Python JSON

Python has a built in package called json, which is used to work with json data.

dumps(data) - This is used to convert python object into json string
To use the json package First we have to import it.

```
import json
python_data = {
    'name': 'Sulav Tamang',
    'roll': 101,
}
json_data = json.dumps(python_data)
print(json_data)

{"name": "Sulav Tamang", "roll": 101}  # Output
```

load(data) - This is used to parse json string.
Example:-

```
import json
json_data = {"name": "Sulav Tamang", "roll": 101}
parsed_data = json.loads(json_data)
print(parsed_data)

{'name': 'Sulav Tamang', 'roll': 101}  #output
```

Serializers

In Django REST Framework, serializers are responsible for converting complex data such as querysets and model instances to native Python datatypes (called serialization) that can then be easily rendered into JSON, XML or other content types which are understandable by Front End.

Serializers are also responsible for deserialization which means it allows parsed data to be converted back into complex types, after first validating the incoming data.

Serializer Class

A serializer class is very similar to a Django Form and ModelForm class, and includes similar validation flags on the various fields, such as required, max_length and default.

DRF provides a Serializer class which gives you a powerful, generic way to control the output of your responses, as well as a ModelSerializer class which provides a useful shortcut for creating serializers that deal with model instances and querysets.

How to Create Serializer class

- Create a separate serializers.py file to write all serializers.

```
from rest_framework import serializers
from .models import Student

class StudentSerializer(serializers.Serializer):
    name = serializers.CharField(max_length=100)
    roll = serializers.CharField()
    city = serializers.CharField(max_length=100)
```

Serialization

The process of converting complex data such as querysets and model instances to native Python datatypes is called Serialization in DRF.

- Creating model instance stu

```
stu = Student.objects.get(id=1)
```

- Converting model instance stu to python Dict/Serializing Object

```
serializer = StudentSerializer(stu)
```

- Creating Query Set

```
stu = Student.objects.all()
```

- Converting Query Set stu to List of Python Dict/Serializing Query Set

```
serializer = StudentSerializer(stu, many=True)
```

serializer.data – This is the serialized data

JSONRenderer – This is used to render Serialized data into JSON which is understandable by Front End.

```
from rest_framework.renderers import JSONRenderer
""" Renderer the Data into Json """
json_data = JSONRenderer().render(serializer.data)
```