1. Why is a unicode function is required in models.py?

def \_\_unicode\_\_(self)

return sumid;

t's not. If you define a \_\_unicode\_\_() method, Django will call it when it needs to render an object in a context where a string representation is needed (e.g. in the model's admin pages).

The [documentation](http://docs.djangoproject.com/en/dev/ref/models/instances/#unicode) says:

The \_\_unicode\_\_() method is called whenever you call unicode() on an object. Since Django's database backends will return Unicode strings in your model's attributes, you would normally want to write a \_\_unicode\_\_() method for your model.

# 2. [differentiate null=True, blank=True in django](https://stackoverflow.com/questions/8609192/differentiate-null-true-blank-true-in-django)

When we add a database field in django we generally write models.CharField(max\_length=100, null=True, blank=True). The same is done with ForeignKey, DecimalField etc. What is the basic difference in having

1. null=True only
2. blank=True only
3. null=True, blank=True

in respect to different (CharField, ForeignKey, ManyToManyField, DateTimeField) fields. What are the advantages/disadvantages of using 1/2/3?

null=True sets NULL (versus NOT NULL) on the column in your DB. Blank values for Django field types such as DateTimeField or ForeignKey will be stored as NULL in the DB.

blank=True determines whether **the field will be required in forms**. This includes the admin and your own custom forms.

If blank=True then **the field will not be required, whereas if it's False the field cannot be blank.**

The combo of the two is so frequent because typically if you're going to allow a field to be blank in your form, you're going to also need your database to allow NULL values for that field. The exception is CharFields and TextFields, which in Django are never saved as NULL. Blank values are stored in the DB as an empty string ('').

A few examples:

models.DateTimeField(blank=True) # raises IntegrityError if blank

models.DateTimeField(null=True) # NULL allowed, but must be filled out in a form

Obviously those two options don't make logical sense to use (though, there might be a use case for null=True, blank=False if you want a field to always be required in forms, but optional when dealing with an object through something like the shell.)

models.CharField(blank=True) # No problem, blank is stored as ''

models.CharField(null=True) # NULL allowed, but will never be set as NULL

CHAR and TEXT types are never saved as NULL by Django, so null=True is unnecessary. However, you can manually set one of these fields to None to force set it as NULL. If you have a scenario where that might be necessary, you should still include null=True.