📝 Tasks on NumPy Arrays

Task 1: Student Grades Analysis

- Create a **2D NumPy array** representing marks of 5 students in 4 subjects.
- Find:
  - Average marks per student
  - Highest marks in each subject
  - Index of student with the lowest total score

```
# task 1
marks = np.array([
    [78, 85, 90, 66],
    [88, 72, 95, 80],
    [60, 70, 65, 75],
    [92, 88, 84, 91],
    [55, 64, 58, 70]
])

print("Average per student:", marks.mean(axis=1))
print("Highest per subject:", marks.max(axis=0))
print("Index of student with lowest total:", marks.sum(axis=1).argmin())
```

✓ 0.0s

```
Average per student: [79.75 83.75 67.5  88.75 61.75]
Highest per subject: [92 88 95 91]
Index of student with lowest total: 4
```

Task 2: Image Reshaping

- Create a **1D array of size 36** with values from 0–35.
- Reshape it into:
  - A 6×6 matrix
  - Then into a 3D array of shape (3, 3, 4)
- Print the shapes at each step.

```
arr = np.arange(36)
arr_6x6 = arr.reshape(6, 6)
arr_3d = arr.reshape(3, 3, 4)

print("6x6 shape:", arr_6x6.shape)
print("3D shape:", arr_3d.shape)
```

✓ 0.0s

```
6x6 shape: (6, 6)
3D shape: (3, 3, 4)
```

Task 3: Random Data Simulation

- Generate a **3×4 NumPy array of random integers (1–100)**.
- Sort each row individually.
- Insert a new column at the end with the **row sums**.

```
random_arr = np.random.randint(1, 101, size=(3, 4))
sorted_arr = np.sort(random_arr, axis=1)
row_sums = sorted_arr.sum(axis=1).reshape(-1, 1)
# print(row_sums)
new_arr = np.concatenate((sorted_arr, row_sums), axis=1)

print("Random array:\n", random_arr)
print("Sorted row-wise:\n", sorted_arr)
print("With row sums:\n", new_arr)
```

✓ 0.0s

```
Random array:
 [[97  4 95 76]
 [95 71 11 97]
 [40 66 22 95]]
Sorted row-wise:
 [[ 4 76 95 97]
 [11 71 95 97]
 [22 40 66 95]]
With row sums:
 [[  4  76  95  97 272]
 [ 11  71  95  97 274]
 [ 22  40  66  95 223]]
```

Task 4: Splitting & Combining Data

- Create a **1D array of numbers from 1 to 20**.
- Split it into 4 equal parts.
- Combine the 2nd and 4th parts into one array.

```
arr = np.arange(1, 21)
new_arr = np.split(arr, 4)
print()
combined_arr = np.concatenate((new_arr[1], new_arr[3]))

print("Split parts:", new_arr)
print("Combined 2nd and 4th:", combined_arr)
```

✓ 0.0s

```
Split parts: [array([1, 2, 3, 4, 5]), array([ 6,  7,  8,  9, 10]), array([11, 12, 13, 14, 15]), array([16, 17, 18, 19, 20])]
Combined 2nd and 4th: [ 6  7  8  9 10 16 17 18 19 20]
```

-

Task 5: Data Cleaning with NumPy

- Given an array:

- `arr = np.array([12, -5, 0, 23, -15, 45, 30, -2])`
- Remove all negative numbers.
- Insert the value `99` at index 2.
- Sort the final array in ascending order.

```
arr = np.array([12, -5, 0, 23, -15, 45, 30, -2])
cleaned = arr[arr >= 0]
cleaned = np.insert(cleaned, 2, 99)
final_cleaned = np.sort(cleaned)

print(final_cleaned)
```

✓ 0.0s

```
[ 0 12 23 30 45 99]
```

Task 6: Stock Price Analysis (3D Array)

- Create a **3D NumPy array of shape (2, 5, 3)** representing stock prices:
  - 2 companies
  - 5 days
  - 3 features: [Open, High, Close]
- Find:
  - Maximum closing price for each company
  - Day index when each company's stock opened lowest

```
print("Stock prices:\n", stock_prices)
print("Max closing price per company:", max_close)
print("Lowest open day per company:", lowest_open_day)
```

✓ 0.0s

```
Stock prices:
 [[[183 140 195]
  [127 134 184]
  [109 196 162]
  [128 121 108]
  [114 105 188]]

 [[170 200 130]
  [144 115 196]
  [112 179 100]
  [183 118 164]
  [138 124 189]]]
Max closing price per company: [195 196]
Lowest open day per company: [2 2]
```
