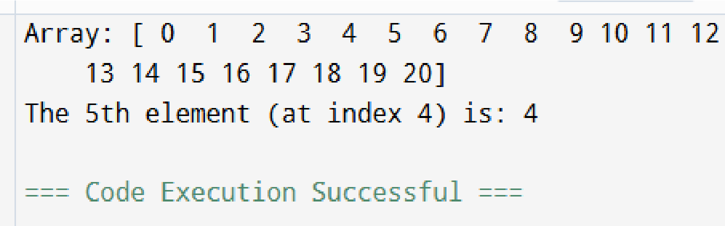
- **Numpy:-**

Vaishnavi Ashok Patil
C07
SPP-II

- **1D array:-**

1. **Create a 1D array of numbers from 0 to 20. Print the 5th element.**

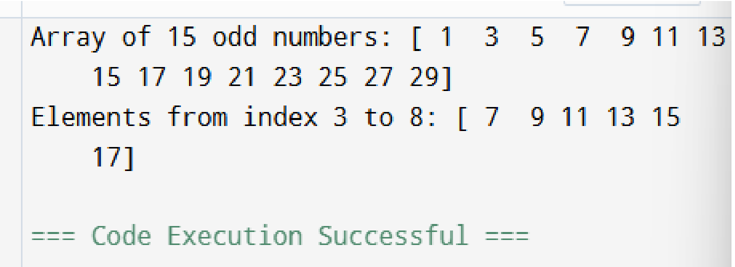
```
import numpy as np

def index():

arr = np.arange(21)

print(f"Array: {arr}")

fifth_element = arr[4]

print(f"The 5th element (at index 4) is: {fifth_element}")

index()
```



```
Array: [ 0  1  2  3  4  5  6  7  8  9 10 11 12
    13 14 15 16 17 18 19 20]
The 5th element (at index 4) is: 4

=== Code Execution Successful ===
```

2. **Create a 1D array of the first 15 odd numbers. Slice elements from index 3 to 8.**

```
import numpy as np

def slice():

odd_numbers = np.arange(1, 30, 2)

print(f"Array of 15 odd numbers: {odd_numbers}")

sliced_array = odd_numbers[3:9]

print(f"Elements from index 3 to 8: {sliced_array}")

slice()
```



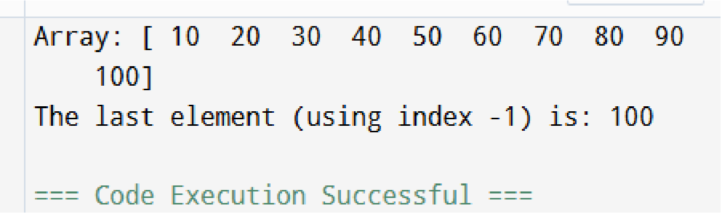
```
Array of 15 odd numbers: [ 1  3  5  7  9 11 13
    15 17 19 21 23 25 27 29]
Elements from index 3 to 8: [ 7  9 11 13 15
    17]

=== Code Execution Successful ===
```

3. **Create a 1D array of numbers from 10 to 100 with a step of 10. Print the last element using negative indexing.**

```
def negative():

arr = np.arange(10, 101, 10)

print(f"Array: {arr}")

last_element = arr[-1]
```

print(f"The last element (using index -1) is: {last_element}")

negative()



```
Array: [ 10  20  30  40  50  60  70  80  90
   100]
The last element (using index -1) is: 100

=== Code Execution Successful ===
```

4. **Create a 1D array of numbers 1 to 12. Reshape it into a (3,4) array.**

import numpy as np

def reshape():

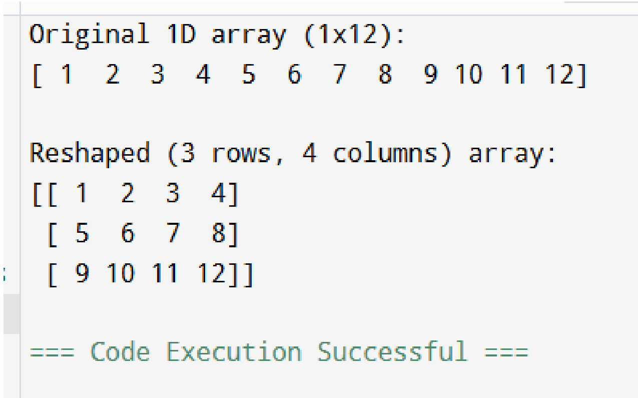
arr_1d = np.arange(1, 13)

print(f"Original 1D array (1x12): \n{arr_1d}")

arr_2d = arr_1d.reshape(3, 4)

print(f"\nReshaped (3 rows, 4 columns) array: \n{arr_2d}")

reshape()



```
Original 1D array (1x12):
[ 1  2  3  4  5  6  7  8  9 10 11 12]

Reshaped (3 rows, 4 columns) array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]

=== Code Execution Successful ===
```

5. **Create an array [5,10,15,20,25]. Broadcast it by adding 5 to all elements.**

import numpy as np

def broadcasting():

arr = np.array([5, 10, 15, 20, 25])

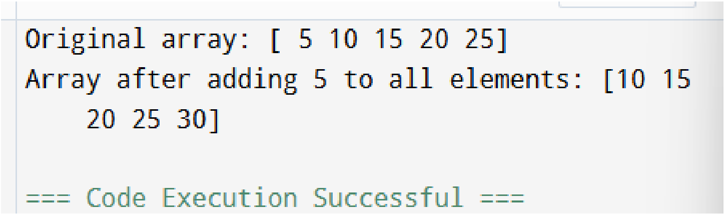
print(f"Original array: {arr}")

result_array = arr + 5

print(f"Array after adding 5 to all elements: {result_array}")

broadcasting()

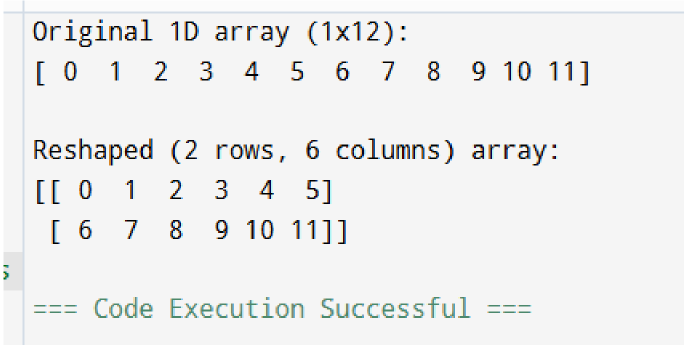
Vaishnavi Ashok Patil
C07
SPP-II



```
Original array: [ 5 10 15 20 25]
Array after adding 5 to all elements: [10 15
    20 25 30]

=== Code Execution Successful ===
```

6. Create a 1D array of 12 elements and reshape it into (2,6).

```
import numpy as np

def reshape():

arr_1d = np.arange(12)

print(f"Original 1D array (1x12): \n{arr_1d}")

arr_2d = arr_1d.reshape(2, 6)

print(f"\nReshaped (2 rows, 6 columns) array: \n{arr_2d}")

reshape()
```

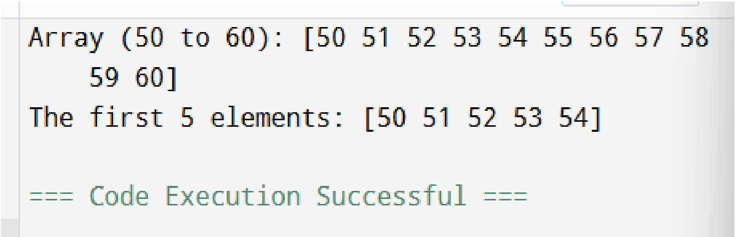


```
Original 1D array (1x12):
[ 0  1  2  3  4  5  6  7  8  9 10 11]

Reshaped (2 rows, 6 columns) array:
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]

=== Code Execution Successful ===
```

7. Create a 1D array of numbers from 50 to 60. Slice the first 5 elements.

```
import numpy as np

def slice_first():

arr = np.arange(50, 61)

print(f"Array (50 to 60): {arr}")

first_five = arr[:5]

print(f"The first 5 elements: {first_five}")

slice_first()
```

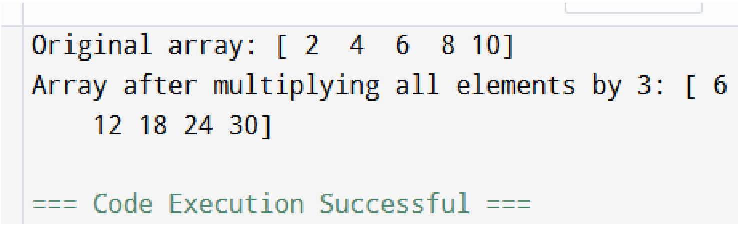
Vaishnavi Ashok Patil
C07
SPP-II



```
Array (50 to 60): [50 51 52 53 54 55 56 57 58
    59 60]
The first 5 elements: [50 51 52 53 54]

=== Code Execution Successful ===
```

8. Create a 1D array [2,4,6,8,10]. Broadcast it by multiplying with 3.

```
import numpy as np

def broad_mult():

arr = np.array([2, 4, 6, 8, 10])

print(f"Original array: {arr}")

result_array = arr * 3

print(f"Array after multiplying all elements by 3: {result_array}")

broad_mult()
```

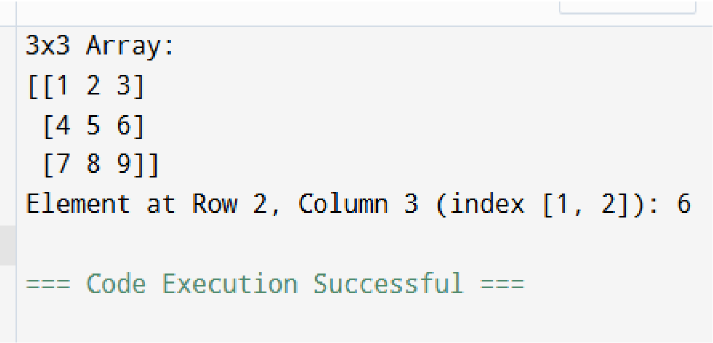


```
Original array: [ 2  4  6  8 10]
Array after multiplying all elements by 3: [ 6
    12 18 24 30]

=== Code Execution Successful ===
```

- **2D array:-**

1. Create a 2D array of shape (3,3) with numbers 1–9. Print the element at row 2, col 3.

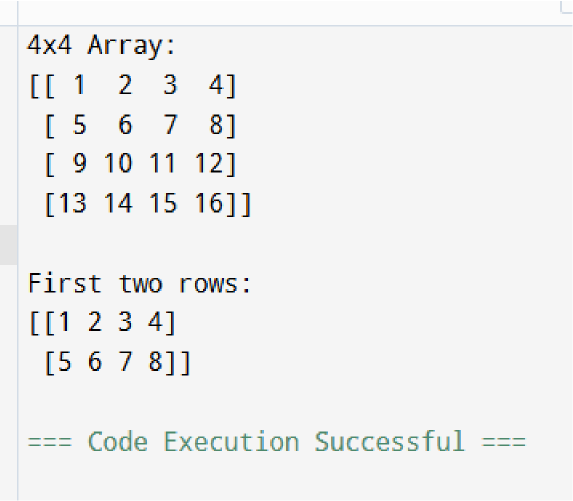
```
import numpy as np

def index():

arr = np.arange(1, 10).reshape(3, 3)

print(f"3x3 Array:\n{arr}")

element = arr[1, 2]

print(f"Element at Row 2, Column 3 (index [1, 2]): {element}")

index()
```



```
3x3 Array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Element at Row 2, Column 3 (index [1, 2]): 6

=== Code Execution Successful ===
```

2. Create a 2D array of shape (4,4) with numbers 1–16. Slice the first two rows.

```
import numpy as np

def slice():

arr = np.arange(1, 17).reshape(4, 4)

print(f"4x4 Array:\n{arr}")

first_two_rows = arr[0:2, :]

print(f"\nFirst two rows:\n{first_two_rows}")

slice()
```



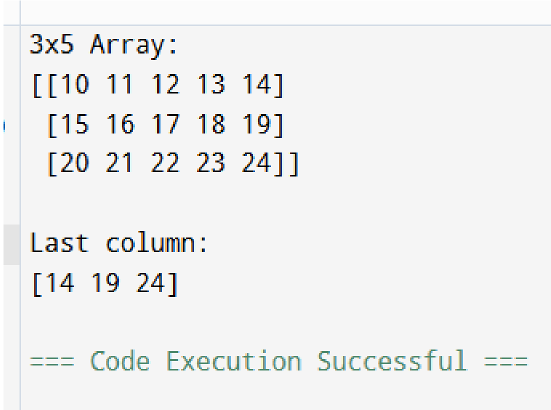
```
4x4 Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

First two rows:
[[1 2 3 4]
 [5 6 7 8]]

=== Code Execution Successful ===
```

3. Create a 2D array of shape (3,5) with numbers from 10 to 24. Slice the last column.

```
import numpy as np

def slicelast():

arr = np.arange(10, 25).reshape(3, 5)

print(f"3x5 Array:\n{arr}")

last_column = arr[:, -1]
```

print(f"\nLast column:\n{last_column}")

slicelast()



```
3x5 Array:
[[10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]

Last column:
[14 19 24]

=== Code Execution Successful ===
```

4. Create a 2D array of shape (2,6). Reshape it into (3,4).
import numpy as np

def reshape():

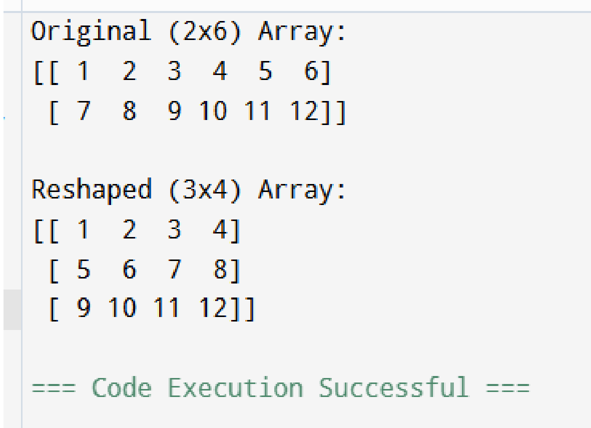
arr_2x6 = np.arange(1, 13).reshape(2, 6)

print(f"Original (2x6) Array:\n{arr_2x6}")

arr_3x4 = arr_2x6.reshape(3, 4)

print(f"\nReshaped (3x4) Array:\n{arr_3x4}")

reshape()

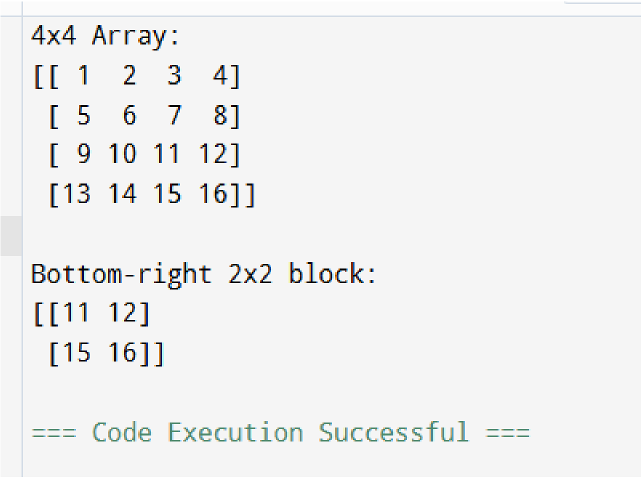


```
Original (2x6) Array:
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]

Reshaped (3x4) Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]

=== Code Execution Successful ===
```

5. Create a 2D array (3×3). Slice the first row.
import numpy as np

def slicefirst():

```
arr = np.arange(1, 10).reshape(3, 3)

print(f"3x3 Array:\n{arr}")

first_row = arr[0, :]

print(f"\nFirst row:\n{first_row}")

slicefirst()
```

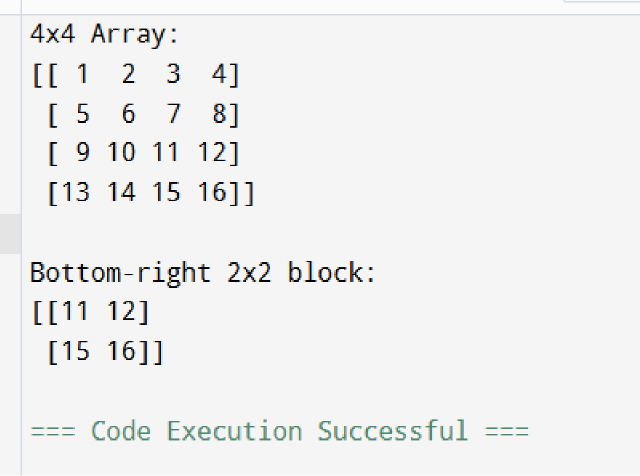


```
4x4 Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

Bottom-right 2x2 block:
[[11 12]
 [15 16]]

=== Code Execution Successful ===
```

6. Create a 2D array (4×4). Slice the last two rows and last two columns (bottom-right block).

```
import numpy as np

def sliceblock():

arr = np.arange(1, 17).reshape(4, 4)

print(f"4x4 Array:\n{arr}")

bottom_right_block = arr[2:, 2:]

print(f"\nBottom-right 2x2 block:\n{bottom_right_block}")

sliceblock()
```

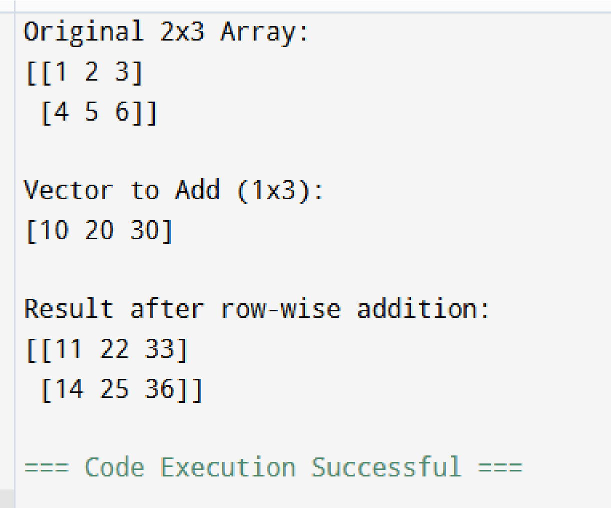
Vaishnavi Ashok Patil
C07
SPP-II



```
4x4 Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

Bottom-right 2x2 block:
[[11 12]
 [15 16]]

=== Code Execution Successful ===
```

7. Create a 2D array of shape (2,3). Broadcast by adding [10,20,30] to each row.

```
import numpy as np

def broadrowadd():

arr_2d = np.array([[1, 2, 3], [4, 5, 6]])

print(f"Original 2x3 Array:\n{arr_2d}")

vector_to_add = np.array([10, 20, 30])

print(f"\nVector to Add (1x3):\n{vector_to_add}")

result_array = arr_2d + vector_to_add

print(f"\nResult after row-wise addition:\n{result_array}")

broadrowadd()
```



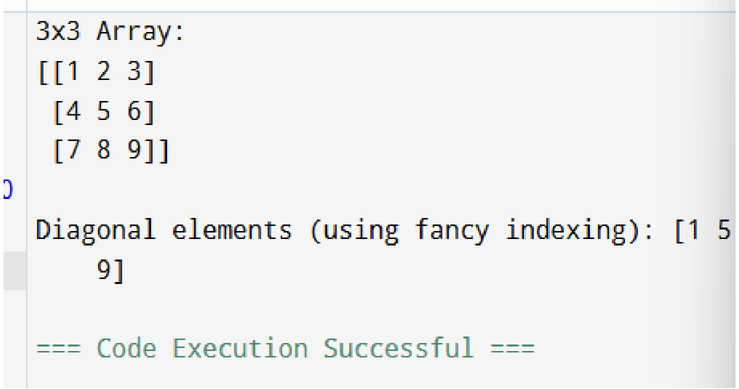
```
Original 2x3 Array:
[[1 2 3]
 [4 5 6]]

Vector to Add (1x3):
[10 20 30]

Result after row-wise addition:
[[11 22 33]
 [14 25 36]]

=== Code Execution Successful ===
```

8. Create a 2D array (3×3). Print the diagonal elements using indexing.

```
import numpy as np
```

Vaishnavi Ashok Patil
C07
SPP-II

```
def diagonal():

arr = np.arange(1, 10).reshape(3, 3)

print(f"3x3 Array:\n{arr}")

diagonal_elements = arr[[0, 1, 2], [0, 1, 2]]

print(f"\nDiagonal elements (using fancy indexing): {diagonal_elements}")

diagonal()
```



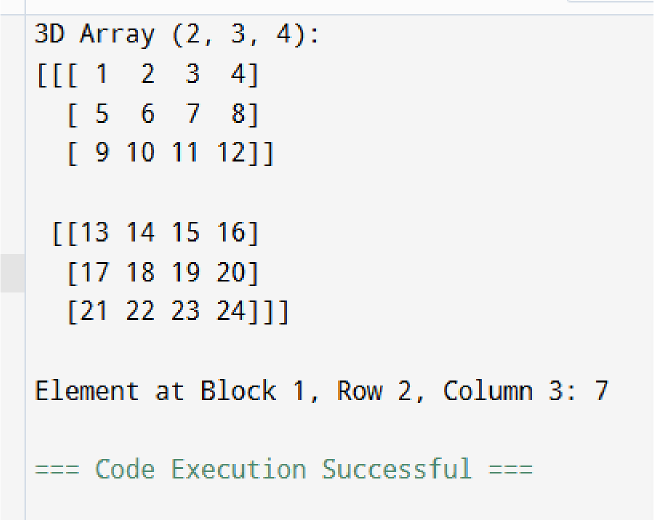
```
3x3 Array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Diagonal elements (using fancy indexing): [1 5
    9]

=== Code Execution Successful ===
```

- **3D array:-**

1. Create a 3D array of shape (2,3,4) with numbers 1–24. Print element at [1,2,3].

```
import numpy as np

def a():

arr = np.arange(1, 25).reshape(2, 3, 4)

print(f"3D Array (2, 3, 4):\n{arr}")

element = arr[0, 1, 2]

print(f"\nElement at Block 1, Row 2, Column 3: {element}")

a()
```

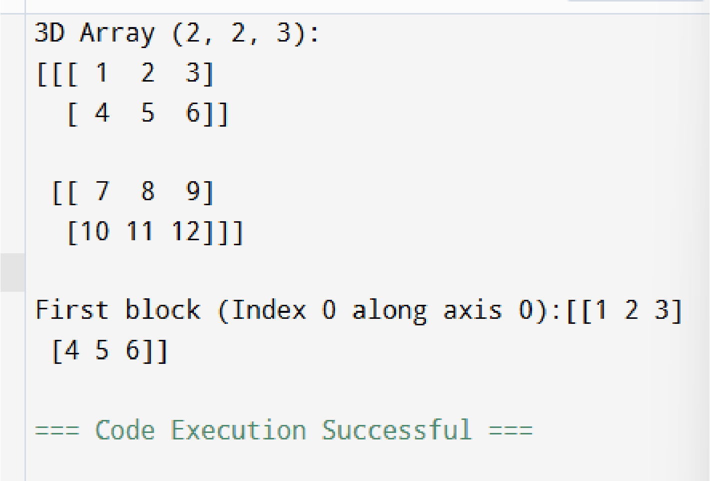
Vaishnavi Ashok Patil
C07
SPP-II



```
3D Array (2, 3, 4):
[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]

Element at Block 1, Row 2, Column 3: 7

=== Code Execution Successful ===
```

2. Create a 3D array of shape (2,2,3). Slice the first “block” (all rows/cols of index 0 along axis 0)

```
import numpy as np

def b():

arr = np.arange(1, 13).reshape(2, 2, 3)

print(f"3D Array (2, 2, 3):\n{arr}")

first_block = arr[0, :, :]

print(f"\nFirst block (Index 0 along axis 0):{first_block}")

b()
```

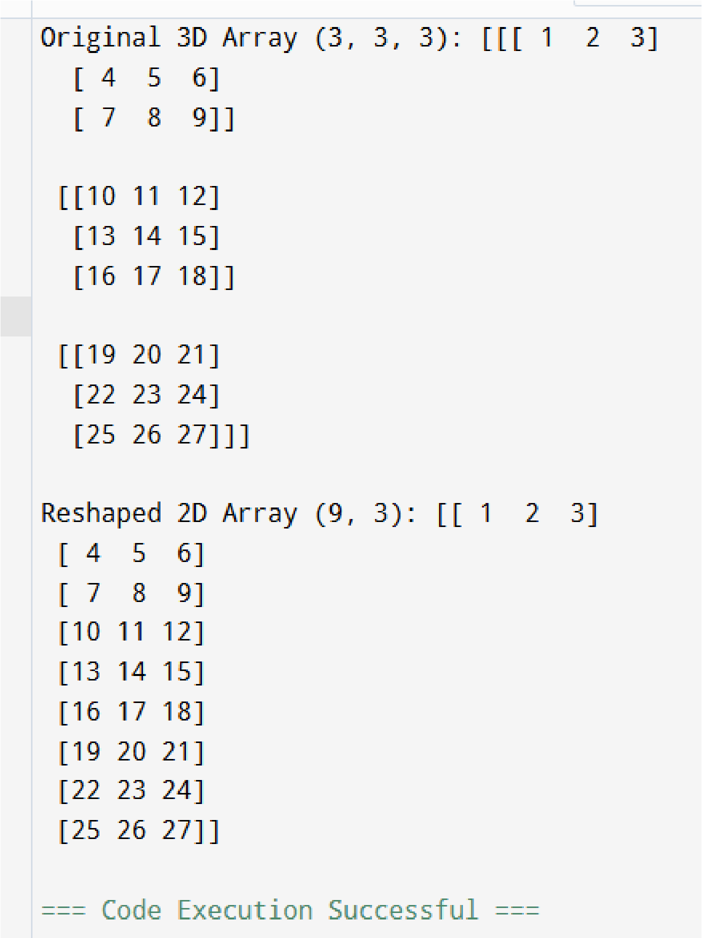


```
3D Array (2, 2, 3):
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]

First block (Index 0 along axis 0):[[1 2 3]
 [4 5 6]]

=== Code Execution Successful ===
```

3. Create a 3D array (3,3,3) with numbers 1–27. Reshape it into (9,3).

```
import numpy as np

def c():

arr_3d = np.arange(1, 28).reshape(3, 3, 3)
```

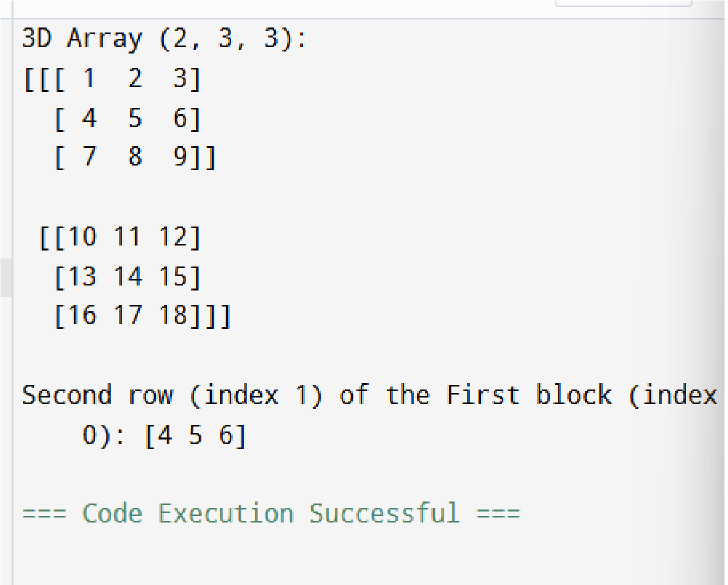
```
print(f"Original 3D Array (3, 3, 3): {arr_3d}")

arr_2d = arr_3d.reshape(9, 3)

print(f"\nReshaped 2D Array (9, 3): {arr_2d}")

c()
```



```
Original 3D Array (3, 3, 3): [[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]]

 [[10 11 12]
  [13 14 15]
  [16 17 18]]

 [[19 20 21]
  [22 23 24]
  [25 26 27]]]

Reshaped 2D Array (9, 3): [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]
 [13 14 15]
 [16 17 18]
 [19 20 21]
 [22 23 24]
 [25 26 27]]

=== Code Execution Successful ===
```

4. Create a 3D array (2,3,3). Slice the second row of the first block.

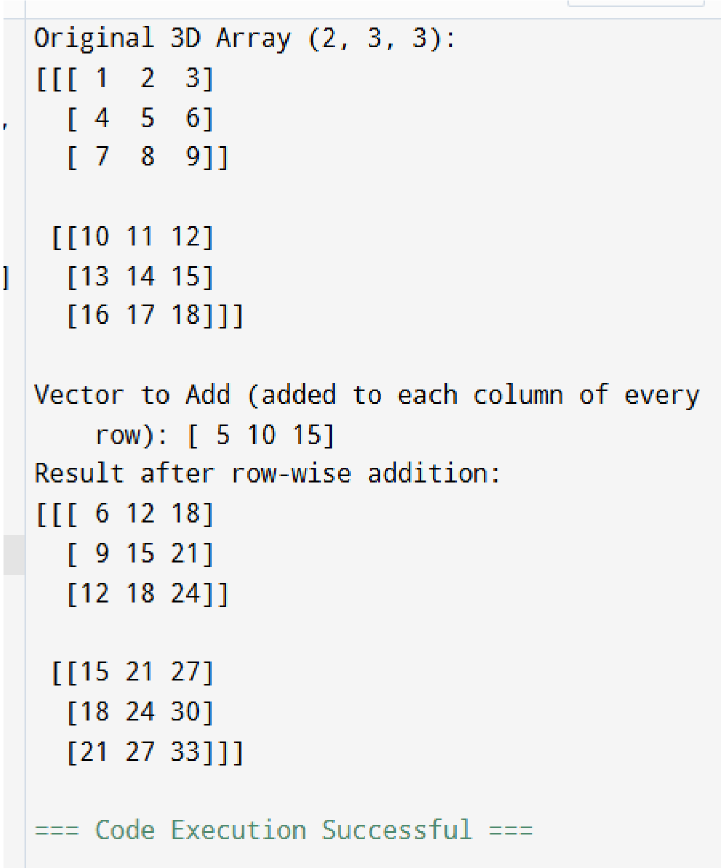
```
import numpy as np

def d():

arr = np.arange(1, 19).reshape(2, 3, 3)

print(f"3D Array (2, 3, 3):\n{arr}")

second_row_first_block = arr[0, 1, :]

print(f"\nSecond row (index 1) of the First block (index 0): {second_row_first_block}")

d()
```



```
3D Array (2, 3, 3):
[[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]]

 [[10 11 12]
  [13 14 15]
  [16 17 18]]]

Second row (index 1) of the First block (index
    0): [4 5 6]

=== Code Execution Successful ===
```

5. Create a 3D array (2,3,3). Broadcast by adding [5,10,15] to each row.

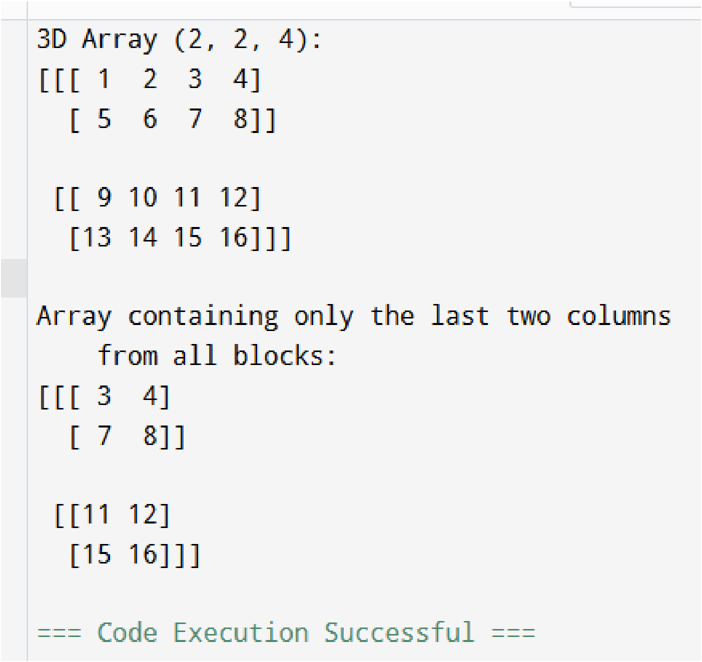
```
import numpy as np

def e():

arr_3d = np.arange(1, 19).reshape(2, 3, 3)

print(f"Original 3D Array (2, 3, 3):\n{arr_3d}")

vector_to_add = np.array([5, 10, 15])

print(f"\nVector to Add (added to each column of every row): {vector_to_add}")

result_array = arr_3d + vector_to_add

print(f"Result after row-wise addition:\n{result_array}")

e()
```



```
Original 3D Array (2, 3, 3):
[[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]]

 [[10 11 12]
  [13 14 15]
  [16 17 18]]]

Vector to Add (added to each column of every
    row): [ 5 10 15]
Result after row-wise addition:
[[[ 6 12 18]
  [ 9 15 21]
  [12 18 24]]

 [[15 21 27]
  [18 24 30]
  [21 27 33]]]

=== Code Execution Successful ===
```

6. Create a 3D array (2,2,4) with numbers from 1 to 16. Slice the last two columns of all blocks.

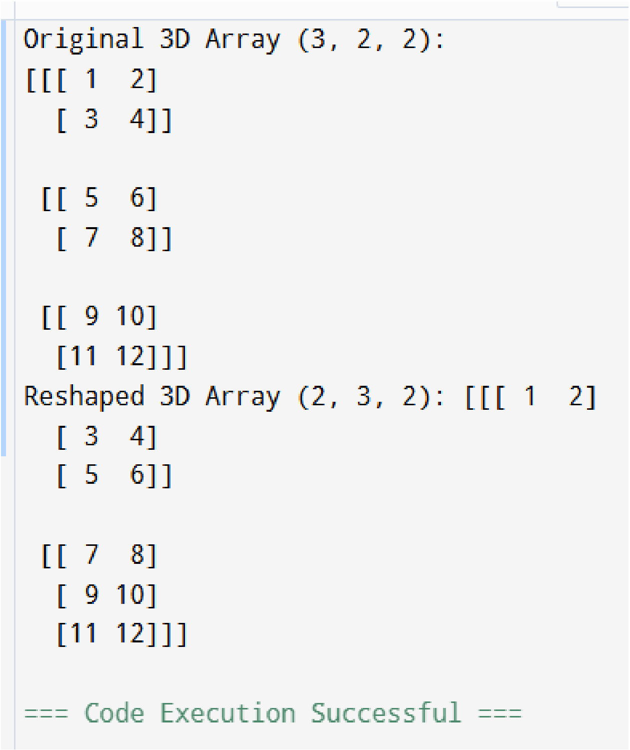
```
import numpy as np

def f():

arr = np.arange(1, 17).reshape(2, 2, 4)

print(f"3D Array (2, 2, 4):\n{arr}")

last_two_columns = arr[:, :, -2:]

print(f"\nArray containing only the last two columns from all blocks:\n{last_two_columns}")

f()
```



```
3D Array (2, 2, 4):
[[[ 1  2  3  4]
  [ 5  6  7  8]]

 [[ 9 10 11 12]
  [13 14 15 16]]]

Array containing only the last two columns
    from all blocks:
[[[ 3  4]
  [ 7  8]]

 [[11 12]
  [15 16]]]

=== Code Execution Successful ===
```

7. Create a 3D array (2,3,2). Print the element at [0,1,1].

```
import numpy as np

def g():

arr = np.arange(1, 13).reshape(2, 3, 2)

print(f"3D Array (2, 3, 2):\n{arr}")

element = arr[0, 1, 1]

print(f"\nElement at Index [0, 1, 1]: {element}")

g()
```

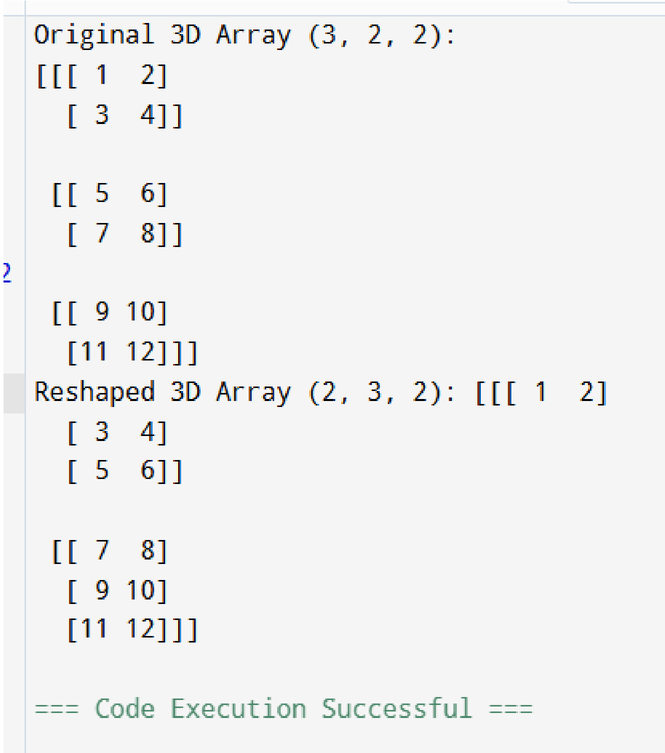


```
Original 3D Array (3, 2, 2):
[[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
Reshaped 3D Array (2, 3, 2): [[[ 1  2]
  [ 3  4]
  [ 5  6]]

 [[ 7  8]
  [ 9 10]
  [11 12]]]

=== Code Execution Successful ===
```

8. Create a 3D array (3,2,2). Reshape it into (2,3,2).

```
import numpy as np

def h():

arr_original = np.arange(1, 13).reshape(3, 2, 2)

print(f"Original 3D Array (3, 2, 2):\n{arr_original}")

arr_reshaped = arr_original.reshape(2, 3, 2)

print(f"Reshaped 3D Array (2, 3, 2): {arr_reshaped}")

h()
```

Vaishnavi Ashok Patil
C07
SPP-II



```
Original 3D Array (3, 2, 2):
[[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
Reshaped 3D Array (2, 3, 2): [[[ 1  2]
  [ 3  4]
  [ 5  6]]

 [[ 7  8]
  [ 9 10]
  [11 12]]]

=== Code Execution Successful ===
```