🐍 Session 7: Dictionaries, Sets & Slicing in Python

🎯 Topics Covered

1. **Dictionaries** (key-value pairs, methods)
2. **Sets** (operations, use cases)
3. **Slicing** (detailed examples)

🔑 1. Dictionaries in Python

What is a Dictionary?

- A dictionary stores data as **key-value pairs**.
- Keys are unique, immutable (string, number, tuple).
- Values can be any data type.

Syntax:

```
dictionary = {key: value}
```

Example:

```
student = {"name": "Bob", "grade": 92, "passed": True}
print(student["name"])       # Access by key
print(student.get("grade"))  # Safer way to access
```

Common Dictionary Methods

- `get(key)` → returns value for the key

- `keys()` → returns all keys
- `values()` → returns all values
- `items()` → returns key-value pairs
- `update()` → updates dictionary
- `pop(key)` → removes a key

Example:

```
student = {"name": "Alice", "age": 21}
print(student.keys())
print(student.values())

student.update({"grade": 88})
print(student)

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

🔑 2. Sets in Python

What is a Set?

- A **set** is a collection of **unique** elements.
- Unordered, mutable.
- Useful for removing duplicates & mathematical operations.

Syntax:

```
myset = {element1, element2, ...}
```

Example:

```
colors = {"red", "green", "blue"}
colors.add("yellow")
```

```
print(colors)
```

Set Operations

- `union()` → combines elements
- `intersection()` → common elements
- `difference()` → elements not in other set

Example:

```
A = {1, 2, 3, 4}
B = {3, 4, 5, 6}

print("Union:", A.union(B))
print("Intersection:", A.intersection(B))
print("Difference:", A.difference(B))
```

🔑 3. Slicing in Python

What is Slicing?

Slicing is a technique to **extract a part of a sequence** (list, tuple, or string).

Syntax:

```
sequence[start:end:step]
```

- `start` → index to begin (default = 0)
- `end` → index to stop (exclusive)
- `step` → interval/jump (default = 1)

Examples:

```
nums = [10, 20, 30, 40, 50, 60]
print(nums[1:4])   # [20, 30, 40]
print(nums[:3])    # [10, 20, 30]
```

```
print(nums[3:])    # [40, 50, 60]
print(nums[::2])   # [10, 30, 50]

# Negative indices & reverse
nums = [10, 20, 30, 40, 50, 60]
print(nums[-3:])   # [40, 50, 60]
print(nums[::-1])  # [60, 50, 40, 30, 20, 10]

# String slicing
text = "Python"
print(text[0:3])   # Pyt
print(text[::-1])  # nohtyP
```

🔗 4. Combining Loops + Dictionaries + Sets + Slicing

We can merge concepts together in problems.

Example 1 – Loop with Dictionary

```
marks = {"Alice": 85, "Bob": 92, "Charlie": 78}
for name, score in marks.items():
    if score > 80:
        print(name, "passed")
```

Example 2 – Loop with Sets

```
nums = [1, 2, 2, 3, 3, 4]
unique_nums = set(nums)
print(unique_nums)
```

Example 3 – Slicing + Dictionary + Set

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Slice even indices
sliced = nums[::2]

# Dictionary of number → square
num_dict = {n: n**2 for n in sliced}
```

```
# Unique squares using set
unique_squares = set(num_dict.values())

print("Sliced:", sliced)
print("Dictionary:", num_dict)
print("Unique Squares:", unique_squares)
```

📝 Assignments for You

1. Create a **student database** using a dictionary (name → marks).
    - Keep adding students until the user types `"stop"`.
    - Print all **unique marks** using a set.
    - Print top 3 students using slicing.
2. Write a program to count **word frequency** in a sentence using a dictionary.
3. Use a set to find **unique characters** in a string.
4. Reverse a string using slicing (no loop).