**Sarah Lorenzen 2-1 Journal: Explore Conditional Statements in Programming**

1. A **real-world scenario** that can use conditional statements is ordering alcoholic beverages online. I'm a Hard Cider connoisseur and like to try different brands and flavors from around the world each month. I noticed when I was ordering my selection for September that the site most likely employs conditional logic for age detection and the legal ordering of their products. I had to enter my birthdate and state my age, along with uploading a copy of my Driver's License for age verification.

Using conditional logic to verify the purchase meets the minimum age for purchasing alcohol would solve the problem of compliance with State and Federal alcohol laws.

2. Explain the **implications**of poorly designed conditional statements:

**Code readability:** Overly complex if statements make it difficult for others (or even my future self) to follow the logic.

**Maintenance:** Adding new rules often requires rewriting large chunks of code if the conditionals were not put in a module/function, increasing the likelihood of introducing bugs.

**Performance:** Although most computers can handle conditionals efficiently, there are certain use cases that can add unnecessary computational overhead, such as Fraud checks and their redundant checks.

**Testing complexity:** When logic is a mess, unit testing every branch becomes challenging and increases the likelihood that an error is missed.

3. Explain how you would **implement**the solution using Python code, including the logic behind **if**, **elif**, and **else**statements.

while True:   
    try:  # to avoid crashing if a string is added instead of a number  
        legal\_age = int(input("How old are you? "))

        if legal\_age >= 21:  # check if the purchaser is legal age to purchase alcohol  
            print("Welcome to Cider House!")  
            # add in an HTML button with link to the store  
            break  # Exit loop once condition is met  
        elif legal\_age >= 90: # message for those entering a preposterous age  
            print("Nice try gaming the system, Access denied!")  
        else: # if not legal age, tell them to leave  
            print("Sorry, come back when you're older and wiser!")

    except ValueError: # catches errors for incorrect input format  
        print("Please enter a valid number.")

4. Discuss the **advantages**of using conditionals for the chosen scenario.

In my chosen scenario, the structure of the decision-making process is simple: if the purchaser is of legal purchasing age, then allow them to purchase the alcoholic beverage. The condition that would need to be evaluated would be if the purchaser's age is 21 or older.

For this scenario, conditionals provide:

* **Flexibility:** Different messages can be encoded for multiple ages.
* **Clarity:** The branching structure mirrors real-world “if this, then that” logic.
* **Error handling:** Users receive appropriate feedback, improving the overall experience.
* **Modularity:** Conditionals allow me to isolate logic, making them reusable in other parts of the application.

5. Discuss any **challenges**that could be encountered when applying these structures.

The potential limitations of the solution are if the person was lying about their age. In the actual implementation, I'm sure they use a much more advanced identity verification system to make sure the purchaser is in fact of legal age.

Several challenges arise when using conditionals:

* **Scalability:** As systems grow, too many conditionals can bloat the code.
* **Readability vs. functionality:** Striking a balance between compact code and understandable branching.
* **Edge cases:** Unexpected input can slip through, requiring ValueError handling.