Key Concepts:

1. **Code Reusability**: Reusing existing code to build new software efficiently. By creating reusable classes, we can create multiple instances with different values, allowing for flexibility and reducing code duplication.
2. **Special Methods in Python**:
   - `__new__` : Responsible for creating and returning a new empty object.
   - `__init__` : The constructor method that initializes the object after it's created by `__new__` . It is typically used to set the initial state of the object.
3. **Class Instantiation**: Creating an object (or instance) from the class, each with its unique set of attributes.

Step-by-Step Breakdown

1. **Creating a Class**:

   Start by creating a class called `Recipe` that helps to represent a dish, its ingredients, and its preparation time.

```
class Recipe:
    def __init__(self, dish, items, time):
        self.dish = dish  # Dish name (e.g., pizza)
        self.items = items  # Ingredients list
        self.time = time  # Preparation time in minutes
```

2. **Adding a Method**:

   Next, we add a method to the class that will return the recipe details in string form:

```
def contents(self):
    print(f"{self.dish} has {', '.join(self.items)} and takes {str(self.time)} min to prep
```

   - `self.dish` , `self.items` , and `self.time` are instance variables that store the specific details for each instance.
   - The method `contents()` uses these variables and formats them into a readable string.

3. **Creating Instances**:

   Now, let's create two instances of the `Recipe` class: one for pizza and one for pasta.

```
pizza = Recipe("Pizza", ["cheese", "bread", "tomato"], 45)
pasta = Recipe("Pasta", ["penne", "sauce"], 55)
```

These instances represent two different recipes with different attributes (dish, items, and preparation time).

4. **Accessing Instance Variables and Methods**:

   You can now access the attributes and methods of each instance separately:

```
print(pizza.items)  # Output: ['cheese', 'bread', 'tomato']
print(pasta.items)  # Output: ['penne', 'sauce']

pizza.contents()  # Output: Pizza has cheese, bread, tomato and takes 45 min to prepare.
pasta.contents()  # Output: Pasta has penne, sauce and takes 55 min to prepare.
```

   - Despite passing the same variable `items`, the two instances (`pizza` and `pasta`) produce different outcomes because their attribute values are different.
   - The `contents()` method is called on each instance, and the corresponding details are printed.

Conclusion

This video demonstrates how you can:

- **Define a class** with variables (attributes) and methods.
- **Instantiate multiple objects** from the same class.
- **Reference instance variables and methods** to get different outcomes based on each instance's data, showcasing the power of **code reusability**.

By following this approach, you can create classes that serve as reusable templates, reducing code duplication and making your code more maintainable. Let me know if you'd like more examples or further clarification!