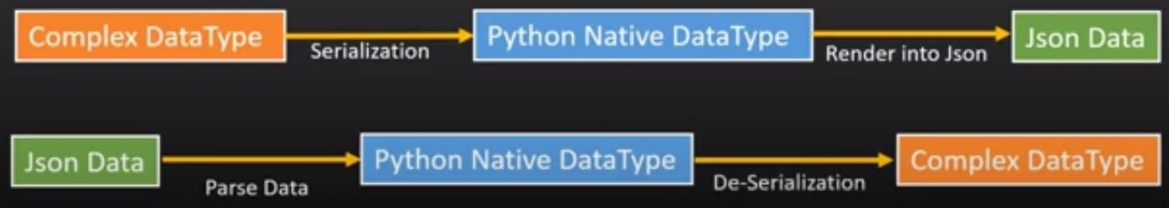
De-serialization

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, validate_data):
        return Student.objects.create(**validate_data)
```