

Name : Parth Nandedkar

Date : 31 Jan 2024

Topics : Python (Unique values, sorting, JSON)

Batch : Data Engineering Batch-1

## Getting Unique Values :

Get Unique Values from a List Using Set Method :

To get unique values from a list in Python, one common approach is to use the set() method. Here's a brief explanation:

Using Set Method to Get Unique Values from a List:

Set Data Structure:

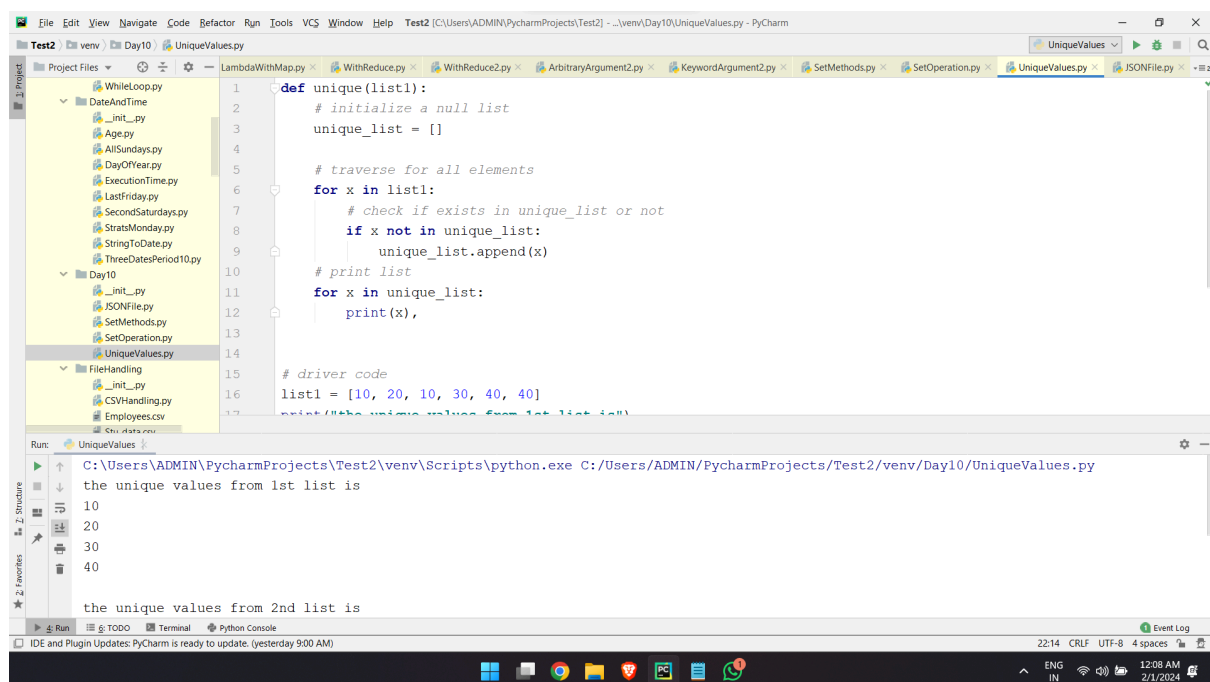
In Python, a set is an unordered collection of unique elements. It does not allow duplicate values.

Sets are defined using curly braces {} or the set() constructor.

Converting List to Set:

To obtain unique values from a list, you can convert the list to a set using the set() method.

This process automatically removes any duplicate elements from the list, leaving only unique values in the set.



```
1 def unique(list1):
2     # initialize a null list
3     unique_list = []
4
5     # traverse for all elements
6     for x in list1:
7         # check if exists in unique_list or not
8         if x not in unique_list:
9             unique_list.append(x)
10
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")
```

the unique values from 1st list is

10  
20  
30  
40

the unique values from 2nd list is

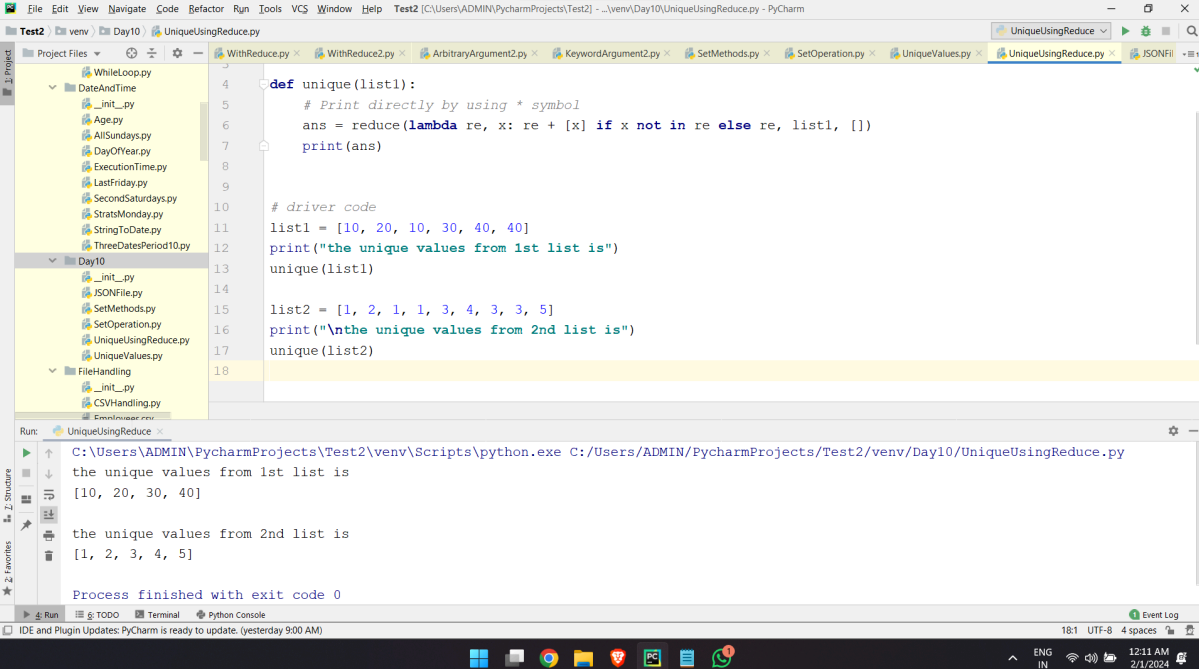
Using reduce() :

The reduce() function will iteratively combine elements while keeping only unique values.

The lambda function takes two parameters: unique\_set (the accumulated unique values) and value (the current value from the list).

unique\_set | {value} performs a union operation, adding the current value to the set of unique values.

The reduce() function is used to successively apply the lambda function to each element of the list, starting with an empty set (set()).



The screenshot shows the PyCharm IDE with a project named 'Test2'. The file explorer on the left shows a directory structure with files like 'WhileLoop.py', 'DateAndTime.py', 'Age.py', 'AllSundays.py', 'DayOfYear.py', 'ExecutionTime.py', 'LastFriday.py', 'SecondSaturdays.py', 'StratMonday.py', 'StringToDate.py', 'ThreeDatesPeriod10.py', 'Day10', 'JSONFile.py', 'SetMethods.py', 'SetOperation.py', 'UniqueUsingReduce.py', 'UniqueValues.py', 'FileHandling.py', and 'CSVHandling.py'. The main editor window displays the code for 'UniqueUsingReduce.py'.

```
1 def unique(list1):
2     # Print directly by using * symbol
3     ans = reduce(lambda re, x: re + [x] if x not in re else re, list1, [])
4     print(ans)
5
6 # driver code
7 list1 = [10, 20, 10, 30, 40, 40]
8 print("the unique values from 1st list is")
9 unique(list1)
10
11 list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
12 print("\nthe unique values from 2nd list is")
13 unique(list2)
```

The Run window at the bottom shows the output of the script:

```
C:\Users\ADMIN\PycharmProjects\Test2\venv\Scripts\python.exe C:/Users/ADMIN/PycharmProjects/Test2/venv/Day10/UniqueUsingReduce.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
```

The status bar at the bottom indicates the IDE and Plugin Updates: PyCharm is ready to update. (yesterday 9:00 AM). The system tray shows the date and time: 12:11 AM 2/1/2024.

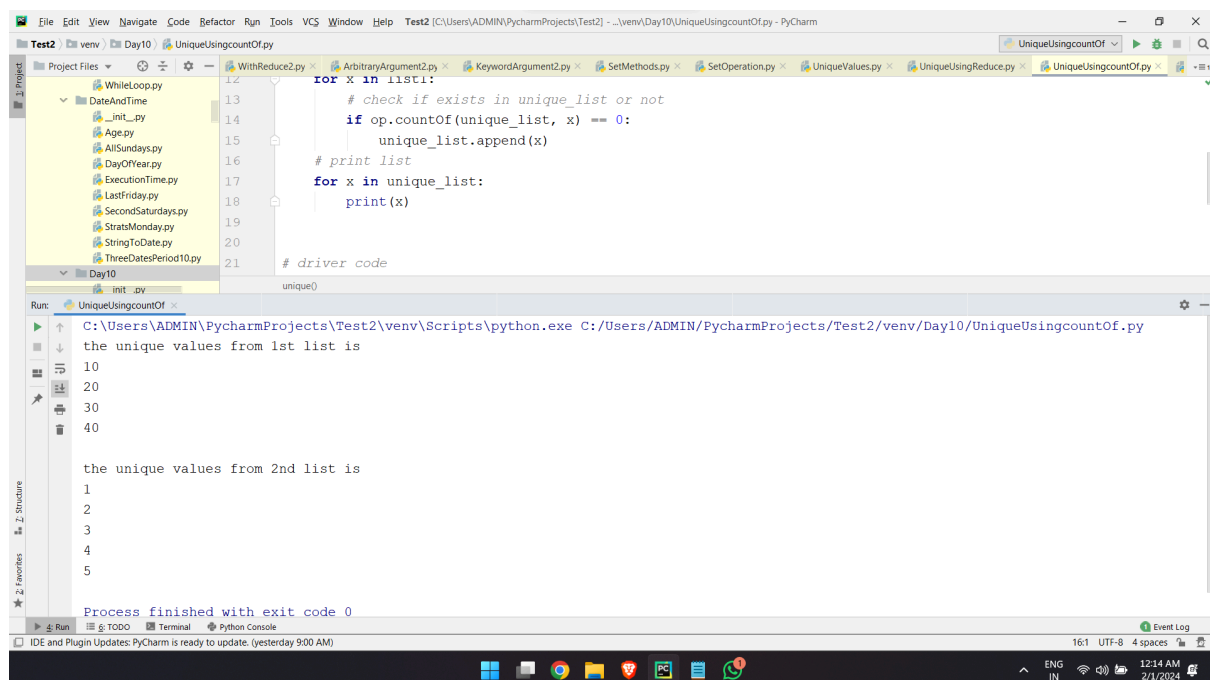
## Getting Unique Values From a List in Python Using Operator.countOf() :

operator.countOf(a, b) returns the number of occurrences of b in the sequence a.

### Usage:

It is part of the operator module in Python.

Commonly used when you want to count occurrences without manually iterating through the sequence.



The screenshot shows the PyCharm IDE with a project named 'Test2'. The file explorer on the left shows a directory structure with files like 'WhileLoop.py', 'DateTime.py', 'Age.py', 'AllSundays.py', 'DayOfYear.py', 'ExecutionTime.py', 'LastFriday.py', 'SecondSaturdays.py', 'StratsMonday.py', 'StringToDate.py', and 'ThreeDatesPeriod10.py'. The main editor window displays a Python script named 'UniqueUsingcountOf.py'. The script defines two lists, 'l1' and 'l2', and uses the 'operator.countOf()' function to find unique values. The output is displayed in the 'Run' console at the bottom.

```
12 for x in l1:
13     # check if exists in unique_list or not
14     if op.countOf(unique_list, x) == 0:
15         unique_list.append(x)
16     # print list
17     for x in unique_list:
18         print(x)
19
20 # driver code
21 unique()
```

Run: UniqueUsingcountOf.py

```
C:\Users\ADMIN\PycharmProjects\Test2\venv\Scripts\python.exe C:/Users/ADMIN/PycharmProjects/Test2/venv/Day10/UniqueUsingcountOf.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 Using numpy.unique :

numpy.unique()

### Definition:

numpy.unique(arr, return\_index=False, return\_inverse=False, return\_counts=False, axis=None) returns the unique elements of an array or a list-like object.

### Parameters:

arr: Input array or list.

return\_index: If True, returns the indices of the first occurrences of unique values.

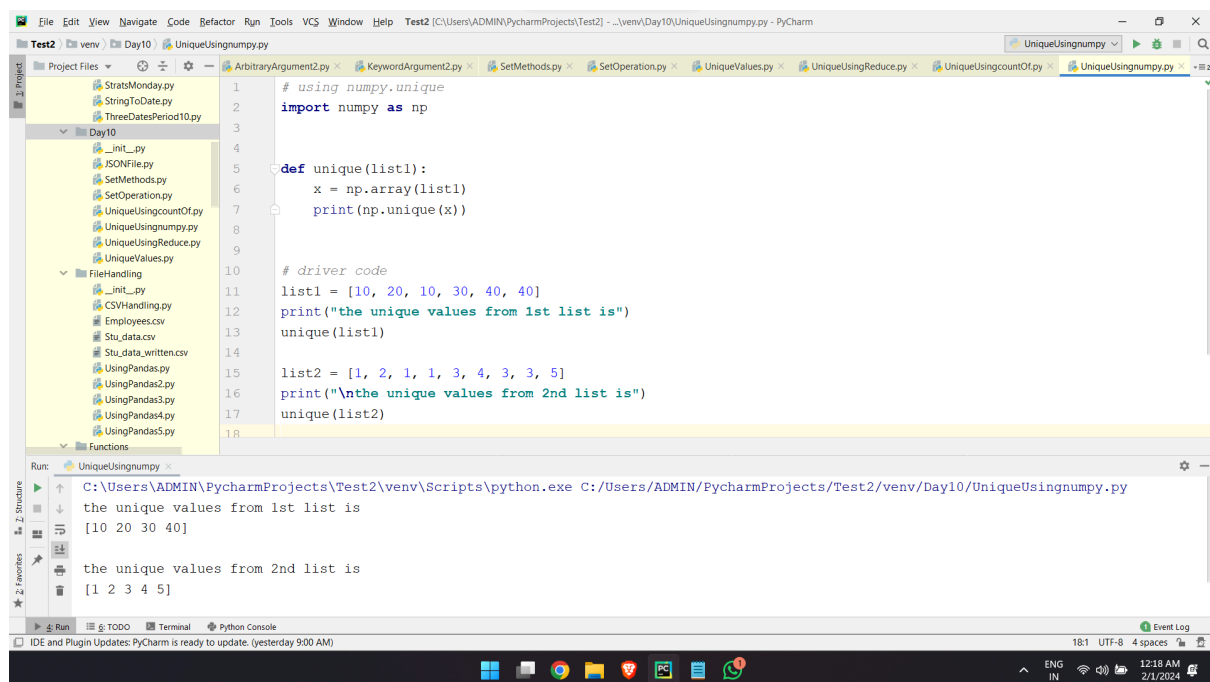
return\_inverse: If True, returns the indices to reconstruct the original array from the unique array.

return\_counts: If True, returns the number of times each unique value appears.

axis: The axis or axes along which unique values should be found.

### Usage:

The primary purpose is to find unique elements in an array or list.



```
1 # using numpy.unique
2 import numpy as np
3
4
5 def unique(list1):
6     x = np.array(list1)
7     print(np.unique(x))
8
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: UniqueUsingnumpy.py

C:\Users\ADMIN\PycharmProjects\Test2\venv\Scripts\python.exe C:/Users/ADMIN/PycharmProjects/Test2/venv/Day10/UniqueUsingnumpy.py

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 collections.Counter() :

### collections.Counter() Overview:

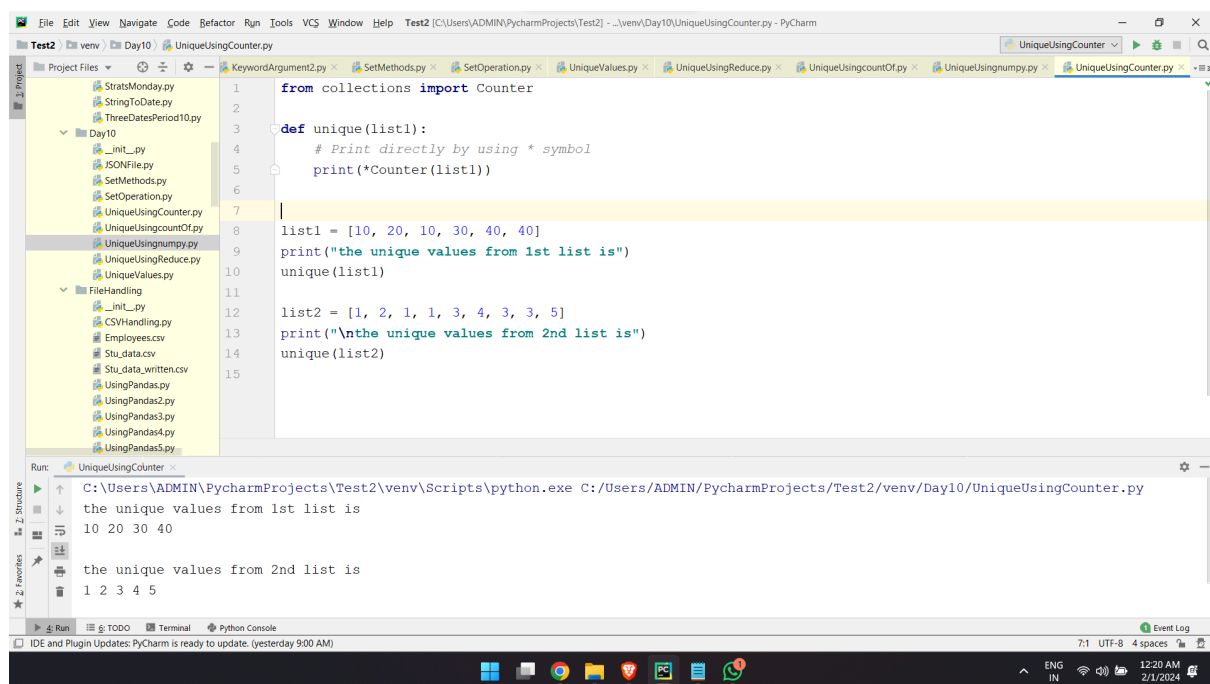
The collections.Counter class is a dictionary subclass in the collections module.

It is designed for counting hashable objects. It returns a dictionary where keys are unique elements, and values are their respective counts.

### Using collections.Counter() for Unique Values:

You can utilize collections.Counter() to count occurrences of each element in the list.

Extract the keys from the resulting Counter dictionary to obtain unique values.



```
1 from collections import Counter
2
3 def unique(list1):
4     # Print directly by using * symbol
5     print(*Counter(list1))
6
7
8 list1 = [10, 20, 10, 30, 40, 40]
9 print("the unique values from 1st list is")
10 unique(list1)
11
12 list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
13 print("\nthe unique values from 2nd list is")
14 unique(list2)
15
```

Run: UniqueUsingCounter

```
C:\Users\ADMIN\PycharmProjects\Test2\venv\Scripts\python.exe C:/Users/ADMIN/PycharmProjects/Test2/venv/Day10/UniqueUsingCounter.py
the unique values from 1st list is
10 20 30 40

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

## Sorting Lists in Python :

The `list.sort()` method in Python is used to modify a list in-place by sorting its elements. Here's an explanation of its key characteristics:

`list.sort()`

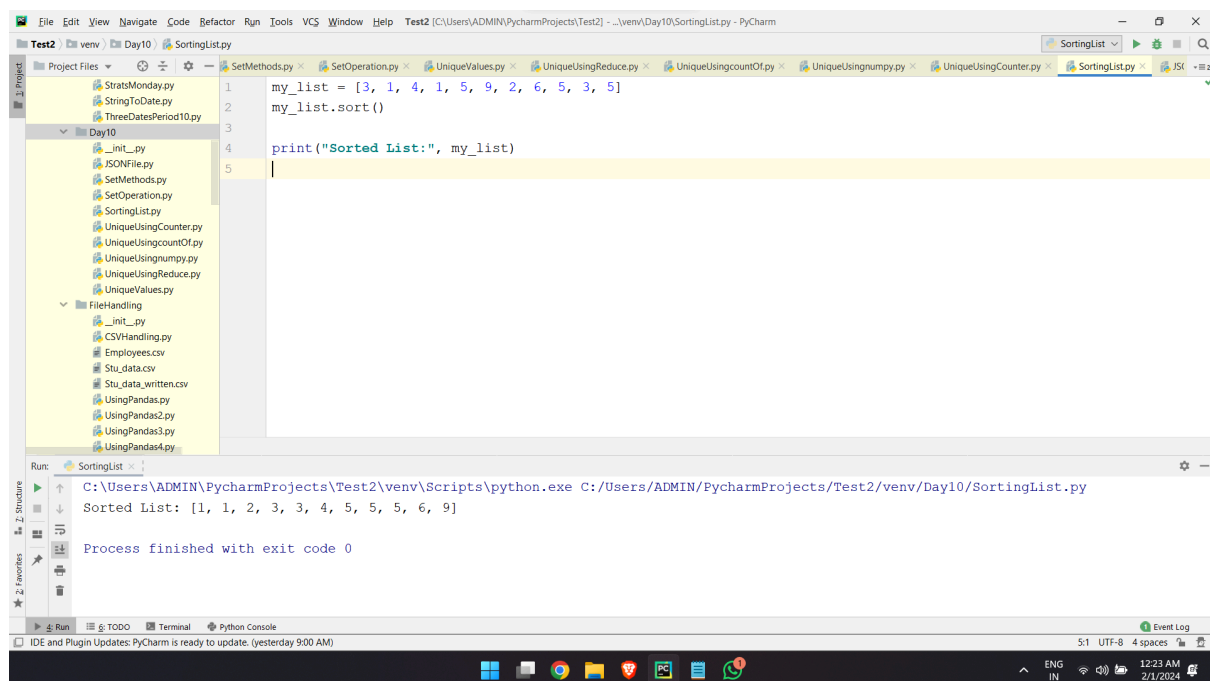
In-Place Sorting:

The `list.sort()` method sorts the elements of a list in ascending order by default. It modifies the original list in-place, meaning it does not create a new sorted list but rather rearranges the elements of the existing list.

Syntax:

The basic syntax of `list.sort()` is:

**`list.sort(key=None, reverse=False)`**



## Introduction to JSON :

JSON, or 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. JSON is widely used as a data format for representing structured data and exchanging information between a server and a web application, as well as between different parts of an application.

Here are key aspects of the JSON data type:

Format and Structure:

JSON data is represented as key-value pairs, similar to Python dictionaries or JavaScript objects. It is a text format that uses human-readable syntax.

Data Types:

JSON supports several data types, including strings, numbers, objects, arrays, booleans, and null.

Syntax Rules:

JSON follows a strict syntax with well-defined rules.

Data is represented in name/value pairs.

Data is separated by commas.

Objects are enclosed in curly braces {}, and arrays are enclosed in square brackets [].

JSON example,

```
{  
  "name": "John Doe",  
  "age": 30,  
  "city": "New York",  
  "isStudent": false,  
  "grades": [85, 90, 78],  
  "address": {  
    "street": "123 Main St",  
    "zip": "10001"  
  },  
  "contact": null  
}
```

`json.load()` :

`json.load()` and `json.loads()` functions in Python are both used for parsing JSON data, but they are applied in slightly different contexts.

`json.load()`

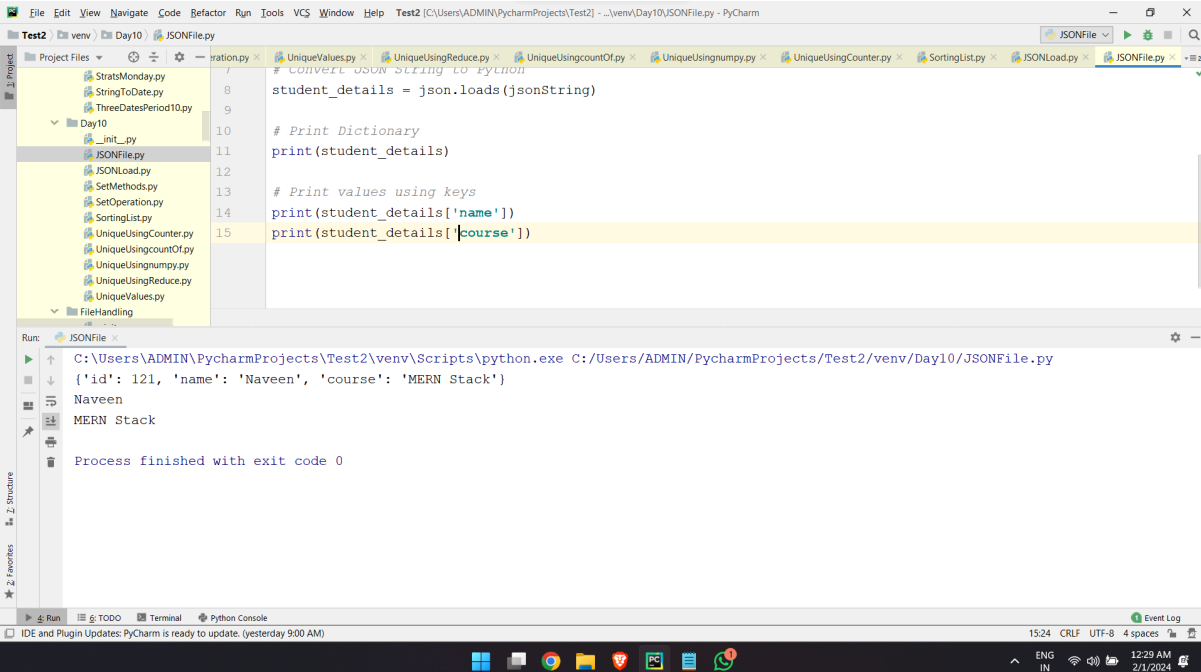
Functionality:

`json.load()` is used to read a JSON file and parse its content.

Syntax:

`import json`

with `open('filename.json', 'r')` as file:  
    `data = json.load(file)`



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script in `JSONFile.py` with the following code:

```
1 # Convert JSON String to Python
2
3 student_details = json.loads(jsonString)
4
5 # Print Dictionary
6 print(student_details)
7
8 # Print values using keys
9 print(student_details['name'])
10
11 print(student_details['course'])
```

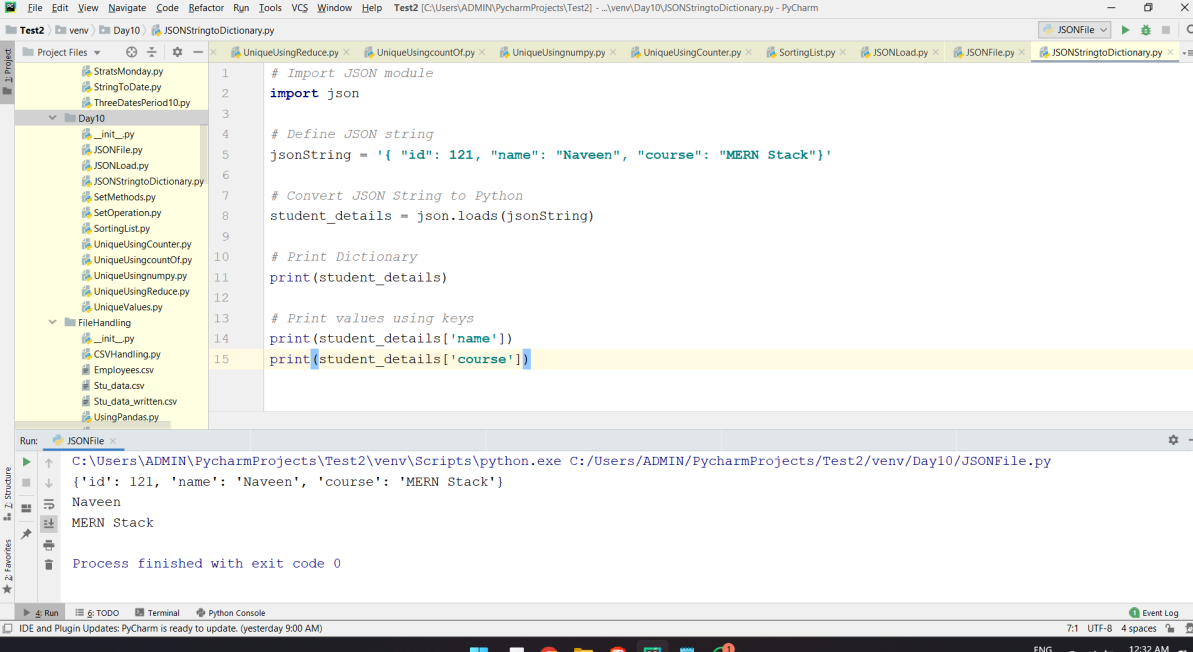
The left sidebar shows the Project Files view with a tree structure including `Day10` and `JSONFile.py`. The bottom Run console shows the output of the script:

```
Run: JSONFile
C:\Users\ADMIN\PycharmProjects\Test2\venv\Scripts\python.exe C:/Users/ADMIN/PycharmProjects/Test2/venv/Day10/JSONFile.py
{'id': 121, 'name': 'Naveen', 'course': 'MERN Stack'}
Naveen
MERN Stack
Process finished with exit code 0
```

The status bar at the bottom indicates the current file is `JSONFile.py`, encoding is `UTF-8`, and the date is `2/1/2024`.



## Convert JSON String to Dictionary Python :



The screenshot shows the PyCharm IDE with a project named 'Test2'. The file explorer on the left shows a directory structure with files like 'StratsMonday.py', 'StringToDate.py', 'ThreeDatesPeriod10.py', 'Day10', and various utility files. The main editor displays the file 'JSONStringtoDictionary.py' with the following Python code:

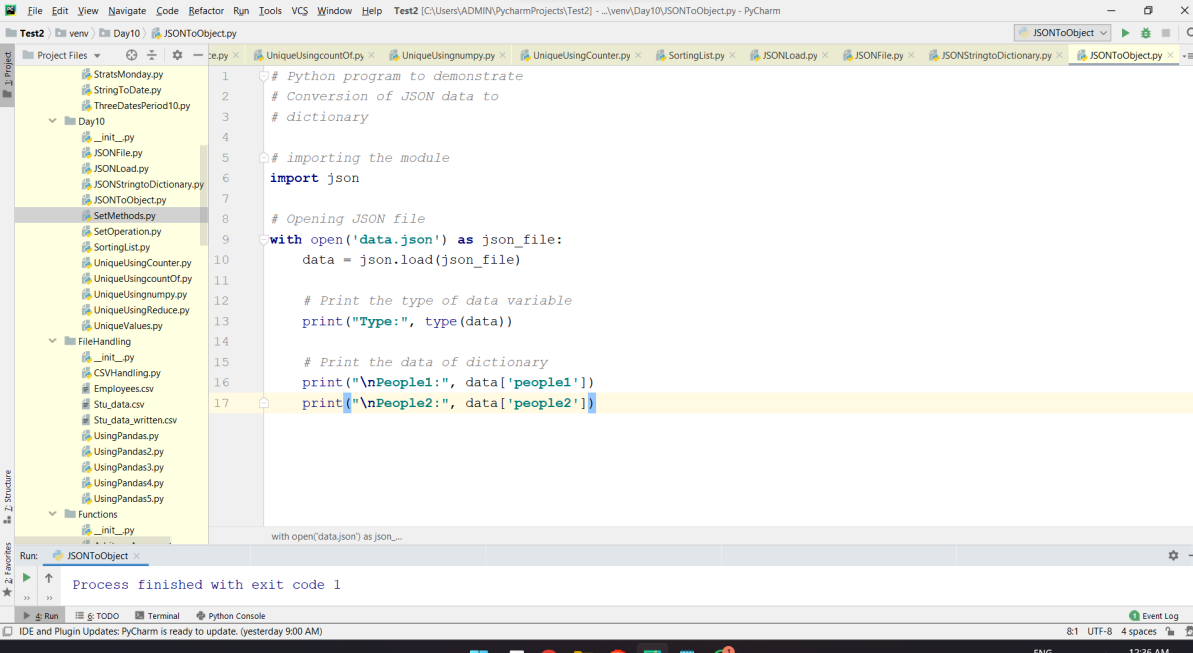
```
1 # Import JSON module
2 import json
3
4 # Define JSON string
5 jsonString = '{ "id": 121, "name": "Naveen", "course": "MERN Stack"}'
6
7 # Convert JSON String to Python
8 student_details = json.loads(jsonString)
9
10 # Print Dictionary
11 print(student_details)
12
13 # Print values using keys
14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

The Run window at the bottom shows the execution output:

```
C:\Users\ADMIN\PycharmProjects\Test2\venv\Scripts\python.exe C:\Users\ADMIN\PycharmProjects\Test2\venv\Day10\JSONFile.py
{'id': 121, 'name': 'Naveen', 'course': 'MERN Stack'}
Naveen
MERN Stack
Process finished with exit code 0
```

The status bar at the bottom indicates the IDE and Plugin Updates: PyCharm is ready to update. (yesterday 9:00 AM).

## Convert JSON File to Python Object :



The screenshot shows the PyCharm IDE with a project named 'Test2'. The file explorer on the left shows a directory structure with files like 'StratsMonday.py', 'StringToDate.py', 'ThreeDatesPeriod10.py', 'Day10', and various utility files. The main editor displays the file 'JSONToObject.py' with the following Python code:

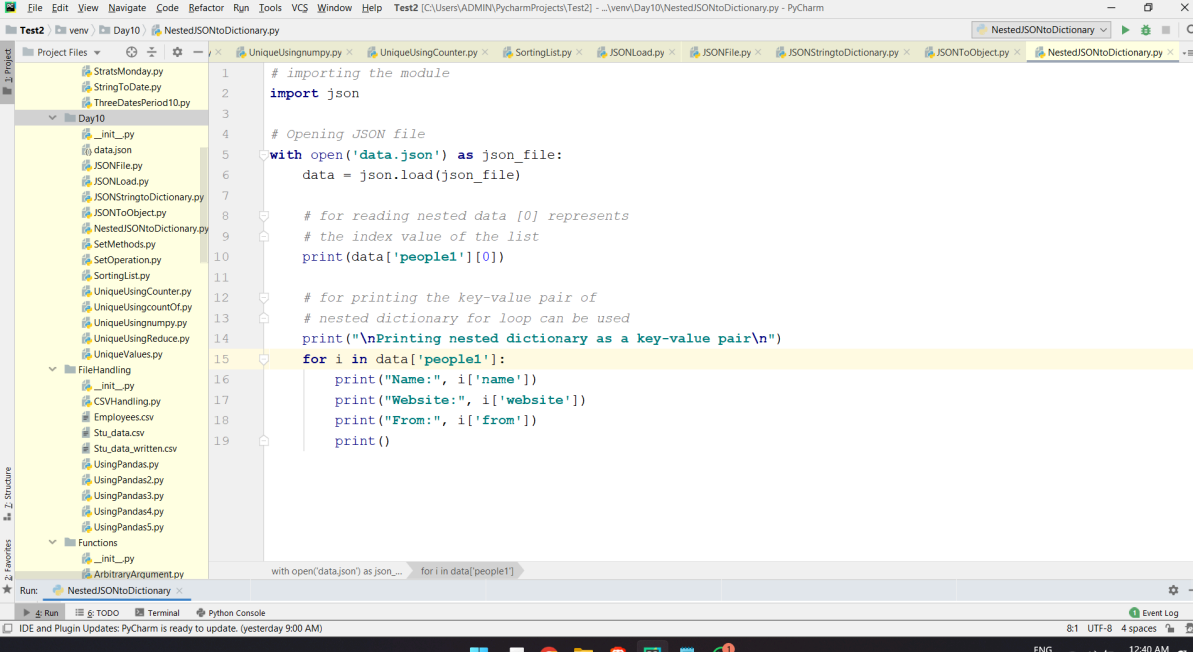
```
1 # Python program to demonstrate
2 # Conversion of JSON data to
3 # dictionary
4
5 # importing the module
6 import json
7
8 # Opening JSON file
9 with open('data.json') as json_file:
10     data = json.load(json_file)
11
12 # Print the type of data variable
13 print("Type:", type(data))
14
15 # Print the data of dictionary
16 print("\nPeople1:", data['people1'])
17 print("\nPeople2:", data['people2'])
```

The Run window at the bottom shows the execution output:

```
Process finished with exit code 1
```

The status bar at the bottom indicates the IDE and Plugin Updates: PyCharm is ready to update. (yesterday 9:00 AM).

## Convert Nested JSON Object to Dictionary :

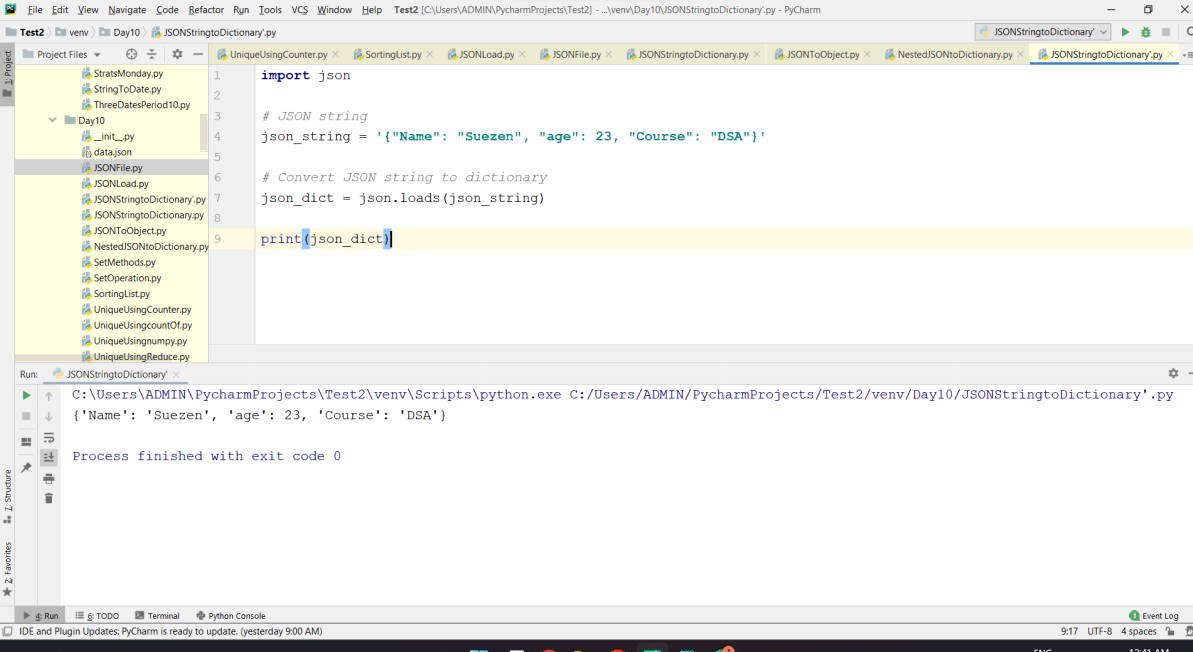


The screenshot shows the PyCharm IDE with a project named 'Test2'. The file explorer on the left shows a directory structure with files like 'StratsMonday.py', 'StringToDate.py', 'ThreeDatesPeriod10.py', and a 'Day10' folder containing various utility files. The main editor displays a Python script named 'NestedJSONtoDictionary.py'. The script imports the 'json' module, opens a file named 'data.json', loads the JSON data, and prints the value of 'people1[0]'. It then iterates over the 'people1' list, printing the 'Name', 'Website', and 'From' fields for each entry.

```
1 # importing the module
2 import json
3
4 # Opening JSON file
5 with open('data.json') as json_file:
6     data = json.load(json_file)
7
8 # for reading nested data [0] represents
9 # the index value of the list
10 print(data['people1'][0])
11
12 # for printing the key-value pair of
13 # nested dictionary for loop can be used
14 print("\nPrinting nested dictionary as a key-value pair\n")
15 for i in data['people1']:
16     print("Name:", i['name'])
17     print("Website:", i['website'])
18     print("From:", i['from'])
19     print()
```

The Run tab at the bottom shows the command: `C:\Users\ADMIN\PycharmProjects\Test2\venv\Scripts\python.exe C:\Users\ADMIN\PycharmProjects\Test2\venv\Day10\NestedJSONtoDictionary.py`. The output shows the first entry of the 'people1' list.

## Convert JSON String to Dictionary in Python :



The screenshot shows the PyCharm IDE with a project named 'Test2'. The file explorer on the left shows a directory structure with files like 'StratsMonday.py', 'StringToDate.py', 'ThreeDatesPeriod10.py', and a 'Day10' folder containing various utility files. The main editor displays a Python script named 'JSONStringtoDictionary.py'. The script imports the 'json' module, defines a JSON string, converts it to a dictionary using 'json.loads()', and prints the resulting dictionary.

```
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 Run tab at the bottom shows the command: `C:\Users\ADMIN\PycharmProjects\Test2\venv\Scripts\python.exe C:\Users\ADMIN\PycharmProjects\Test2\venv\Day10\JSONStringtoDictionary.py`. The output shows the dictionary: `{'Name': 'Suezen', 'age': 23, 'Course': 'DSA'}`. The process finished with exit code 0.

