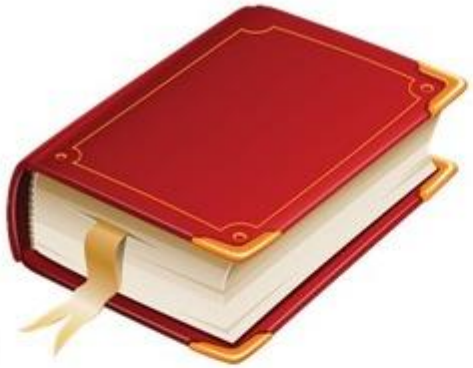
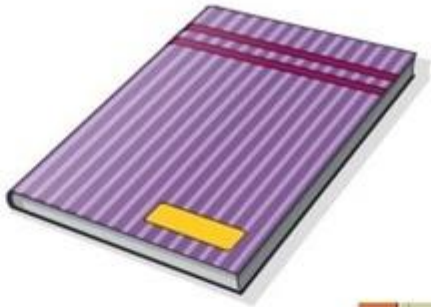
Class 04

Class Plan

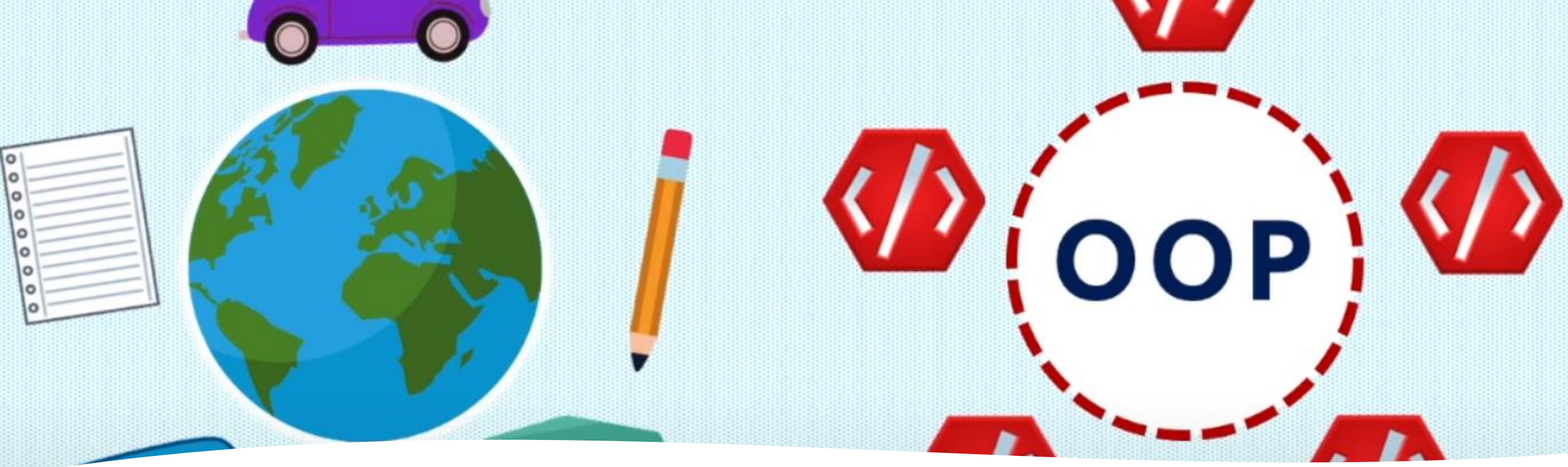
- Questions
- Object-Oriented Programming
- Stretch and Flex
- Quiz Talk
- Debugging with PyCharm

The world is made up of individual **objects**!





https://www.youtube.com/watch?v=X3cFiJnxUBY



- **Object-Oriented Programming (OOP)** is a programming paradigm that organizes code around objects, which are instances of classes.

https://www.youtube.com/watch?v=X3cFiJnxUBY

Class & object

- Class is a blueprint or template for creating objects. It defines the attributes of an object and methods or functions that objects can perform.
- Object is an actual realization of a class.

https://www.youtube.com/watch?v=X3cFiJnxUBY

Class & **object** for better programming

1. Modularity and reusability
2. Maintainability and Extensibility
3. Collaboration and Teamwork

https://www.youtube.com/watch?v=X3cFiJnxUBY

OOP in MIS: Real-World Applications

Examples:

- **Employee Management System**
- You can have classes like **Employee, Manager,** and **Department**

- **Inventory Management System**
- You can have classes like **Product, Supplier,** and **Inventory**

OOP in MIS: Real-World Applications

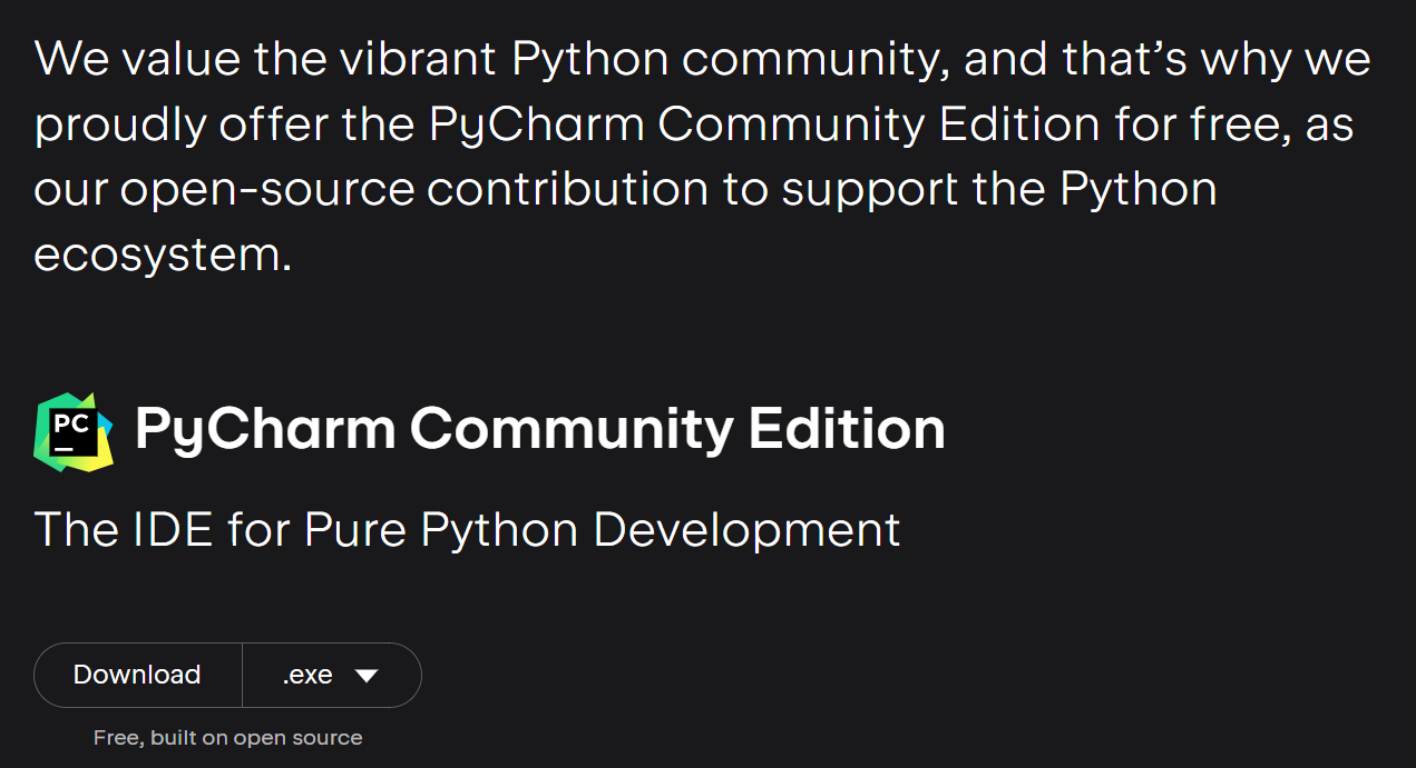
- The special method in Python used for initializing an object's attributes when it is created is called __init__().
- The __init__() method is a constructor method that gets automatically called when an instance of a class is created. It allows you to specify the initial values for the object's attributes. This method is essential for initializing the state of the object and preparing it for use.

```
class MyClass:
        def __init__(self, attribute1, attribute2):
                self.attribute1 = attribute1
                self.attribute2 = attribute2
```

Notebook Time!

Stretch & Flex

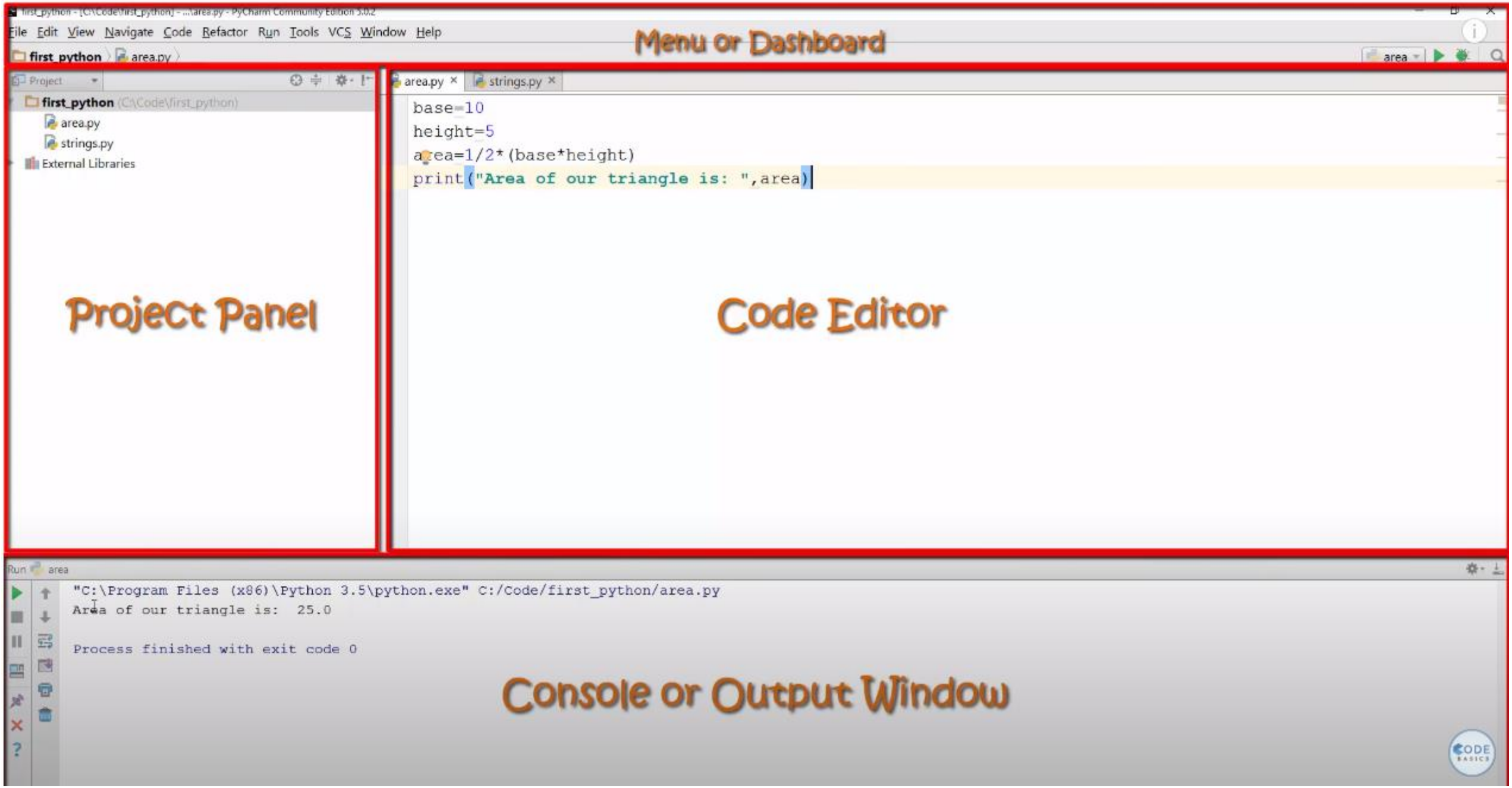
Quiz Talk!



PyCharm and Debugging

- **Debugging**: A process of identifying and fixing problems (bugs) in your code
- **Install PyCharm**:

https://www.jetbrains.com/pycharm/download/?section=windows



[7. Debug Python code using PyCharm [Python 3 Programming Tutorials] - YouTube]

Debugging

- Setting breakpoints
- Set variables to watch
- Step into: if there is a function call, it goes inside the function, and you can see how the function is executing line by line till it returns and you go back to the next line right after the function call.

- Step over: if there is a function call, it just executes it like a black box and returns the result, but you cannot see how the function was executed.

- Step out: if you have Stepped into a function and now you want to skip seeing how the rest of the function is going to execute, you Step out and the function returns. Then, you go back to the next line, that is the line right after the function call.