Enhancement Narrative: Car Rental Program

Troy Rushing

Southern New Hampshire University

Enhancement Narrative: Car Rental Program

1. Software Engineering/Design
2. Artifact

The artifact for my software engineering/design enhancement is a rental car cost calculating program. This program is written in Python and was originally created as part of my course work for IT-140 during term 18EW2 in November of 2018.

1. Justification

I selected this component because it is a useful program, but lacked the nuance and failure handling required to make it fully function as intended. In reviewing the original code for this assignment, I realized there was a deficiency in the design that allowed for only 3 selections in the original prompt ((B)udget, (D)aily, or (W)eekly). Any selection other than “B”, “D”, or “W” would result in an error in the program. To remedy this deficiency, I made the selection non-case sensitive, added failure handling to the code that will prompt the user to “try again” if an invalid selection is made, and limited the number of invalid selections to 5 before exiting the program.

The above enhancement demonstrates an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals. This is achieved by illustrating the ability to identify shortcomings in a program, design a solution for the shortcoming, and implement advanced programming using multiple types of loop logic including embedded loops to solve the problem.

1. Objective Outcome

In designing and implementing the enhancement for this assignment, I did meet the planned goal defined in the software design document plan submitted in module 1. In addition to the enhancement described in the plan, I implemented an additional failure handling enhancement in this segment of code. Originally, after selecting the rental type, you were asked to enter a rental period (either weeks or days). The original code would allow for entering non-numeric characters that would cause the program to crash when attempting to make calculations later in the program. I added failure handling to ensure that the entered rental period value is numeric and not allow the user to continue until a numeric value was entered. Because the failure handling code falls within the original planed loop to limit invalid selections to 5, 5 invalid rental period selections will also inform the user that they have made too many invalid selections and exit the program.

1. Reflection

In implementing this enhancement, I learned a number of lessons as well as enhanced skills that I already had learned. I learned how to critically analyze all aspects of a segment of code. I also learned how to write code that limit retries of invalid inputs. I had not previously done this, so it took some research and trial and error to figure out how to implement it properly. Additionally, I learned how to ensure an entered value is a specific type (using .isdigit() in my case). This had been briefly covered in early coding classes, but because I had not used this function, I forgot how to code it. I also improved my skills in using loop logic.

The first challenge that I faced was planning a design that would be fully functional and include all necessary failure handling branches to ensure the code would not crash. While I was implementing and testing my originally planned enhancement, I realized the deficiency in the segment of code that gets the rental period input. I Once I implemented my originally planned enhancement, a challenge I had was to implement the additional enhancement of adding failure handling to ensure the rental period input is a numeric value.

1. Algorithms & Data Structures
2. Artifact

The artifact for my algorithms and data structures enhancement is a rental car cost calculating program. This program is written in Python and was originally created as part of my course work for IT-140 during term 18EW2 in November of 2018.

1. Justification

I selected this component because it is a useful program, but lacked the nuance and failure handling required to make it fully function as intended. In reviewing the original code for this assignment, I realized there were multiple deficiencies in the design. First, the program did not allow for mileage based charges that would be common in most rental agreements. Additionally, there was a segment of code that would allow for division by 0 which needed to be remedied. I also added a fair amount of code cleanup that provides more detail in the rental summary and makes the program more visually appealing both in a code review setting and when actively running the code. In order to properly implement these enhancements, I had to create a list to contain the valid rental type selections for calculations. This helps to make the code more modular. I also had to write code to calculate mileage, as well as create loop logic algorithms to calculate additional mileage charges based on type of rental and number of total miles traveled.

Once I had properly implemented the enhancements above and tested them to satisfaction, I decided to add further refinements and enhancements. The last time I traveled, an additional fuel charge was added to my car rental (even though I returned the car with a full tank), so I decided I should add a fuel charge assessment function. To do so, I added a “while True” loop with nester if/else logic to determine if the vehicle had been returned with a full tank and add a fuel charge or not accordingly. I also decided to added logic to calculate a 10% tourist tax, and an 8% state/local tax.

The implemented solution demonstrates the ability to design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices. This is done by designing and implementing advanced programing concepts such as nested loops in algorithms that calculate cost based on provided variables.

1. Objective Outcome

In designing and implementing the enhancement for this assignment, I did meet the planned goal defined in the software design document plan submitted in module 1. I added data structures and algorithms to perform advanced calculations based on user input and miles traveled. In addition to the enhancement described in the plan, I implemented additional failure handling enhancements in this segment of code. Initially, when getting the odometer readings I did not consider failure handling to ensure these values are numeric. I had to add a “while True” loop to make sure the user entered a number. Additionally, the original plan did not include failure handling to avoid potential division by zero errors. I added code that evaluated the rental period, and if zero, exit the program because if the rental period is zero, no rental occurred. Finally, I added code to assess fuel level and add a fuel charge accordingly as well as code to calculate taxes.

1. Reflection

In implementing this enhancement, I learned a number of lessons as well as enhanced skills that I already had learned. I learned how to critically analyze all aspects of a segment of code. I also learned how to ensure code does not allow for division by zero. I had not previously done this, so it took some research and trial and error to figure out how to implement properly. Additionally, I learned how to ensure an entered value is a specific type using a “while True” loop until the proper data type is entered. I had not previously used “while True” loops, so it took some trial and error to get it to function correctly. I also improved my skills in creating case specific logic in the code that calculates and adds mileage charges based on different mileage traveled scenarios.

The first challenge that I faced was learning how to properly use a “while True” loop. Through research, trial and error, and testing, I was able to properly implement this code. The second challenge I faced was in implementing a failure handling to avoid division by zero errors. With this challenge, I was initially looking at adding additional “while True” loops that would make the code more complex than necessary because I was looking at the code from too low of a level. I stepped back and realized that if the rental period is zero, then by definition no rental had occurred. I was able to simply inform the user that no rental had occurred and gracefully exit the program. This is a good lesson to learn. Sometimes on projects we can get too involved in the low level details of our tasks. Sometimes it is useful to step back and take a more global view of the project and how your task at hand fits in the overall scope of the project for better perspective on the proper solution.