University Of Loralai

Name:	Sajad ullah .
Semester	3nd semester .
Dept	Computer Science .
Course	Data Engineering .
Course instructor	Sir.Hazrat_Bilal .
Lecture no	3rd Lecture.

1. Lists in Python

◆ What is a List?

A **list** is a collection of items that is:

- **Ordered**
- **Mutable** (changeable)
- Allows **duplicate values**
- Written using **square brackets** **[ ]**

◆ Example:

```
# Creating a list
lst = [1, 2, 3, 4]

print(lst)
```

◆ Accessing Elements:

```
print(lst[0])    # First element
print(lst[-1])   # Last element
```

◆ Modifying a List (Mutable):

```
lst[1] = 99
print(lst)
```

◆ Common List Methods:

```
lst.append(5)         # Add element at end
lst.insert(1, 10)     # Insert at index
lst.remove(3)         # Remove value
lst.pop()             # Remove last element
lst.sort()            # Sort list
lst.reverse()        # Reverse list

print(lst)
```

✅ **Use lists when data needs to change**.

---

2. Tuples in Python

◆ What is a Tuple?

A **tuple** is a collection of items that is:

- **Ordered**
- **Immutable** (cannot be changed)
- Allows **duplicate values**
- Written using **round brackets `()`**

◆ Example:

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

print(tup)
```

◆ Accessing Elements:

```
print(tup[0])
print(tup[-1])
```

◆ Immutability (Cannot Change):

```
# This will cause an error
# tup[0] = 100
```

◆ **Tuple with One Element:**

```
single = (5,)  # comma is important
print(single)
```

✓ **Use tuples when data should not change** (e.g., fixed records).

---

3. Dictionaries in Python

◆ **What is a Dictionary?**

A **dictionary** stores data in **key : value** pairs.
It is:

- **Unordered** (in concept)
- **Mutable**
- Keys must be **unique**
- Written using **curly braces { }**

◆ **Example:**

```
# Creating a dictionary
student = {
    "name": "Sajad Ullah",
    "father_name": "Kamal Khan",
    "field": "Computer Science"
}

print(student)
```

◆ **Accessing Values:**

```
print(student["name"])
print(student.get("field"))
```

◆ **Modifying Dictionary:**

```
student["field"] = "Data Engineering"
student["age"] = 20

print(student)
```

◆ **Removing Items:**

```
student.pop("age")
print(student)
```

◆ **Looping Through Dictionary:**

```
for key, value in student.items():
    print(key, ":", value)
```

✅ **Use dictionaries when data has meaning (key-value relationship).**

🔁 Summary Table

Feature	List	Tuple	Dictionary
Definition	Collection of elements	Collection of elements	Collection of key-value pairs
Syntax	`[ ]`	`( )`	`{ key : value }`
Example	`[1, 2, 3]`	`(1, 2, 3)`	`{"a":1, "b":2}`
Ordered	✅ Yes	✅ Yes	✅ Yes

Feature	List	Tuple	Dictionary
Mutable (Changeable)	✅ Yes	❌ No	✅ Yes
Duplicate Values	✅ Allowed	✅ Allowed	❌ Keys not allowed
Indexing	✅ Yes	✅ Yes	❌ No (keys used)
Key-Value Support	❌ No	❌ No	✅ Yes
Access Method	`lst[0]`	`tup[0]`	`d["key"]`
Add Item	`append()`	❌ Not allowed	`d[key]=value`
Remove Item	`remove()`, `pop()`	❌ Not allowed	`pop(key)`
Use Case	Dynamic data	Fixed data	Structured data
Memory Efficient	❌ Less	✅ More	❌ Less
Real-Life Example	Shopping list	Coordinates	Student record

✅ Final Tip for You (as a CS student & future Data Engineer):

- Use **lists** for dynamic data
- Use **tuples** for fixed data
- Use **dictionaries** for structured data (very important in Data Engineering)