PROGRAMMING 0
LAB-02

Dr Richard Holden

Table of Contents

Summary

This lab focuses on teaching the fundamentals of binary string conversion to and from decimal values. It consists of a series of exercises designed to help students understand the use of if statements, loops, and user input handling in Python. The exercises range from specific binary-to-decimal conversion with constraints (less than or equal to 8 bits), to more generalized solutions, then decimal-to-binary conversion. Additionally, students are encouraged to practice writing clean, well-commented code and sharing their solutions with peers for collaborative learning.

Learning Objectives:

By the end of this lab, you should be able to:

- Understand and Use If Statements in Python:
    - Learn how to implement conditional logic to handle binary string conversion with `if` statements.
- Handle User Input:
    - Gain experience in prompting for and validating user input, especially for binary strings and integers.
- Differentiate Between Specific and General Solutions:
    - Appreciate the differences between specific (constrained) solutions and more flexible (general) solutions when converting binary strings to decimal integers.
- Gain an Initial Understanding of Loops (for and while):
    - Use `for` loops to traverse binary strings for conversion.
    - Implement `while` loops for decimal-to-binary conversion using repeated division.
- Comment and Document Code:
    - Develop the habit of writing clear and descriptive comments in code to improve readability and maintainability.

Exercise 1: binary string I (8 bits or less)

Create a Python script named *to_integer_1.py* that converts a binary string into its corresponding integer value using conditional logic (specifically, using **if** statements).

Details:

- The script should handle binary strings with a maximum length of 8 bits (or less).
- Pad the string with additional bits, if the length of the string is less than 8
- For binary strings, the leftmost bit represents the most significant bit (MSB), and the rightmost bit represents the least significant bit (LSB).
- You could use if statements to check each bit's position and calculate its contribution based on the associated exponent.
- Ensure you sum up the contributions to obtain the final integer value.

This script will allow you to convert binary strings into integers, considering the significance of each bit position, as described in the task.

Exercise 2: binary string II (8 bits or less)

Create a Python script named *to_integer_2.py* that converts an 8-bit binary string into its corresponding integer value. The functionality of this script will be the same as *to_integer_1.py*. However, the code will look very different; rather than using if logic, please use a for loop to convert the string.

You might find this script more difficult to implement?

Exercise 3: binary string III (arbitrary length string)

Create a Python script named *to_integer_3.py* that converts an n-bit binary string into its corresponding integer value. Again, the binary string will represent a binary number, where each bit's position determines its significance in the number, similar to Exercise 2. However, this time check that the input string contains characters of ‘0’ or ‘1’, thereby handing inappropriate input, then implement the for loop on the string and print the value.

Exercise 4: convert a decimal number to a binary number

Inside a script called *to_binary.py*, convert a decimal integer to a binary number. Receive input from the user and ensure that the input is positive and then convert it to its binary representation. If the input is not positive, it displays an error message.

- Start with the decimal number.
- Initialize an empty binary string.
- Perform Repeated Division by 2:
    - Divide the decimal number by 2.
    - Record the remainder.
    - Append the remainder to the beginning of the binary string.
    - Repeat until the decimal number becomes 0.

This process converts a positive decimal to binary by repeatedly dividing by 2 and recording remainders until the decimal number becomes 0.

Example, using the decimal number 10:

1. Start with the decimal number 10.
2. Initialize an empty binary string: ""
3. Perform Repeated Division by 2:
    a. 10 divided by 2 = 5, remainder 0. Insert "0" to the beginning of string.
        i. Resulting Binary string: "0"
    b. 5 divided by 2 = 2, remainder of 1. Insert "1" to the beginning of string.
        i. Resulting Binary string: "10"
    c. 2 divided by 2 = 1, remainder of 0. Insert "0" to the binary string.
        i. Resulting Binary string: "010"

        d. 1 divided by 2 = 0, remainder 1. Insert "1" to the binary string.
            i. Resulting Binary string: "1010"

4. Binary Representation is Complete: The binary representation of 10 is "1010"

Exercise 5: Comment all your code

It is important to comment code! You should ensure that you 1. go over each script, 2. clean up any code and 3. add comments.

Exercise 6: Share you code

Share your code with other students; you can learn how they approached the problem.