Manual Task A1.(a) - Explain how 2D slicing works in Numpy

Answer:

In Numpy, 2D slicing allows us to extract a smaller rectangular section (subarray) from a 2-dimensional array by specifying row and column index ranges.

Example:

$$A = \begin{bmatrix} 10 & 20 & 30 & 40 \\ 50 & 60 & 70 & 80 \\ 90 & 100 & 110 & 120 \\ 130 & 140 & 150 & 160 \end{bmatrix}$$

Basic Syntax:

array[row_start : row_end, col_start : col_end]

If we write: A[1:3, 1:3]. It means - take rows 1 and 2, take columns 1 and 2.

So the result is:

$$\begin{bmatrix} 60 & 70 \\ 100 & 110 \end{bmatrix}$$

Simple Diagram:

Full Array (A):

Row ↓ \ col →	0	1	2	3
0	10	20	30	40
1	50	[60]	[70]	80
2	90	[100]	[110]	120
3	130	140	150	160

The boxed part [] is selected by A[1:3,1:3]

2D slicing extracts a block of data from an array by specifying row and column index range.

Manual Task A1 (b) – Pseudocode to Compute Student-wise Averages Using Array Slicing

Answer:

```
BEGIN
    INPUT marks // 2D array of size [num_students] ← [num_subjects]
    num_students ← number of rows in marks
    num_subjects ← number of columns in marks
    CREATE array student_avg of size [num_students]
    For i ← 0 To num_students - 1 Do
        student_marks ← marks[i, 0 : num_subjects] // slice each student's marks
        total ← SUM(student_marks)
        avg ← total / num_subjects
        student_avg[i] ← avg
```

```
    END FOR
    OUTPUT "Student-wise averages:",
student_avg
END.
```

Manual Task B1.(a) - How Pandas Series Allow Easier Label-Based Analysis than Numpy Arrays.

Answers:

Pandas Series make data analysis easier because they use labels (indexes) instead of just numeric positions. Each value in a Series is associated with a label, which helps identify and access data more intuitively. For example, you can write attendance["S1"] instead of attendance[0] which is clearer and less error-prone.

Series also automatically align data by labels when performing operations, making it easy to combine or compare data from different sources.

In contrast, NumPy arrays rely only on numeric indexes, so you must remember the positions of each element manually.

Manual Task B1(b) - ~~Show~~ How a Boolean Mask Works on a Series

Answer:

A boolean mask is a set of True or False values that we apply to a Pandas Series to filter data. Only the elements corresponding to True values in the mask are selected.

Let's take a series of attendance percentages:

attendance =

s1	92
s2	80
s3	95
s4	70
s5	88
s6	85

dtype: int64

Now, created a boolean mask for students who have attendance ≥ 85:

```
mask = attendance >= 85
mask =
S1  True
S2  False
S3  True
S4  False
S5  True
S6  True
dtype: bool
```

Applying the mask:

```
attendance[mask] =
S1  92
S3  95
S5  88
S6  85
dtype: int64
```

Explanation:

The mask keeps only the entries where the condition is True. So students S1, S3, S5, and S6 are displayed, while others are filtered out.