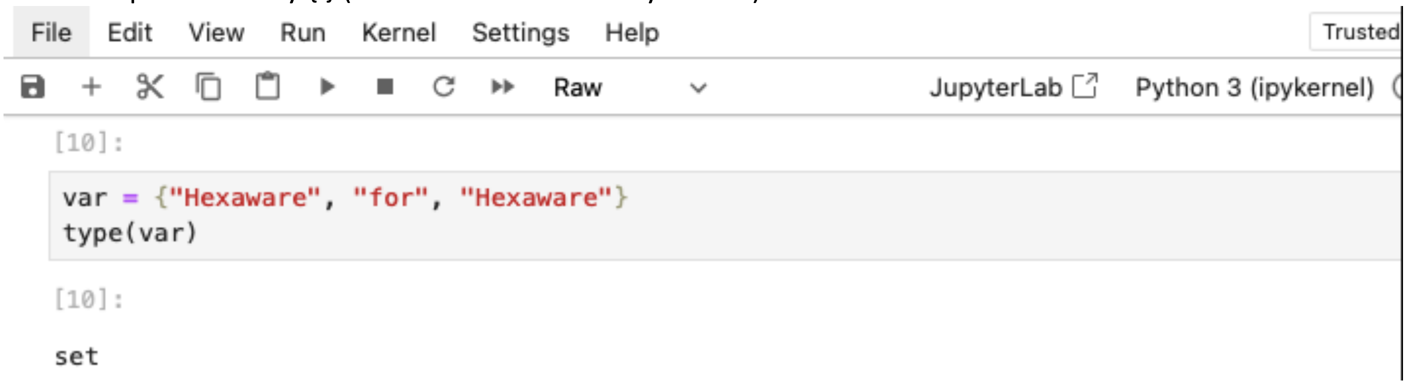
Assignment-9

Name-Ajay Chaudhary
Batch-Data Engineering(Batch-1)

SET

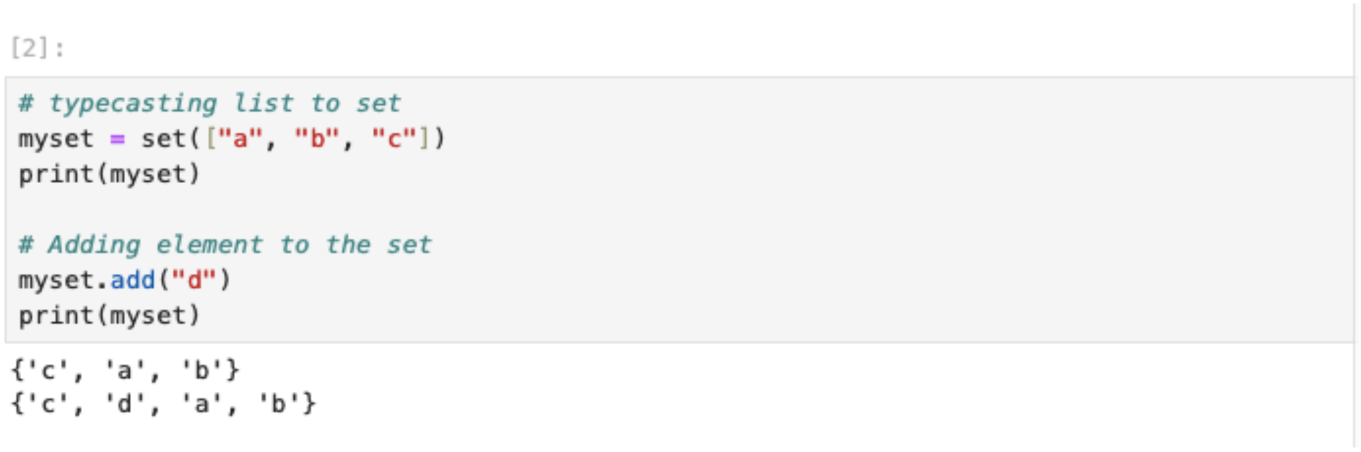
A Set in Python programming is an unordered collection data type that is iterable, mutable and has no duplicate elements.
Set are represented by { } (values enclosed in curly braces)



```
[10]:
var = {"Hexaware", "for", "Hexaware"}
type(var)
```

```
[10]:
set
```

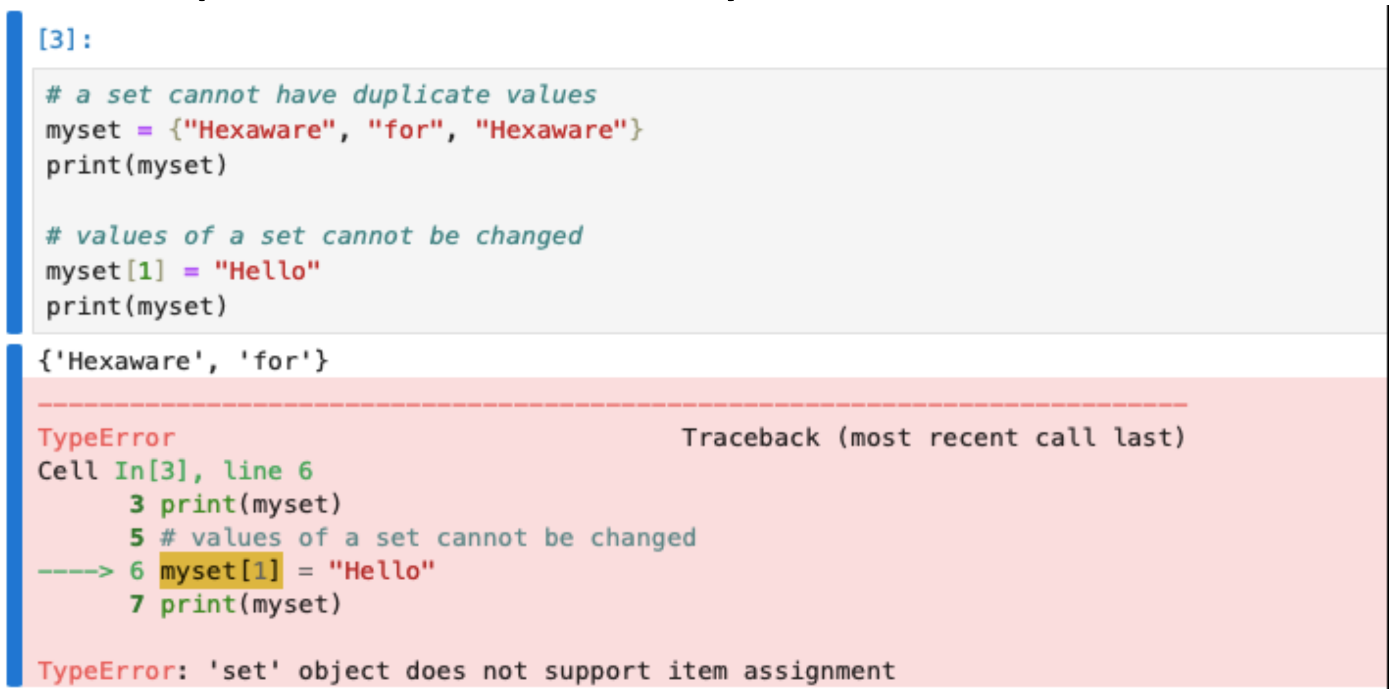
Type Casting with Python Set method



```
[2]:
# typecasting list to set
myset = set(["a", "b", "c"])
print(myset)

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

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

Check unique and Immutable with Python Set

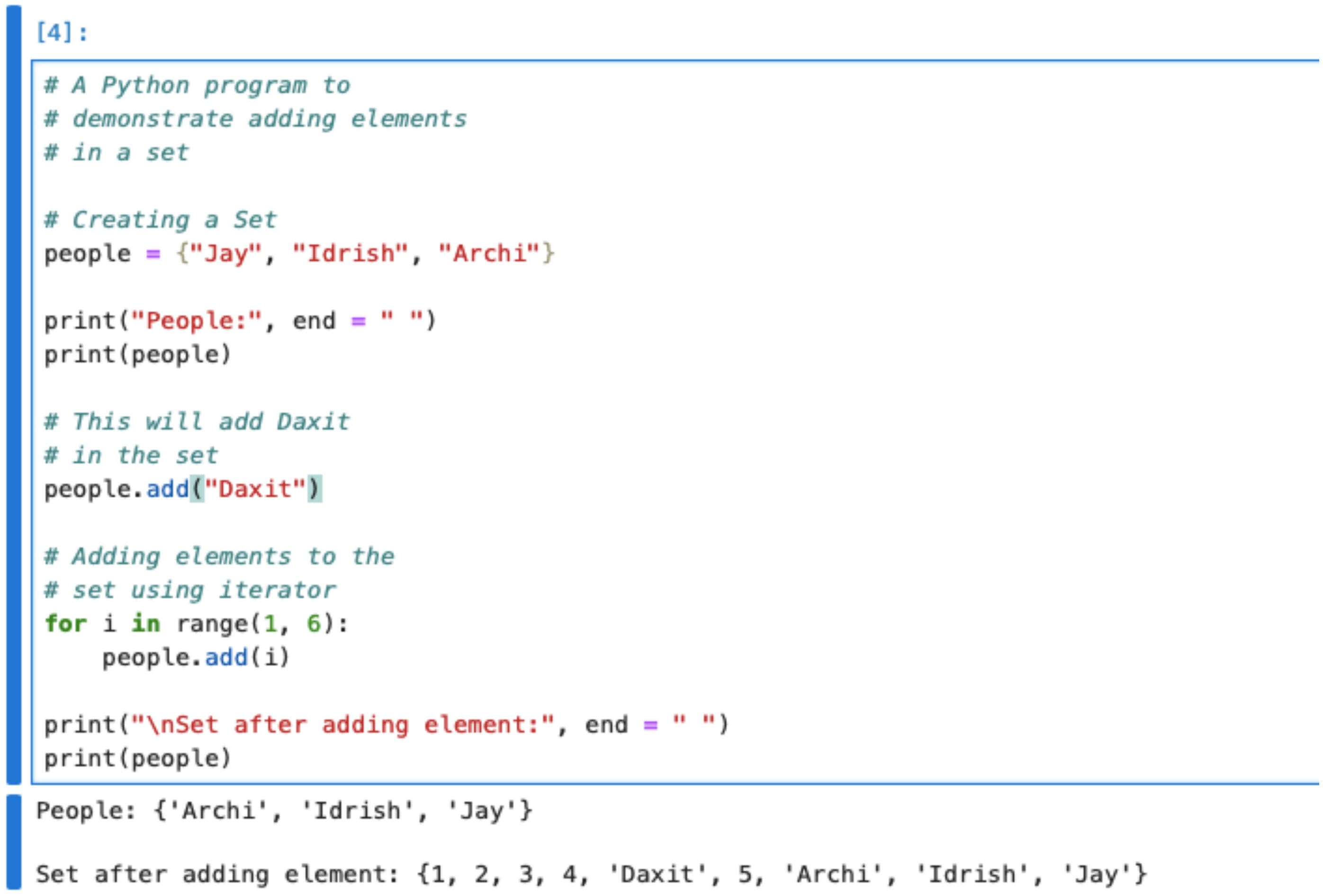


```
[3]:
```

```python
# a set cannot have duplicate values
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)
Cell In[3], line 6
      3 print(myset)
      5 # values of a set cannot be changed
----> 6 myset[1] = "Hello"
      7 print(myset)

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

Adding elements to Python Sets

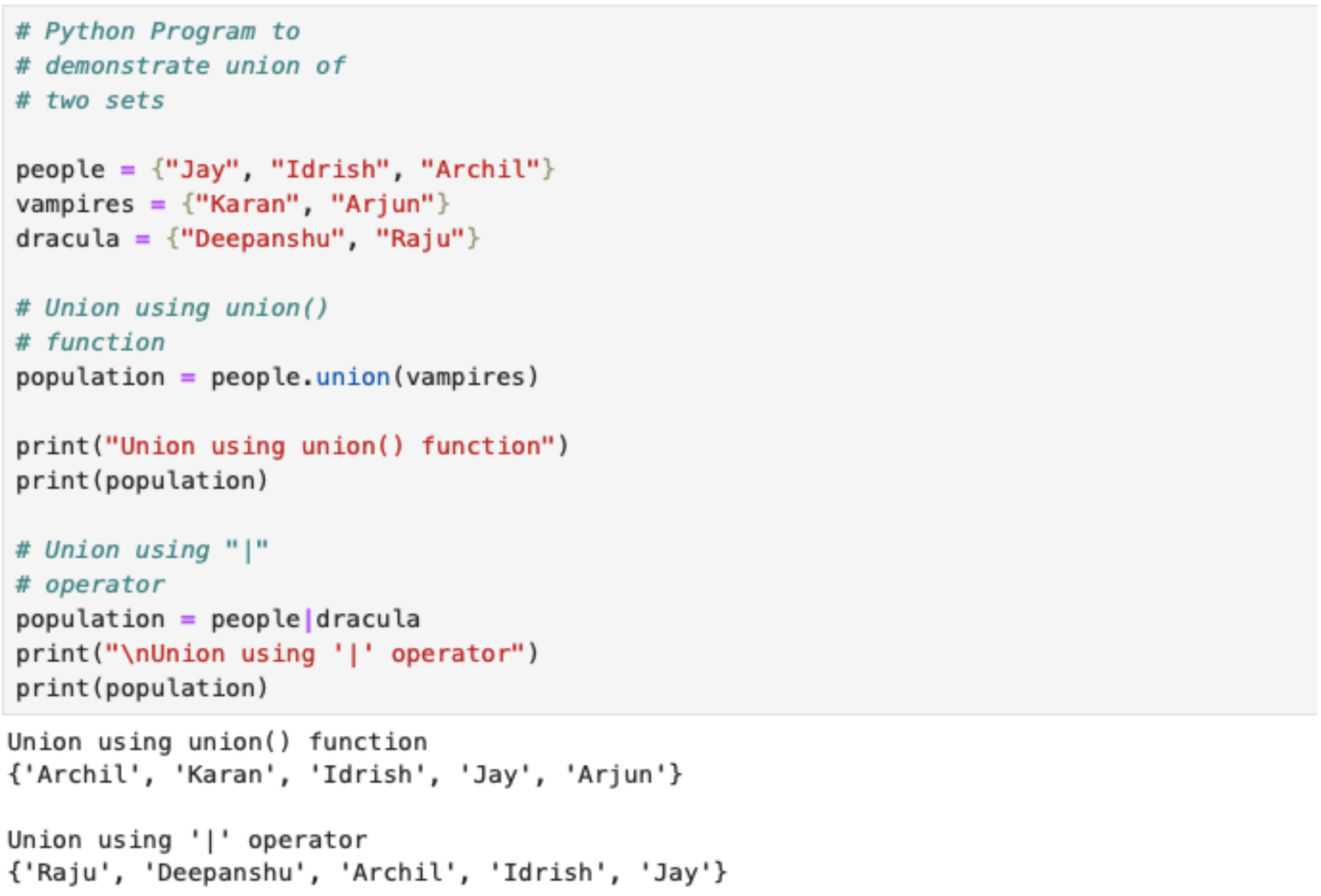


```
[4]:
```

```python
# A Python program to
# demonstrate adding elements
# in a set

# Creating a Set
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: {'Archi', 'Idrish', 'Jay'}

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

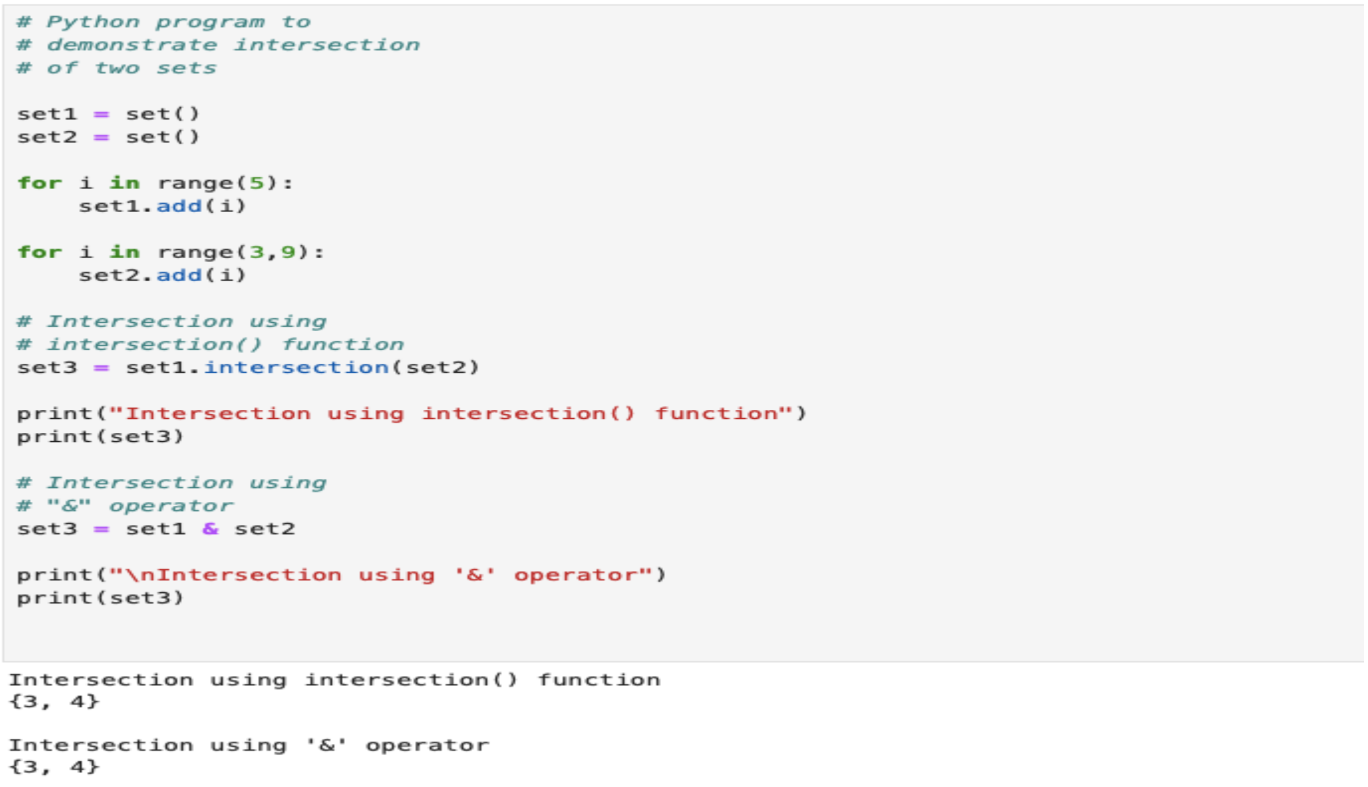
Union operations on Python sets



```
# 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', 'Karan', 'Idrish', 'Jay', 'Arjun'}

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

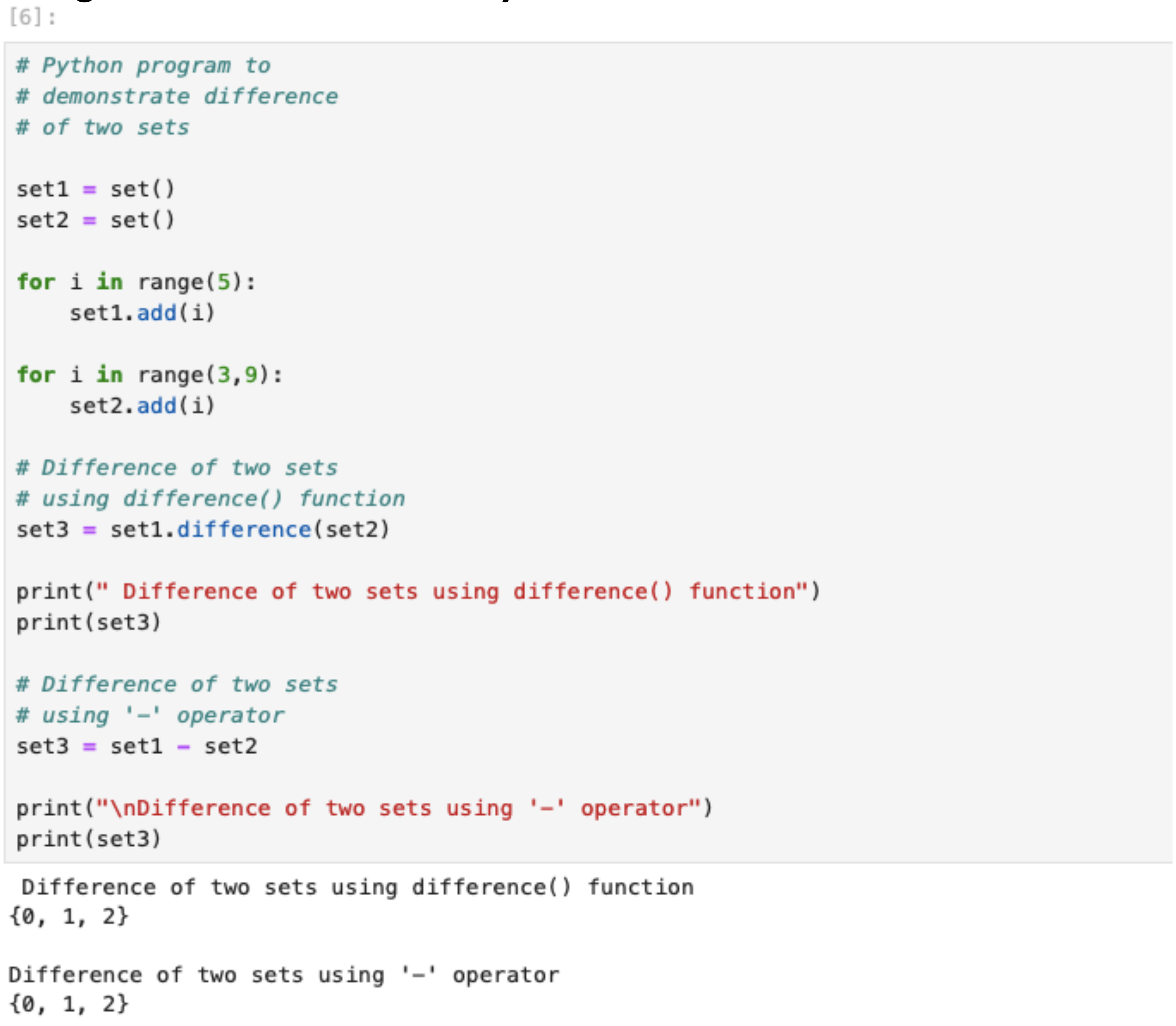
Intersection operation on Python Sets



```
# 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)
```

```
Intersection using intersection() function
{3, 4}

Intersection using '&' operator
{3, 4}
```

Finding Differences of Sets in Python



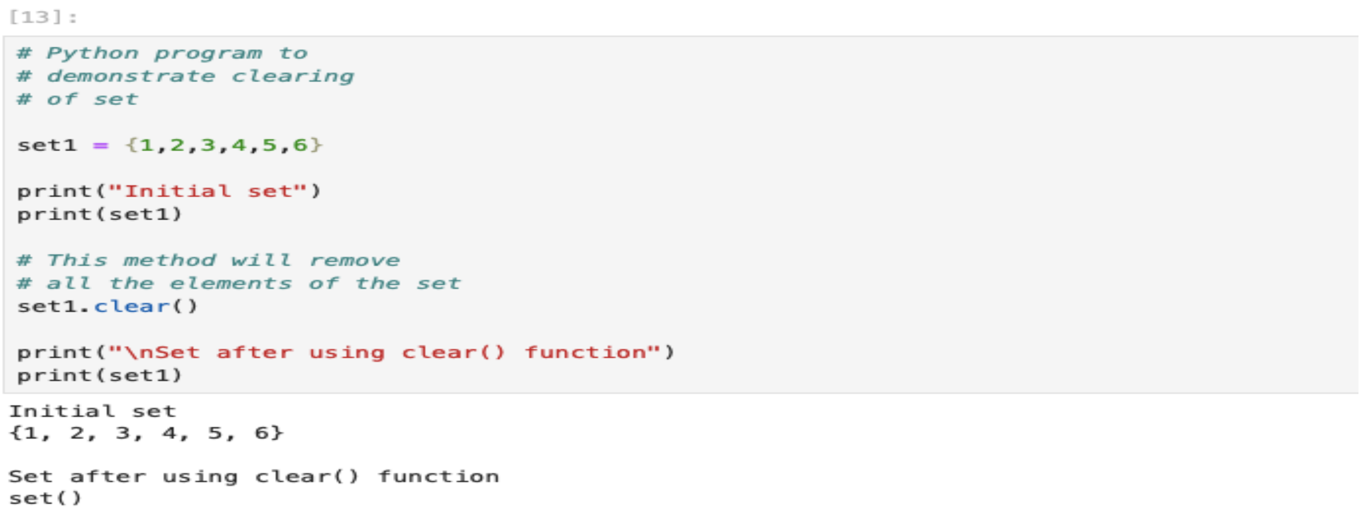
```
[6]:
```

```python
# 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
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.



```
[13]:
```

```python
# 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}

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.

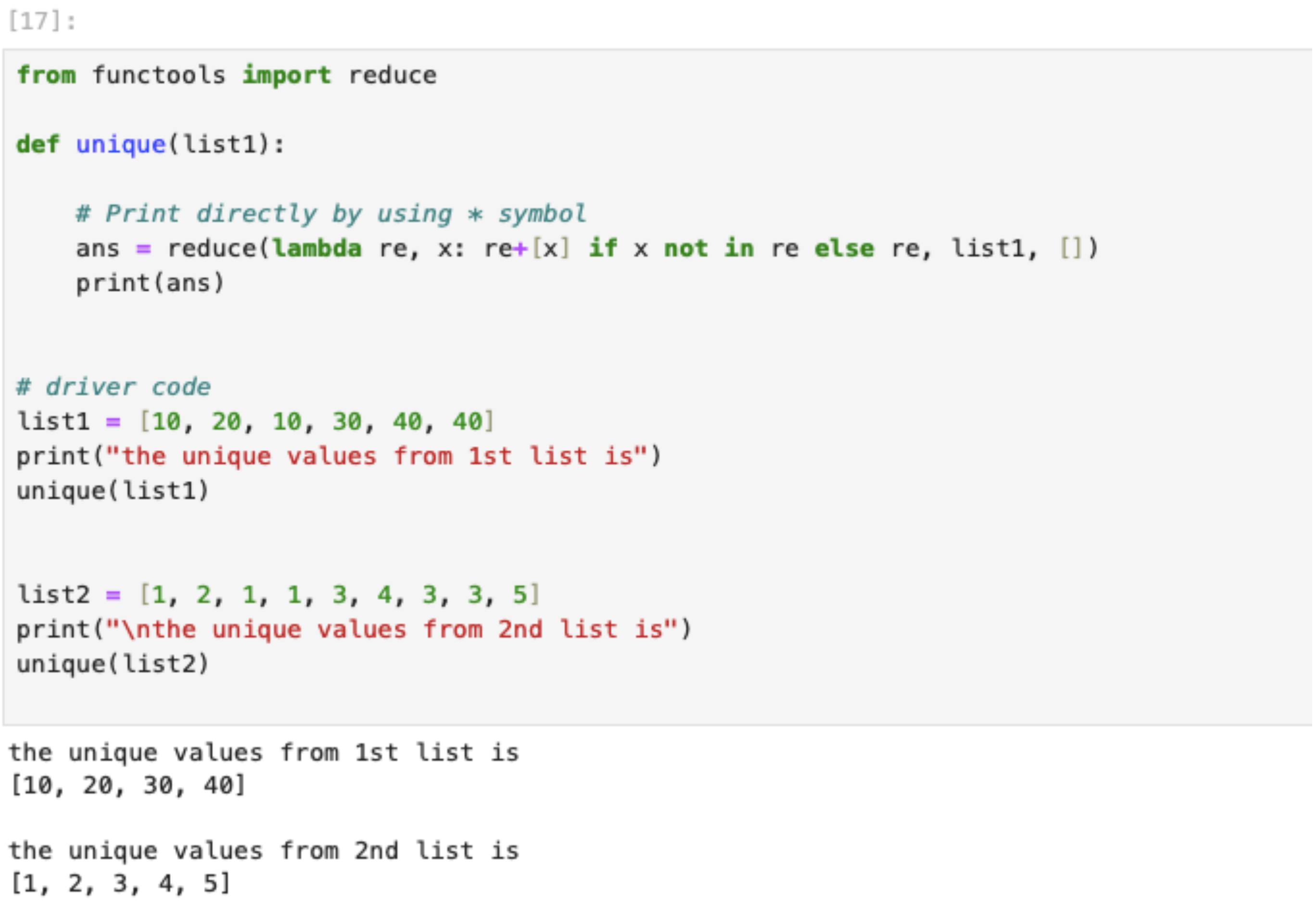
[16]:

```python
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)
```

```
the unique values from 1st list is
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



```
[17]:
from functools import reduce

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)


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

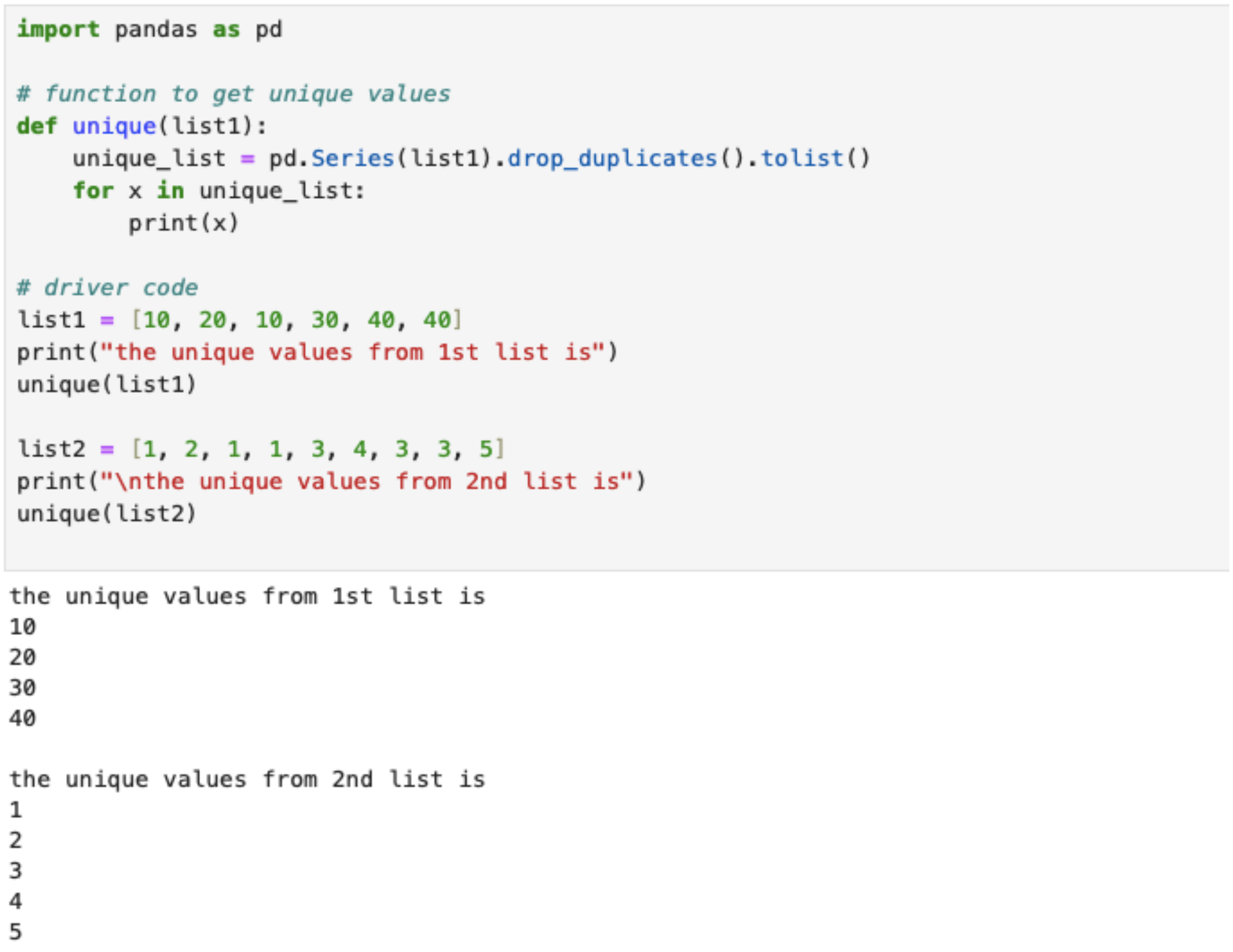
```
[18]:
```

```python
import operator as op
# 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
    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.

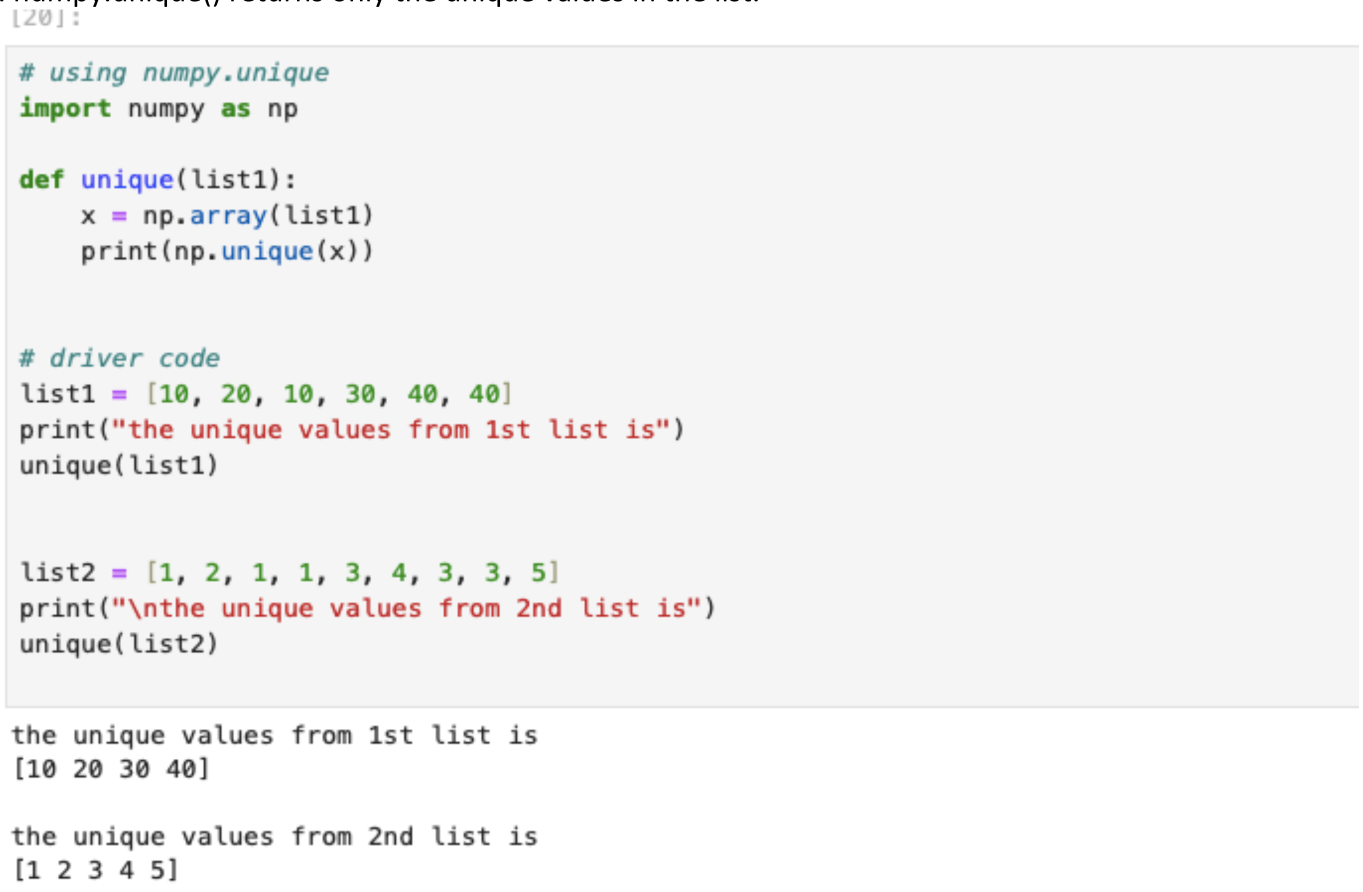


```python
import pandas as pd

# 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

Using Python’s import numpy, the unique elements in the array are also obtained. In the first step convert the list to x=numpy.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.

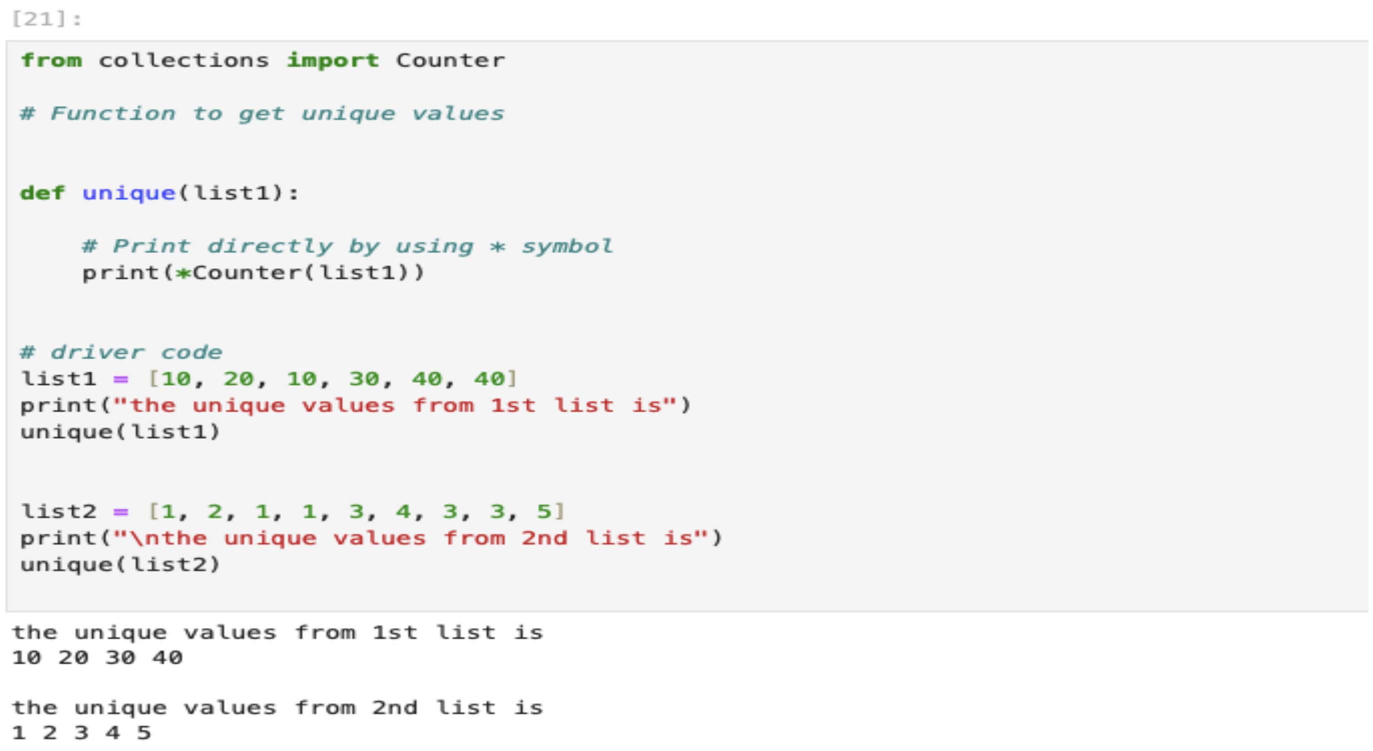


```
[20]:
# using numpy.unique
import numpy as np

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.



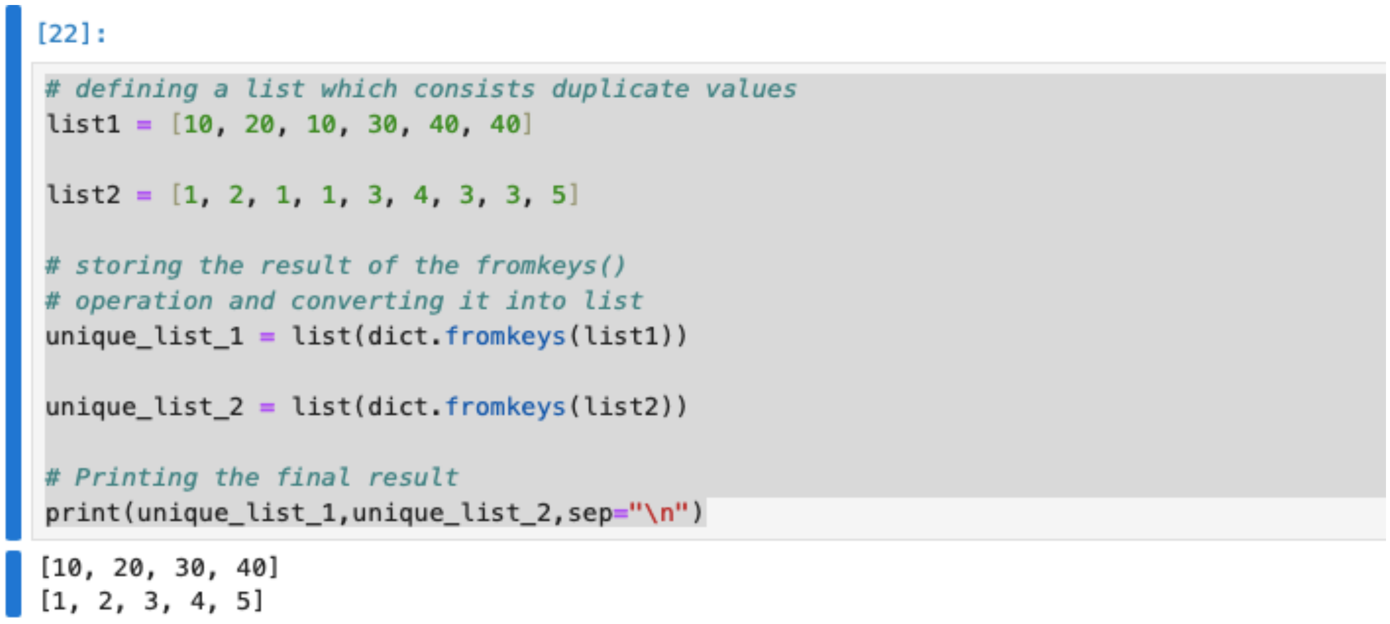
```
[21]:
from collections import Counter

# 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**.**

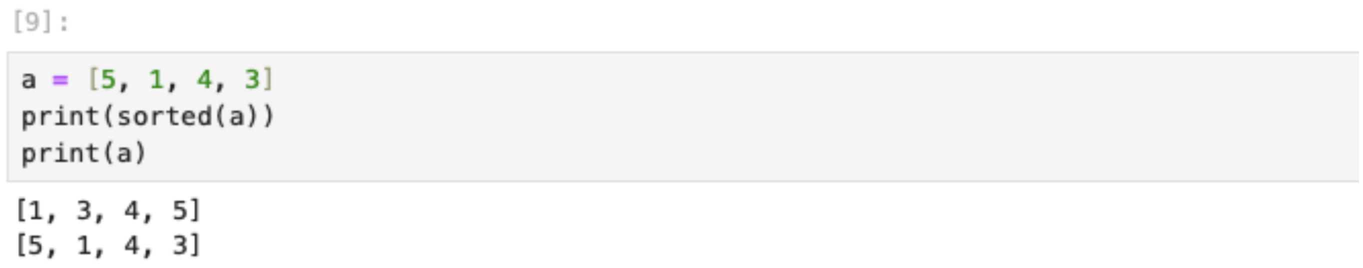


```
[22]:
# defining a list which consists duplicate values
list1 = [10, 20, 10, 30, 40, 40]

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]
```

Sorted() function

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.



```
[9]:
a = [5, 1, 4, 3]
print(sorted(a))
print(a)
```

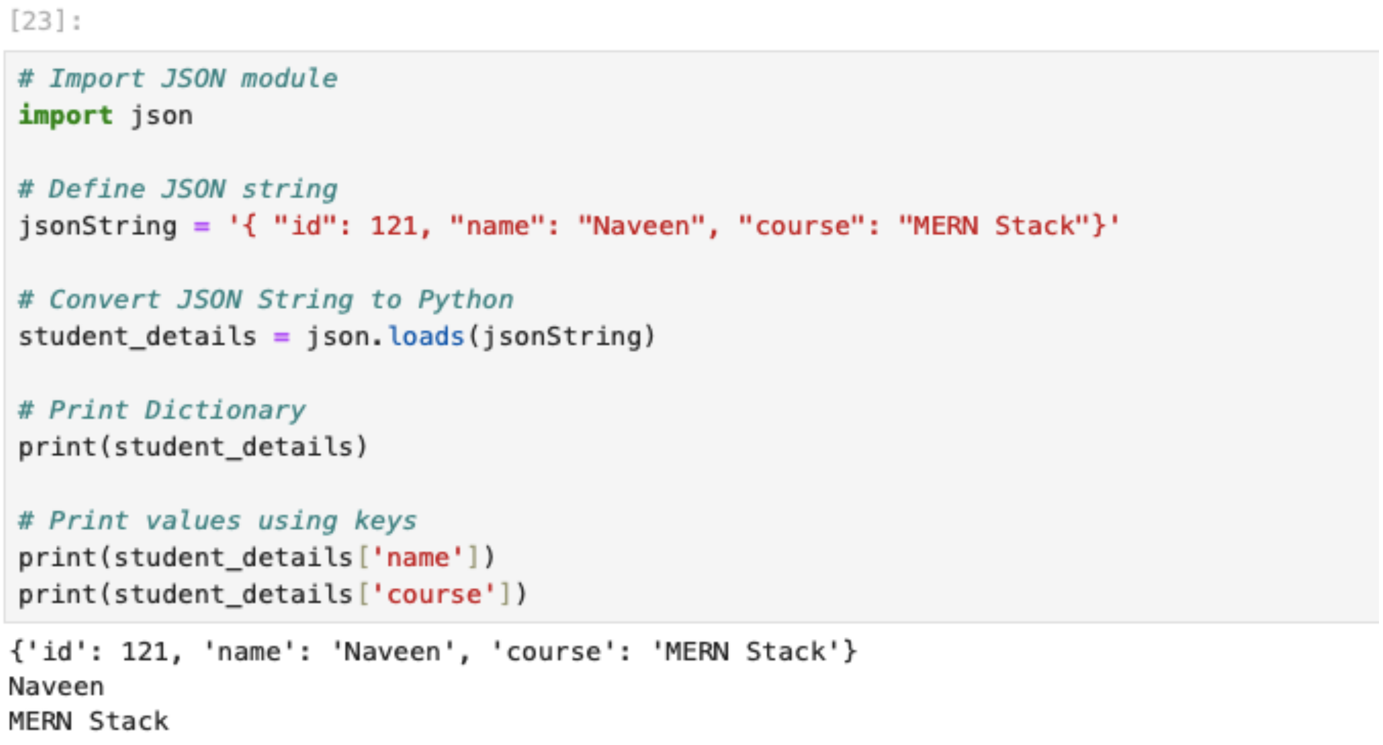
```
[1, 3, 4, 5]
[5, 1, 4, 3]
```

JSON Introduction

JSON stands for **J**ava**S**cript **O**bject **N**otation. 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.



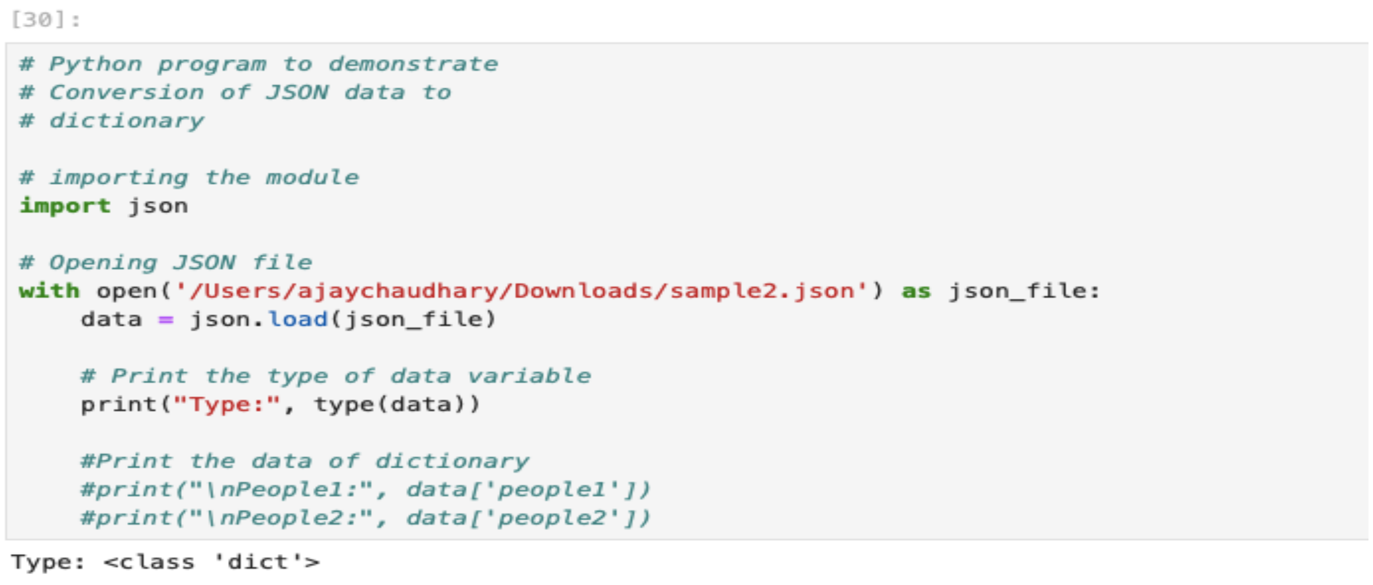
```
[23]:
```

```python
# Import JSON module
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() mehtod.
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.



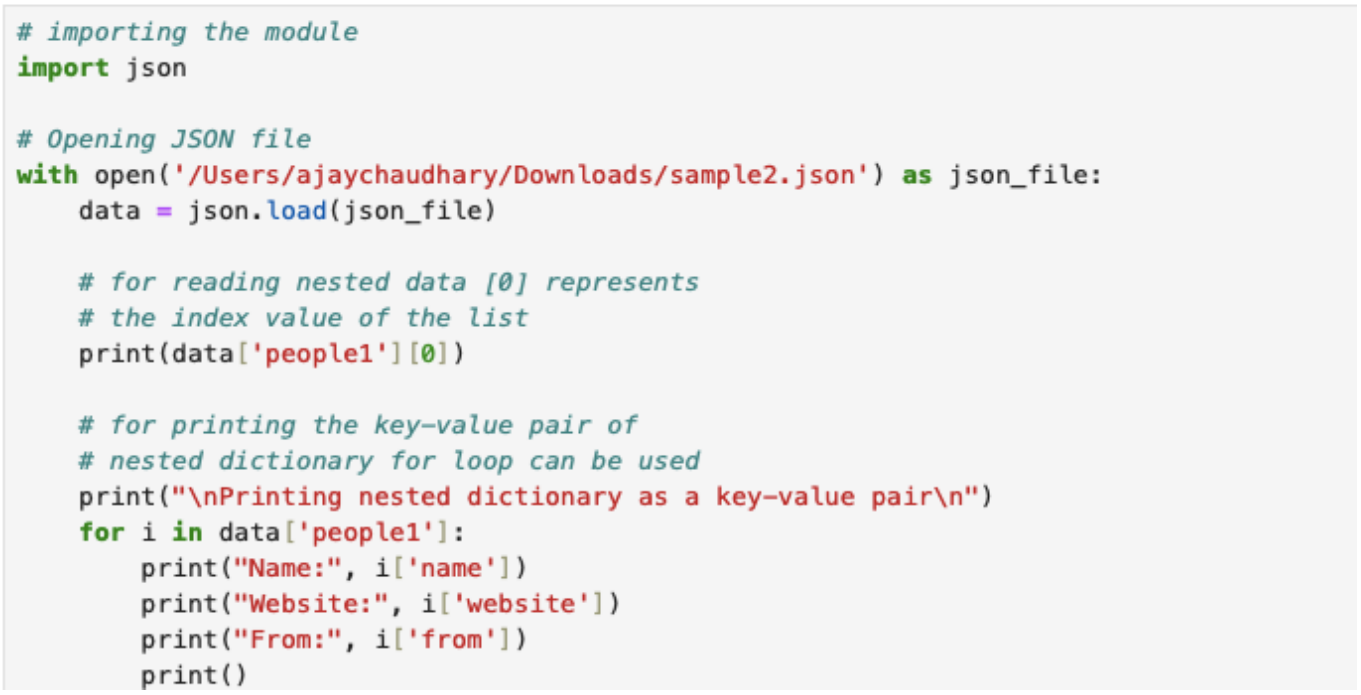
```
[30]:
```

```python
# Python program to demonstrate
# Conversion of JSON data to
# dictionary

# importing the module
import json

# Opening JSON file
with open('/Users/ajaychaudhary/Downloads/sample2.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'])
```

```
Type: <class 'dict'>
```

Convert Nested JSON Object to Dictionary

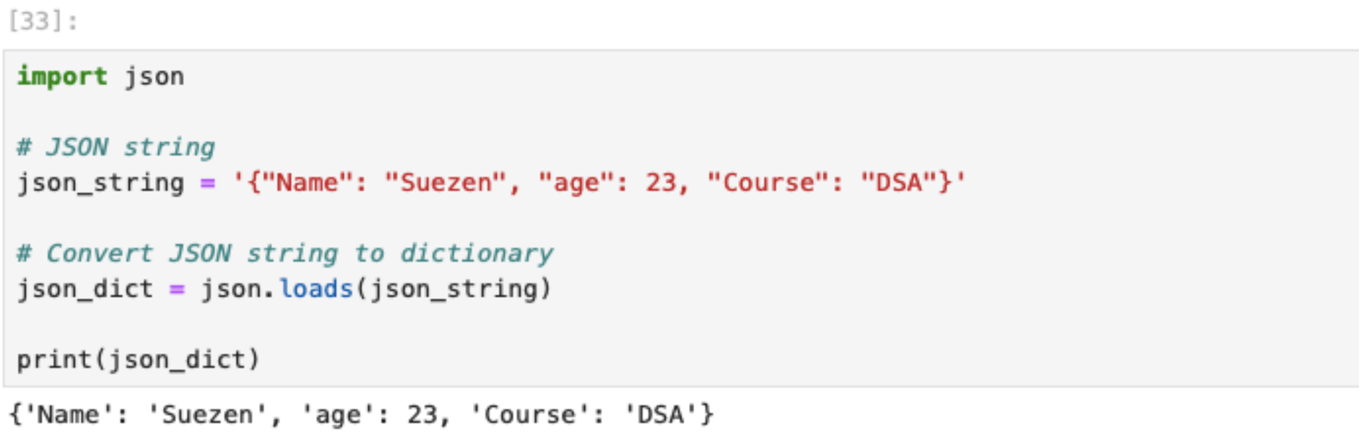
In this example, we will convert the nested JSON into a Python dictionary. For JSON data we will use the same JSON file used in the above example.



```
# importing the module
import json

# Opening JSON file
with open('/Users/ajaychaudhary/Downloads/sample2.json') as json_file:
    data = json.load(json_file)

    # for reading nested data [0] represents
    # the index value of the list
    print(data['people1'][0])

    # for printing the key-value pair of
    # nested dictionary for loop can be used
    print("\nPrinting nested dictionary as a key-value pair\n")
    for i in data['people1']:
        print("Name:", i['name'])
        print("Website:", i['website'])
        print("From:", i['from'])
        print()
```

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.



```
[33]:
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 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’ .

```
[34]:
```

```
# Python program to convert JSON to Python
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
```

Read, Write and Parse JSON using Python

Python read JSON file

Let’s suppose we have a JSON file that looks like this.
Here, we have used the open() function to read the JSON file. Then, the file is parsed using json.load() method which gives us a dictionary named data.

```
[35]:
```

```
import json

# Opening JSON file
f = open('data.json',)

# returns JSON object as
# a dictionary
data = json.load(f)

# Iterating through the json
# list
for i in data['emp_details']:
    print(i)

# Closing file
f.close()
```

Convert Python Dict to JSON

In the below code, we are converting a Python dictionary to a JSON object using json.dumps() method of JSON module in Python.
We first import the JSON module and then make a small dictionary with some key-value pairs and then passed it into json.dumps() method with ‘indent=4’ to convert this Python dictionary into a JSON object.

[36]:

```python
# Python program to convert
# Python to JSON
import json

# Data to be written
dictionary = {
  "id": "04",
  "name": "sunil",
  "department": "HR"
}

# Serializing json
json_object = json.dumps(dictionary, indent = 4)
print(json_object)
```

```
{
    "id": "04",
    "name": "sunil",
    "department": "HR"
}
```

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.

```python
# Python program to write JSON
# to a file
import json

# Data to be written
dictionary ={
    "name" : "sathiyajith",
    "rollno" : 56,
    "cgpa" : 8.6,
    "phonenumber" : "9976770500"
}
with open("sample.json", "w") as outfile:
    json.dump(dictionary, outfile)
```

Python Pretty Print 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.

[39]:

```python
# Python program to convert JSON to Python
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"
}
```