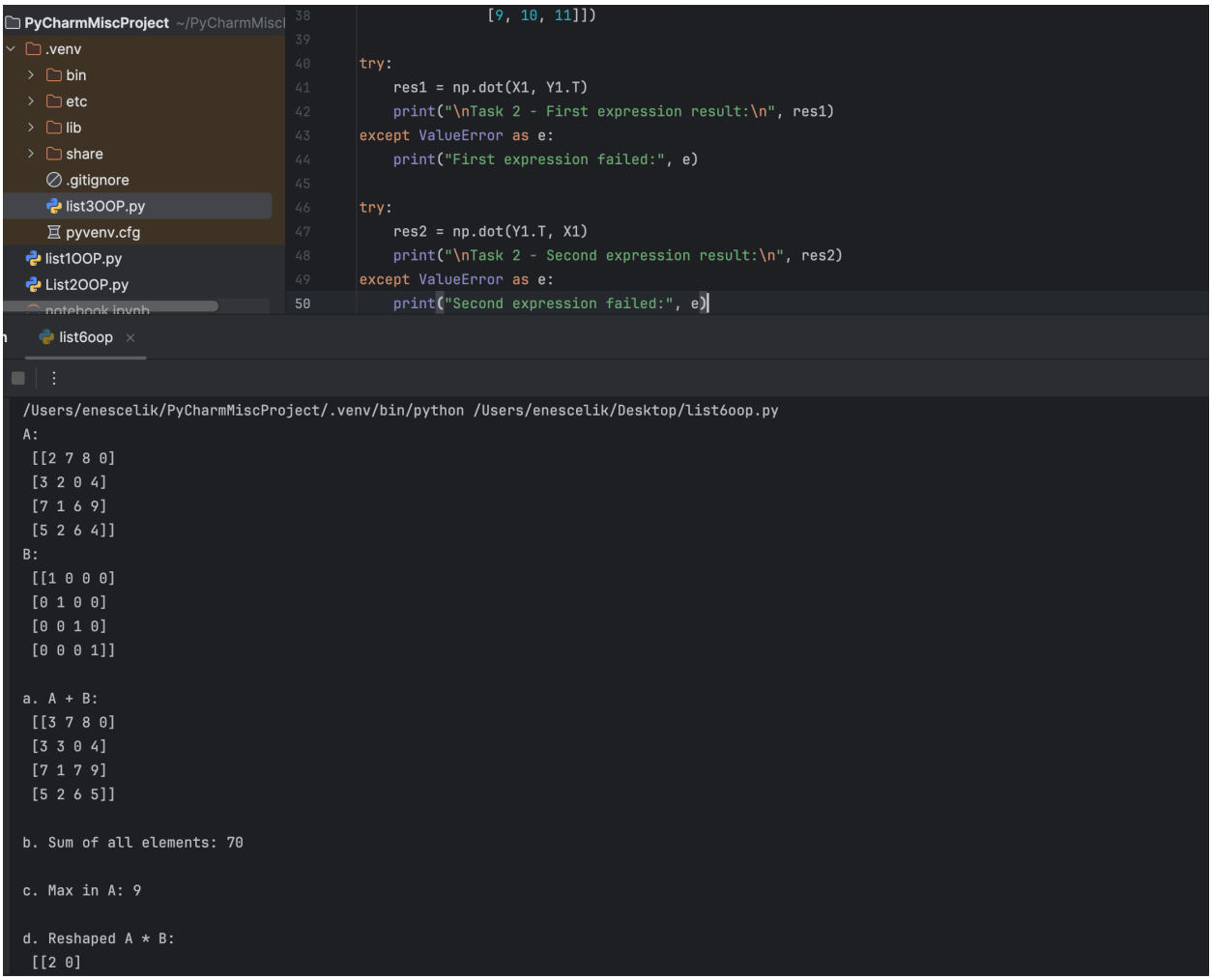
In this NumPy exercise, matrix operations is performed to demonstrate fundamental skills in numerical computing. A 4x4 matrix with random integers (A) and a 4x4 identity matrix (B) were created. Element-wise addition, total summation of all elements, and maximum value extraction from matrix A were accurately executed. The matrices were reshaped into 8x2 form and multiplied element-wise, and specific column and row sums were calculated. Additionally, the second column of A was squared, and both matrices were horizontally concatenated. They were also converted to string type for element-wise string concatenation. Finally, two valid matrix multiplication expressions were evaluated in Task 2, both yeilding correct results. After correcting a minor mistake in the total element summation, all outputs matched the expected results. This exercise provided solid practice in reshaping, indexing, broadcasting, type conversion, and matrix multiplication using NumPy.

Code&Output:

```python
import numpy as np
# A: 4x4 random integers (0-9), B: 4x4 identity matrix
A = np.random.randint( low: 0,  high: 10,  size: (4, 4))
B = np.eye( N: 4, dtype=int)
print("A:\n", A)
print("B:\n", B)
# a. A + B
print("\na. A + B:\n", A + B)
# b. Sum of all elements in A and B
print("\nb. Sum of all elements:", A.sum() + B.sum())
# c. Max in A
print("\nc. Max in A:", A.max())
# d. Reshape to 8x2 and multiply
A_r = A.reshape(8, 2)
B_r = B.reshape(8, 2)
print("\nd. Reshaped A * B:\n", A_r * B_r)
# e. Sum of 3rd column in A and 3rd row in B
print("\ne. Sum of A[:,2] + B[2,:]:", A[:, 2].sum() + B[2, :].sum())
# f. Square second column in A
A_f = A.copy()
A_f[:, 1] = A_f[:, 1] ** 2
print("\nf. A with 2nd column squared:\n", A_f)
# g. Horizontal join (4x8)
AB = np.hstack((A, B))
print("\ng. A and B joined:\n", AB)
# h. Convert to string and add
A_str = A.astype(str)
B_str = B.astype(str)
print("\nh. A + B (as strings):\n", A_str + B_str)

# Task 2 - Matrix Expressions
X1 = np.array([[2, 2, 2],
               [2, 2, 2],
               [2, 2, 2]])

Y1 = np.array([[3, 4, 5],
               [6, 7, 8],
               [9, 10, 11]])
```


PyCharmMiscProject ~/PyCharmMisc
.venv
bin
etc
lib
share
.gitignore
list3OOP.py
pyvenv.cfg
list1OOP.py
List2OOP.py
notebook.ipynb
[9, 10, 11]])
try:
res1 = np.dot(X1, Y1.T)
print("\nTask 2 - First expression result:\n", res1)
except ValueError as e:
print("First expression failed:", e)
try:
res2 = np.dot(Y1.T, X1)
print("\nTask 2 - Second expression result:\n", res2)
except ValueError as e:
print("Second expression failed:", e)
list6oop
/Users/enescelik/PyCharmMiscProject/.venv/bin/python /Users/enescelik/Desktop/list6oop.py
A:
[[2 7 8 0]
[3 2 0 4]
[7 1 6 9]
[5 2 6 4]]
B:
[[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]]
a. A + B:
[[3 7 8 0]
[3 3 0 4]
[7 1 7 9]
[5 2 6 5]]
b. Sum of all elements: 70
c. Max in A: 9
d. Reshaped A * B:
[[2 0]

```
d. Reshaped A * B:
 [[2 0]
 [0 0]
 [0 2]
 [0 0]
 [0 0]
 [6 0]
 [0 0]
 [0 4]]

e. Sum of A[:,2] + B[2,:]: 21

f. A with 2nd column squared:
 [[ 2 49  8  0]
 [ 3  4  0  4]
 [ 7  1  6  9]
 [ 5  4  6  4]]

g. A and B joined:
 [[2 7 8 0 1 0 0 0]
 [3 2 0 4 0 1 0 0]
 [7 1 6 9 0 0 1 0]
 [5 2 6 4 0 0 0 1]]

h. A + B (as strings):
 [['21' '70' '80' '00']
 ['30' '21' '00' '40']
 ['70' '10' '61' '90']
 ['50' '20' '60' '41']]

Task 2 - First expression result:
 [[24 42 60]
 [24 42 60]
 [24 42 60]]

Task 2 - Second expression result:
 [[36 36 36]
```

```
Task 2 - Second expression result:
 [[36 36 36]
 [42 42 42]
 [48 48 48]]

Process finished with exit code 0
```