UNIT 9 — DICTIONARIES

What is a Dictionary?

A Dictionary in Python is an unordered collection of items. It is stored as a Key:Value pair. Think of it like a real-world dictionary where you look up a 'Word' (Key) to find its 'Meaning' (Value).

 Memory Trick: Dictionaries use Curly Braces { } like a bag of secrets!

Key Characteristics:

1. Mutable: You can change values after creation.
2. Indexed by Keys: Not by numbers like lists.
3. Keys must be Unique: No duplicates allowed for keys!

Syntax & Example

```
# Creating a dictionary
student = {
    "name": "Rahul",
    "age": 20,
    "course": "Python"
}
print(student)
# Output: {'name': 'Rahul', 'age': 20, 'course': 'Python'}
```

Real-World Example:

Think of a Contact List: Name -> Phone Number

Accessing & Modifying Data

1. Accessing Values:

Use the key inside square brackets [ ] or use .get()

```
# Accessing data
print(student["name"])  # Rahul
print(student.get("age")) # 20
```

2. Adding/Updating Items:

Just assign a value to a key. If key exists, it updates; else, it adds.

```
student["grade"] = "A"  # Adds new key
student["age"] = 21     # Updates existing key
```

~~ Common Mistake~~

Accessing a key that doesn't exist using [] will raise a KeyError!
Fix: Use .get() - it returns 'None' instead of crashing.

Mini Exercise:

Create a dict 'car' with brand, model, and year. Update the year to 2024.

Essential Dictionary Methods

- **keys()**: Returns a list of all keys in the dict.
- **values()**: Returns a list of all values.
- **items()**: Returns a list of (key, value) tuples.
- **pop(key)**: Removes the item with the specified key.
- **clear()**: Removes all items from the dictionary.
- **update(dict2)**: Merges another dictionary into the current one.

Method Demonstration

```
d = {"a": 1, "b": 2, "c": 3}

print(d.keys())   # dict_keys(['a', 'b', 'c'])
print(d.values()) # dict_values([1, 2, 3])
print(d.items())  # dict_items([('a', 1), ('b', 2), ('c', 3)])

d.pop("b")
print(d)          # {'a': 1, 'c': 3}

d.update({"d": 4})
print(d)          # {'a': 1, 'c': 3, 'd': 4}
```

Iterating Through Dictionaries

We can loop through keys, values, or both!

A) Loop through Keys:

```
for key in student:
    print(key)
```

B) Loop through Values:

```
for val in student.values():
    print(val)
```

C) Loop through Key-Value Pairs (Best Way!):

```
for k, v in student.items():
    print(f"{k} -> {v}")
```

~~How it works?~~

Dictionary.items() ---> Returns Tuples ---> Unpacked into k, v

[('name', 'Rahul'), ('age', 20)]

Nested Dictionaries

A dictionary can contain another dictionary. This is useful for complex data.

Example Structure

```
# Nested Dictionary Example
class_data = {
    "student1": {"name": "Aman", "marks": 85},
    "student2": {"name": "Priya", "marks": 92}
}

# Accessing Nested Data
print(class_data["student1"]["name"]) # Aman
```

Visual Representation:

```
class_data {
    student1: { name: Aman, marks: 85 }
    student2: { name: Priya, marks: 92 }
}
```

Pro Tip: Use Nested Dicts for JSON data in Web APIs!

Interview Qs & Practice

Interview Questions:

Q1. Difference between List and Dictionary?

Q2. Can a list be used as a key in a dictionary? (Ans: No, keys must be immutable!)

Q3. How do you merge two dictionaries in Python 3.9+? (Ans: Using | operator)

Practice Questions:

- 1. Write a program to sum all the values in a dictionary.
- 2. Create a dictionary from two lists using zip().
- 3. Count the frequency of each character in a string using a dict.

HAPPY LEARNING!

Keep Practicing, Stay Curious!