**Minor in AI**

**Master Session Notes**

Batch 4

3 Feb 2025

**Title: Python's Fantastic Four: Unlocking the Power of Lists, Tuples, Sets, and Dictionaries**

**Concepts Covered:**

Overview and characteristics of each data structure: List, Tuple, Dictionary, Set  
Operations and methods for:  
- Lists: Indexing, len(), slicing, comprehension  
- Tuples: Immutability, Indexing, len(), slicing  
- Dictionary: Key-value pairs, methods - get(), items(), keys(), values()  
- Sets: Union, intersection, difference, len()

Point to ponder on:

What do you understand by

“Tail Wags the Dog”

**Overview**

When we have a collection of data, say integers, we work on them with several purposes. We might want them ordered or not, we might want to change them or not, we might want duplicates or not. Based on the purpose and on set python allows us to operate on this collection using lists, tuples, dictionaries and sets.

**Summary Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Structure** | **Ordered?** | **Mutable?** | **Allow Duplicates?** |
| List | Yes |  |  |
| Tuple |  | No |  |
| Dictionary | Yes |  | No |
| Set |  | Yes |  |

Can you all fill up the remaining ones?

**Lists and Operations**

# Demonstrating Lists and operations

# Creating a list

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Indexing

print("First element is:", numbers[0])

print("Fifth element is:", numbers[4])

print("Last element is:", numbers[-1])

print("Last but one element is:", numbers[-2])

# Length of list

print("Length of list is:", len(numbers))

# Slicing

print("Slice first 5 elements:", numbers[:5])

print("Slice last 3 elements:", numbers[-3:])

print("Elements from index 2 to 4:", numbers[2:5])

print("Get every second element:", numbers[::2])

print("Reverse the list:", numbers[::-1])

# List Comprehension

added\_numbers = [x+1 for x in numbers]

print("Doing + 1 to each member:", added\_numbers)

odd\_numbers = [x for x in numbers if x % 2 != 0]

print("Odd numbers in list:", odd\_numbers)

We can change the items from the list.

days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]

print(days)

days[6] = "Fun"

print(days)

For below case, we get an error. Why?

days = ("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")

print(days)

days[6] = "Fun"

print(days)

**Tuple and Operations:**

days = ("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")

print(days)

print("length of days is", len(days))

print("Weekdays are:", days[:5])

print("Weekends are:", days[-2:])

print("Reversing:", days[::-1])

**Dictionary and Operations**

# Demonstrating Dictionary and operations

# Creating a dictionary - associative data

phone= {

    "name": "Samsung M31",

    "price": 9000,

    "Battery": 6000,

    "CPU": "Octa-core"

}

# Accessing values using get()

print("Mobile name:", phone.get("name"))

print("CPU:", phone.get("CPU"))

# Getting all key-value pairs using items()

print("\nDictionary Items:")

for key, value in phone.items():

    print(key, "->", value)

# Getting all keys using keys()

print("All Keys:", phone.keys())

# Getting all values using values()

print("All Values:", phone.values())

Can you add one more entry to above code? How would you do it?

**Set and Operations**

# Creating two sets

setA = {1, 2, 3}

setB = {3, 4, 5}

print(setA)

print(setB)

# Union: Combines elements from both sets

print("Union Operation")

union\_set = setA | setB

print(union\_set)

uset = setA.union(setB)

print(uset)

# Intersection: Common elements in both sets

intersection\_set = setA & setB

print("Intersection:", intersection\_set)

# Difference: Elements in set\_A but not in set\_B

difference\_set = setA - setB

print("Difference (A - B):", difference\_set)

# Length of a set

print("Size of set\_A:", len(setA))

print("Size of set\_B:", len(setB))