I
E & C
T
IIT Guwahati

Advanced Certification Programme in Data Science Business Analytics

Week 15
Case Study
NumPy

Topics Covered

- Pandas Exercise
- Async Recap
- Q and A

Async Recap

1. Getting Started with Python:
Set up Python and explore tools like VS Code, PyCharm and Jupyter to begin writing and running code easily.

2. Python Fundamentals:
Learn basic syntax, data types and conditional statements to build logical Python programs.

3. Working with Loops and Functions:

Understand how to automate tasks using for and while loops, and define reusable blocks using def.

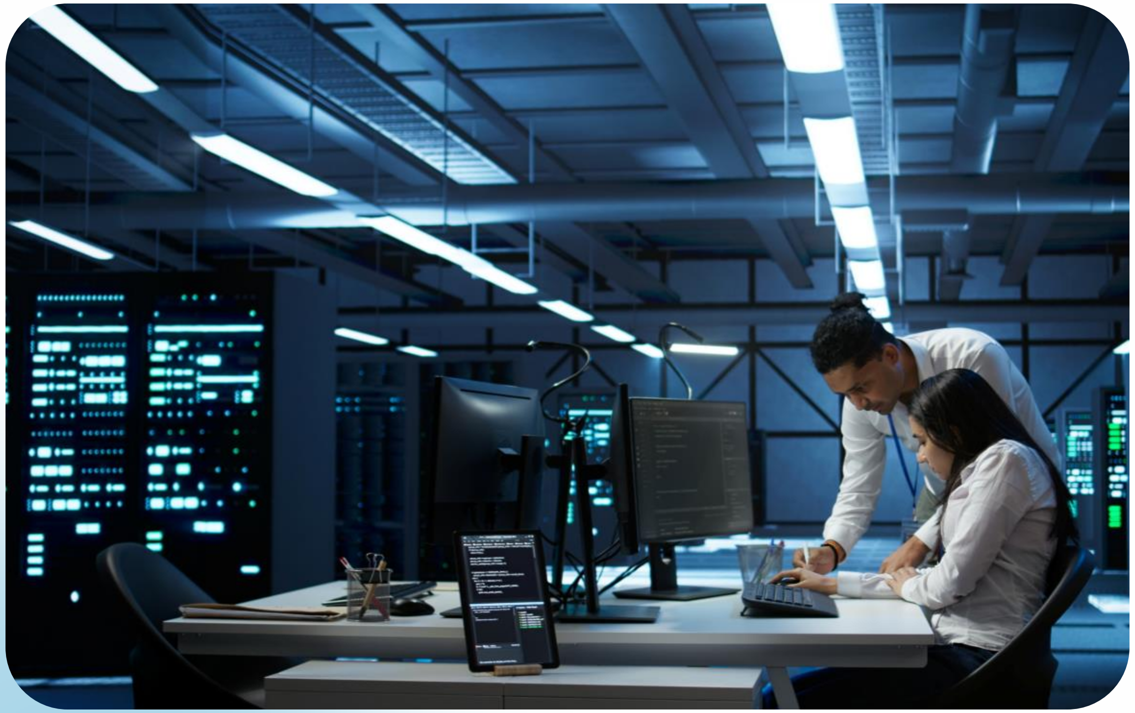
4. Data Structures in Python:
Use lists, tuples, sets and dictionaries to organise and manage collections of data efficiently.

5. NumPy & Pandas for Data Analysis:

Leverage NumPy arrays for numerical operations and use Pandas for reading, cleaning and manipulating structured datasets.

Getting Started with NumPy Arrays

Efficient Data Structures for Numerical Computing



- NumPy is a Python library for fast numerical computations
- Arrays are grid-like and more efficient than Python lists
- NumPy arrays are homogeneous (same data type)
- Ideal for data analysis, machine learning and scientific computing
- Widely used in statistical modelling for speed and flexibility

Core NumPy Array Concepts

Essential Tools for Efficient Data Handling

- **Array creation**: Use np.array() to create arrays
- **Element-wise operations**: Apply functions to each element
- **Multidimensional arrays**: Work with 1D, 2D or higher
- **Indexing and slicing**: Access and manipulate data subsets

Exercise 1: NumPy Array Creation & Operations

Hands-On with 1D and 2D Arrays

- **Create 1D array:** [10, 20, 30, 40, 50]
- **Add elements:** Add 10 to each item in the array
- **Create 2D array:** 2×3 grid with numbers 1 to 6
- **Indexing:** Access the second row of the 2D array
- **Slicing:** Get first 3 items of the 1D array
- **Multiply elements:** Use [2, 3, 4, 5, 6] for element-wise multiplication

Exercise 2: NumPy Array Operations

Practising Random Generation and Statistical Tasks

- **Create 1D array:** 10 random numbers using np.random.randint()
- **Find statistics:** Calculate sum, mean and standard deviation
- **Create 2D array:** 3×3 array of random integers between 1 and 100
- **Column and row operations**: Sum of each column and mean of each row
- **Reshape array**: Convert 2D array into 1D and print it

Exercise3 : Reshaping and Broadcasting

Working with Array Shapes and Advanced Operations

Create 1D array:

- Numbers from 1 to 12

Reshape array:

- Convert to a 3×4 2D array and print

Broadcasting exercise:

- Generate random 3×4 2D array
- Add 1D array of shape (4,) to it and print result

Exercise 4: Analysing Data with NumPy

Use Built-in Functions to Explore Arrays

Create 1D array:
- 100 random numbers from 0–50 using np.random.randint()

Print statistics:
- Minimum, maximum, mean, median and standard deviation

Create 2D array: shape (5, 4)
- Compute total sum
- Find mean of each column

Challenge Exercise: Real-World Data Analysis

Apply NumPy Stats on a Sample Dataset

Create a 2D NumPy array

- Represent a dataset with 5 rows and 3 columns
- Row 1: [25, 175, 70]
- Row 2: [30, 180, 80]
- Row 3: [22, 165, 65]
- Row 4: [28, 172, 75]
- Row 5: [35, 178, 85]

Compute statistics

- For each feature (Age, Height, Weight)
- Mean
- Standard deviation
- Minimum and maximum values

Find correlations

- Between age, height, and weight using np.corrcoef()

Pandas Exercise

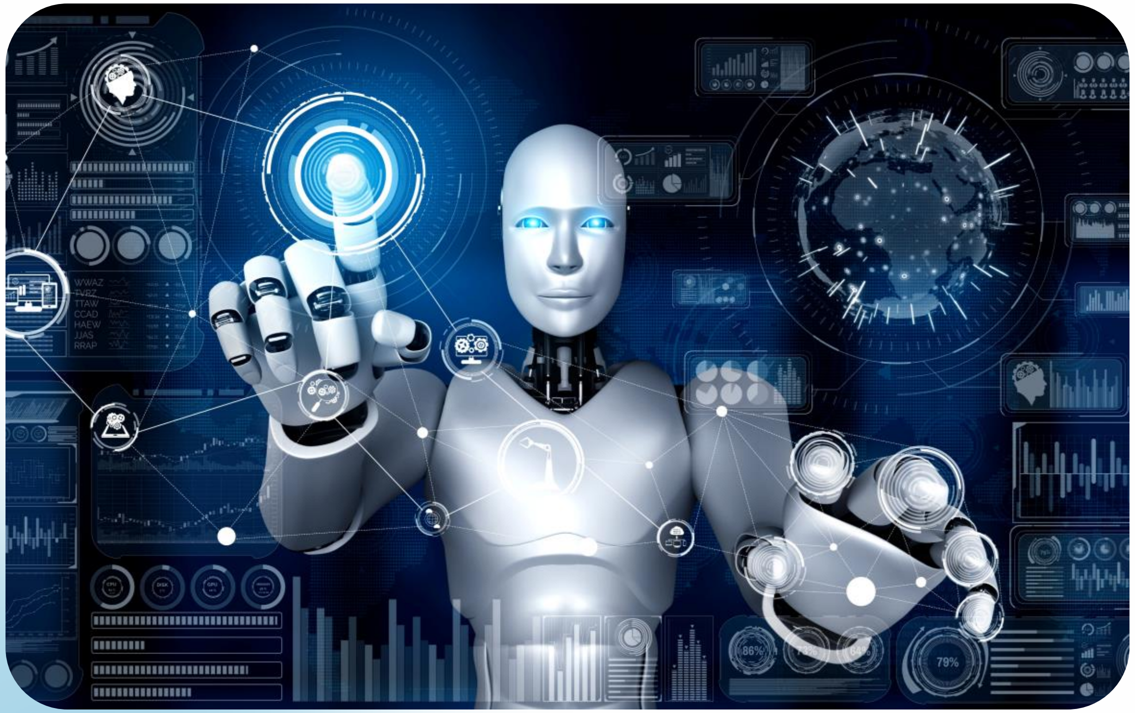
Pandas: Data Handling Essentials

Learn Core Operations for Effective Data Manipulation

- Reading files (CSV)
- Accessing columns and rows
- Filtering data based on conditions
- Using describe and info functions
- Adding a new column
- Handling missing values
- Dropping columns
- Combining NumPy functions for data manipulation

Exercise 1: Loading Data

Steps to Begin Working with a Dataset



Load CSV

- Load a dataset from a CSV file (sales_data.csv) into a Pandas DataFrame

Use info()

- Use the info() method to get a summary of the DataFrame (check the number of rows, columns, and data types)

Use describe()

- Use the describe() method to get a summary of the numeric columns

Exercise 2: Accessing and Filtering Data

Explore Rows and Columns in a DataFrame Efficiently

Access first 5 rows

- Access the first 5 rows of the dataset using the head() method

Access specific column

- Access a specific column (Product or Price) and display its unique values

Access rows by condition

- Access rows where the Sales column is greater than 500

Exercise 3: Filtering Data

Use Logical Filters to Extract Targeted Rows

Filter by region

- Filter the data to get all rows where the Region is "East"

Filter by price and sales

- Filter the dataset to show rows where the Price is greater than 100 and Sales is less than 1000

Exercise 4: Handling Missing Values

Detect, Fill and Drop Null Values Efficiently

Check for missing values

- Check if there are any missing values in the dataset using isnull() and sum()

Fill missing values

- Fill missing values in the Sales column with the median of the column

Drop rows with missing values

- Drop any rows that have missing values in the Product column

Exercise 5: Adding New Column

Create and Verify Derived Columns in Pandas

Add Discounted_Price column

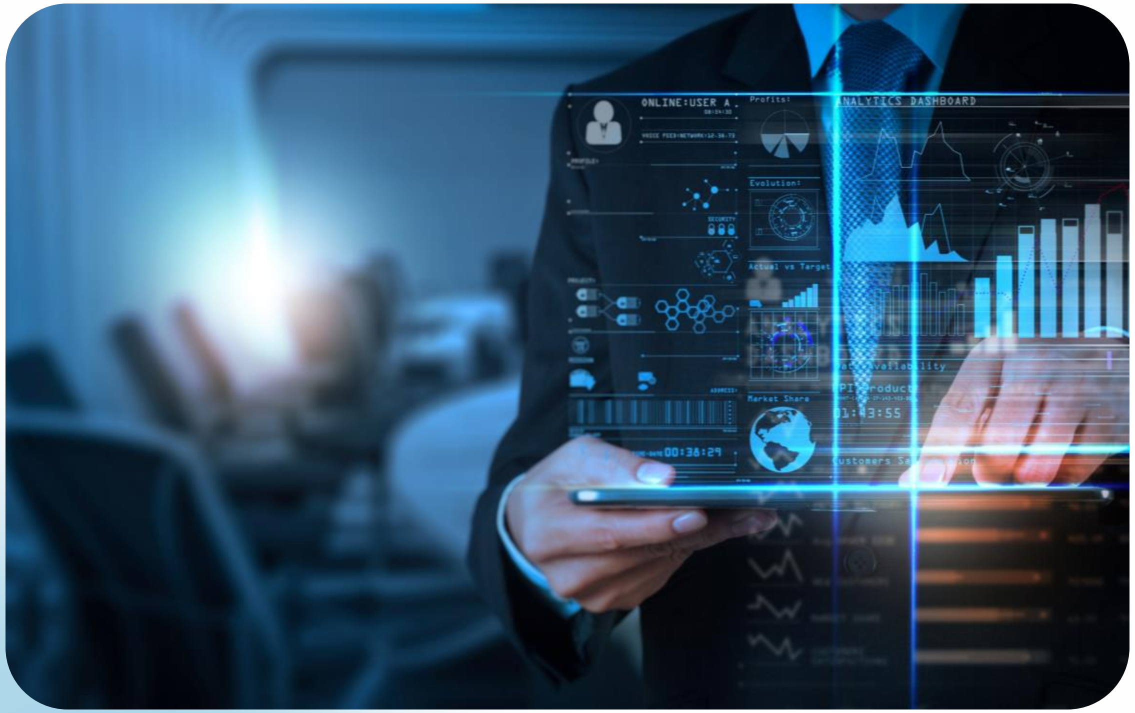
- Add a Discounted_Price column with 10% off the original price

Display results

- Show the first 5 rows to confirm the column was added

Exercise 6: Dropping Columns

Removing Unwanted Columns



Drop column

- Drop the discounted_price column from the DataFrame

Confirm deletion

- Confirm that the column has been dropped by printing the column names using columns

Exercise 7: Combining NumPy Functions

Using NumPy for Column Calculations

Create profit column

- Create a new column Profit by Subtracting the cost from the sales

Create log_profit column

- Use np.log() to create a new column Log_Profit that stores the logarithm of the Profit

Q & A

Thank you