**Python Conditional Logic Challenges (if/elif/else)**

1. **Ticket Price Calculator:**
   * **Task:** Write a program that asks the user for their age.
   * **Logic:** Implement the following pricing tiers:
     + Age 0 to 5: "Free"
     + Age 6 to 18: "5.00"
     + Age 19 to 60: "10.00"
     + Age 61 and above: "Free"
   * **Output:** Print the appropriate ticket price based on the age.
2. **Number Sign Detector:**
   * **Task:** Ask the user to input any integer.
   * **Logic:** Use an if/elif/else structure to determine if the number is **positive**, **negative**, or **zero**.
   * **Output:** Print the corresponding classification (e.g., "The number is positive.").
3. **Password Strength Checker (Length):**
   * **Task:** Ask the user to input a password string.
   * **Logic:**
     + Password length ≥12: "Strong"
     + Password length ≥8 but <12: "Medium"
     + Password length <8: "Weak"
   * **Output:** Print the strength classification.
4. **Traffic Light Simulator:**
   * **Task:** Ask the user to input one of three colors: "red", "yellow", or "green".
   * **Logic:**
     + "red": Print "Stop"
     + "yellow": Print "Proceed with caution"
     + "green": Print "Go"
     + Any other input: Print "Invalid color for a traffic light"
5. **Leap Year Check (Simple):**
   * **Task:** Write a function that takes a year (integer) as input.
   * **Logic:** A year is a leap year if it is divisible by 4.
   * **Output:** Print whether the given year "is a leap year" or "is not a leap year". (Ignore the 100 and 400 rules for this simple exercise).
6. **Case-Sensitive Input:**
   * **Task:** Ask the user to input a programming language name.
   * **Logic:** Use conditional logic to check if the input is exactly "Python". If it is, print "Correct!". If the input is "python" (lowercase), print "Close, but capitalization matters here.". Otherwise, print "Incorrect.".
7. **Odd/Even and Range:**
   * **Task:** Ask the user for a number between 1 and 100.
   * **Logic:**
     + If the number is even and ≥50: Print "High Even"
     + If the number is odd and <50: Print "Low Odd"
     + Otherwise: Print "Other"
     + (Ensure you handle cases where the input is outside 1 to 100 by printing "Out of Range").
8. **Nested Conditional (Login):**
   * **Task:** Define variables username = "admin" and password = "secret". Ask the user for both.
   * **Logic:** Use a **nested if structure**:
     + **Outer if:** Check if the username matches.
     + **Inner if:** If the username is correct, check if the password matches.
     + **Output:** Print "Login Successful" only if both are correct. Otherwise, print "Invalid Username" or "Invalid Password" (depending on where the failure occurs).
9. **Voter Eligibility:**
   * **Task:** Ask the user for their country of residence and age.
   * **Logic:**
     + If the age is <18: Print "Too young to vote."
     + If the age is ≥18:
       - If the country is "USA" or "Canada": Print "Eligible to vote in North America."
       - Otherwise: Print "Eligible to vote, but check local laws."
10. **Tuple Comparison:**
    * **Task:** Given two fixed tuples, t1 = (1, 2, 3) and t2 = (1, 2, 4).
    * **Logic:** Use an if/elif/else chain with direct comparison operators to determine which of the following is true (and print the corresponding message):
      + t1 == t2: "Tuples are identical."
      + t1 < t2: "Tuple 1 is lexicographically smaller."
      + Else: "Tuple 1 is lexicographically greater."

**Practice Questions: For Loops**

1. **Basic List Iteration:**
   * **Task:** Given the list colors = ['red', 'green', 'blue', 'yellow'], write a for loop to print each color on a separate line.
2. **Using range() for Simple Count:**
   * **Task:** Write a for loop using the range() function to print all the integers from 1 up to and including 5.
3. **Using range() with Step Value:**
   * **Task:** Write a for loop using range() to print all the **even numbers** starting from 2 and going up to 20.
4. **String Iteration and Conditional Break:**
   * **Task:** Given the word word = "programming", write a for loop to iterate through the letters. Use a **break** statement to stop the loop and print a message as soon as the letter 'g' is encountered.
5. **Dictionary Iteration (Keys and Values):**
   * **Task:** Given the dictionary grades = {'Alice': 90, 'Bob': 85, 'Charlie': 92}, write a for loop using the .items() method to print each name and their score in the format: "[Name] scored [Score]".
6. **Counting with enumerate():**
   * **Task:** Given the list tasks = ["buy bread", "clean car", "write report"], write a for loop using **enumerate()** to print the task number (starting from 1) and the task name (e.g., "Task 1: buy bread").
7. **Calculating Sum (Accumulator Pattern):**
   * **Task:** Given the list data = [10, 20, 30, 40], initialize a variable total = 0. Write a for loop to iterate over the list and add each number to total. Print the final total.
8. **List Filtering with continue:**
   * **Task:** Given numbers = [1, 5, 8, 12, 15, 20], write a for loop that uses the **continue** statement to skip and not print any number that is divisible by 5. Print all other numbers.
9. **Nested Loops for Grid Printing:**
   * **Task:** Use **nested for loops** to print a 3×3 grid of asterisks (\*). The output should look like this:
   * \*\*\*
   * \*\*\*
   * \*\*\*
10. **Looping Over a Tuple and Conditional Output:**
    * **Task:** Given the tuple items = ('pen', 'pencil', 'eraser', 'sharpener'). Write a for loop to check if the item contains the letter 'e'. If it does, print the item followed by " (has 'e')". Otherwise, print the item alone.

**Practice Questions: While Loops**

1. **Simple Counter:**
   * **Task:** Use a while loop to print the numbers from 1 to 5.
   * **Constraint:** Initialize a variable to 1 and increment it inside the loop.
2. **Counting Down:**
   * **Task:** Use a while loop to print a countdown from 10 down to 1, then print "Liftoff!".
3. **User Input Accumulator:**
   * **Task:** Write a program that repeatedly asks the user to "Enter a number (or 0 to stop):".
   * **Logic:** Use a while loop to keep accepting numbers and add them to a running total until the user enters 0. Print the final total.
4. **Sentinel Value (Quit Command):**
   * **Task:** Write a program that prompts the user for a city name.
   * **Logic:** Use a while loop to continue prompting until the user enters the specific sentinel value "quit". Print a friendly message after the loop ends.
5. **Guessing Game (Limited Tries):**
   * **Task:** Fix a secret number (e.g., 7). Allow the user only 3 attempts to guess it.
   * **Logic:** Use a while loop that continues as long as the user hasn't guessed the number **AND** the number of tries is less than 3. Use break if the user guesses correctly.
6. **Input Validation with while True and break:**
   * **Task:** Write a program that uses an infinite while True loop to ask the user to input an **even number**.
   * **Logic:** If the input is valid (even), use break to exit the loop. If it's invalid (odd), print an error and continue the loop. (Assume the input is always a valid integer).
7. **Skipping with continue:**
   * **Task:** Use a while loop to print the numbers from 1 to 10.
   * **Logic:** Use a **continue** statement to skip and **not print** the number 5.
8. **List Processing with pop():**
   * **Task:** Given a list of names, names = ['Alice', 'Bob', 'Charlie', 'David'].
   * **Logic:** Use a while loop that continues as long as the list is **not empty**. Inside the loop, use the .pop(0) method to remove and print the first name in the list.
9. **Factorial Calculation:**
   * **Task:** Write a program that calculates the factorial of a positive integer N (e.g., 5!=5×4×3×2×1).
   * **Logic:** Use a while loop starting from N and decrementing down to 1, accumulating the product in a result variable.
10. **Generating Powers of Two:**
    * **Task:** Write a program to print the first 5 powers of 2 (20,21,22,23,24).
    * **Logic:** Initialize a variable to 1 and multiply it by 2 in each iteration of a while loop, keeping a separate counter to track the number of iterations.

**Practice Questions: match/case Statement**

1. **Simple Value Matching:**
   * **Task:** Write a function day\_of\_week(day\_num) that takes an integer (1 to 7) and uses a match statement to print the corresponding day name (e.g., 1 is "Monday", 7 is "Sunday").
   * **Default:** If the input is outside 1−7, use a wildcard case (\_) to print "Invalid day number."
2. **Color Code Check (OR Pattern):**
   * **Task:** Write a function check\_traffic\_signal(color) that takes a string color.
   * **Pattern:** Use a single case pattern to match either "red" or "stop". If matched, print "Halt!". If the color is "green", print "Go!".
   * **Default:** Use the wildcard to print "Check light".
3. **Point Coordinate Matching (Tuple Pattern):**
   * **Task:** Write a function describe\_point(point) that takes a 2D tuple (e.g., (x,y)).
   * **Pattern:**
     + If the point is the origin (0,0): Print "Origin".
     + If the point is on the X-axis (e.g., (5,0)): Print "On X-axis".
     + If the point is on the Y-axis (e.g., (0,5)): Print "On Y-axis".
     + Otherwise: Print "In a quadrant".
4. **Action Command Matching (List Pattern):**
   * **Task:** Write a function execute\_command(command) that takes a list of strings representing a command (e.g., ["move", "north"]).
   * **Pattern:**
     + Match the list ["move", direction], binding the second element to the variable direction. Print "Moving [direction]".
     + Match the list ["open", "door"]. Print "Door opened!".
     + Otherwise: Print "Unknown command."
5. **Data Type Validation (Class Pattern):**
   * **Task:** Write a function check\_data\_type(data) that takes any variable.
   * **Pattern:** Use match with a class pattern to check the type:
     + Match if the data is an int: Print "Integer value."
     + Match if the data is a list: Print "List structure."
     + Match if the data is a str: Print "String sequence."
     + Otherwise: Print "Other data type."
6. **HTTP Status Check (Guard Clause if):**
   * **Task:** Write a function handle\_http\_status(status\_code) that takes an integer.
   * **Pattern:** Use a match case that binds the status code to a variable (e.g., case code:), and then use a **guard clause (if)** to check for specific ranges:
     + Match any code if code >= 200 and code < 300: Print "Success".
     + Match any code if code >= 400 and code < 500: Print "Client Error".
     + Otherwise: Print "Informational or Server Error".
7. **Single/Multiple Items (List Length Pattern):**
   * **Task:** Write a function check\_items(items) that takes a list.
   * **Pattern:**
     + Match an empty list []: Print "No items."
     + Match a list with one item [item]: Print "One item: [item]".
     + Match a list with two or more items [a, b, \*rest]: Print "Multiple items."
8. **Dictionary Matching (Key Existence):**
   * **Task:** Write a function process\_user(user\_dict) that takes a dictionary.
   * **Pattern:**
     + Match if the dictionary contains the key "role" with the value "admin": Print "Access granted."
     + Match if the dictionary contains the key "role" with any other value: Print "Access denied."
     + Otherwise (no "role" key): Print "Role undefined."
9. **Function Call Dispatcher:**
   * **Task:** Write a function run\_operation(operation\_tuple) that takes a tuple (e.g., ("add", 10, 5)).
   * **Pattern:**
     + Match ("add", x, y): Print the result of x+y.
     + Match ("subtract", x, y): Print the result of x−y.
     + Otherwise: Print "Unsupported operation."
10. **Tuple Range Matching (Guard with Variable Binding):**
    * **Task:** Write a function classify\_tuple(data) that takes a 2D tuple.
    * **Pattern:**
      + Match (x, y) and use a **guard clause** to check if *both* x and y are greater than 10. If so, print "High Range".
      + Otherwise (wildcard): Print "Mixed or Low Range".