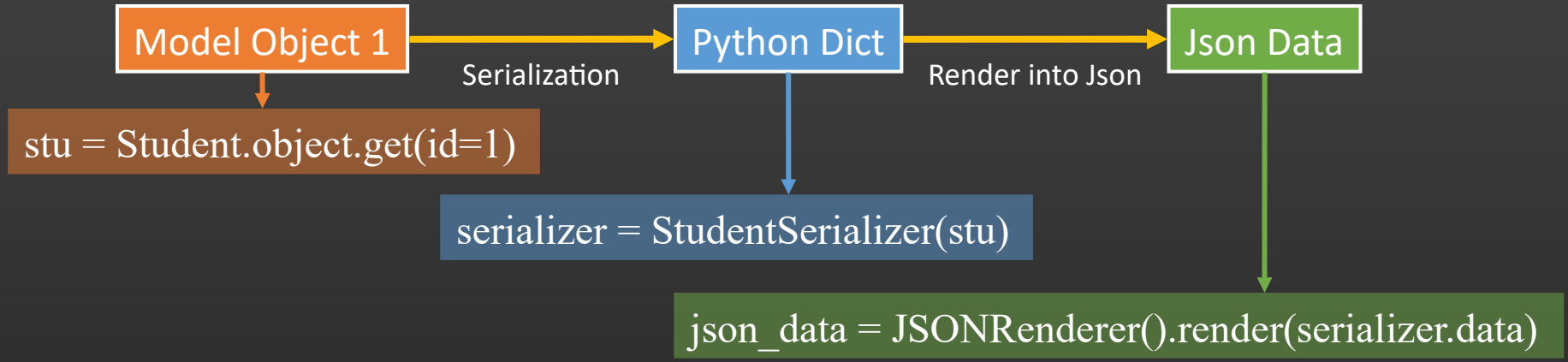
Serialization

ID	NAME	ROLL	CITY
1	Sonam	101	Ranchi
2	Rahul	102	Ranchi
3	Raj	103	Bokaro

Model Object 1

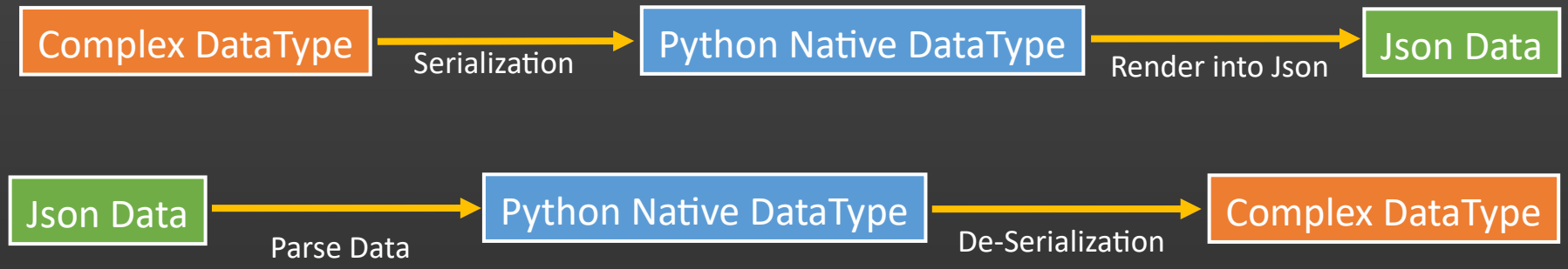
Model Object 2

Model Object 3



De-serialization

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.



BytesIO( )

A stream implementation using an in-memory bytes buffer. It inherits BufferedIOBase. The buffer is discarded when the close() method is called.

```
import io

stream = io.BytesIO(json_data)
```

JSONParser( )

This is used to parse json data to python native data type.

```
from rest_framework.parsers import JSONParser
parsed_data = JSONParser().parse(stream)
```

De-serialization

Deserialization allows parsed data to be converted back into complex types, after first validating the incoming data.

Creating Serializer Object

serializer = StudentSerializer(data = parsed_data)

Validated Data

serializer.is_valid()

serializer.validated_data

serializer.errors

serializer.validated_data

This is the Valid data.

serializer.validated_data

Create Data/Insert Data

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

   def create(self, validated_data):
       return Student.objects.create(**validated_data)
```

Update Data

```
from rest_framework import serializers
class StudentSerializer(serializers.Serializer):
    name = serializers.CharField(max_length=100)
    roll = serializers.IntegerField()
    city = serializers.CharField(max_length=100)
    def update(self, instance, validated_data):
        instance.name = validated_data.get('name', instance.name)
        instance.roll = validated_data.get('roll', instance.roll)
        instance.city = validated_data.get('city', instance.city)
        instance.save()
        return instance
```

New Data from user for updation

Old Data stored in Database

Complete Update Data

By default, serializers must be passed values for all required fields or they will raise validation errors.

Required All Data from Front End/Client

```
serializer = StudentSerializer(stu, data=pythondata)
if serializer.is_valid():
    serializer.save()
```

Partial Update Data

Partial Update - All Data not required

```
serializer = StudentSerializer(stu, data=pythondata, partial=True)
if serializer.is_valid():
    serializer.save()
```