Apr 10

1.Explicitly by the programmer by using clean methods
Syntax for clean method:clean_fieldname(self)

-->In the Form class, for any field, if we define clean method, then at the time of submit this form, django will call this method automatically to perform validations. If clean method wont raise any error then only request will be processed.

- forms.py

```
    def clean_name(self):
  print('Validating name field')
  inputname = self.cleaned_data['name']
  if len(inputname) < 4:
     raise forms.ValidationError('The minimum number of characters in the name field should be 4')
  return inputname
    def clean_rollno(self):
  print('Validating rollno field')
  inputrollno = self.cleaned_data['rollno']
  return inputrollno
def clean_email(self):
  print('Validating email field')
  inputrollno = self.cleaned_data['email']
  return inputrollno
def clean_feedback(self):
  print('Validating feedback field')
  inputrollno = self.cleaned_data['feedback']
  return inputrollno
```

<u>Note:</u>
Django will call these field level clean methods automatically and we are not required to call these methods explicitly. The names are fixed because these are understandable by Django.

2).By using django inbuilt validators:
Django provides several inbuilt validators to perform very common validations. We can use directly and we are not responsible to implement those.
All inbuilt validators present in django.core module

- **forms.py**

```
from django.core import validators
class FeedBackForm(forms.Form):
   feedback = forms.CharField(widget=forms.Textarea,validators=
          [validators.MaxLengthValidator(40),validators.MinLengthValidator(10)
])
```

How to implement custom validators by using same validators parameter?

Ex:The name should starts with 's'

- **forms.py**

```
class FeedBackForm(forms.Form):
   def starts_with_s(value):
      print('starts_with_s function execution')
      if value[0].lower() != 's':
         raise forms.ValidationError('Name should be starts with s or S')

   name = forms.CharField(validators=[starts_with_s])
```

- **mail should contains @gmail.com**

```
mail = 'mahesh@gmail.com'
print(mail[-10:])

def gmail_validator(value):
      print('Checking for gmail validation')
      if value[-10:] != '@gmail.com':
         raise forms.ValidationError('Mail extension should be gmail')

email = forms.EmailField(validators=[gmail_validator])
```

- **Validation of total form directly by using single clean() method**

We are not required to write separate field level methods. Inside single clean method all validations we can perform.

```
        def clean(self):
      print('Total form validation.....')
      total_cleaned_data = super().clean()
      print('Validating Name')
      inputname = total_cleaned_data['name']
```

```
        if inputname[0].lower() != 's':
            raise forms.ValidationError('Name should be starts with s')
        print('Validating Rollno')
        inputrollno = total_cleaned_data['rollno']
        if inputrollno <= 0:
            raise forms.ValidationError('Rollno should be > 0')
        print('Validating email')
        inputemail = total_cleaned_data['email']
        if inputemail[-10:] != '@gmail.com':
            raise forms.ValidationError('Email extension should be gmail')
```

-->If we want to validate multiple field values together, then single clean() method is the best choice.

- **<u>How to check original pwd and re-entered pwd are same or not:</u>**

```
class FeedBackForm(forms.Form):
    password = forms.CharField(label='Enter Password',
widget=forms.PasswordInput)
    rpassword =
forms.CharField(label='Password(Again)',widget=forms.PasswordInput)

def clean(self):
        total_cleaned_data = super().clean()
        pwd = total_cleaned_data['password']
        rpwd = total_cleaned_data['rpassword']
        if pwd != rpwd:
            raise forms.ValidationError('Both passwords must be same.....')
```