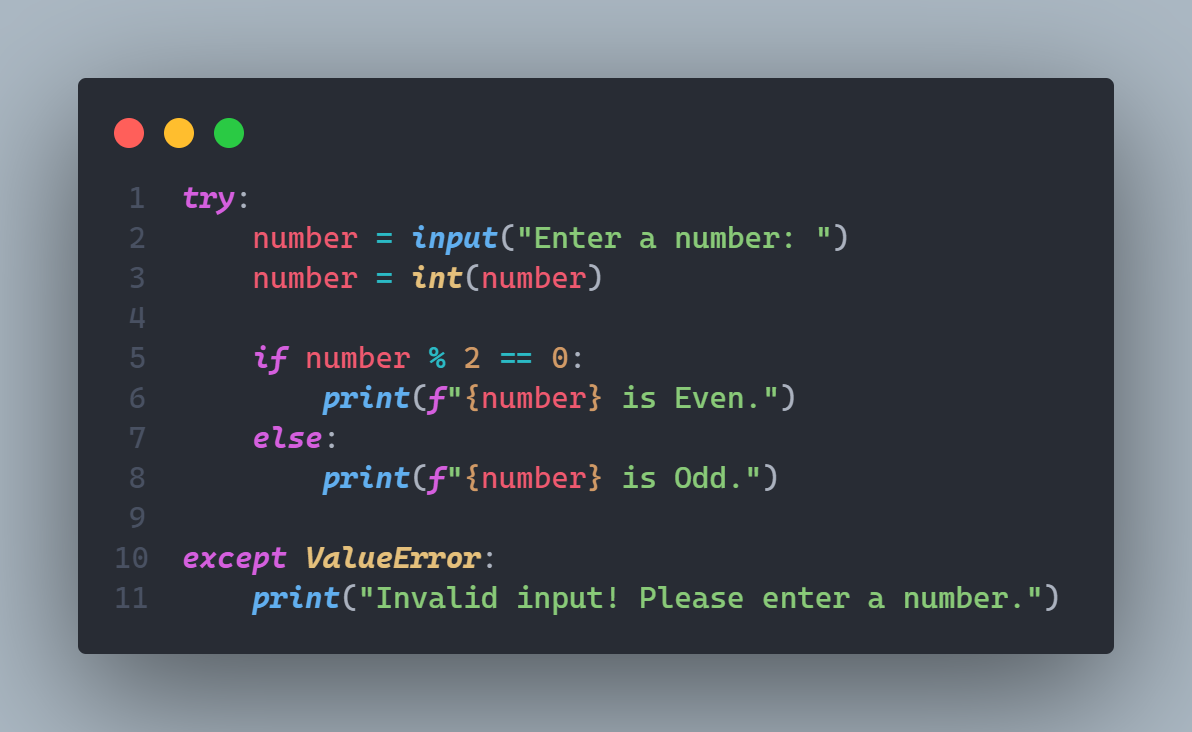
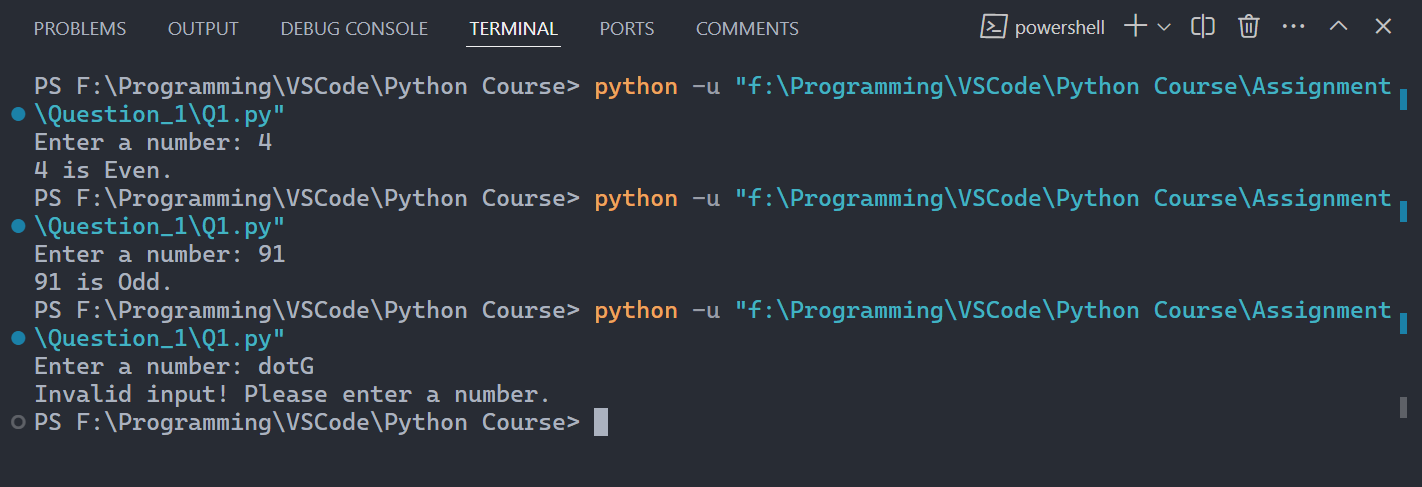
# Question #1

Write a Python program that takes input from the user. Use a try-except block to handle invalid inputs (e.g., non-numeric values) and display the message: "Invalid input! Please enter a number." If the input is valid, use an if-else statement to check whether the number is even or odd and print the result accordingly.

**Code:**



**Output:**



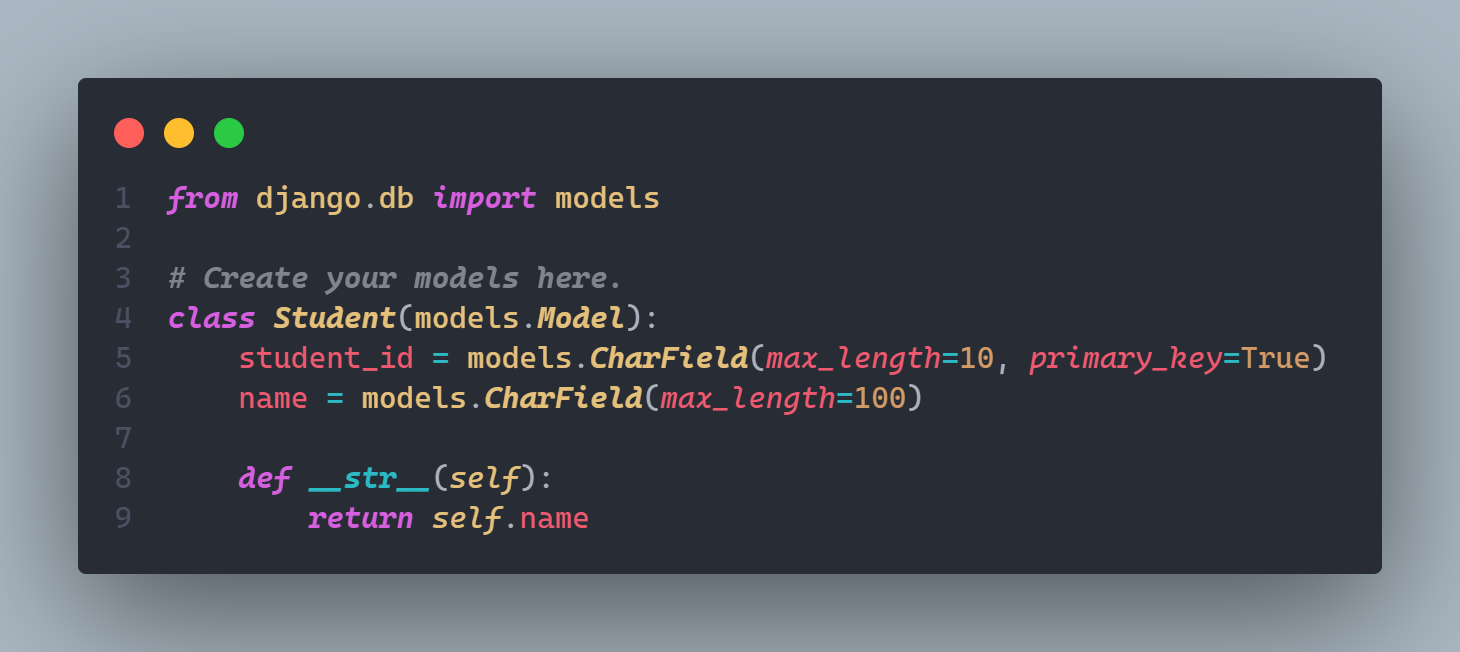
**Explanation:**

The program prompts the user to input a number and uses a try-except block to handle invalid inputs, such as non-numeric values. If the input cannot be converted to an integer, a ValueError is caught, and the program displays the message: "Invalid input! Please enter a number." If the input is valid, the program uses an if-else statement to check whether the number is even (divisible by 2 without a remainder) or odd and prints the result. This ensures the program is robust and user-friendly.

# Question #2

Create a Django application with basic CREATE and READ functionality for managing student records. Define a Student model with name and student\_id fields. Implement a form to add new students (CREATE) and a view to display all students in a table (READ). Ensure the form saves data to the database.

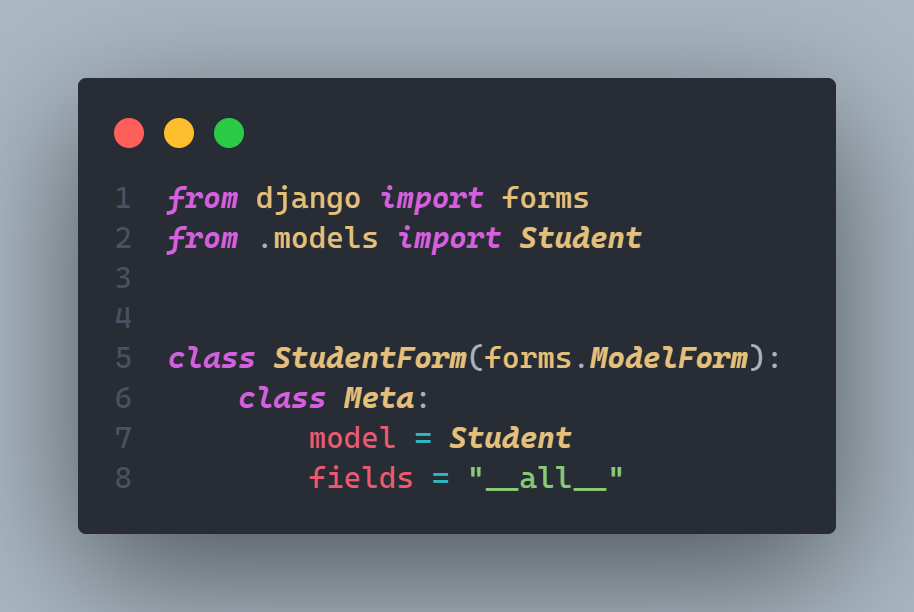
**Code:**



***models.py***

**Explanation:**

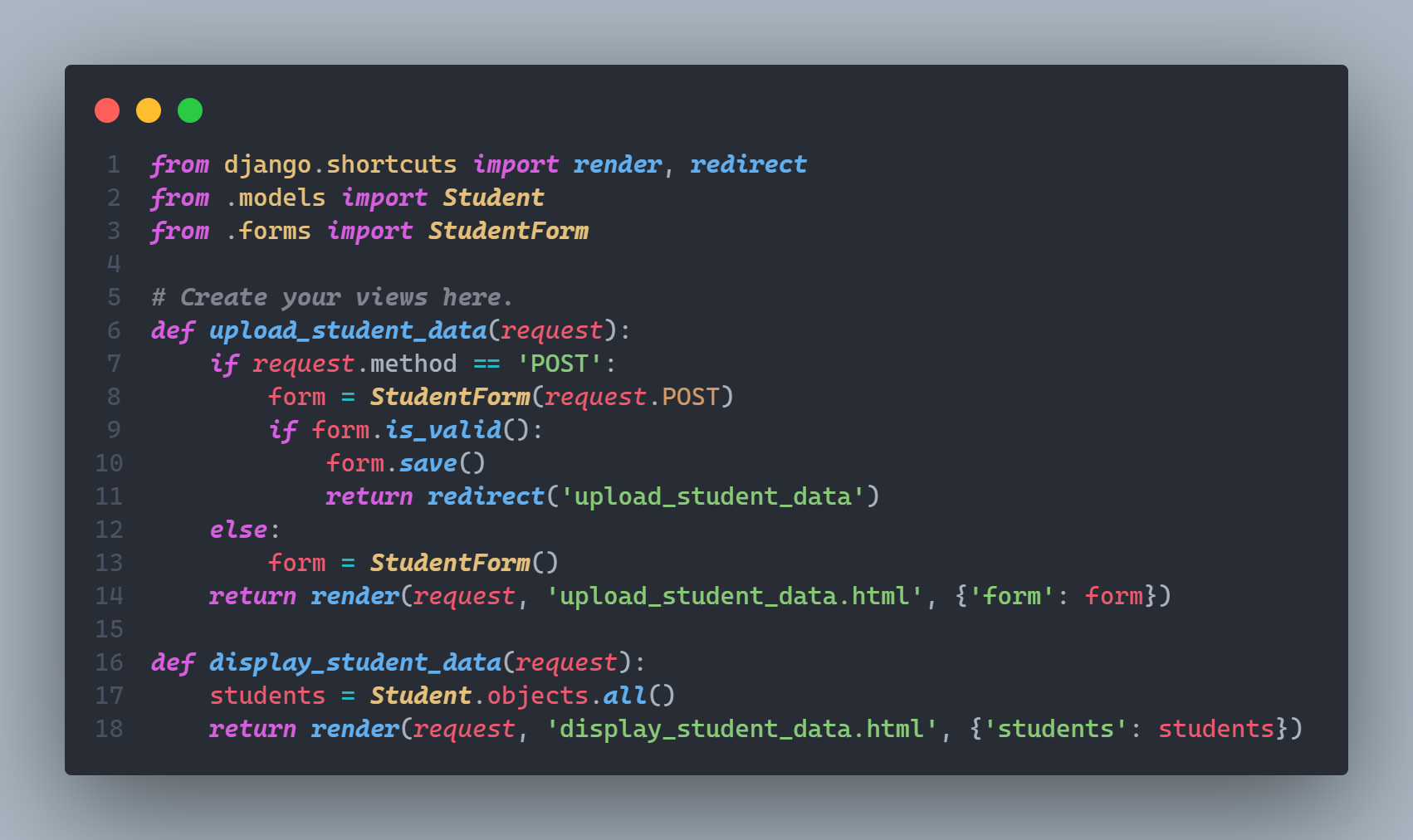
This file is where we define the **Student model**. Think of it as the blueprint for the student records we want to store in the database. Each student will have a student\_id (a unique identifier) and a name. The \_\_str\_\_ method just makes sure that when we print a student object, it shows their name instead of some technical stuff.



***forms.py***

**Explanation:**

Here, we create a **form** for adding new students. Django makes this super easy with ModelForm, which automatically generates a form based on the Student model. By saying fields = "\_\_all\_\_", we’re including all the fields from the model in the form—so it will have both student\_id and name.



***views.py***

**Explanation:**

This file is where the main functionality happens:

1. **upload\_student\_data**: This is for adding new students. When someone fills out the form and submits it, the view checks if the data is valid and saves it to the database. If the user hasn’t submitted anything yet, it just shows an empty form.
2. **display\_student\_data**: This is for showing all the students we’ve added so far. It fetches all student records from the database and sends them to the template, where they’re displayed in a table.

Together, these files handle adding students (CREATE) and viewing the list of students (READ).