

Day 9 Assessment on Set, Unique, JSON

Name: Krushnakumar Patle

Email: krishnapatle128@gmail.com

Batch: Data Engineering Batch-1

SET

A Set in Python programming is an unordered collection data type that is iterable, mutable and has no duplicate elements. Set are represented by {} (values enclosed in curly braces)

```
✓ [1] var = {"Hexaware", "for"}
Ds type(var)

set
```

Type Casting with Python Set method

```
✓ # typecasting list to set
Ds myset = set(["a", "b", "c"])
print(myset)

# Adding element to the set
myset.add("d")
print(myset)

{ 'a', 'b', 'c' }
{ 'd', 'a', 'b', 'c' }
```

Check unique and Immutable with Python Set

```
! # a set cannot have duplicate values
Ds myset = {"Hexaware", "for", "Hexaware"}
print(myset)

# values of a set cannot be changed
myset[1] = "Hello"
print(myset)

{ 'Hexaware', 'for' }
```

TypeError Traceback (most recent call last)

```
<ipython-input-3-94b71b90bcdb> in <cell line: 6>()
4
5 # values of a set cannot be changed
----> 6 myset[1] = "Hello"
7 print(myset)

TypeError: 'set' object does not support item assignment
```

EXPLAIN ERROR

Day 9 Assessment on Set, Unique, JSON

Adding elements to Python Sets

```
✓ 0s ▶ people = {"Jay", "Idrish", "Archi"}

print("People:", end = " ")
print(people)

# This will add Daxit
# in the set
people.add("Daxit")

# Adding elements to the
# set using iterator
for i in range(1, 6):
    people.add(i)

print("\nSet after adding element:", end = " ")
print(people)
```

```
➡ People: {'Jay', 'Archi', 'Idrish'}

Set after adding element: {1, 'Jay', 2, 3, 4, 5, 'Idrish', 'Archi', 'Daxit'}
```

Union operations on Python sets

```
✓ 0s [5] # Python Program to
# demonstrate union of
# two sets

people = {"Jay", "Idrish", "Archil"}
```

Union operations on Python sets

```
✓ 0s ▶ # Python Program to
# demonstrate union of
# two sets

people = {"Jay", "Idrish", "Archil"}
vampires = {"Karan", "Arjun"}
dracula = {"Deepanshu", "Raju"}

# Union using union()
# function
population = people.union(vampires)

print("Union using union() function")
print(population)

# Union using "|"
# operator
population = people|dracula
print("\nUnion using '|' operator")
print(population)
```

```
➡ Union using union() function
{'Archil', 'Jay', 'Idrish', 'Arjun', 'Karan'}

Union using '|' operator
{'Archil', 'Jay', 'Idrish', 'Deepanshu', 'Raju'}
```

Day 9 Assessment on Set, Unique, JSON

Intersection operation on Python Sets

✓
0s



```
# Python program to
# demonstrate intersection
# of two sets

set1 = set()
set2 = set()

for i in range(5):
    set1.add(i)

for i in range(3,9):
    set2.add(i)

# Intersection using
# intersection() function
set3 = set1.intersection(set2)

print("Intersection using intersection() function")
print(set3)

# Intersection using
# "&" operator
set3 = set1 & set2

print("\nIntersection using '&' operator")
print(set3)
```

Day 9 Assessment on Set, Unique, JSON

✓
0s

[6]

Intersection using intersection() function
{3, 4}

Intersection using '&' operator
{3, 4}

Finding Differences of Sets in Python

✓
0s



```
# Python program to  
# demonstrate difference  
# of two sets
```

```
set1 = set()  
set2 = set()
```

```
for i in range(5):  
    set1.add(i)
```

```
for i in range(3,9):  
    set2.add(i)
```

```
# Difference of two sets  
# using difference() function  
set3 = set1.difference(set2)
```

```
print(" Difference of two sets using difference() function")  
print(set3)
```

```
# Difference of two sets  
# using '-' operator
```

Day 9 Assessment on Set, Unique, JSON

```
✓ 0s [7] set3 = set1 - set2

print("\nDifference of two sets using '-' operator")
print(set3)
```

Difference of two sets using difference() function
{0, 1, 2}

Difference of two sets using '-' operator
{0, 1, 2}

Clearing Python Sets Set Clear() method empties the whole set inplace.

```
✓ 0s ▶ # Python program to
# demonstrate clearing
# of set

set1 = {1,2,3,4,5,6}

print("Initial set")
print(set1)

# This method will remove
# all the elements of the set
set1.clear()

print("\nSet after using clear() function")
print(set1)
```

Initial set
{1, 2, 3, 4, 5, 6}

0s [8] Set after using clear() function
set()

Get Unique Values from a List Using Set Method

Using set() property of Python, we can easily check for the unique values. Insert the values of the list in a set. Set only stores a value once even if it is inserted more than once. After inserting all the values in the set by list_set=set(list1), convert this set to a list to print it.

```
✓ 0s ▶ def unique(list1):

    # insert the list to the set
    list_set = set(list1)
    # convert the set to the list
    unique_list = (list(list_set))
    for x in unique_list:
        print(x)

# driver code
list1 = [10, 20, 10, 30, 40, 40]
print("the unique values from 1st list is")
unique(list1)

list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
print("\nthe unique values from 2nd list is")
unique(list2)
```

Day 9 Assessment on Set, Unique, JSON

```
✓ [9] the unique values from 1st list is
0s 40
    10
    20
    30

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

Get Unique Values From a List in Python Using reduce() function

Using Python import reduce() from functools and iterate over all element and checks if the element is a duplicate or unique value. Below is the implementation of the above approach.

```
✓ [9] from functools import reduce
0s

def unique(list1):

    # Print directly by using * symbol
    ans = reduce(lambda re, x: re+[x] if x not in re else re, list1, [])
    print(ans)

    # driver code
    list1 = [10, 20, 10, 30, 40, 40]
    print("the unique values from 1st list is")
    unique(list1)

[10] list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
    print("\nthe unique values from 2nd list is")
    unique(list2)
```

```
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 Operator.countOf() method

The 'unique' function initializes an empty 'unique_list', then iterates through 'list1'. For each element 'x', it employs 'op.countOf()' to check if 'x' is present in 'unique_list'. If not found (count is 0), 'x' is appended to 'unique_list'. The final unique values are printed using a loop.


```
✓ [9] import operator as op
0s # function to get unique values

def unique(list1):

    # initialize a null list
    unique_list = []


    # traverse for all elements
    for x in list1:
        # check if exists in unique_list or not
        if op.countOf(unique_list, x) == 0:
            unique_list.append(x)
    # print list
```

Day 9 Assessment on Set, Unique, JSON

```
05  for x in unique_list:
    print(x)

# driver code
list1 = [10, 20, 10, 30, 40, 40]
print("the unique values from 1st list is")
unique(list1)


list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
print("\nthe unique values from 2nd list is")
unique(list2)
```

```
 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 'unique' function utilizes Pandas to create a Series from 'list1', then employs 'drop_duplicates()' to eliminate duplicates and obtain a list of unique values. Subsequently, it iterates through the unique list and prints each element.

```
15  import pandas as pd

# function to get unique values
def unique(list1):
    unique_list = pd.Series(list1).drop_duplicates().tolist()
    for x in unique_list:
        print(x)

# driver code
list1 = [10, 20, 10, 30, 40, 40]
print("the unique values from 1st list is")
unique(list1)

list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
print("\nthe unique values from 2nd list is")
unique(list2)
```


```
 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 Using numpy.unique

Day 9 Assessment on Set, Unique, JSON


Using Python's import numpy, the unique elements in the array are also obtained. In the first step convert the list to `x=np.array(list)` and then use `numpy.unique(x)` function to get the unique values from the list. `numpy.unique()` returns only the unique values in the list.

```
0s  # using numpy.unique
import numpy as np

def unique(list1):
    x = np.array(list1)
    print(np.unique(x))

# driver code
list1 = [10, 20, 10, 30, 40, 40]
print("the unique values from 1st list is")
unique(list1)


list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
print("\nthe unique values from 2nd list is")
unique(list2)
```

```
 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()`

Using Python to import `Counter()` from `collections` print all the keys of `Counter` elements or we print directly by using the `"*"` symbol. Below is the implementation of the above approach.

```
0s  from collections import Counter


# Function to get unique values

def unique(list1):

    # Print directly by using * symbol
    print(*Counter(list1))

# driver code
list1 = [10, 20, 10, 30, 40, 40]
print("the unique values from 1st list is")
unique(list1)

list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]
print("\nthe unique values from 2nd list is")
unique(list2)
```

```
 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 Using `dict.fromkeys()`

Using the `fromkeys()` method of dictionary data structure we can fetch the unique elements. Firstly we need to define a list that consists of duplicate elements. Then we need to use a variable in which we will store the result after using the `fromkeys()` method.

Day 9 Assessment on Set, Unique, JSON

```
✓ [15] # defining a list which consists duplicate values
0s list1 = [10, 20, 10, 30, 40, 40]

list2 = [1, 2, 1, 1, 3, 4, 3, 3, 5]

# storing the result of the fromkeys()
# operation and converting it into list
unique_list_1 = list(dict.fromkeys(list1))

unique_list_2 = list(dict.fromkeys(list2))

# Printing the final result
print(unique_list_1, unique_list_2, sep="\n")

[10, 20, 30, 40]
[1, 2, 3, 4, 5]
```

The easiest way to sort is with the `sorted(list)` function, which takes a list and returns a new list with those elements in sorted order. The original list is not changed.

```
✓ [16] a = [5, 1, 4, 3]
0s print(sorted(a))
print(a)

[1, 3, 4, 5]
[5, 1, 4, 3]
```

JSON Introduction

JSON stands for JavaScript Object Notation. It is a format for structuring data. This format is used by different web applications to

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 file `.json` extension.

Convert JSON String to Dictionary Python

In this example, we are going to convert a JSON string to Python Dictionary using `json.loads()` method of JSON module in Python. Firstly, we import json module and then define JSON string after that converting JSON string to Python dictionary by passing it to `json.loads()` in parameter. We have print the dictionary and their values using the keys as seen in the output.

```
✓ [17] import json
0s # Define JSON string
jsonString = '{ "id": 121, "name": "Naveen", "course": "MERN Stack" }'

# Convert JSON String to Python
student_details = json.loads(jsonString)

# Print Dictionary
print(student_details)

# Print values using keys
print(student_details['name'])
print(student_details['course'])
```

Day 9 Assessment on Set, Unique, JSON

```
✓ 0s ▶ import json

# Define JSON string
jsonString = '{ "id": 121, "name": "Naveen", "course": "MERN Stack"}'

# Convert JSON String to Python
student_details = json.loads(jsonString)

# Print Dictionary
print(student_details)

# Print values using keys
print(student_details['name'])
print(student_details['course'])
```

```
{'id': 121, 'name': 'Naveen', 'course': 'MERN Stack'}
Naveen
MERN Stack
```

Convert JSON File to Python Object

Below is the JSON file that we will convert to Python dictionary using `json.load()` method. In the below code, firstly we open the "data.json" file using file handling in Python and then convert the file to Python object using the `json.load()` method we have also print the type of data after conversion and print the dictionary.

```
[ ] import json

# Opening JSON file
with open('data.json') as json_file:

[ ]     data = json.load(json_file)

# Print the type of data variable
print("Type:", type(data))

# Print the data of dictionary
print("\nPeople1:", data['people1'])
print("\nPeople2:", data['people2'])
```

Convert JSON String to Dictionary in Python

In this example, we will convert the json string into Python dictionary using `json.loads()` method. Firstly, we will import JSON module. Create a json string and store it in a variable 'json_string' after that we will convert the json string into dictionary by passing 'json_string' into `json.loads()` as argument and store the converted dictionary in 'json_dict'. Finally, print the Python dictionary.

```
✓ 0s ▶ import json

# JSON string
json_string = '{"Name": "Suezen", "age": 23, "Course": "DSA"}'

# Convert JSON string to dictionary
json_dict = json.loads(json_string)

print(json_dict)
```

```
{'Name': 'Suezen', 'age': 23, 'Course': 'DSA'}
```

Python Parse JSON String

In the below code, we are going to convert JSON to a Python object. To parse JSON string Python firstly we import the JSON module. We have a

Day 9 Assessment on Set, Unique, JSON

In the below code, we are going to convert JSON to a Python object. To parse JSON string Python firstly we import the JSON module. We have a JSON string stored in a variable 'employee' and we convert this JSON string to a Python object using `json.loads()` method of JSON module in Python. After that, we print the name of an employee using the key 'name'.

```
0s [20] import json

# JSON string
employee = '{"id": "09", "name": "Nitin", "department": "Finance"}'

# Convert string to Python dict
employee_dict = json.loads(employee)
print(employee_dict)

print(employee_dict['name'])
```

```
{'id': '09', 'name': 'Nitin', 'department': 'Finance'}
Nitin
```

Convert from Python to JSON

If you have a Python object, you can convert it into a JSON string by using the `json.dumps()` method.

```
0s [20] import json

# a Python object (dict):
x = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

# convert into JSON:
y = json.dumps(x)

# the result is a JSON string:
print(y)
```

```
{"name": "John", "age": 30, "city": "New York"}
```

Writing JSON to a file in Python

We can write JSON to file using `json.dump()` function of JSON module and file handling in Python. In the below program, we have opened a file named `sample.json` in writing mode using 'w'. The file will be created if it does not exist. `json.dump()` will transform the Python dictionary to a JSON string and it will be saved in the file `sample.json`.

```
0s [21] import json

# Data to be written
dictionary = {
    "name": "sathiyajith",
    "rollno": 56,
    "cgpa": 8.6,
    "phonenum": "9976770500"
}

with open("sample.json", "w") as outfile:
    json.dump(dictionary, outfile)
```

Python Pretty Print JSON

Day 9 Assessment on Set, Unique, JSON

When we convert a string to JSON the data is in a less readable format. To make it more readable we can use pretty printing by passing additional arguments in `json.dumps()` function such as `indent` and `sort_keys` as used in the below code.

```
0s  import json

# JSON string
employee = '{"id": "09", "name": "Nitin", "department": "Finance"}'

# Convert string to Python dict
employee_dict = json.loads(employee)


# Pretty Printing JSON string back
print(json.dumps(employee_dict, indent = 4, sort_keys= True))
```

```
 {
    "department": "Finance",
    "id": "09",
    "name": "Nitin"
}
```


You can convert Python objects of the following types, into JSON strings:

- dict
- list
- tuple
- string
- int

- float
- True
- False
- None

```
0s  import json

print(json.dumps({"name": "John", "age": 30}))
print(json.dumps(["apple", "bananas"]))
print(json.dumps(("apple", "bananas")))
print(json.dumps("hello"))
print(json.dumps(42))
print(json.dumps(31.76))
print(json.dumps(True))
print(json.dumps(False))
print(json.dumps(None))
```

```
 {"name": "John", "age": 30}
["apple", "bananas"]
["apple", "bananas"]
"hello"
42
31.76
true
false
null
```

Convert a Python object containing all the legal data types:

Day 9 Assessment on Set, Unique, JSON

✓
0s

```
import json

x = {
    "name": "John",
    "age": 30,
    "married": True,
    "divorced": False,
    "children": ("Ann", "Billy"),
    "pets": None,
    "cars": [
        {"model": "BMW 230", "mpg": 27.5},
        {"model": "Ford Edge", "mpg": 24.1}
    ]
}

print(json.dumps(x, indent=4, sort_keys=True))
```

```
{
  "age": 30,
  "cars": [
    {
      "model": "BMW 230",
      "mpg": 27.5
    },
    {
      "model": "Ford Edge",
      "mpg": 24.1
    }
  ],
  "children": [
    "Ann",
    "Billy"
  ],
  "divorced": false,
  "married": true,
  "name": "John",
  "pets": null
}
```