Common Fields

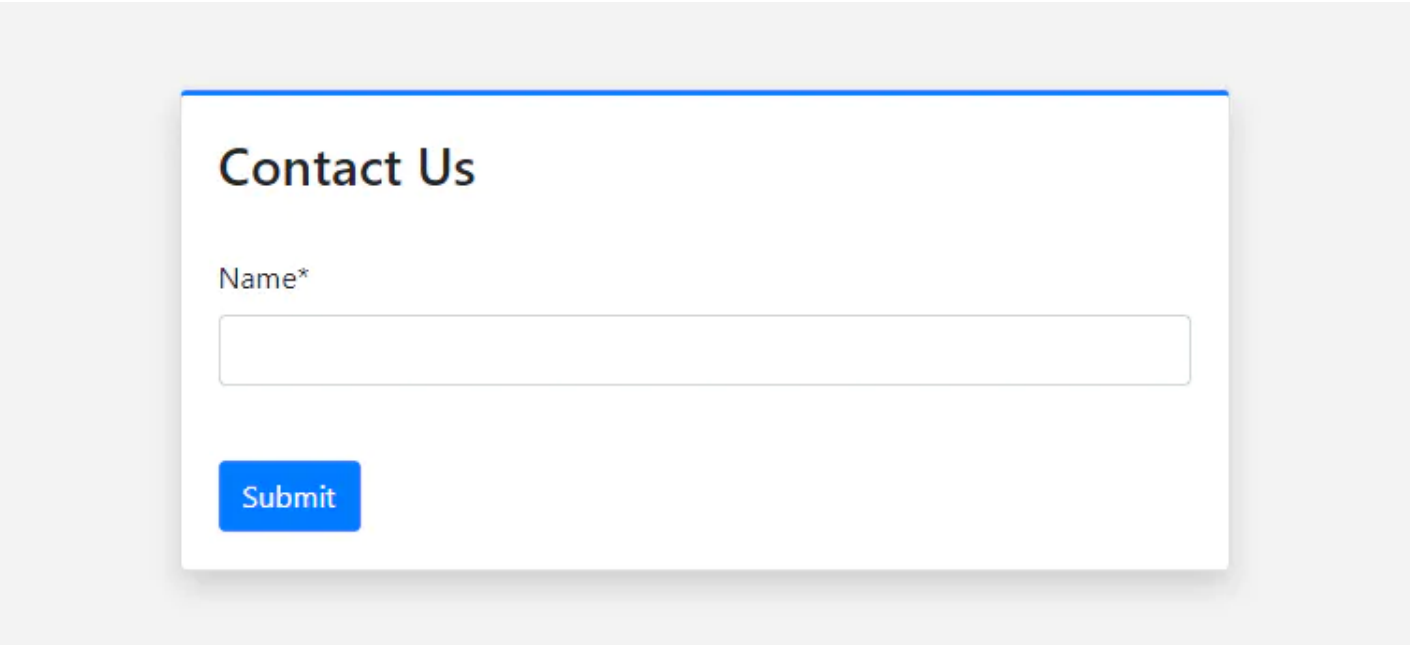
Let's start with some common fields, all of which have built-in widgets. These built-in widgets represent HTML elements.

CharField()

```
from django import forms

# Create your forms here.

class ExampleForm(forms.Form):
    name = forms.CharField()
```

CharField() is a Django form field that takes in a text input. It has the default widget TextInput, the equivalent of rendering the HTML code <input type="text" ...>.

This field works well for collecting one line inputs such as name or address.



CharField() with Textarea widget

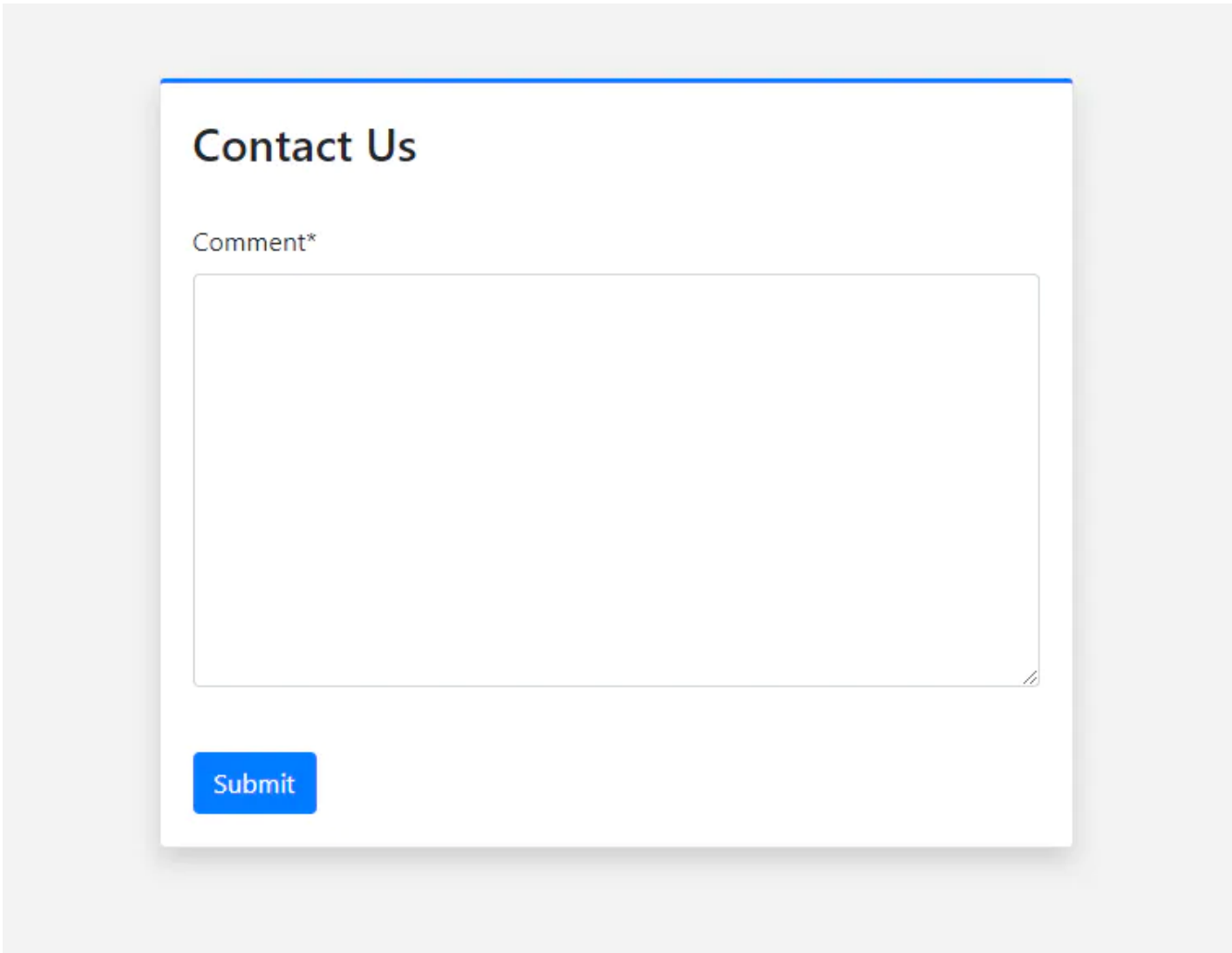
```
from django import forms

# Create your forms here.
```

```
class ExampleForm(forms.Form):
    comment = forms.CharField(widget=forms.Textarea)
```

If you are looking to add a multi-line input field to your form, add the Textarea widget to CharField().

The Textarea widget renders the field as <textarea>...</textarea>, a multi-line text input.

Create this multi-line text input if you are looking to have a comment or message field.



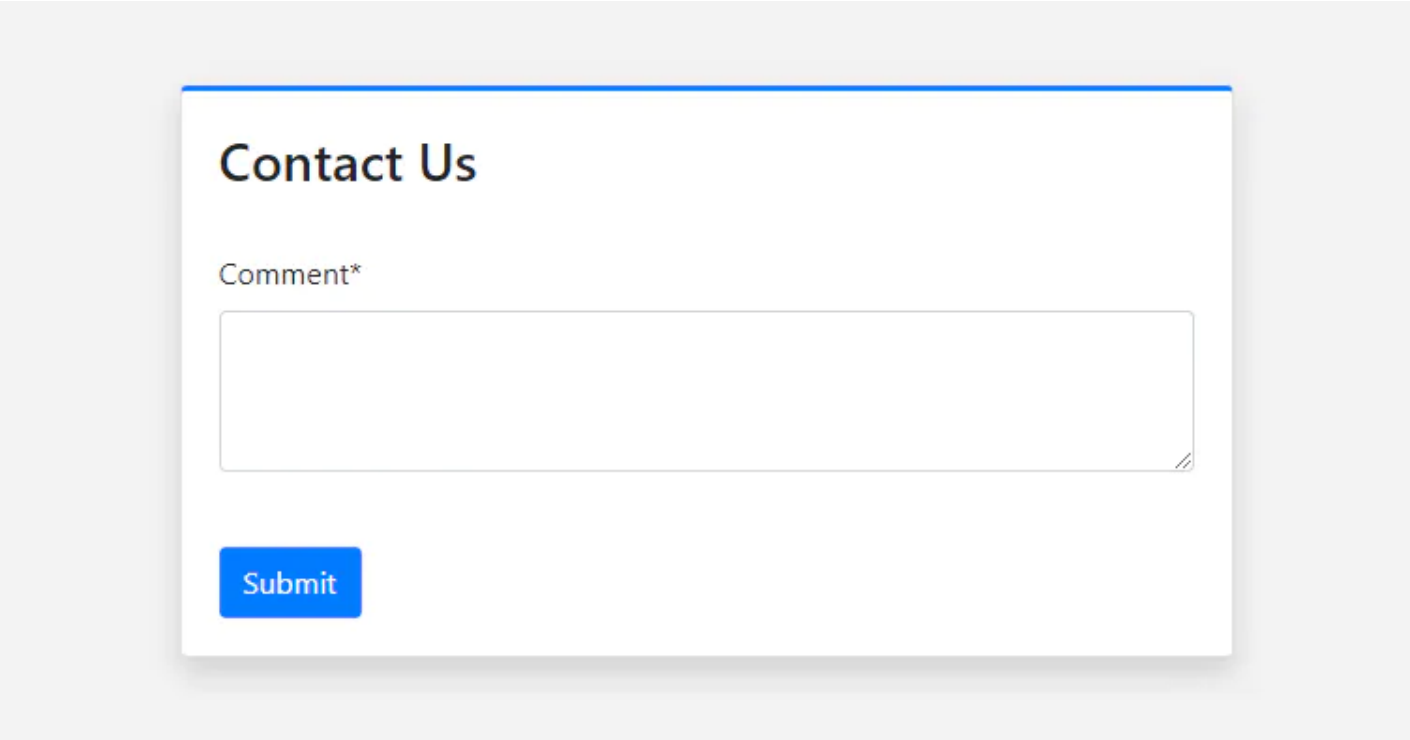
CharField() with Textarea widget attribute

```
from django import forms

# Create your forms here.
```

```
class ExampleForm(forms.Form):
    comment = forms.CharField(widget=forms.Textarea(attrs={'rows':3}))
```

Finally, if you wish to increase or decrease the height of the Textarea, specify the attribute 'rows' in the widget.

Please note the default number of rows is 10.

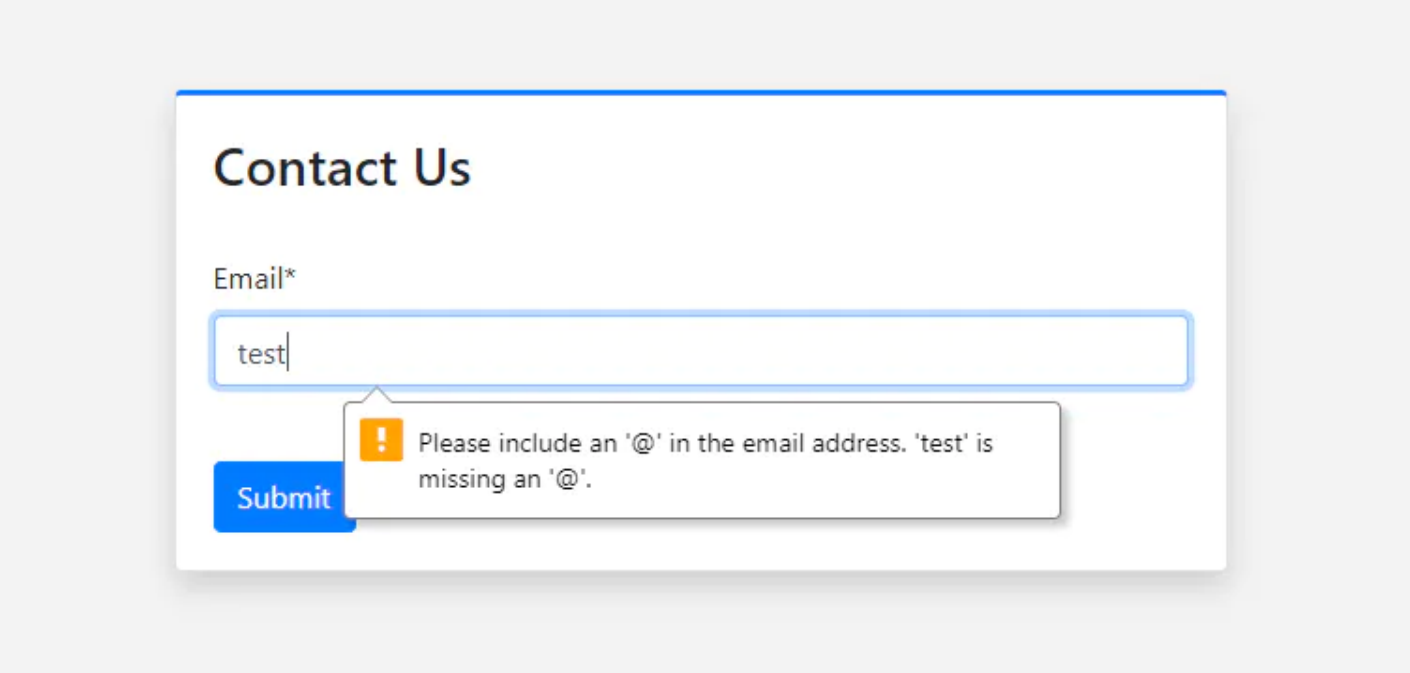


EmailField()

```
from django import forms

# Create your forms here.

class ExampleForm(forms.Form):
    email = forms.EmailField()
```

EmailField() has the default widget of EmailInput and renders as <input type="email" ...> in plain HTML.

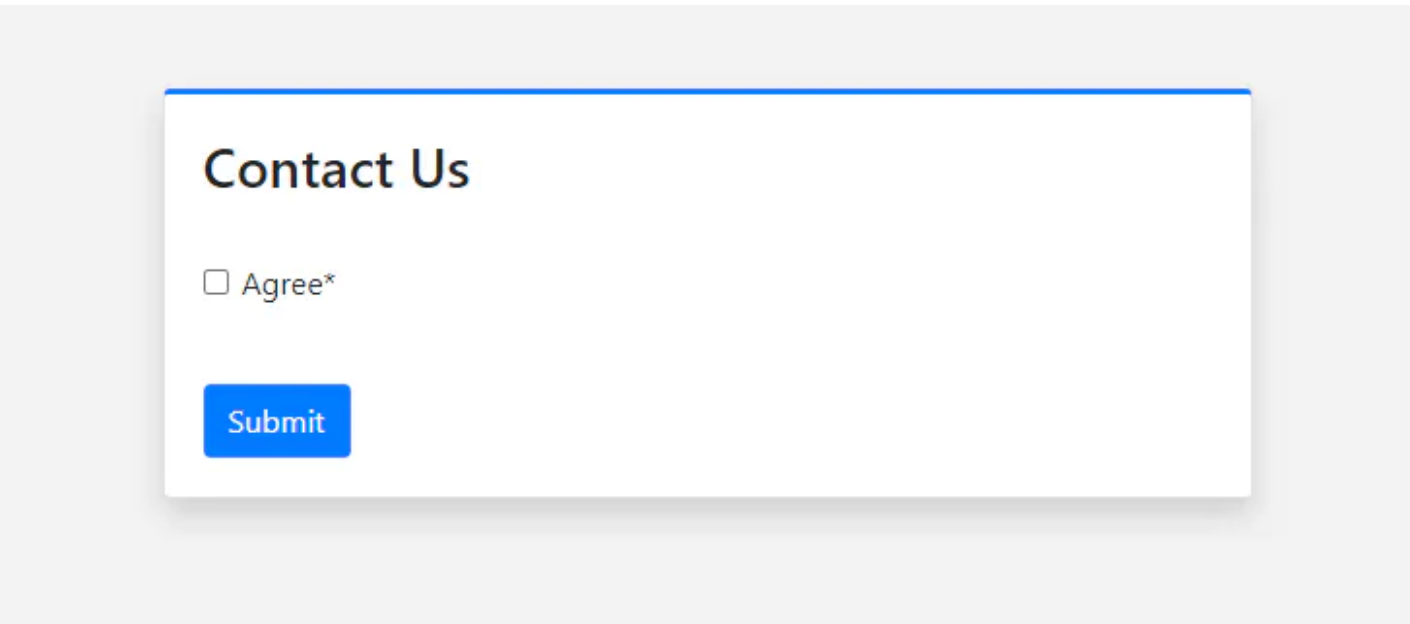
This field also uses the built-in Django validation EmailValidator that requires an @ symbol within the input for it to be considered valid.



BooleanField()

```
from django import forms

# Create your forms here.

class ExampleForm(forms.Form):
    agree = forms.BooleanField()
```

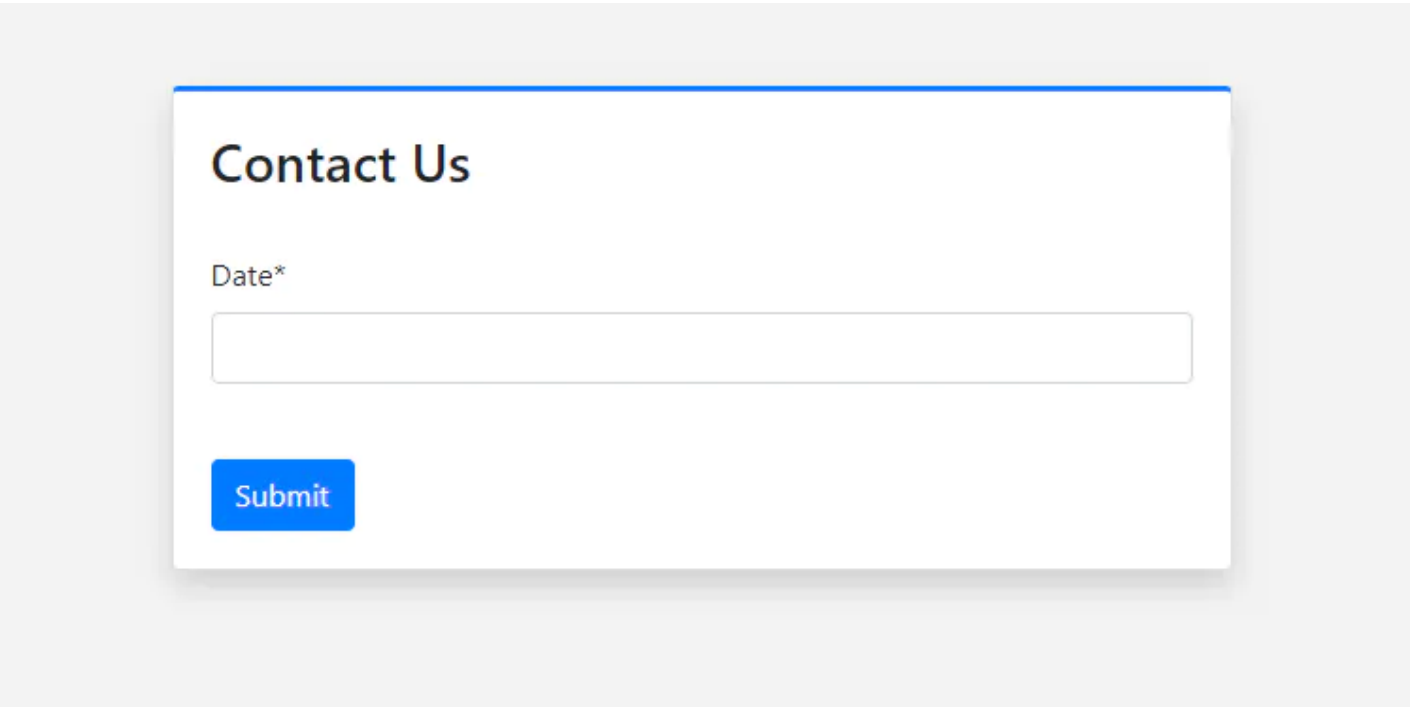
A BooleanField, as the name implies, takes in a boolean data type of either True or False. The default is False and renders an unclicked checkbox in the HTML template.



DateField()

```
from django import forms

# Create your forms here.

class ExampleForm(forms.Form):
    date = forms.DateField()
```

DateField() only accepts date formatted values such as 2020-07-30. The default widget is DateInput but it renders just like a CharField with <input type="text" ...>.

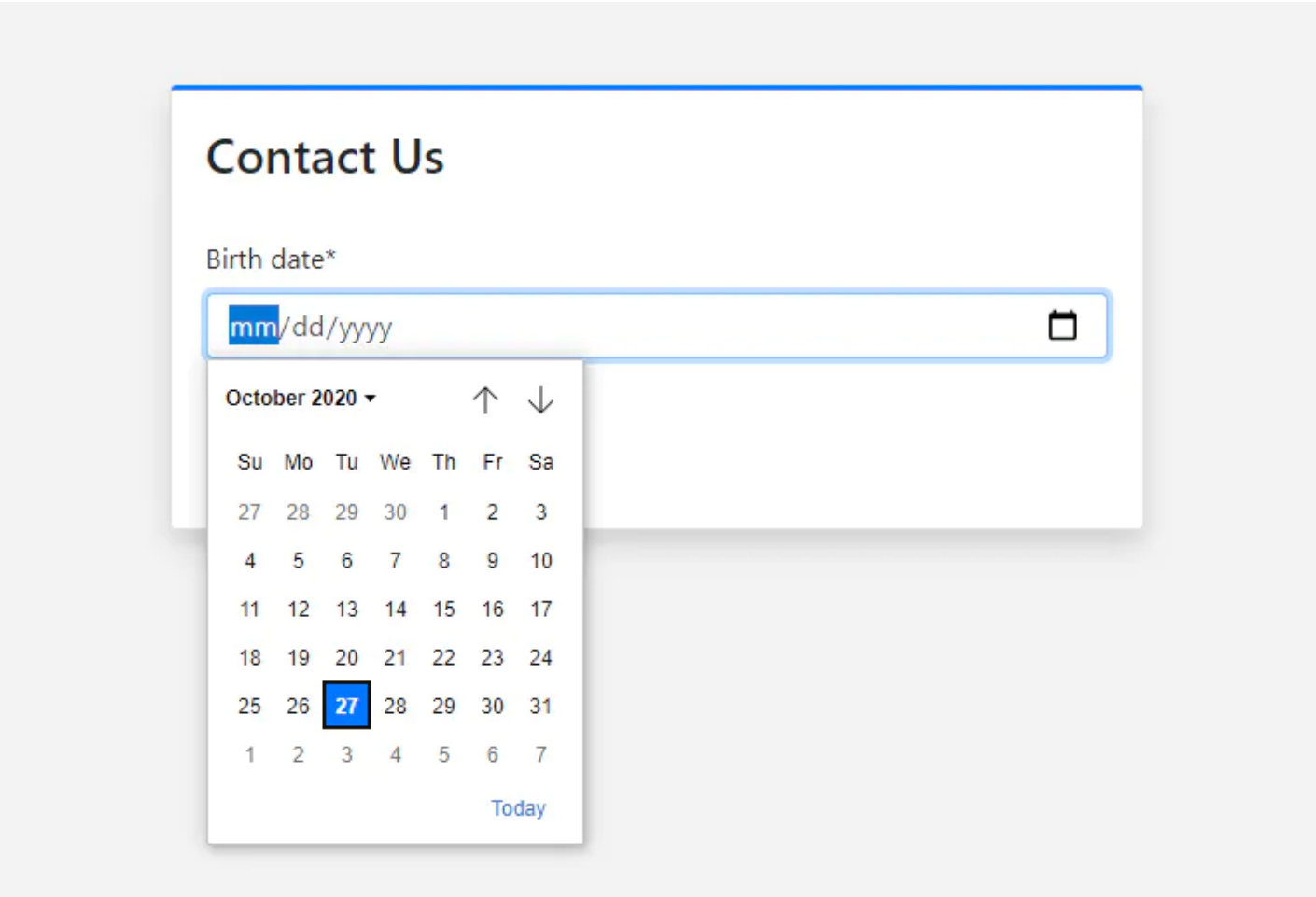


DateField() with NumberInput widget attribute

```
from django import forms
from django.forms.widgets import NumberInput

# Create your forms here.


class ExampleForm(forms.Form):
    birth_date = forms.DateField(widget=NumberInput(attrs={'type': 'date
```

If you are looking to add a calendar, import NumberInput at the top of the file then add the NumberInput widget with the attribute 'type' : 'date'.

This renders the equivalent of the HTML calendar <input type="date"> in plain HTML.

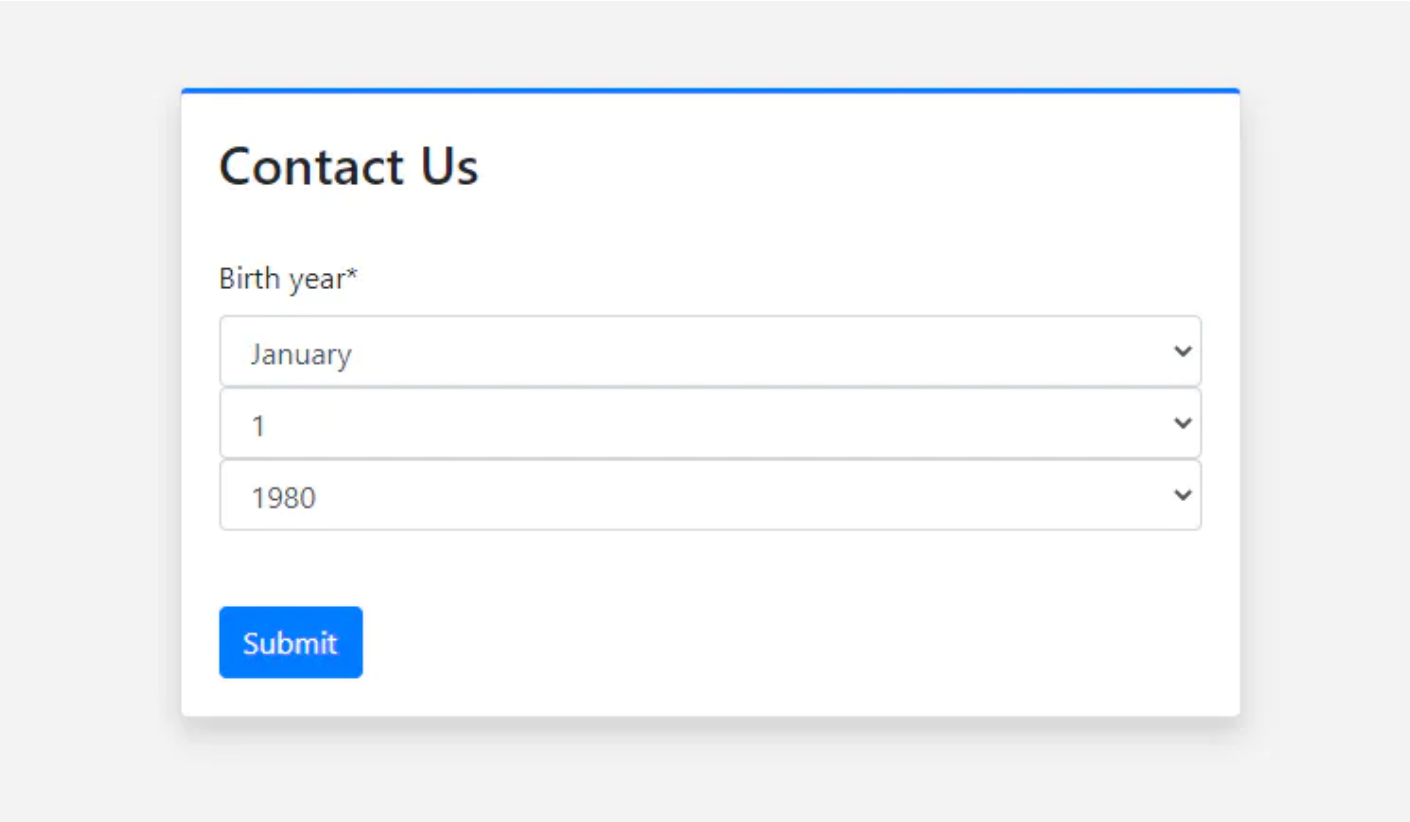


DateField() with SelectDateWidget widget

```
from django import forms


# Create your forms here.

BIRTH_YEAR_CHOICES = ['1980', '1981', '1982']

class ExampleForm(forms.Form):
        birth_year = forms.DateField(widget=forms.SelectDateWidget(years
```

Django also comes with the build-in SelectDateWidget which displays three drop-down menus for month, date, and year.

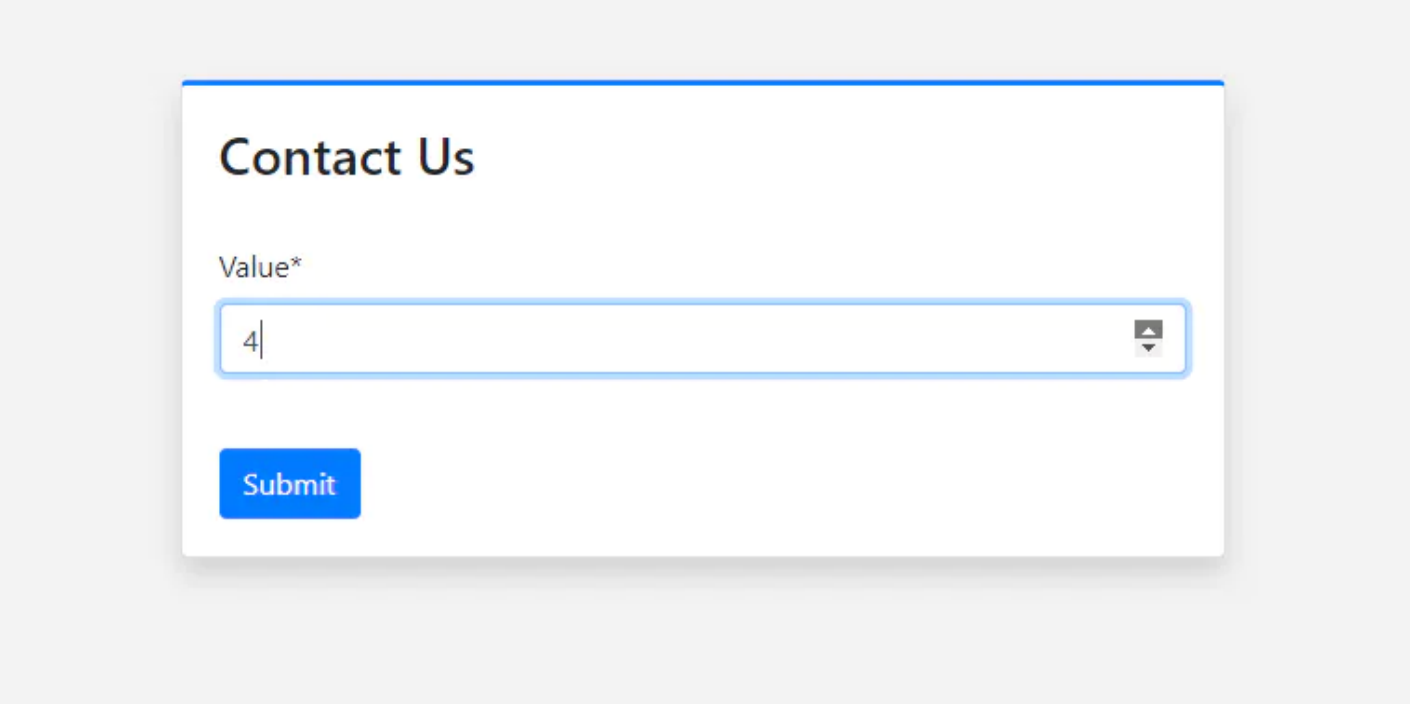
We will discuss more drop-down menus below.



DecimalField()

```
from django import forms

# Create your forms here.

class ExampleForm(forms.Form):
    value = forms.DecimalField()
```

If you are looking to collect number value inputs, use a DecimalField(). The default widget is NumberInput.



Core Arguments

Now on to the core arguments. These are non-specific arguments accepted by all fields.

required (Boolean)

```
email_address = forms.EmailField(
    required = False,
)
```

The required argument determines if the fields are required for form submission. It is a boolean (True or False only) and creates the asterisk mark next to the field.

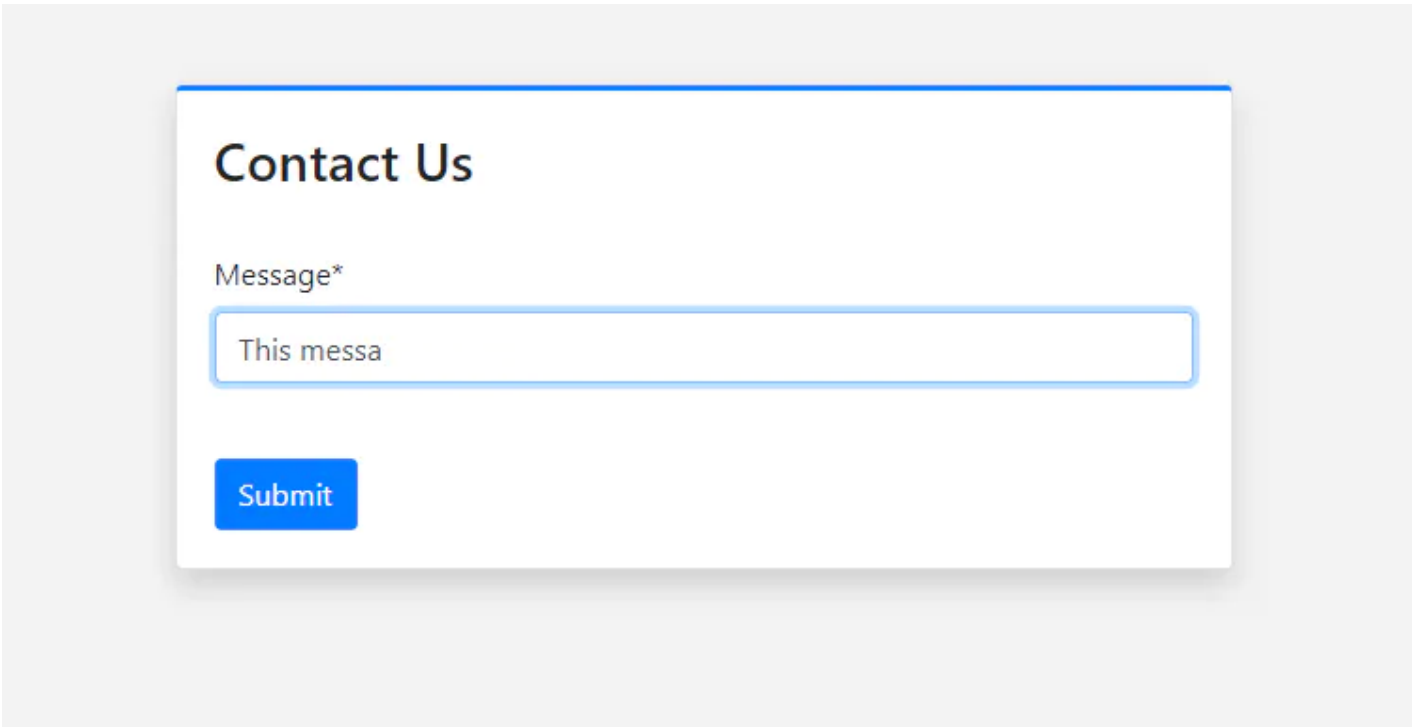
The argument is assigned to every field and is True by default.

max_length and min_length

```
message = forms.CharField(
    max_length = 10,
)
```

A maximum character length is assigned to the user input using max_length.

A minimum character length is assigned using min_length.

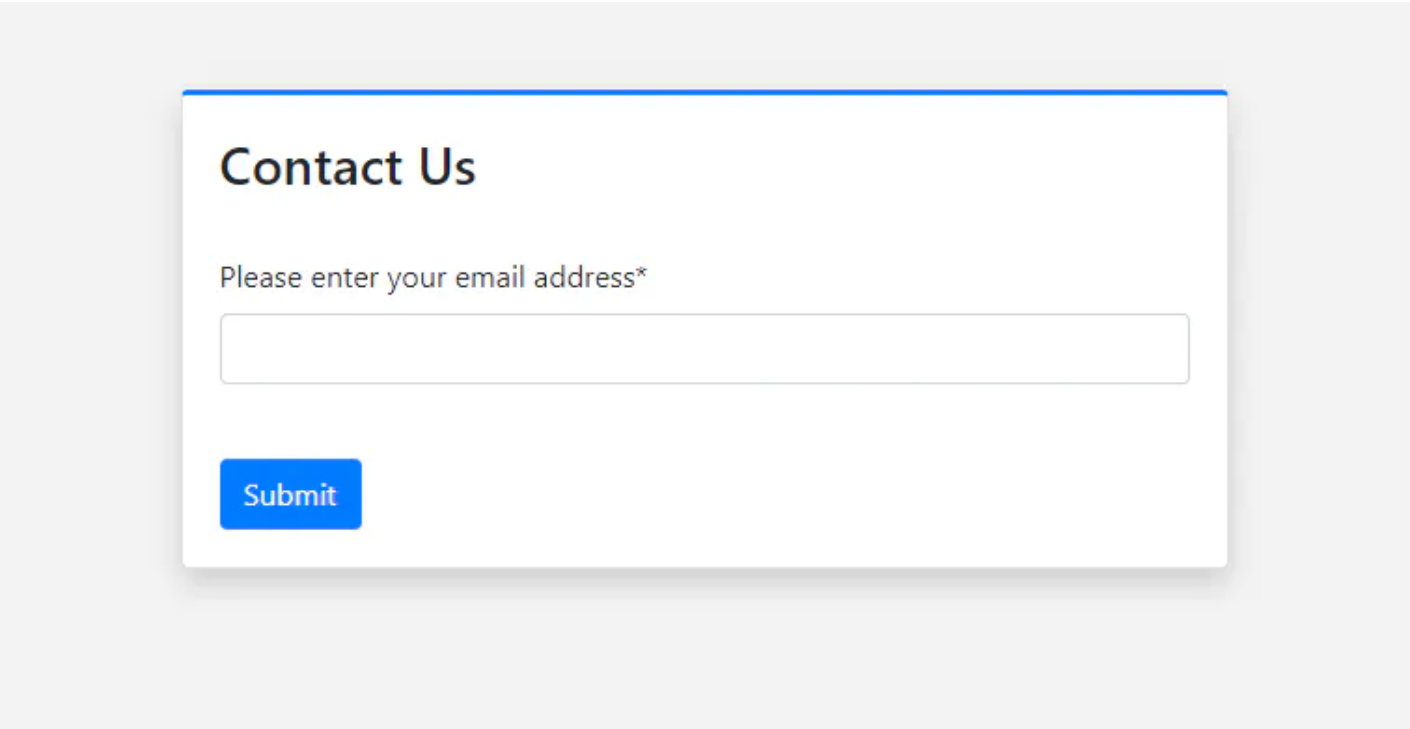


label (String)

```
email_address = forms.EmailField(
    label="Please enter your email address",
)
```

Django creates the field label by taking the designated field name, changing underscores to spaces, and the first letter to uppercase.

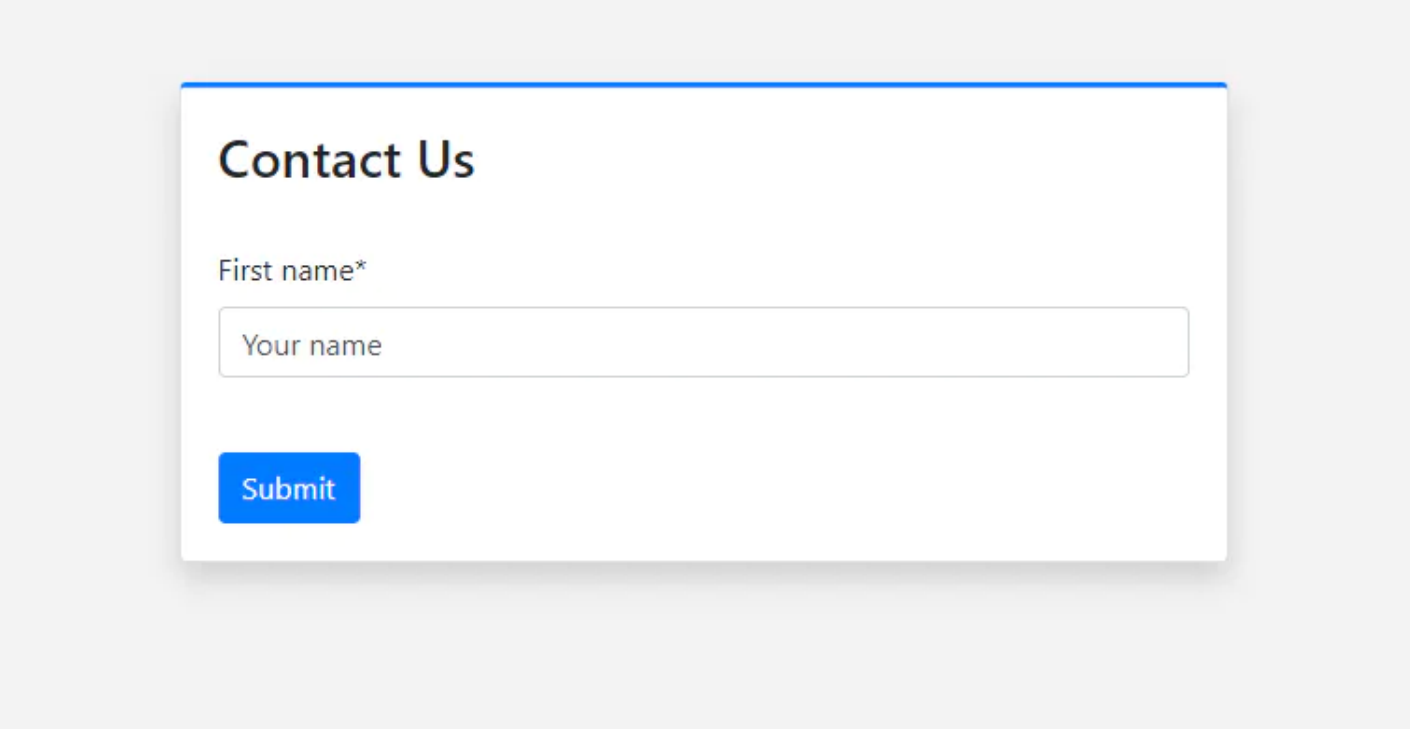
If you wish to create a custom label of a field, add the label argument.



initial (String) for CharField()

```
from django import forms

# Create your forms here.

class ExampleForm(forms.Form):
    first_name = forms.CharField(initial='Your name')
```

To add pre-loaded information to input, use the initial argument.



initial (Boolean) for BooleanField()

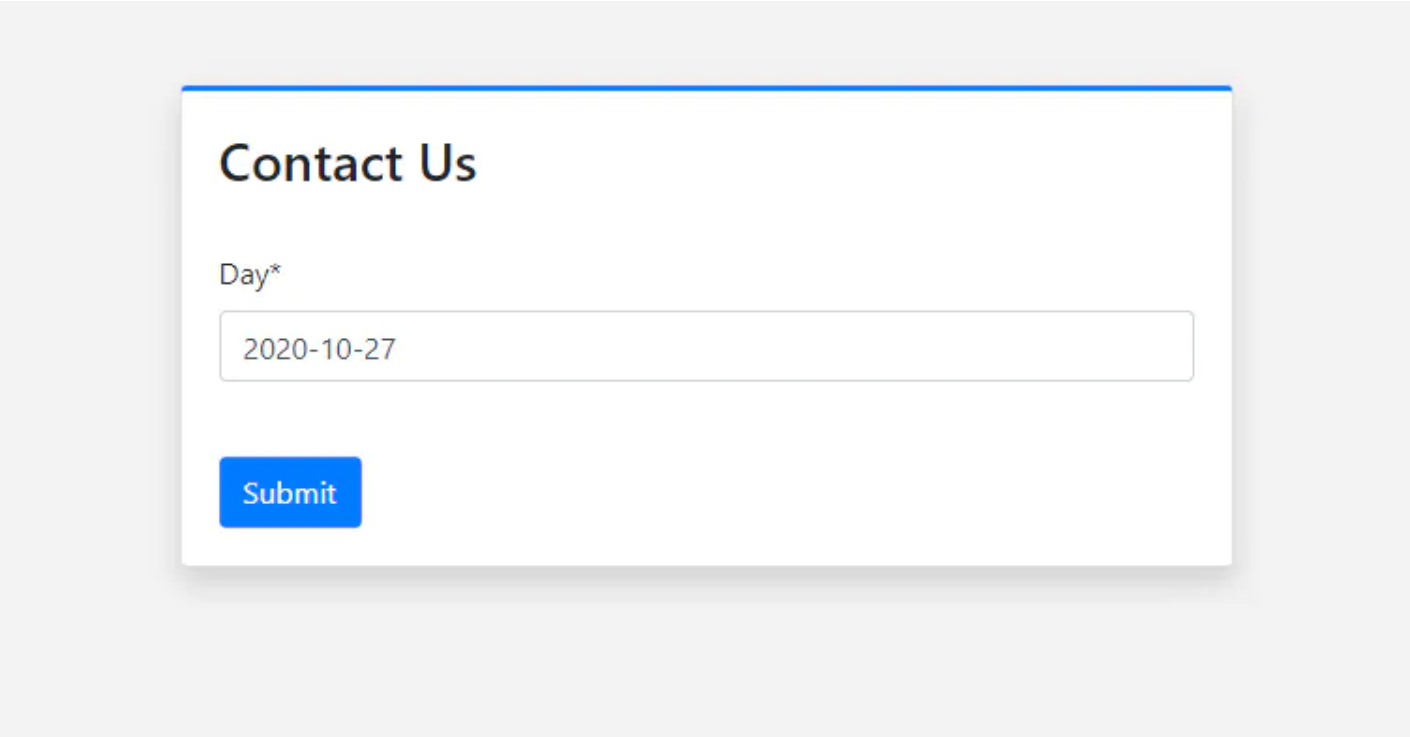
```
from django import forms

# Create your forms here.

class ExampleForm(forms.Form):
    agree = forms.BooleanField(initial=True)
```

Add initial=True to the BooleanField() to set the default as a clicked checkbox in the HTML template.

initial (Datetime) for DateField()

```
from django import forms
import datetime

# Create your forms here.

class ExampleForm(forms.Form):
    day = forms.DateField(initial=datetime.date.today)
```

Import datetime at the top of the file. Then you can set the initial input as the current date by using the import datetime.



ChoiceField, MultipleChoiceField, ModelChoiceField, and ModelMultipleChoiceField

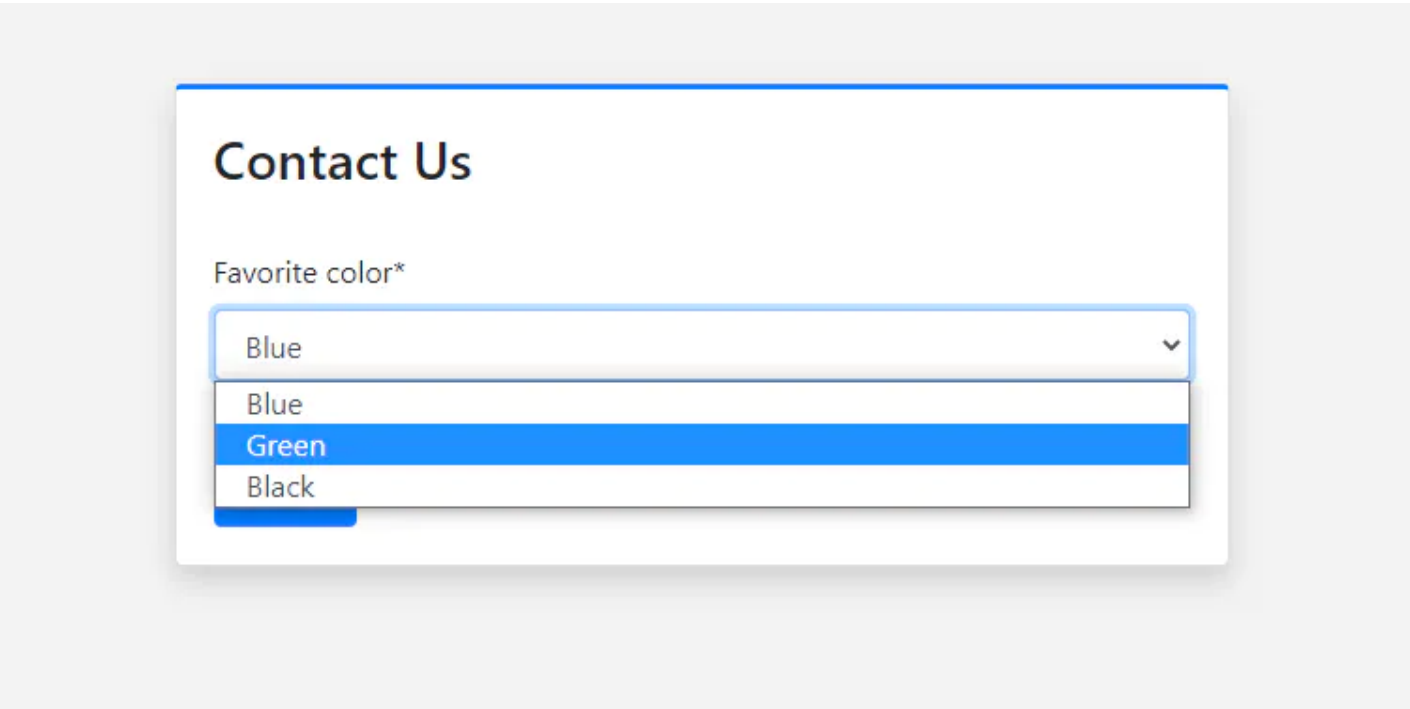
Create pre-built choice fields so the user does not have to enter any information and only selects from the predefined choices.

ChoiceField()

```
from django import forms

# Create your forms here.


FAVORITE_COLORS_CHOICES = [
    ('blue', 'Blue'),
    ('green', 'Green'),
    ('black', 'Black'),
]

class ExampleForm(forms.Form):
    favorite_color = forms.ChoiceField(choices=FAVORITE_COLORS_CHOICES)
```

Use the Django ChoiceField to create a drop-down menu of choices. The choices argument and the options are specified in the field.

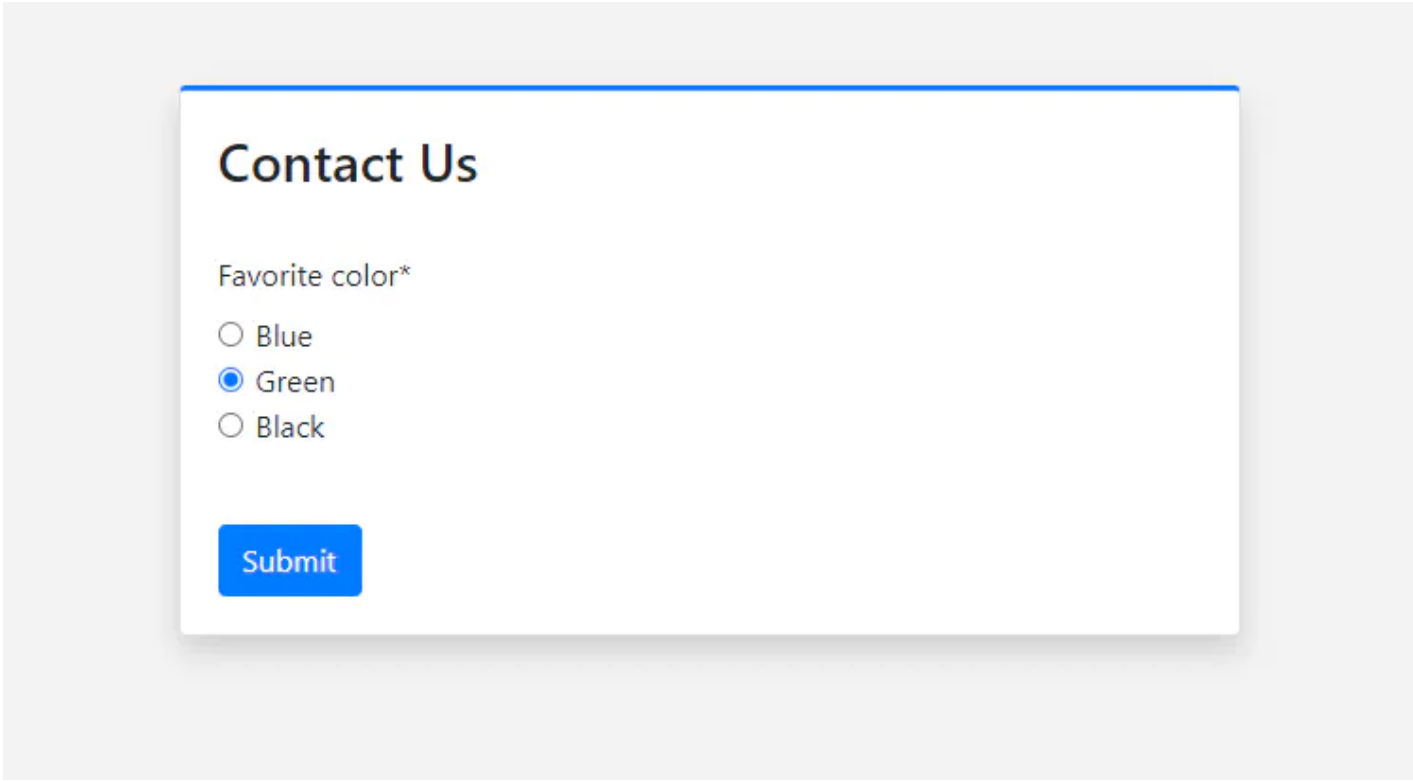
Each choice requires a key and a value, with the value being the option displayed to the user. To call an initial value, specify the name of the key of the value you wish to appear on the page load.



ChoiceField() with RadioSelect widget

```
from django import forms

# Create your forms here.

FAVORITE_COLORS_CHOICES = [
    ('blue', 'Blue'),
    ('green', 'Green'),
    ('black', 'Black'),
]

class ExampleForm(forms.Form):
    favorite_color = forms.ChoiceField(widget=forms.RadioSelect, choices
```

If you wish to have a radio select option, add the widget RadioSelect to the ChoiceField. This creates a radio, or small circle, buttons that are filled in when selected.

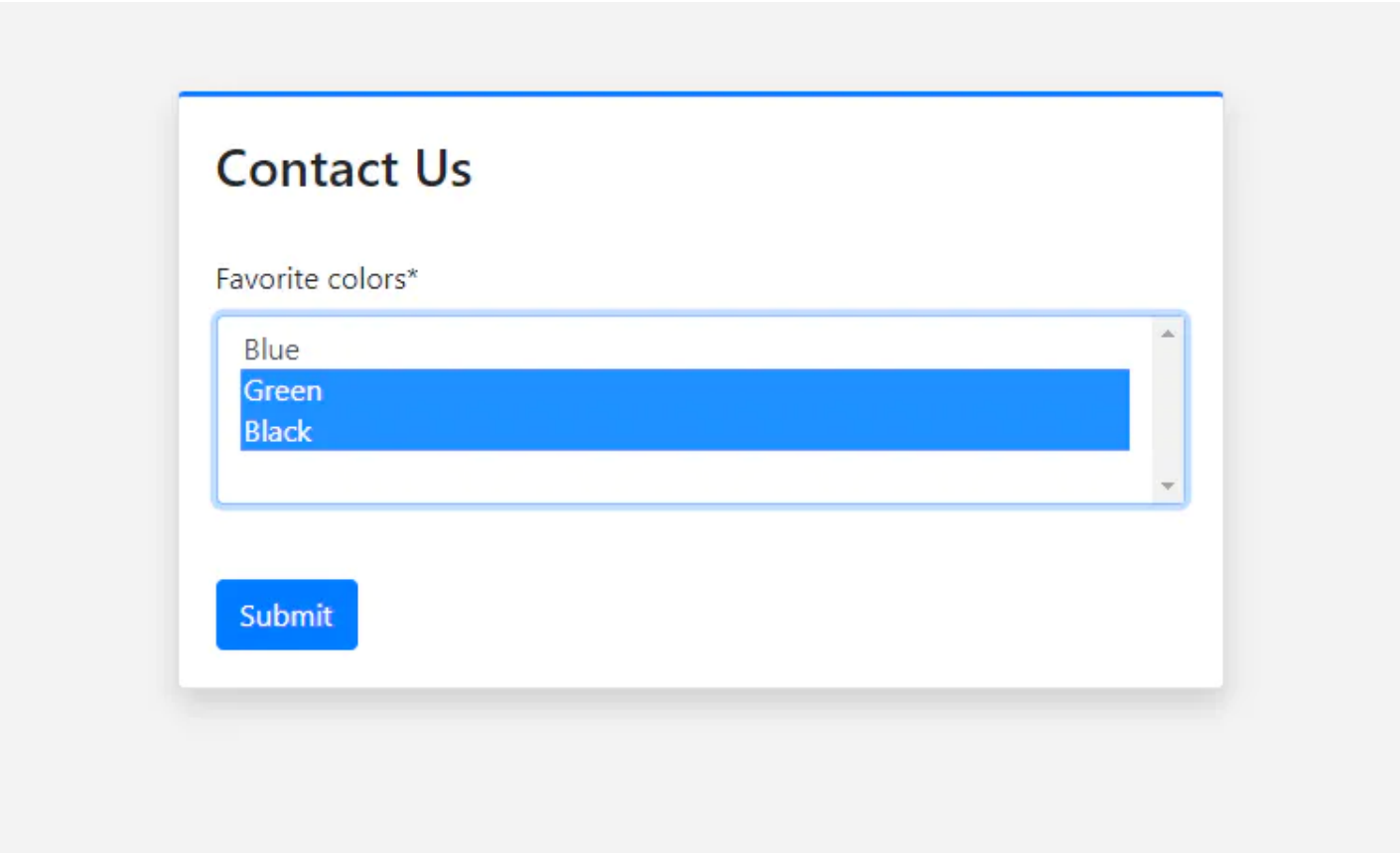


MultipleChoiceField()

```
from django import forms


# Create your forms here.
```

```
FAVORITE_COLORS_CHOICES = [
    ('blue', 'Blue'),
    ('green', 'Green'),
    ('black', 'Black'),
]

class ExampleForm(forms.Form):
    favorite_colors = forms.MultipleChoiceField(choices=FAVORITE_COLORS_
```

If you wish for the user to select multiple, use the MultipleChoiceField.

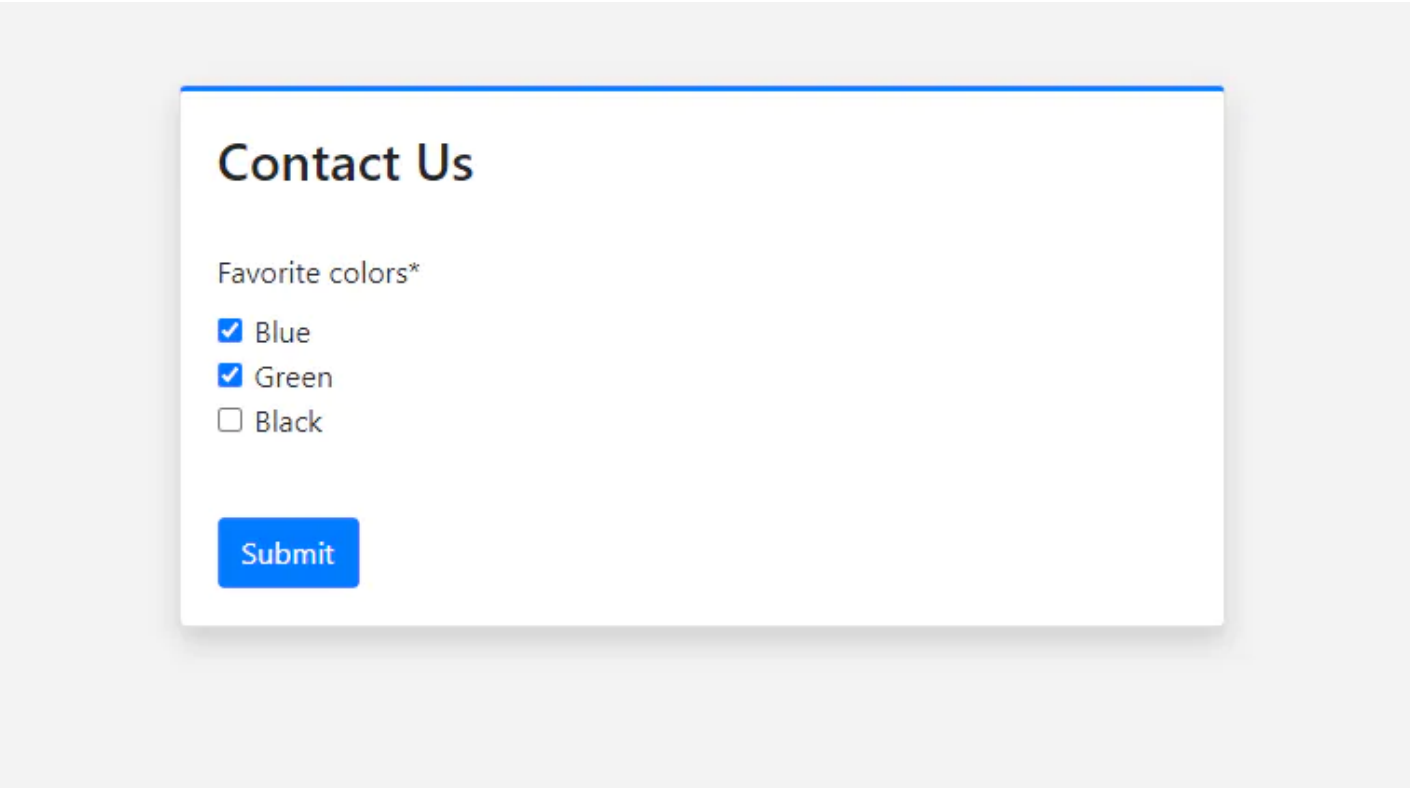


MultipleChoiceField() with CheckboxSelectMultiple widget

```
from django import forms


# Create your forms here.

FAVORITE_COLORS_CHOICES = [
    ('blue', 'Blue'),
    ('green', 'Green'),
    ('black', 'Black'),
```

```
]

class ExampleForm(forms.Form):
    favorite_colors = forms.MultipleChoiceField(widget=forms.CheckboxSel
```

You can also add the widget CheckboxSelectMutiple to have the choices render next to checkboxes.



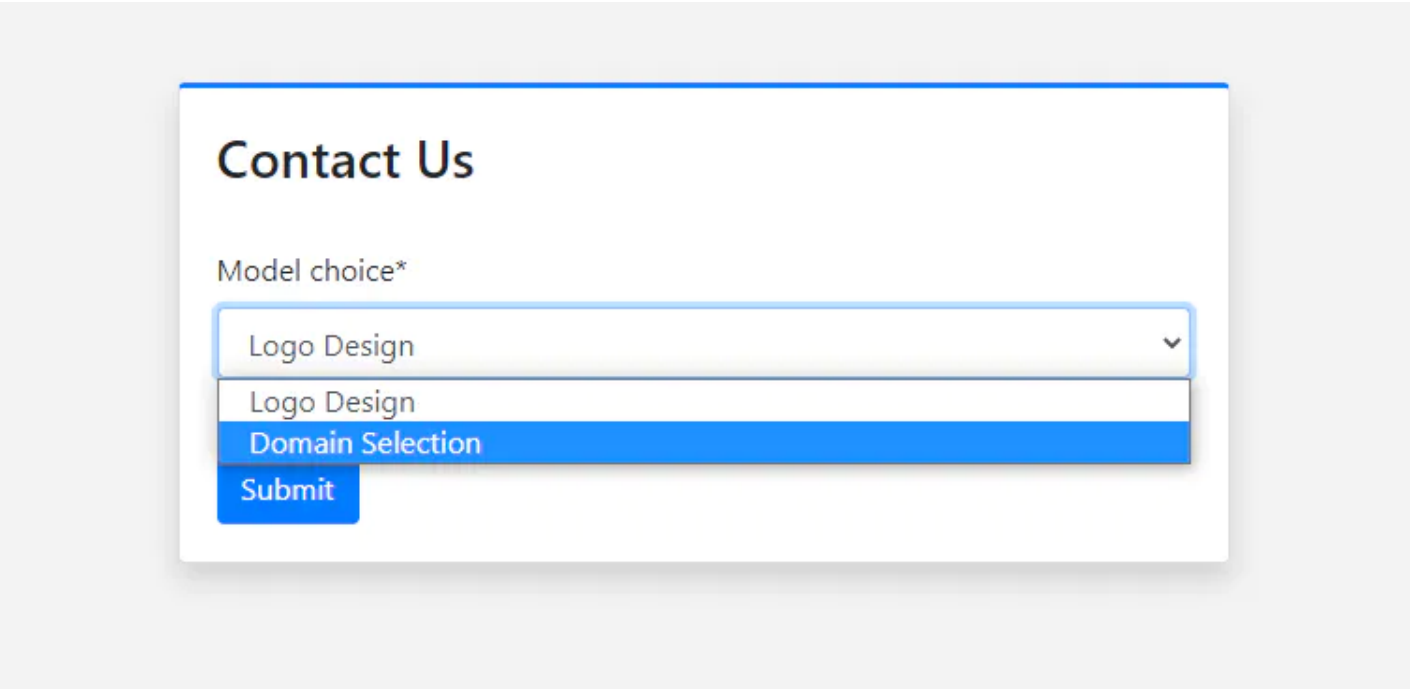
ModelChoiceField()

```
from django import forms
from .models import MyModel

# Create your forms here.

class ExampleForm(forms.Form):
    model_choice = forms.ModelChoiceField(
        queryset = MyModel.objects.all(),
        initial = 0
        )
```

If you wish to connect a Django model to the form, import the model from the *models.py* file and use the ModelChoiceField() and specify the queryset in the form field.

Set the initial to 0 if you wish for the first model object value to be the initial value of the drop-down menu.

For more information on creating a model, visit the article How to use Django Models.



ModelMultipleChoiceField() with CheckboxSelectMultiple widget

```
from django import forms
from .models import MyModel

# Create your forms here.

class ExampleForm(forms.Form):
    model_choices = forms.ModelMultipleChoiceField(
        widget = forms.CheckboxSelectMultiple,
        queryset = MyModel.objects.all(),
        initial = 0
        )
```

To allow for multiple-choice selection, use ModelMultipleChoiceField. If you wish to change the drop-down menu to checkboxes, add the widget CheckboxSelectMultiple.

Contact Us

Model choices*

- [x] Logo Design
- [x] Domain Selection

Submit