Introduction to NumPy & Pandas for Data Analysis

Module 1.3

Sunit Bhattacharya

July 2025

Introduction to NumPy

- **NumPy** = Numerical Python — a library for high-performance numerical operations.
- Key features:
  - Multidimensional arrays (`ndarray`)
  - Fast vectorized operations
  - Linear algebra, random number generation, FFT, etc.
- Why NumPy?
  - Faster than Python lists for large-scale numerical tasks
  - Basis for many data science libraries (Pandas, SciPy, scikit-learn)

Creating and Using NumPy Arrays

```
import numpy as np

# 1D array
arr1 = np.array([1, 2, 3, 4])
print(arr1)

# 2D array
arr2 = np.array([[1, 2], [3, 4]])
print(arr2)

# Basic operations
print(arr1 + 10)
print(arr1 * 2)
```

Basic NumPy Operations

```
a = np.array([1, 2, 3, 4, 5])
print(np.mean(a))  # Average
print(np.std(a))   # Standard deviation
print(np.sum(a))   # Sum
print(a.shape)     # Shape of array
```

- Vectorized: operations apply to all elements without explicit loops.
- Efficient memory and speed.

Introduction to Pandas

- **Pandas** = Data analysis and manipulation library.
- Core data structures:
  - **Series**: 1D labeled array.
  - **DataFrame**: 2D labeled table with columns of potentially different types.
- Built on top of NumPy.
- Common tasks:
  - Load data from CSV, Excel, SQL
  - Clean and filter data
  - Summarize and analyze

Creating Series and DataFrames

```
import pandas as pd

# Series
s = pd.Series([10, 20, 30], index=['a', 'b', 'c'])
print(s)

# DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Grade': [85, 90, 78]}
df = pd.DataFrame(data)
print(df)
```

Basic DataFrame Operations

```
# Selection
print(df['Name'])
print(df[['Name', 'Grade']])

# Filtering
print(df[df['Grade'] > 80])

# Sorting
print(df.sort_values(by='Grade', ascending=False))
```

Loading Data from CSV and Excel

```
# Load CSV
grades_df = pd.read_csv("grades.csv")

# Load Excel
econ_df = pd.read_excel("economics.xlsx")

# Inspect data
print(grades_df.head())
print(econ_df.info())
```

Exploring Data

```
# Summary statistics
print(grades_df.describe())

# Column selection
print(grades_df['Math'])

# Unique values
print(grades_df['Grade'].unique())

# Value counts
print(grades_df['Grade'].value_counts())
```

Toy Problem: Predicting Student Dropouts

- **Scenario:** University wants to identify students who might drop out.
- **Features available:**
  - GPA
  - Major
  - Number of clubs joined
  - Attendance percentage
- We will create a **synthetic dataset** for exploration.

Step 1: Defining the Variables

- **GPA:** Normal distribution (mean ≈ 3.0, std ≈ 0.5).
- **Major:** Randomly chosen from:
  - Computer Science, Math, Economics, History
- **Clubs:** Poisson distribution (average $= 2$ clubs).
- **Attendance:** Uniform distribution between 50% and 100%.

Step 2: Generating the Data

```
import numpy as np
import pandas as pd

np.random.seed(42)
n = 200

gpa = np.round(np.random.normal(3.0, 0.5, n), 2)
majors = np.random.choice(
    ['CS', 'Math', 'Economics', 'History'], n
)
clubs = np.random.poisson(2, n)
attendance = np.random.uniform(50, 100, n)
```

Step 3: Simulating Dropouts

```
# Dropout rule: low GPA & low attendance
dropout_prob = (gpa < 2.5) & (attendance < 70)
dropout = np.where(dropout_prob, 1, 0)

df = pd.DataFrame({
    'GPA': gpa,
    'Major': majors,
    'Clubs': clubs,
    'Attendance': np.round(attendance, 1),
    'Dropout': dropout
})

print(df.head())
```

Step 4: First Look at the Data

```
# Basic summary
print(df.describe())

# Dropout counts
print(df['Dropout'].value_counts())
```

- `describe()` shows mean, min, max, etc.
- `value_counts()` counts how many dropped out.

Step 5: Grouping and Filtering

```
# Average GPA by major
df.groupby('Major')['GPA'].mean()

# High-risk students
high_risk = df[
    (df['GPA'] < 2.5) & (df['Attendance'] < 70)
]
print(high_risk.head())
```

Visualizing GPA vs Attendance

```
import matplotlib.pyplot as plt

plt.figure(figsize=(6,4))

colors = df['Dropout'].map({0: 'green', 1: 'red'})

plt.scatter(df['GPA'], df['Attendance'],
            c=colors, alpha=0.6)

plt.xlabel("GPA")
plt.ylabel("Attendance (%)")
plt.title("Student Dropout Risk")
plt.show()
```

Understanding the Plot

- **Green points** = Students who stayed.
- **Red points** = Students who dropped out.
- Dropouts cluster in the **low GPA + low attendance** region.
- Higher GPA and higher attendance $\rightarrow$ much lower dropout risk.

Step 6: Next Steps

- Check dropout rate per major.
- Compare summary stats for dropouts vs non-dropouts.
- In future sessions: build a simple model.

Practical Exercises

1. Install and configure Python environment.
2. Practice basic Python syntax and data structures.
3. Load and explore sample datasets:
   - Student grades
   - Simple economic indicators
4. Use Pandas to:
   - Calculate mean, median, min, max
   - Filter data based on conditions
   - Sort and select relevant columns

Summary

- **NumPy** for numerical operations — fast and vectorized.
- **Pandas** for tabular data — flexible, labeled structures.
- Learned:
  - Creating arrays, Series, and DataFrames.
  - Loading data from CSV/Excel.
  - Basic selection, filtering, and sorting.
  - Summary statistics and exploration.