DATA SCIENCE

DL-2001

Lab 02

Python Programming

National University of Computer & Emerging Sciences – NUCES – Karachi

National University of Computer & Emerging Sciences - NUCES - Karachi

FAST School of Computing

Course Code: DL-2001	Data Science Lab

Objectives

1. Understand and implement Python's core data structures.
2. Perform Create, Read, Update, and Delete operations.
3. Apply iteration and sorting methods on collections.
4. Handle files for reading and writing data.
5. Load, preview, and explore datasets from CSV, JSON, and Excel files.

List Tuple Set and dictionary

Feature	List	Tuple	Set	Dictionary
Order	Ordered	Ordered	Unordered (no indexing)	Ordered (Python 3.7+)
Mutability	Mutable (can be changed)	Immutable (cannot be changed)	Mutable (can add/remove elements)	Mutable (can add/remove key-value pairs)
Duplicates	Allowed	Allowed	Not Allowed	Keys: Not Allowed, Values: Allowed
Indexing	Supported	Supported	Not supported	By keys (not by position)
Data Types	Mixed data types allowed	Mixed data types allowed	Mixed data types allowed	Keys: immutable, Values: any type
Use Case	Dynamic collections, e.g., student list	Fixed data, e.g., coordinates	Unique collections, e.g., IDs	Key-value mapping, e.g., roll no → name
Syntax Example	fruits = ["Apple", "Banana"]	point = (10, 20)	ids = {101, 102, 103}	student = {1: "Ali", 2: "Sara"}

List example

```
fruits = ["Apple", "Banana", "Mango", "Orange"]
fruits.append("Grapes")          # Create (Add new item)
print(fruits[1])                 # Read (Access element at index 1)
fruits[0] = "Pineapple"          # Update (Change value at index 0)
fruits.remove("Mango")           # Delete (Remove item)
   # Iteration
for fruit in fruits:
    print(fruit)
# Sorting
print(sorted(fruits))            # Ascending order
print(sorted(fruits, reverse=True))  # Descending order
```

tuple example

```python
# Creating a tuple
coordinates = (10, 20, 30)

# Accessing elements
print(coordinates[0])        # First element
print(coordinates[1:])       # Slicing

# Iteration
for value in coordinates:
    print(value)

# Tuple with different data types
student = ("Ali", 21, "BSCS")
print(student)

```

Sets example

```python
# Creating sets
set1 = {"Ali", "Sara", "Ahmed"}
set2 = {"Ahmed", "Fatima", "Usman"}


set1.add("Ayesha")            # Add element
set1.remove("Sara")           # Remove element

# Iteration
for student in set1:
    print(student)

# Set Operations
print(set1.union(set2))       # Union
print(set1.intersection(set2))# Intersection
print(set1.difference(set2))  # Difference

```

Dict example

```python
student = {"name": "Ali", "age": 21, "course": "AI"}
print("Dictionary:", student)

# CRUD Operations
student["grade"] = "A"             # Create
print("After Create:", student)
print("Read course:", student["course"])  # Read
student["age"] = 22                # Update
print("After Update:", student)
del student["course"]              # Delete
print("After Delete:", student)

# Iteration
print("Keys:", list(student.keys()))
print("Values:", list(student.values()))
print("Items:")
for k, v in student.items():
    print(k, ":", v)

# Dictionary Methods
print("Safe Access:", student.get("name"))
print("With Default:", student.get("marks", 0))
```

```
print("Pop Grade:", student.pop("grade"))
print("After Pop:", student)

# Nested Dictionary
students = {
    1: {"name": "Ali", "age": 20, "marks": 85},
    2: {"name": "Sara", "age": 21, "marks": 90},
    3: {"name": "Ahmed", "age": 22, "marks": 78}
}
print("Nested Dict (Student 1 Name):", students[1]["name"])
print("Nested Dict (Student 2 Marks):", students[2]["marks"])
```

Files and CRUD operations

```
from pathlib import Path
import os

# Choose a working file
fname = Path("students.txt")

# --------- CREATE (write new / overwrite) ----------
print("\n[CREATE]")
with open(fname, "w", encoding="utf-8") as f:
    f.write("Ali\nSara\nAhmed\n")
print(f"Created {fname} with 3 lines.")

# --------- READ (full + line-by-line) --------------
print("\n[READ - full]")
with open(fname, "r", encoding="utf-8") as f:
    content = f.read()
print(content, end="")

print("\n[READ - line by line]")
with open(fname, "r", encoding="utf-8") as f:
    for i, line in enumerate(f, start=1):
        print(f"Line {i}: {line.strip()}")

# --------- UPDATE (append) -------------------------
print("\n[UPDATE - append]")
with open(fname, "a", encoding="utf-8") as f:
    f.write("Fatima\nUsman\n")
print("Appended Fatima and Usman.")

print("[READ after append]")
with open(fname, "r", encoding="utf-8") as f:
    print(f.read(), end="")

# --------- UPDATE (modify specific content) --------
# Example: Replace 'Sara' -> 'Zara'
print("\n[UPDATE - modify specific line: Sara -> Zara]")
with open(fname, "r", encoding="utf-8") as f:
    lines = f.readlines()

with open(fname, "w", encoding="utf-8") as f:
    for line in lines:
        name = line.strip()
        if name == "Sara":
            f.write("Zara\n")
        else:
            f.write(line)
print("Replaced 'Sara' with 'Zara'.")

```

```
print("[READ after modify]")
with open(fname, "r", encoding="utf-8") as f:
    print(f.read(), end="")

# --------- SAFE DELETE -----------------------------
print("\n[DELETE]")
if fname.exists():
    os.remove(fname)
    print(f"Deleted {fname}.")
else:
    print(f"{fname} not found; nothing to delete.")

# --------- VERIFY DELETE ---------------------------
print("\n[VERIFY]")
print("Exists after delete? ->", fname.exists())

```

File read write on datasets of different types

```
import pands ad pd
print("\n[CSV DATA]")
try:
    df_csv = pd.read_csv("data.csv")
    print("CSV Shape:", df_csv.shape)
    print("First 5 rows:\n", df_csv.head())
except FileNotFoundError:
    print("CSV file not found. Place 'data.csv' in the working directory.")

# --------- Load JSON ----------
print("\n[JSON DATA]")
try:
    df_json = pd.read_json("data.json")
    print("JSON Shape:", df_json.shape)
    print("First 5 rows:\n", df_json.head())
except FileNotFoundError:
    print("JSON file not found. Place 'data.json' in the working directory.")

# --------- Load Excel ----------
print("\n[EXCEL DATA]")
try:
    df_excel = pd.read_excel("data.xlsx")
    print("Excel Shape:", df_excel.shape)
    print("First 5 rows:\n", df_excel.head())
except FileNotFoundError:
    print("Excel file not found. Place 'data.xlsx' in the working directory.")

# --------- Common Data Preview Commands ----------
# If any file is loaded, show quick summary
if 'df_csv' in locals():
    print("\nCSV Columns:", df_csv.columns.tolist())
    print("Missing Values:\n", df_csv.isnull().sum())

if 'df_json' in locals():
    print("\nJSON Columns:", df_json.columns.tolist())
    print("Missing Values:\n", df_json.isnull().sum())

if 'df_excel' in locals():
    print("\nExcel Columns:", df_excel.columns.tolist())
    print("Missing Values:\n", df_excel.isnull().sum())

```

Lab tasks

Q1 : Create a list of 10 student names and perform the following:

1. Add two more names using append() and insert().
2. Update the 3rd element with a new name.
3. Remove one student using remove() and another using pop().
4. Access elements using positive and negative indexing.
5. Slice the first 5 students and display them.

Q2 :Create a list of numbers and demonstrate the use of the following methods:

1. len(), min(), max(), and sum().
2. sort() and reverse().
3. count() and index() for repeated values.
4. Convert the list to a set to remove duplicates.

Q3. Create a tuple with mixed data types (string, int, float).

1. Access elements by index and slicing ([:3], [-2:]).
2. Demonstrate count() and index() methods.
3. Access values from a nested tuple.
4. Try to update an element and explain what happens.

Q4. Given two sets:

- A = {"AI", "ML", "DS"}
- B = {"ML", "NLP", "CV"}

Perform the following:

1. Find Union, Intersection, Difference, and Symmetric Difference.
2. Add a new subject to a set.
3. Remove an element using remove() and discard() and explain the difference.
4. Check if "AI" exists in the set using in.

Q5. Create a dictionary of 5 students with roll numbers as keys and names as values.
Perform the following:

1. Add a new student.
2. Access a student name using roll number.
3. Update an existing student's name.
4. Delete one student record.
5. Iterate through the dictionary using .keys(), .values(), and .items().

Q6. Demonstrate the use of get() with and without default values.

1. Use pop() and popitem() on a dictionary.
2. Add multiple student records using update().
3. Create a nested dictionary storing roll no, name, and marks for each student.
4. Use dictionary comprehension to generate {x: x**2} for numbers 1–5.

Q7.

1. Create a list of tuples: [(1, "Ali"), (3, "Sara"), (2, "Ahmed")].
    - Sort the list by roll number.
    - Sort the list by name using key=lambda.
2. Create a dictionary of {student: marks} and sort the dictionary by marks.

Q8.

1. Create a file records.txt and write names of 5 students into it.
2. Read and display the file content.
3. Append 2 more student names to the file.
4. Replace one student's name with another (update operation).
5. Delete the file after completing the operations.

Q9.

1. Load a dataset from data.csv, data.json, and data.xlsx using pandas.
2. Display:
    - Shape of the dataset.
    - Column names and data types (dtypes).
    - First 10 rows (head(10)).
    - Last 5 rows (tail()).
    - Missing values per column (isnull().sum()).

Q10.

Develop a Student Record Manager that can:

1. Store student records (roll no, name, marks) in a dictionary.
2. Save records into a text file.
3. Read and display records from the file.
4. Add, update, and delete records (CRUD operations).
5. Export records into a CSV file and reload them using pandas.