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

**Topic 30 - Removing and Changing Items in Python Dictionaries**

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

**What**

Python dictionaries allow you to **remove** or **update** key-value pairs, providing flexibility in managing stored information. The del keyword removes items, and reassigning a key lets you change its value.

**Why**

Removing and updating items enables you to keep your data relevant and accurate. If a customer’s address changes, for example, you can update the "address" key without creating a new dictionary. Similarly, removing data that is no longer needed keeps the dictionary manageable and optimized.

**How**

To **remove a key-value pair** from a dictionary, use the del statement, followed by the dictionary name and key in square brackets [].

Example:

python

Copy code

# Dictionary with customer info

customer\_29876 = {

"first name": "David",

"last name": "Elliott",

"address": "4803 Wellesley St."

}

# Removing the address

del customer\_29876["address"]

To **change a value** for an existing key, simply reassign the key to the new value.

Example:

python

Copy code

customer\_29876["city"] = "Winnipeg"

In the code above, customer\_29876["city"] = "Winnipeg" changes the "city" key’s value.

**Summary**

Deleting or updating dictionary entries is easy and essential for managing data accurately. The del keyword removes items, and reassigning keys allows you to adjust values, keeping your dictionary up to date and efficient.

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