

Name: Pradip Bochare

JSON

- JSON stands for JavaScript Object Notation. It is a format for structuring data. This format is used by different web applications to communicate with each other.
- JSON is the replacement of the XML data exchange format in JSON. It is easy to struct the data compare to XML. It supports data structures like arrays and objects and the JSON documents that are rapidly executed on the server.
- It is also a Language-Independent format that is derived from JavaScript. The official media type for the JSON is application/json and to save those files .json extension.

❖ Features of JSON:

- **Easy to understand:** JSON is easy to read and write.
- **Format:** It is a text-based interchange format. It can store any kind of data in an array of video, audio, and image anything that you required.
- **Support:** It is light-weighted and supported by almost every language and OS. It has a wide range of support for the browsers approx each browser supported by JSON.
- **Dependency:** It is an independent language that is text-based. It is much faster compared to other text-based structured data.

❖ JSON Syntax Rules:

Data is in name/value pairs and they are separated by commas. It uses curly brackets to hold the objects and square brackets to hold the arrays.

❖ Advantages of JSON:

- JSON stores all the data in an array so data transfer makes easier. That's why JSON is the best for sharing data of any size even audio, video, etc.
- Its syntax is very easy to use. Its syntax is very small and light-weighted that's the reason that it executes and response in a faster way.
- JSON has a wide range for the browser support compatibility with the operating systems, it doesn't require much effort to make it all browser compatible.

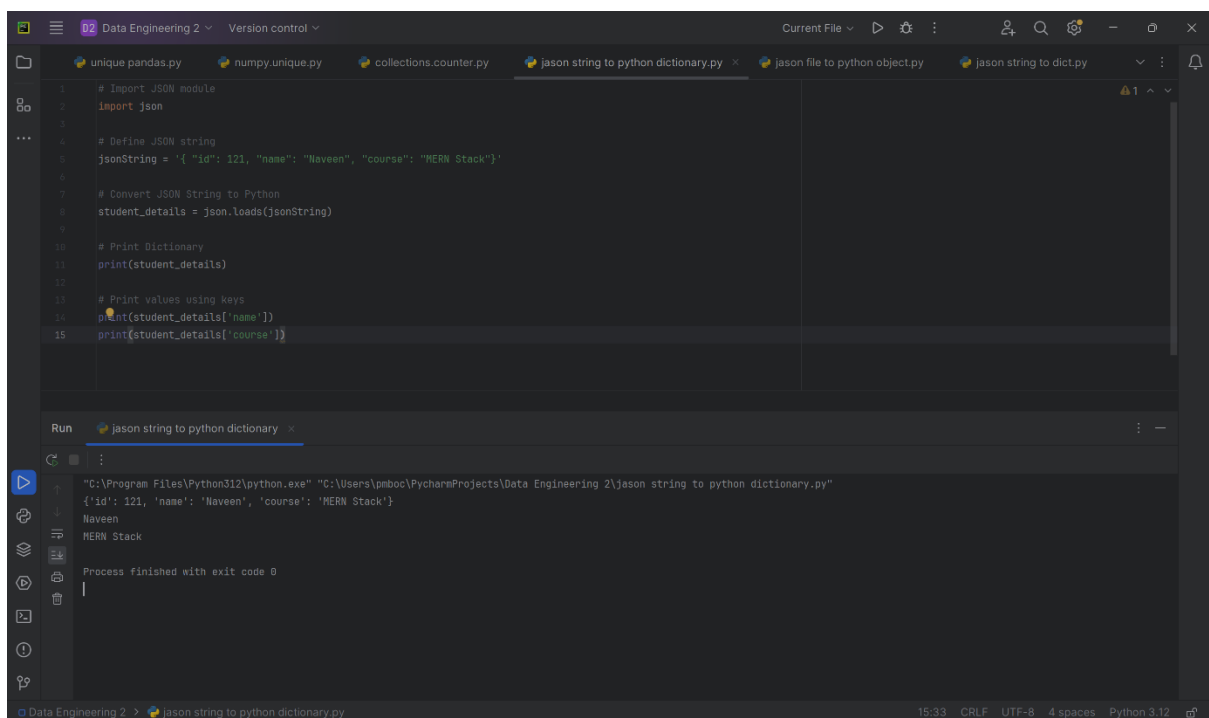
- On the server-side parsing the most important part that developers want, if the parsing will be fast on the server side then the user can get the fast response, so in this case JSON server-side parsing is the strong point compare tot others.

❖ Disadvantages of JSON:

- The main disadvantage for JSON is that there is no error handling in JSON, if there was a slight mistake in the JSON script then you will not get the structured data.
- JSON becomes quite dangerous when you used it with some unauthorized browsers. Like JSON service return a JSON file wrapped in a function call that has to be executed by the browsers if the browsers are unauthorized then your data can be hacked.
- JSON has limited supported tools that we can use during JSON development.

❖ Function Used

- **json.load():** json.load() function is present in Python built-in 'JSON' module. This function is used to parse the JSON string.
- **json. loads():** json.loads() function is present in Python built-in 'json' module. This function is used to parse the JSON string.



```
1 # Import JSON module
2 import json
3
4 # Define JSON string
5 jsonString = '{"id": 121, "name": "Naveen", "course": "MERN Stack"}'
6
7 # Convert JSON String to Python
8 student_details = json.loads(jsonString)
9
10 # Print Dictionary
11 print(student_details)
12
13 # Print values using keys
14 print(student_details['name'])
15 print(student_details['course'])
```

Run json string to python dictionary

```
"C:\Program Files\Python312\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering 2\json string to python dictionary.py"
{'id': 121, 'name': 'Naveen', 'course': 'MERN Stack'}
Naveen
MERN Stack
Process finished with exit code 0
```

The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named "Data Engineering 2" with several files. The active file is "json file to python object.py". The code in the editor is as follows:

```
1 import json
2
3 # Opening JSON file
4 json_file_path = r"C:\Users\paboc\Downloads\sample1.json"
5
6 with open(json_file_path, 'r') as json_file:
7     data = json.load(json_file)
8
9     # Print the type of data variable
10    print("Type:", type(data))
11
12    print(data)
```

The terminal output shows the execution of the script:

```
Run json file to python object
"C:\Program Files\Python312\python.exe" "C:\Users\paboc\PycharmProjects\Data Engineering 2\json file to python object.py"
Type: <class 'dict'>
{'fruit': 'Apple', 'size': 'Large', 'color': 'Red'}
Process finished with exit code 0
```

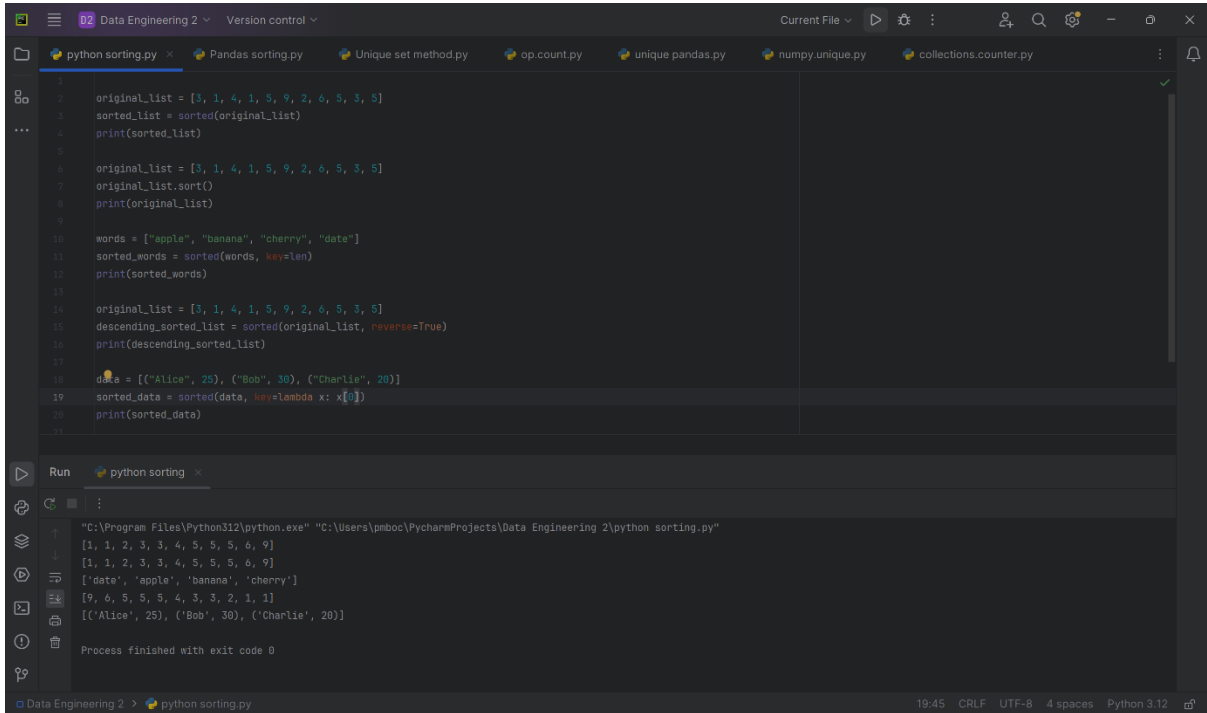
The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named "Data Engineering 2" with several files. The active file is "json string to dict.py". The code in the editor is as follows:

```
1 import json
2
3 # JSON string
4 json_string = '{"Name": "Suezen", "age": 23, "Course": "DSA"}'
5
6 # Convert JSON string to dictionary
7 json_dict = json.loads(json_string)
8
9 print(json_dict)
```

The terminal output shows the execution of the script:

```
Run json string to dict
"C:\Program Files\Python312\python.exe" "C:\Users\paboc\PycharmProjects\Data Engineering 2\json string to dict.py"
{'Name': 'Suezen', 'age': 23, 'Course': 'DSA'}
Process finished with exit code 0
```

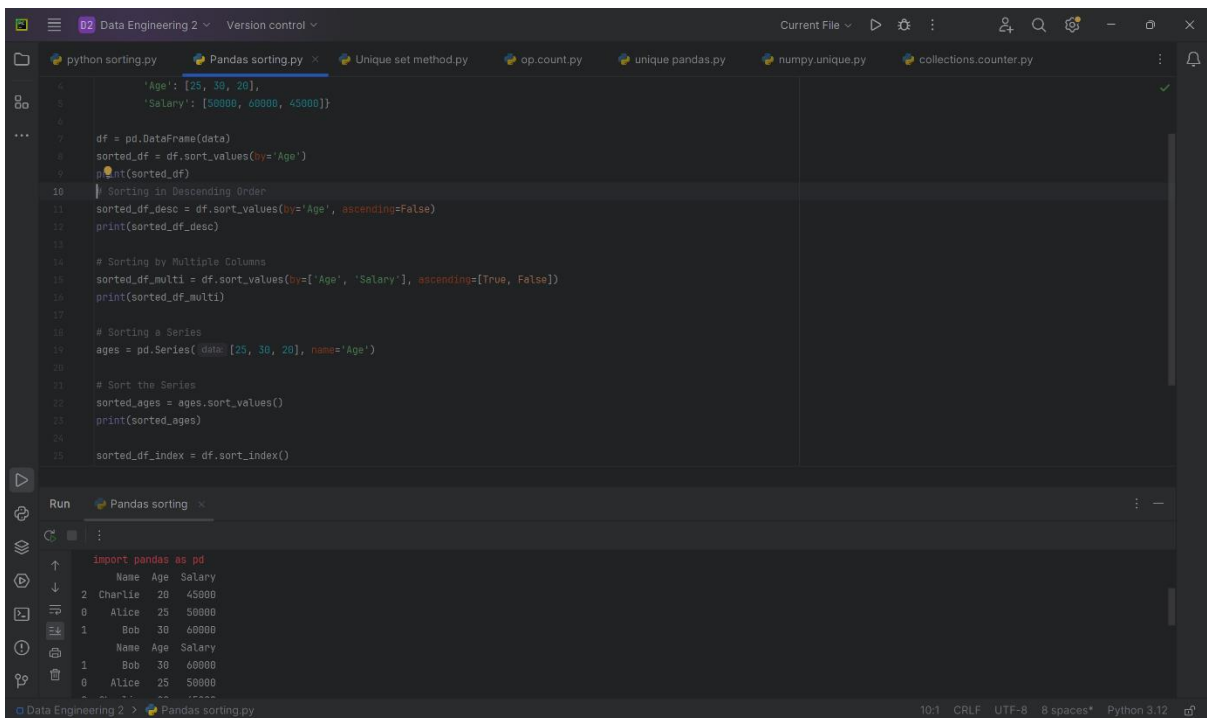
Sorting in python



```
1 original_list = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
2 sorted_list = sorted(original_list)
3 print(sorted_list)
4
5 original_list = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
6 original_list.sort()
7 print(original_list)
8
9 words = ["apple", "banana", "cherry", "date"]
10 sorted_words = sorted(words, key=len)
11 print(sorted_words)
12
13 original_list = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
14 descending_sorted_list = sorted(original_list, reverse=True)
15 print(descending_sorted_list)
16
17 data = [{"Alice": 25}, {"Bob": 30}, {"Charlie": 20}]
18 sorted_data = sorted(data, key=lambda x: x[0])
19 print(sorted_data)
20
```

Run python sorting

```
"C:\Program Files\Python312\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering 2\python sorting.py"
[1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
[1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
['date', 'apple', 'banana', 'cherry']
[9, 6, 5, 5, 5, 4, 3, 3, 2, 1, 1]
[('Alice', 25), ('Bob', 30), ('Charlie', 20)]
Process finished with exit code 0
```



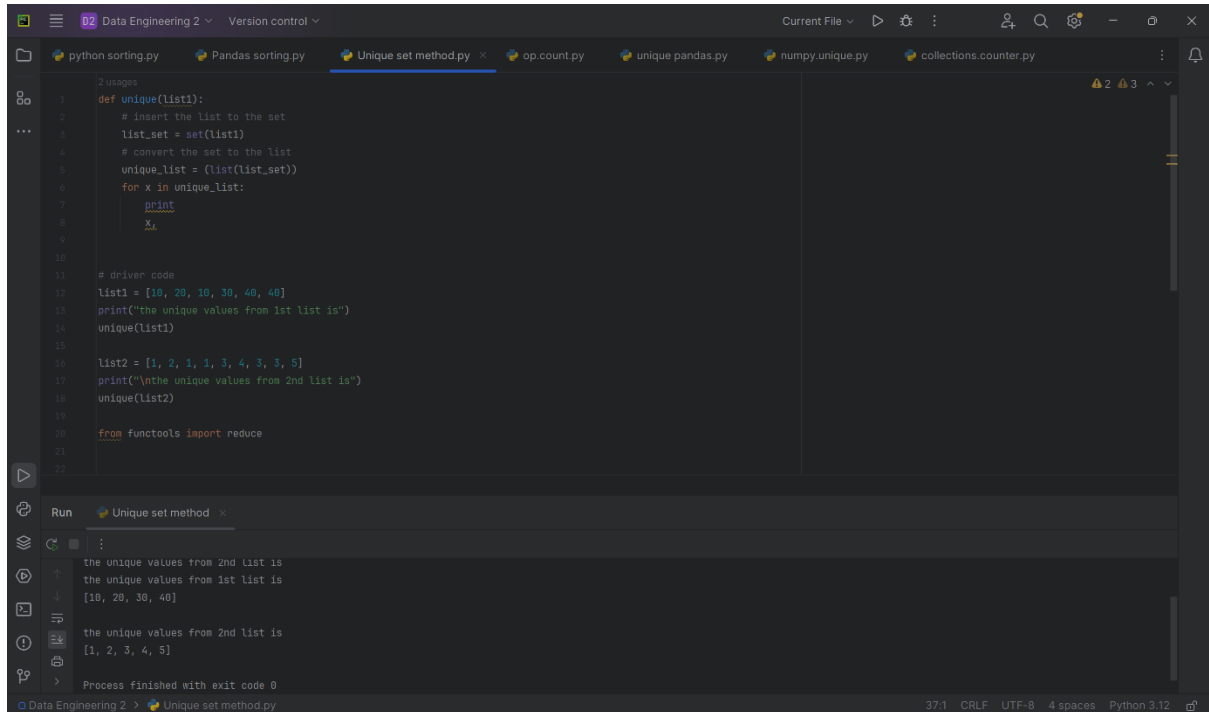
```
4 data = {'Age': [25, 30, 20],
5         'Salary': [50000, 60000, 45000]}
6
7 df = pd.DataFrame(data)
8 sorted_df = df.sort_values(by='Age')
9 print(sorted_df)
10
11 # Sorting in Descending Order
12 sorted_df_desc = df.sort_values(by='Age', ascending=False)
13 print(sorted_df_desc)
14
15 # Sorting by Multiple Columns
16 sorted_df_multi = df.sort_values(by=['Age', 'Salary'], ascending=[True, False])
17 print(sorted_df_multi)
18
19 # Sorting a Series
20 ages = pd.Series([25, 30, 20], name='Age')
21
22 # Sort the Series
23 sorted_ages = ages.sort_values()
24 print(sorted_ages)
25
26 sorted_df_index = df.sort_index()
```

Run Pandas sorting

```
import pandas as pd
  Name Age Salary
2 Charlie 20 45000
0 Alice 25 50000
1 Bob 30 60000
  Name Age Salary
1 Bob 30 60000
0 Alice 25 50000
```

Set Methods

❖ Get Unique Values from a List Using Set Method



```
1 def unique(list1):
2     # insert the list to the set
3     list_set = set(list1)
4     # convert the set to the list
5     unique_list = (list(list_set))
6     for x in unique_list:
7         print
8         x
9
10
11 # driver code
12 list1 = [10, 20, 10, 30, 40, 40]
13 print("the unique values from 1st list is")
14 unique(list1)
15
16 list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
17 print("\nthe unique values from 2nd list is")
18 unique(list2)
19
20 from functools import reduce
21
22
```

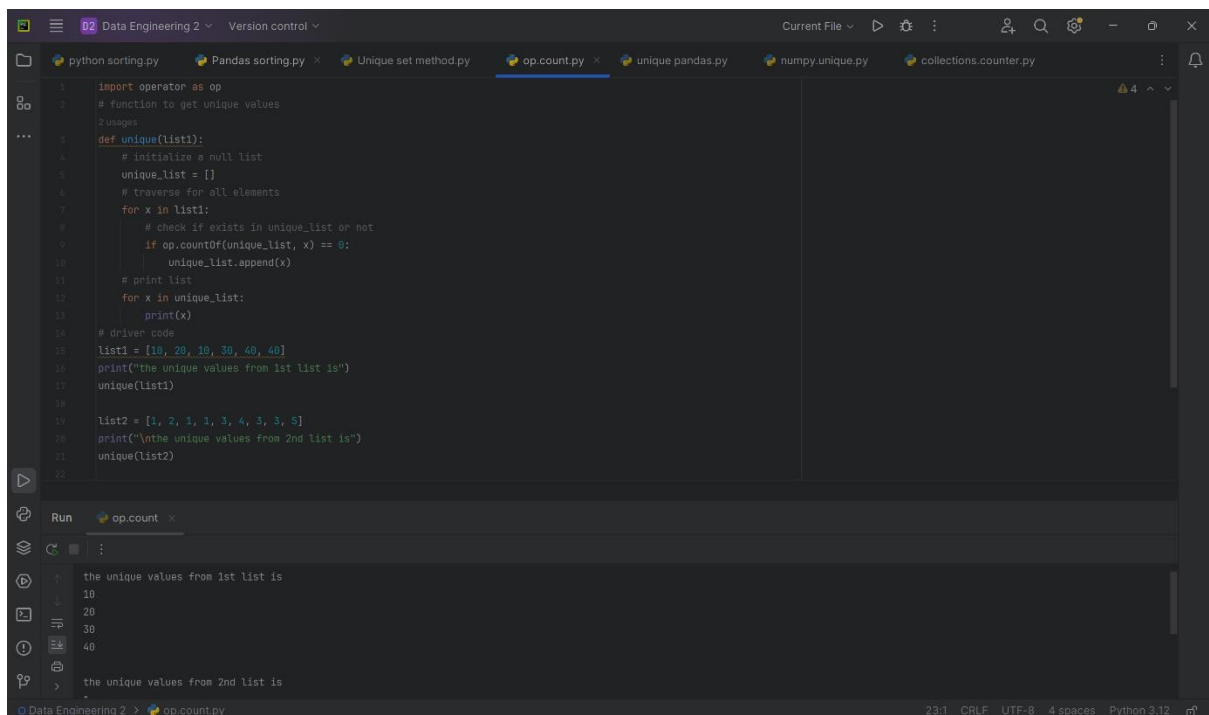
Run Unique set method

```
the unique values from 2nd list is
the unique values from 1st list is
[10, 20, 30, 40]

the unique values from 2nd list is
[1, 2, 3, 4, 5]

Process finished with exit code 0
```

❖ Get Unique Values From a List in Python Using Operator.countOf() method



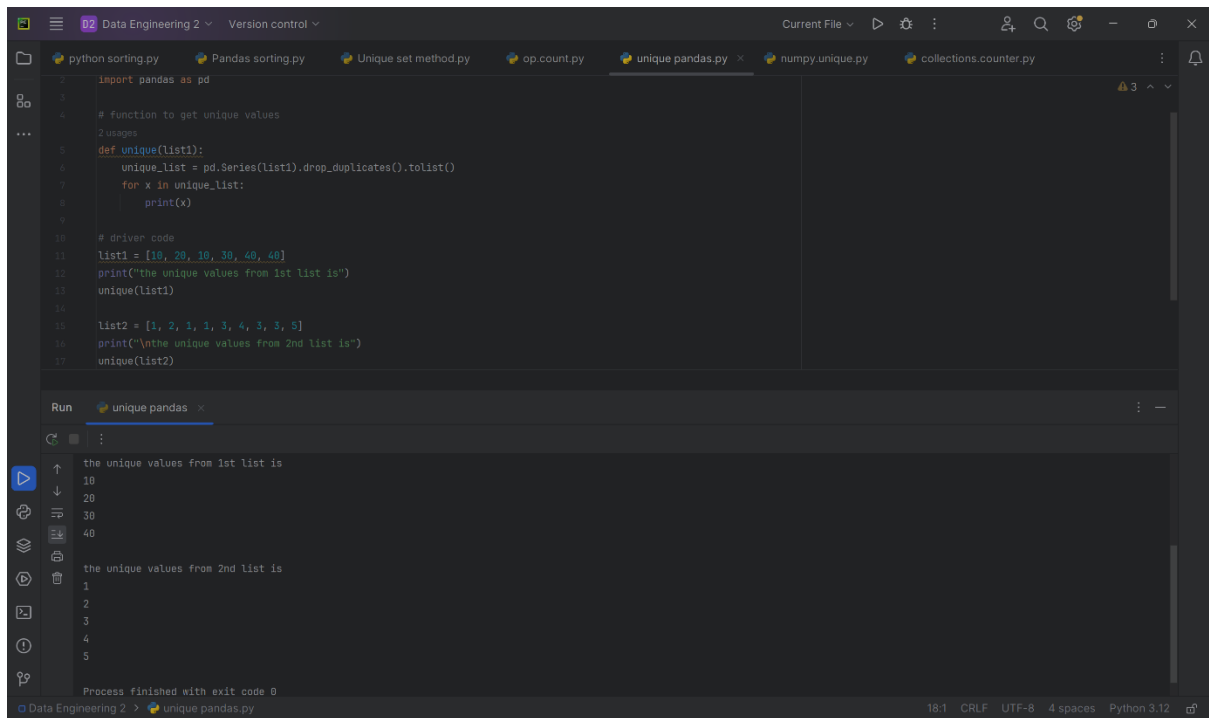
```
1 import operator as op
2 # function to get unique values
3 def unique(list1):
4     # initialize a null list
5     unique_list = []
6     # traverse for all elements
7     for x in list1:
8         # check if exists in unique_list or not
9         if op.countOf(unique_list, x) == 0:
10             unique_list.append(x)
11     # print list
12     for x in unique_list:
13         print(x)
14
15 # driver code
16 list1 = [10, 20, 10, 30, 40, 40]
17 print("the unique values from 1st list is")
18 unique(list1)
19
20 list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
21 print("\nthe unique values from 2nd list is")
22 unique(list2)
23
```

Run op.count

```
the unique values from 1st list is
10
20
30
40

the unique values from 2nd list is
1
2
3
4
5
```

❖ Get Unique Values From a List in Python Using pandas module



The screenshot shows a Jupyter Notebook interface with a file explorer at the top displaying several Python files. The active file is 'unique pandas.py'. The code in the notebook defines a function 'unique(list1)' that uses 'pd.Series(list1).drop_duplicates().tolist()' to get unique values. It then prints the unique values for two lists: list1 = [10, 20, 10, 30, 40, 40] and list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]. The output in the console shows the unique values for each list: [10, 20, 30, 40] and [1, 2, 3, 4, 5].

```
1 import pandas as pd
2
3 # function to get unique values
4
5 def unique(list1):
6     unique_list = pd.Series(list1).drop_duplicates().tolist()
7     for x in unique_list:
8         print(x)
9
10 # driver code
11 list1 = [10, 20, 10, 30, 40, 40]
12 print("the unique values from 1st list is")
13 unique(list1)
14
15 list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
16 print("\nthe unique values from 2nd list is")
17 unique(list2)
```

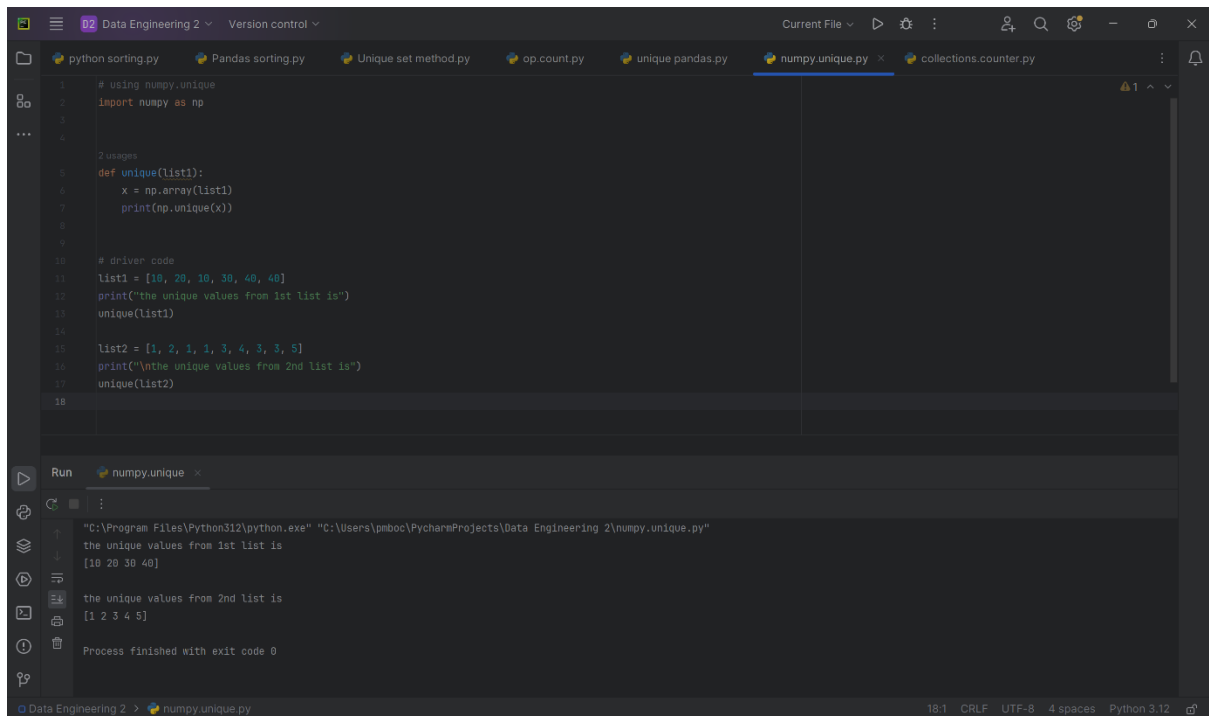
Run unique pandas

the unique values from 1st list is
10
20
30
40

the unique values from 2nd list is
1
2
3
4
5

Process finished with exit code 0

❖ Get Unique Values From a List Using numpy.unique



The screenshot shows a Jupyter Notebook interface with a file explorer at the top displaying several Python files. The active file is 'numpy.unique.py'. The code in the notebook defines a function 'unique(list1)' that uses 'np.array(list1)' and 'np.unique(x)' to get unique values. It then prints the unique values for two lists: list1 = [10, 20, 10, 30, 40, 40] and list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]. The output in the console shows the unique values for each list: [10 20 30 40] and [1 2 3 4 5].

```
1 # using numpy.unique
2 import numpy as np
3
4
5 # function to get unique values
6 def unique(list1):
7     x = np.array(list1)
8     print(np.unique(x))
9
10 # driver code
11 list1 = [10, 20, 10, 30, 40, 40]
12 print("the unique values from 1st list is")
13 unique(list1)
14
15 list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
16 print("\nthe unique values from 2nd list is")
17 unique(list2)
18
```

Run numpy.unique

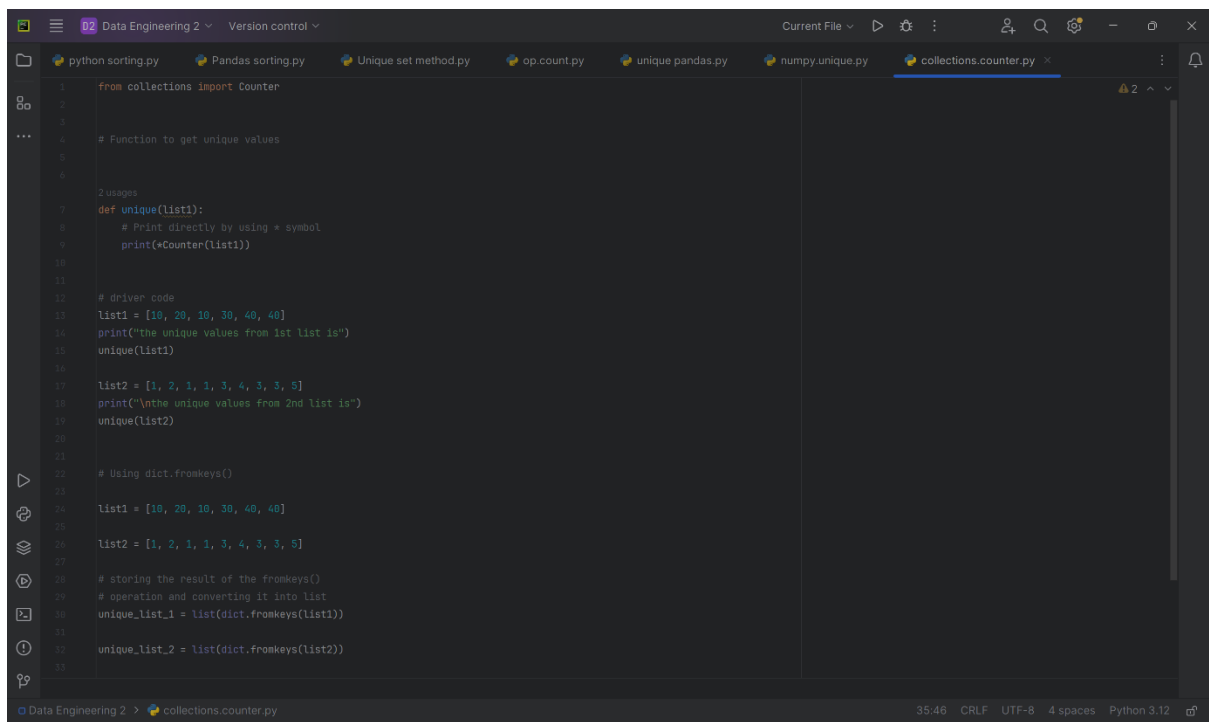
"C:\Program Files\Python312\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering 2\numpy.unique.py"

the unique values from 1st list is
[10 20 30 40]

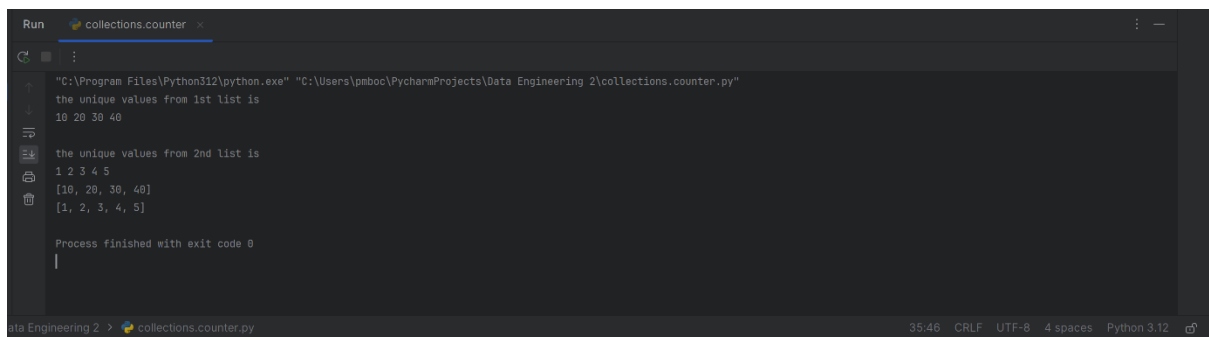
the unique values from 2nd list is
[1 2 3 4 5]

Process finished with exit code 0

❖ Get Unique Values From a List in Python Using collections.Counter()



```
1 from collections import Counter
2
3
4 # Function to get unique values
5
6
7 # Usage
8 def unique(List1):
9     # Print directly by using * symbol
10    print(*Counter(List1))
11
12 # driver code
13 List1 = [10, 20, 10, 30, 40, 40]
14 print("the unique values from 1st list is")
15 unique(List1)
16
17 List2 = [1, 2, 1, 1, 3, 4, 3, 5]
18 print("\nthe unique values from 2nd list is")
19 unique(List2)
20
21
22 # Using dict.fromkeys()
23
24 List1 = [10, 20, 10, 30, 40, 40]
25
26 List2 = [1, 2, 1, 1, 3, 4, 3, 5]
27
28 # storing the result of the fromkeys()
29 # operation and converting it into list
30 unique_list_1 = list(dict.fromkeys(List1))
31
32 unique_list_2 = list(dict.fromkeys(List2))
33
```



```
Run collections.counter
"C:\Program Files\Python312\python.exe" "C:\Users\pmboc\PycharmProjects\Data Engineering 2\collections.counter.py"
the unique values from 1st list is
10 20 30 40

the unique values from 2nd list is
1 2 3 4 5
[10, 20, 30, 40]
[1, 2, 3, 4, 5]

Process finished with exit code 0
```