

## Flask Python (8) – Form data validation

### wtform validators

Step 1: Currently we can add any book data – even blank titles and incorrect ISBN numbers. Fortunately, flask has a validators library which we can import into the form.py file.

```
from wtforms import StringField, SubmitField, PasswordField, validators
```

We can add these to our class to ensure that data is exists nd of the correct format.

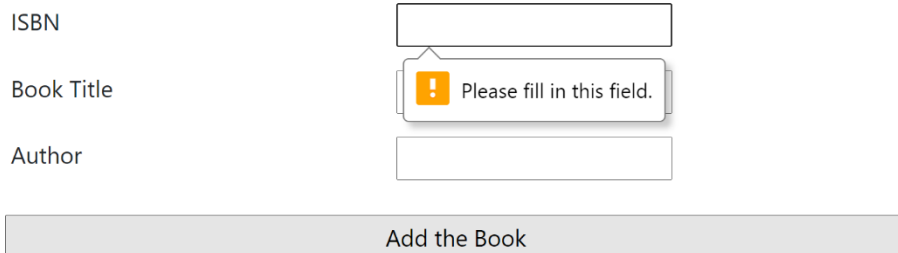
### Required Data

Step 2: With this library we can declare an attribute as follows:

```
isbn = StringField('ISBN',[validators.DataRequired()])
```

so that an ISBN is needed. If the user doesn't enter data then a warning comes up and the form is not validated so it isn't submitted.

## Add a Book



ISBN

Book Title

Author

Add the Book

### Length of data

Step 3: An isbn number is usually between 10 and 13 numbers so we can add further validation:

```
isbn = StringField  
( 'ISBN',  
    [validators.DataRequired()],  
    [validators.Length(min=10, max=13)]  
)
```

Step 4: However, this new validator will pass the data in the filed isbn to the `/library` route, so we need to change our flask\_app.py code as follows:

```
@app.route('/library',methods=["GET","POST"])  
def library():  
    form = AddBook()  
    if form.validate_on_submit():  
        add_book(form.isbn.data, form.title.data, form.author.data)  
    all_books = get_books()  
    return render_template('library.html', all_books=all_books, form=form)
```

The if statement is now waiting for a valid form. If our isbn number isn't valid it won't proceed and will say there is an error.

Step 5: We can list these errors on our library.html page using this code just above the form:

```
<div class="row" style="background-color:#ffb3b3">
    {% for field, errors in form.errors.items() %}
    {{ ', '.join(errors)}}
    {% endfor %}
</div>
```

Field must be between 10 and 13 characters long.

ISBN	<input type="text" value="234"/>
Book Title	<input type="text" value="My Way"/>
Author	<input type="text" value="Dennis"/>

### Customising Validation

Step 6: It is also possible to customise validation. Let's check whether an ISBN number already exists in the database; remember it is a primary key so trying to add it to the database will throw an error. We write a function as part of the `AddBook` class:

```
def validate_isbn(self,isbn):
    if isbn_exists(self.isbn.data) == True:
        raise ValidationError('ISBN already exists')
```

Step 7: Write your own function `isbn_exists(isbn)` in the `models.py` file which is a query. It will return True if there is a book with the given ISBN otherwise false. Import it so the above code runs correctly and throws the error in the error field if the user tries to add a book that already exists.

#### **Add a Book**

ISBN already exists

ISBN	<input type="text" value="9781853260285"/>
Book Title	<input type="text" value="Emma"/>
Author	<input type="text" value="Test"/>