Assignment: Attributes and Methods

Instructions:

1. Complete the tasks in Python using a code editor of your choice.
2. Submit the screenshots of the completed Python file with output.
3. Deadline:
4. Make sure to write clear, readable code with proper indentation.

Part 1: Class Variables and Instance Variables

Task:
Create a `Car` class with the following:

1. A **class variable** `wheels` initialized to 4.
2. **Instance variables** for `brand` and `model`.
3. A method `display_info()` that prints the car's brand, model, and the number of wheels.

Example Input/Output:

```
car1 = Car("Toyota", "Corolla")
car1.display_info()
# Output: Brand: Toyota, Model: Corolla, Wheels: 4
```

Part 2: Types of Methods

Task:
Create a `Rectangle` class with:

1. **Instance variables** `length` and `width`.

2. A **static method** `is_square(length, width)` that checks if the rectangle is a square.
3. A **class method** `description()` that returns a string describing the class.
4. A method `area()` that calculates and returns the area of the rectangle.

Example Input/Output:

```
rect1 = Rectangle(5, 5)
print(Rectangle.is_square(5, 5))  # Output: True
print(Rectangle.description())   # Output: This class handles
rectangles.
print(rect1.area())              # Output: 25
```

Part 3: Decorators

Task 1: `@property` Decorator

Create a `Temperature` class with:

1. An **instance variable** `celsius`.
2. A `@property` method `fahrenheit` that converts the Celsius temperature to Fahrenheit.

Example Input/Output:

```
temp = Temperature(0)
print(temp.fahrenheit)  # Output: 32.0
```

Task 2: Custom Decorator

Create a custom decorator `log_message` that prints **"Calling function [function_name]"** before calling any function. Apply it to a method `greet()` in a class `Greeter`.

Example Input/Output:

```
greeter = Greeter("Alice")
greeter.greet()
# Output:
# Calling function greet
```

```
# Hello, Alice!
```

Part 4: Special Methods

Task:
Create a `Book` class with:

1. Instance variables `title` and `author`.
2. A `__str__` method to display the book in the format: `"[Title] by [Author]"`.
3. A `__repr__` method to display the book in the format: `Book(title='[Title]', author='[Author]')`.

Example Input/Output:

```
book1 = Book("1984", "George Orwell")
print(book1)         # Output: 1984 by George Orwell
print(repr(book1))   # Output: Book(title='1984', author='George Orwell')
```

Part 5: Hands-On Challenge

Task:
Create an `Employee` class with:

1. Instance variables `name`, `position`, and `salary`.
2. A method `display_details()` to print the employee's details.
3. A method `increase_salary(percentage)` to increase the salary by a given percentage.
4. A **staticmethod** `tax_bracket(salary)` that prints the tax bracket based on the salary:
    - Below $40,000: Low
    - $40,000–$80,000: Medium
    - Above $80,000: High
5. A **custom decorator** `log_execution_time` that calculates and prints the time taken to execute the `increase_salary()` method.

Example Input/Output:

```
emp1 = Employee("John", "Developer", 50000)
emp1.display_details()
# Output: Name: John, Position: Developer, Salary: 50000

emp1.increase_salary(10)
# Output: Salary increased by 10%
# Execution time: 0.002 seconds

print(Employee.tax_bracket(emp1.salary))
# Output: Medium
```

Submission Checklist:

- Code for all parts is completed.
- Comments explaining each section of the code.
- Python file is named: `Assignment_OOP_Class2_part_?.py`.