Saving After Editing Function

- first we should call it an action key with its url name to be executed after user submission:

```
<body>
    <p>Edit Book Form</p>
    <form method="POST" action="{% url 'new-save-book' book_data.id %}">
        {% csrf_token %}
        Book name: <input type="text" name="book_name" value="{{ book_data.name }}"> <br>
        Book description: <input type="text" name="book_desc" value="{{ book_data.description }}
        Book Price: <input type="number" name="book_price" value="{{ book_data.price }}"> <br>
        <button type="submit"> Edit Book </button>
    </form>
</body>
</html>
```

- then we make its URL in urls.py file as follows:

```
urlpatterns = [
    path('list', home, name='book-list'),
    path('read', read),
    path('', index),
    path('create', create_book, name='create-book'),
    path('save', save_book, name='save-book'),
    path('edit/<book_id>', edit_book, name='edit-book'),
    path('delete/<book_id>', delete_book, name='delete-book'),
    path('new_save/<book_id>',new_save,name='new-save-book')
]
```

here I make the URL with a variable that takes its value from action caller which is the id of the book that I want to save its new data.

- then we go to views.py and write new_save function:

```
63  def new_save(request,book_id):
64      if request.method == "POST":
65          edited_book = Book()
66          edited_book.id = book_id
67          edited_book.name = request.POST.get('book_name')
68          edited_book.description = request.POST.get('book_desc')
69          edited_book.price = request.POST.get('book_price')
70          edited_book.save()
71          return redirect('book-list')
```

I make a new object from interface class Book to pass new data to its attributes. Then I pass book id to this object and save.
When I save this object it will check if this id is already exist or not. And if it is, It will overwrite on this record data with the new data.