**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Topic 33 - Looping Through Key-Value Pairs in Python Dictionaries**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**What**

Python allows you to iterate over both **keys** and **values** in a dictionary simultaneously using the .items() method. This method returns each key-value pair in the dictionary as a tuple, making it easy to work with both elements in a single loop.

**Why**

Looping through key-value pairs allows you to access both the label (key) and the data (value) at once. This technique is efficient for displaying or processing structured information, such as user profiles, product attributes, or configuration settings.

**How**

To loop through key-value pairs in a dictionary, use the .items() method within a for loop. Here’s an example:

python

Copy code

customer\_29876 = {

"first name": "David",

"last name": "Elliott",

"address": "4803 Wellesley St."

}

# Loop through and print each key-value pair

for each\_key, each\_value in customer\_29876.items():

print("The customer's " + each\_key + " is " + each\_value)

In this code:

* **Line 6**: The for loop iterates over each key-value pair in customer\_29876.
* **Line 7**: print() outputs each pair in a readable sentence format, resulting in:

python

Copy code

The customer's first name is David

The customer's last name is Elliott

The customer's address is 4803 Wellesley St.

Key points:

1. **Two Variables**: Use two variables (e.g., each\_key, each\_value) to represent keys and values.
2. **.items() Method**: This returns both the key and value in each pair.

**Summary**

Using .items() to loop through both keys and values in a dictionary is a powerful approach when you need to access or display a dictionary’s complete data. This method is straightforward and useful for handling structured data in Python.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**