TOPS TECHNOLOGY

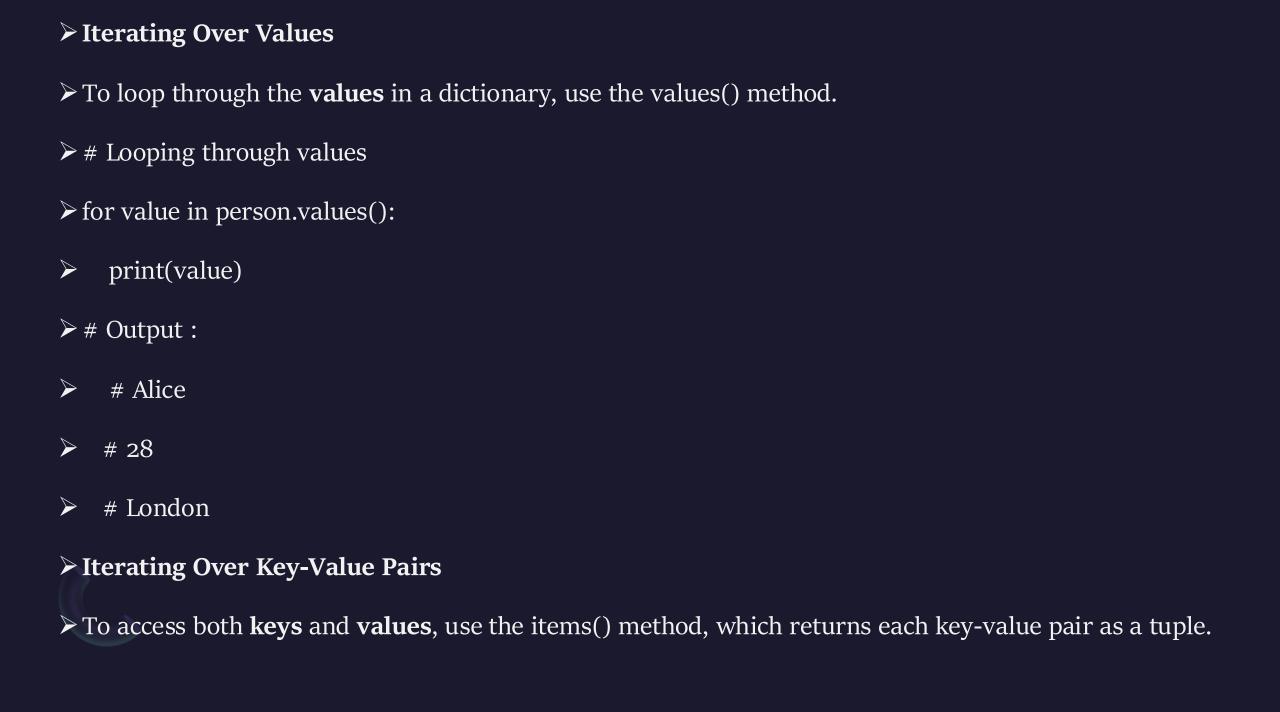


Presented By: Nandni Vala



Working with Dictionaries

- 1.Iterating over a dictionary using loops.
- ➤ Dictionaries store data as **key-value pairs**, and Python provides several ways to iterate over these pairs effectively. You can loop through **keys**, **values**, or both.
- > Iterating Over Keys :
- > By default, iterating over a dictionary loops through its keys.
- person = {"name": "Alice", "age": 28, "city": "London"}
- ➤ Looping through key:
- > for key in person:
- print(key)
- Output:
- > # name
- > # age
- > # city



Example:

- > # Looping through key-value pairs
- ➤ for key, value in person.items():
- print(f"{key}: {value}")
- ># Output:
- > # name: Alice
- > # age: 28
- ># city: London

- 2. Merging two lists into a dictionary using loops or zip().
- > Using a Loop:
- > Iterate through both lists simultaneously and construct the dictionary.
- keys = ["name", "age", "city"]
- ➤ values = ["Alice", 30, "New York"]
- # Create an empty dictionary
- merged_dict = {}
- # Add key-value pairs using a loop
- for i in range(len(keys)):
- merged_dict[keys[i]] = values[i]
- print(merged_dict)
- # Output: {'name': 'Alice', 'age': 30, 'city': 'New York'}

- > Using zip():
- ➤ The zip() function pairs elements from both lists into tuples, which can be directly converted into a dictionary.
- ▶ keys = ["name", "age", "city"]
- > values = ["Alice", 30, "New York"]
- > # Use zip() to merge into a dictionary
- merged_dict = dict(zip(keys, values))
- print(merged_dict)
- > # Output: {'name': 'Alice', 'age': 30, 'city': 'New York'}

3. Counting occurrences of characters in a string using dictionaries.

- > Explanation:
- ➤ **Initialization**: An empty dictionary char_count is created to store each character as a key and its count as the value.
- > Iteration: Loop through each character in the string.
- Updating the dictionary:
- > If the character is already a key in the dictionary, its value is incremented.
- If not, the character is added as a new key with the value set to 1.
- Output: The dictionary containing the character counts is returned.

```
Example:
def count_characters(string):
      char_count = {}
for char in string:
char_count[char] = char_count.get(char, 0) + 1
return char_count
print(count_characters("hello world"))
  Output:
{'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'w': 1, 'r': 1, 'd': 1}
```