1. Run the Django project and create a field object

class SerializerMetaclass(type):

def \_\_new\_\_(cls, name, bases, attrs):

data\_dict = {}

for k, v in list(attrs.items()): # {"v1":123,"v2":123,"v3":123}

if isinstance(v, int):

data\_dict[k] = attrs.pop(k)

attrs['\_declared\_fields'] = data\_dict

return super().\_\_new\_\_(cls, name, bases, attrs)

class BaseSerializer(object):

pass

class Serializer(BaseSerializer, metaclass=SerializerMetaclass):

pass

class ModelSerializer(Serializer):

pass

class XXXX(Serializer):

xx = serializers.CharField()

class UserSerializer(serializers.ModelSerializer,XXXX):

v1 = 123

id = serializers.IntegerField() # {max\_value:111, \_creation\_counter:0}

title = serializers.CharField() # {allow\_blank:Fakse, \_creation\_counter:1}

order = serializers.IntegerField() # {max\_value:111, \_creation\_counter:2}

class Meta:

model = models.Depart

fields = "\_\_all\_\_" # ["xx","xx","xx"]

2. Run the Django project to create a class (using metaclass)

# All field objects in the current serialized class (parent class + itself)

InfoSerializer.\_declared\_fields

3. User request arrives, database obtains data + serialization class

instance = models.UserInfo.objects.all().first()

ser = UserSerializer(instance=instance, many=False) # UserSerializer

queryset = models.UserInfo.objects.all()

ser = UserSerializer(instance=queryset, many=True) # Instantiate ListSerializer( child=obj = UserSerializer() )

4. Trigger serialization-current class ser.data

ser.data

5. Trigger serialization-ListSerializer ser.data

ser.data => loop + step 4 based