Python Merlin’s MPG GUI

Contents

Time required: 120 minutes

- Comment each line of code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Pseudocode

1. Write pseudocode for the exercise
2. Save it in a document
3. Submit with the assignment

Assignment Options

You may want to do your own calculator. That is perfectly fine. Here are some examples of alternatives.

Grocery Bill Calculator

Create a Tkinter program where users enter the prices of up to five grocery items and their quantities. The program calculates and displays the total cost and applies a discount if the total exceeds a certain amount.

Loan Payment Calculator

Develop a Tkinter app that accepts the loan amount, annual interest rate, and loan term in years. The program calculates and displays the monthly payment and total repayment amount.

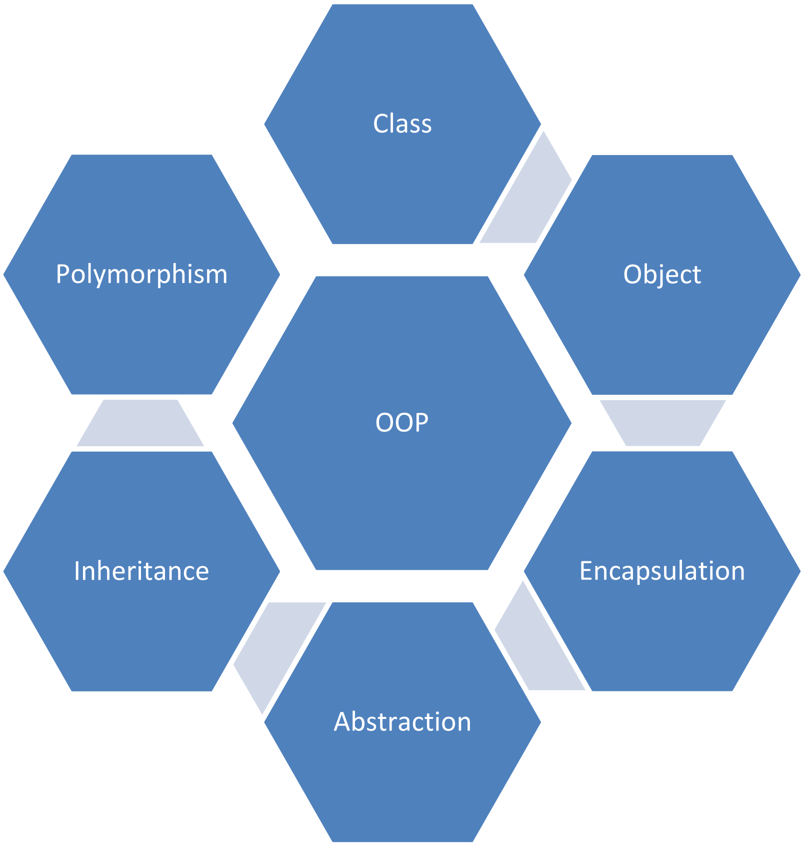
Currency Converter

Build a Tkinter program that accepts an amount in one currency (e.g., USD) and converts it to another currency (e.g., EUR) using a predefined exchange rate. Display the converted amount in the interface. You could add an API that uses real time currency exchange information.

Each of these assignments introduces user input handling, basic calculations, and GUI design concepts, making them similar in complexity to the mileage calculator. You can use AI to come up with other ideas if you wish.

OOP Mind Map

The following mind map gives a high level view of the different concepts that are part of Object Oriented programming.

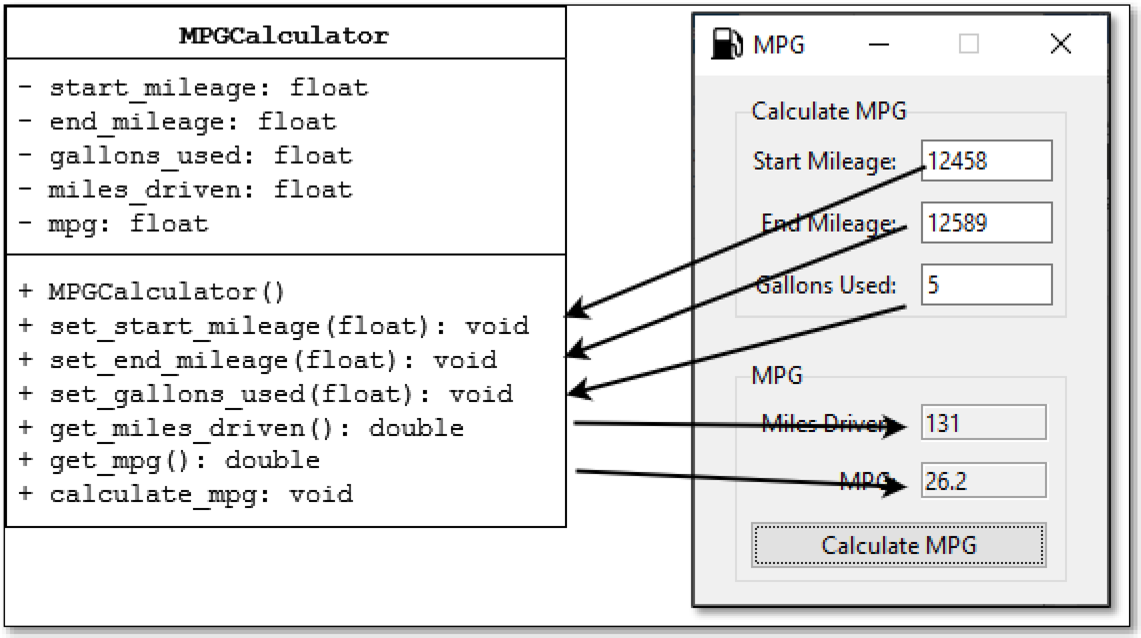


1. **Class**: A blueprint for creating objects, defining properties and behaviors (attributes and methods).
2. **Object**: An instance of a class, representing a specific entity with defined data and behavior.
3. **Encapsulation**: Restricting direct access to an object's data, bundling data with methods for controlled interaction.
4. **Abstraction**: Hiding complex implementation details, showing only essential features to the user.
5. **Inheritance**: Enabling a class (child) to acquire properties and behaviors of another class (parent).
6. **Polymorphism**: Allowing entities to take on multiple forms, enabling methods to perform differently based on context.

Assignment 1: UML Diagram

Draw a UML diagram of your class structure. Your class should be a separate file from your application.

The following diagram illustrates how the data flows between the class and the user interface. You only need to draw the class diagram.



```
MPGCalculator
- start_mileage: float
- end_mileage: float
- gallons_used: float
- miles_driven: float
- mpg: float

+ MPGCalculator()
+ set_start_mileage(float): void
+ set_end_mileage(float): void
+ set_gallons_used(float): void
+ get_miles_driven(): double
+ get_mpg(): double
+ calculate_mpg: void
```

MPG

Calculate MPG

Start Mileage: 12458

End Mileage: 12589

Gallons Used: 5

MPG

Miles Driven: 131

MPG: 26.2

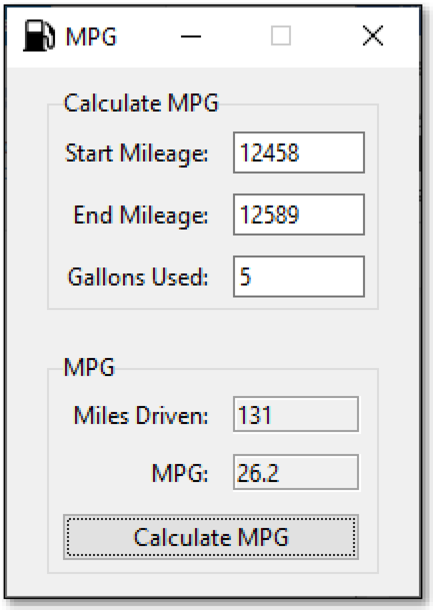
Assignment 2: Requirements

Merlin the Magician would like a program to calculate the MPG of his ride. He likes GUI programs in OOP.

- Object Oriented Programming in the same format as we have used in the past.
- Use a separate class file for the calculations.
- All user input and output will be in the main app file.
- Use object variables.
- Create a method for each calculation.
- Create getters and setters for the needed variables.

Here is an example Python program named **mpg_gui.py** that calculates the miles per gallon.

Example design:



Start Mileage: 12458

End Mileage: 12589

Gallons Used: 5

Miles Driven: 131

MPG: 26.2

Assignment Submission

1. Attach the pseudocode.
2. Attach the class diagram.
3. Attach the program files.

4. Attach screenshots showing the successful operation of the program.

5. Submit in Blackboard.