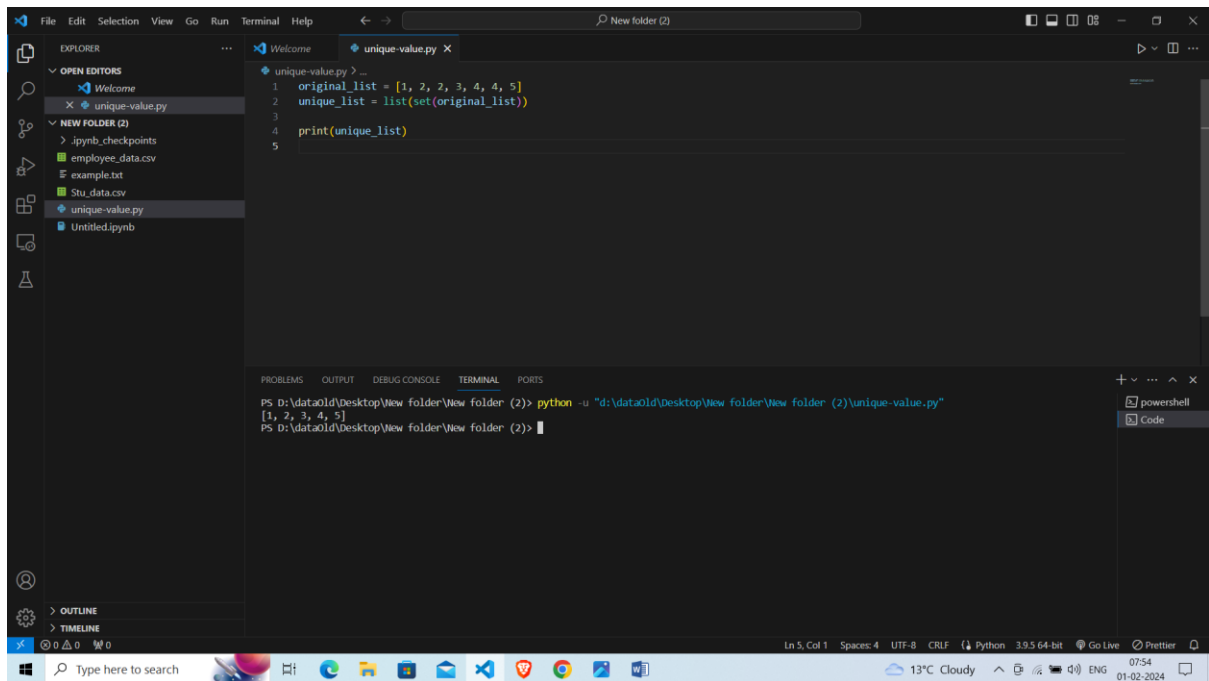


Assignment – 9

In Python, we can get unique values from a list using various approaches. Here are a few methods along with examples:

Using set():

The set() data type automatically eliminates duplicate values. You can convert a list to a set and then convert it back to a list.



The screenshot shows a Visual Studio Code editor window with a file named 'unique-value.py' open. The code in the file is as follows:

```
1 original_list = [1, 2, 2, 3, 4, 4, 5]
2 unique_list = list(set(original_list))
3
4 print(unique_list)
5
```

The Explorer sidebar on the left shows the project structure with files like 'unique-value.py' and 'Untitled.ipynb'. The Terminal at the bottom shows the command 'python -u "d:\dataold\Desktop\New folder\New folder (2)\unique-value.py"' being executed, with the output '[1, 2, 3, 4, 5]' displayed.

Using a loop:

You can iterate through the original list and build a new list with only unique values.

The screenshot shows the Visual Studio Code editor with a file named `unique-value.py` open. The code in the editor is as follows:

```
5  
6 original_list = [1, 2, 2, 3, 4, 4, 5]  
7 unique_list = []  
8  
9 for item in original_list:  
10     if item not in unique_list:  
11         unique_list.append(item)  
12  
13 print(unique_list)  
14  
15
```

The terminal at the bottom shows the command `python -u "d:\dataold\Desktop\New folder\New folder (2)\unique-value.py"` being executed, resulting in the output `[1, 2, 3, 4, 5]`.

Using List Comprehension:

You can use list comprehension to create a new list with unique values.

The screenshot shows the Visual Studio Code editor with a file named `unique-value.py` open. The code in the editor is as follows:

```
15  
16 original_list = [1, 2, 2, 3, 4, 4, 5]  
17 unique_list = [item for item in original_list if original_list.count(item) == 1]  
18  
19 print(unique_list)  
20  
21  
22
```

The terminal at the bottom shows the command `python -u "d:\dataold\Desktop\New folder\New folder (2)\unique-value.py"` being executed, resulting in the output `[1, 3, 5]`.

JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is a common format for representing

structured data and is widely used for data exchange between a server and a web application, among other purposes.

JSON Strings:

A JSON string is a sequence of characters representing a valid JSON value. JSON values can be objects, arrays, strings, numbers, booleans, or null. Strings in JSON are enclosed in double quotation marks.

Here's an example of a JSON string representing a person:

```
{  
  "name": "John Doe",  
  "age": 30,  
  "city": "New York",  
  "is_student": false,  
  "grades": [95, 88, 75]  
}
```

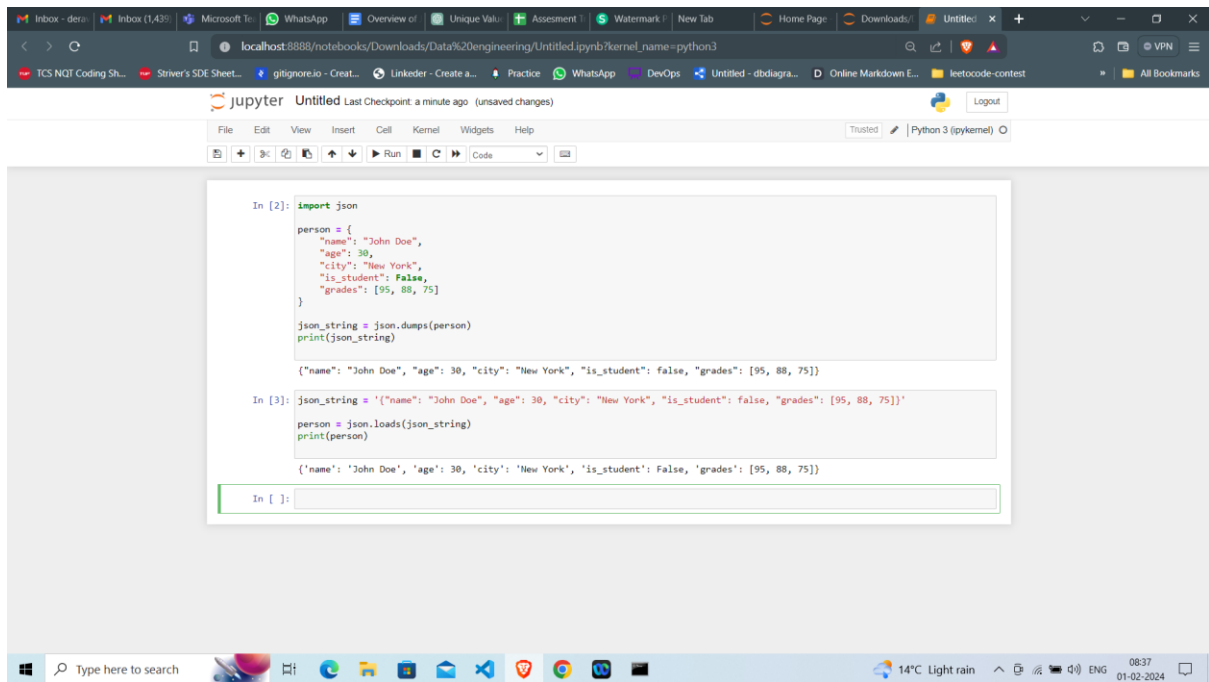
JSON Files in Python:

Python provides a built-in module called `json` for encoding and decoding JSON data. You can use this module to work with JSON strings and files.

Encoding (Python Object to JSON String):

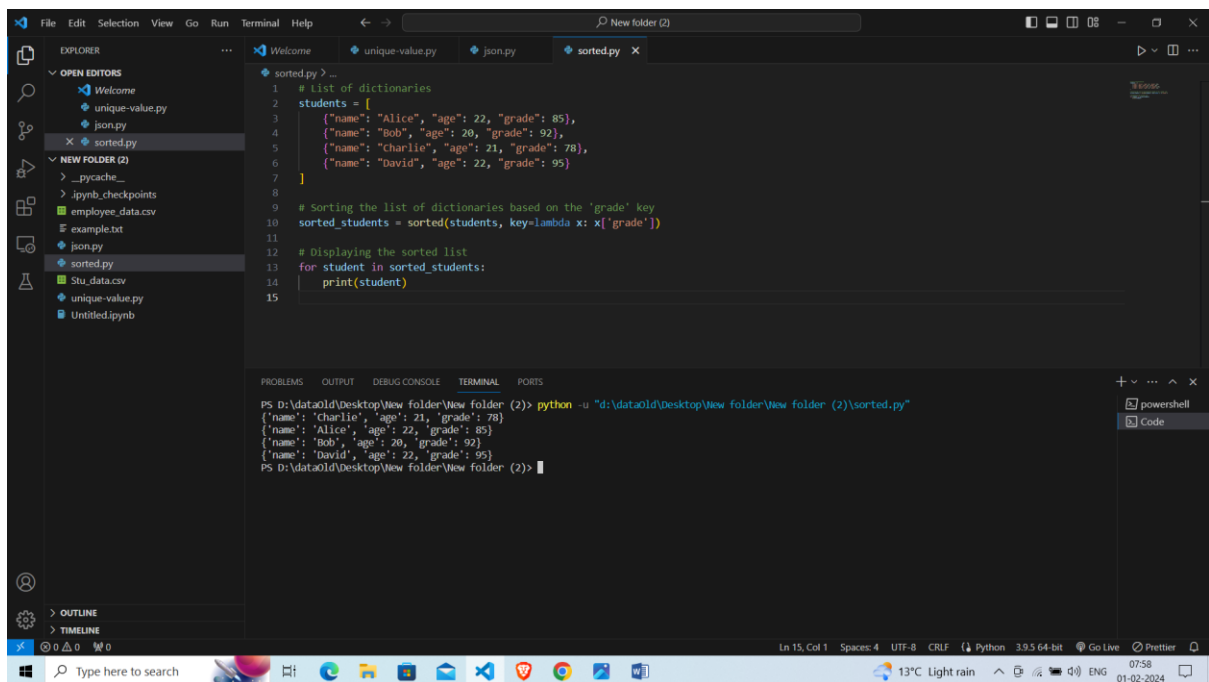
We can use `json.dumps()` to convert a Python object (such as a dictionary) into a JSON-formatted string.

We can use `json.loads()` to convert a JSON-formatted string into a Python object.



In Python, you can use the **sorted()** function to sort a list based on specified keys. The **sorted()** function allows you to provide a custom sorting key using the `key` parameter. The key is a function that takes an element from the **iterable** (list) and returns a value that will be used for sorting.

Here's an example to illustrate how to sort a list of dictionaries based on a specific key:



We can also use the `reverse` parameter to sort the list in descending order:

