“**Module 2: Lab 1 - Numbering System Conversion”**

Harsh Siddhapura

“Department of Information Technology, Arizona State University”

“IFT 510: Principles of Computer Information and Technology”

“Dr. Dinesh Sthapit”

“August 26, 2023”

Summary

Numbering System Conversion

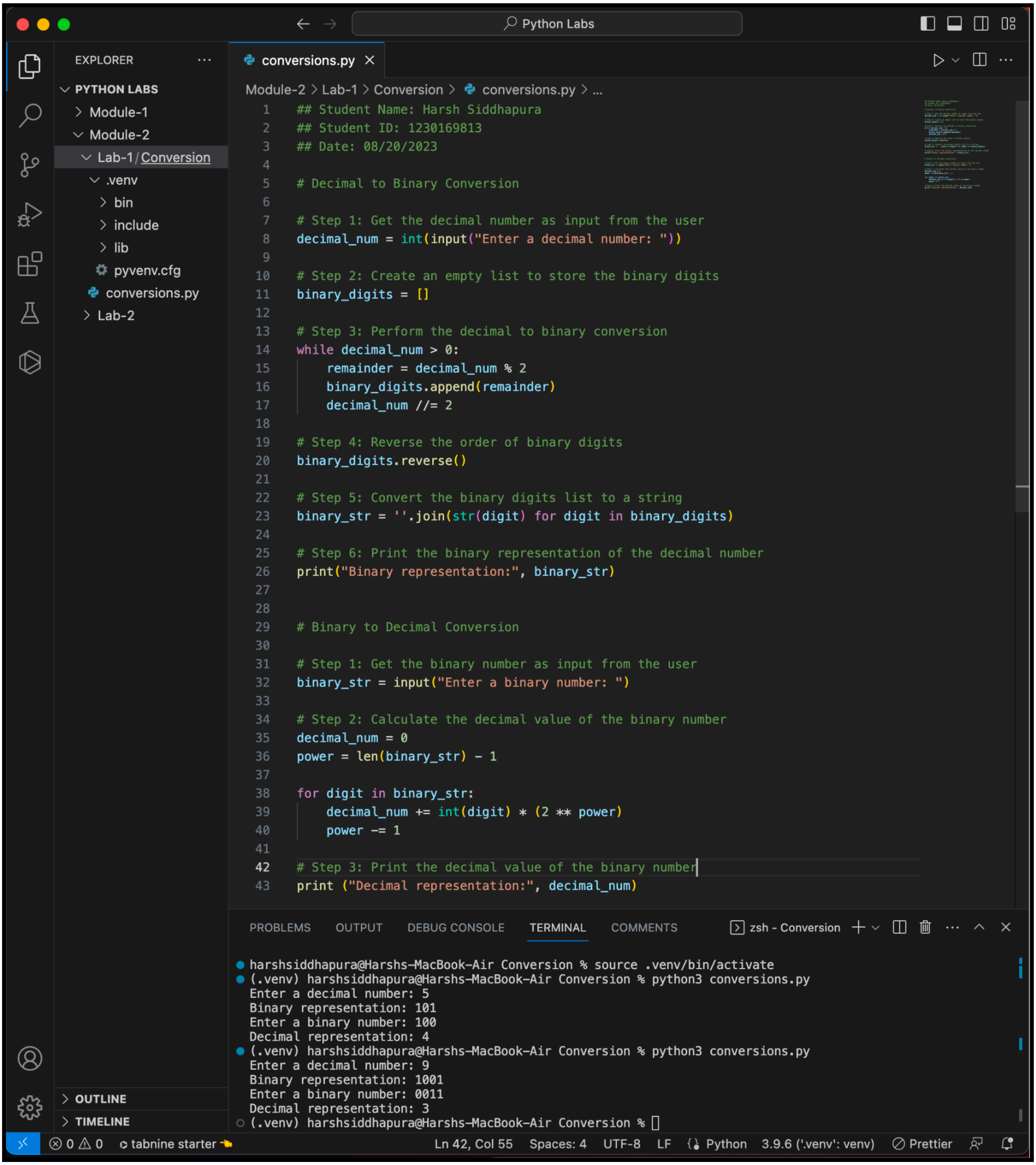
The provided code offers a practical implementation of converting numbers between the decimal and binary number systems. The code is divided into two distinct sections, each responsible for a specific conversion direction: from decimal to binary and from binary to decimal.

The decimal to binary conversion process begins with user input, where the user is prompted to provide a decimal number. To facilitate the conversion, an empty list named `binary_digits` is introduced to hold the binary representation's individual digits. Employing a loop, the code iterates by dividing the decimal number by 2 repeatedly. The remainder of each division, which can be either 0 or 1, is appended to the `binary_digits` list. Once the loop concludes, the order of these binary digits is reversed to ensure the correct sequence. The final binary representation is achieved by joining the binary digits into a single string, which is then presented as the converted binary form of the originally input decimal number. This method showcases the systematic process of converting decimal numbers into binary format using fundamental arithmetic operations and organised data structures.

The binary to decimal conversion procedure commences by prompting the user to input a binary number, presented as a string. Within this process, two variables come into play: `decimal_num`, which serves as the repository for the resultant decimal value, and `power`, initialised to indicate the exponent of 2 during the conversion. The essence of the conversion lies in the sequential examination of each digit within the binary string. The code progresses from the leftmost to the rightmost digit, wherein the integer value of each digit is multiplied by 2 raised to the power of `power`. The resulting product is then added to the ongoing accumulation in `decimal_num`. Following the processing of each digit, the `power` variable is decremented, facilitating the transition to the subsequent lower exponent of 2. Ultimately, the converted decimal value is produced as output, effectively representing the equivalent decimal interpretation of the originally supplied binary number. This methodology underscores the systematic strategy employed to convert binary sequences into their corresponding decimal counterparts, utilising basic mathematical operations and structured variable management.

This code presents valuable insights into the practical application of fundamental arithmetic operations, loops, and string manipulation techniques to facilitate seamless conversions between distinct numerical systems. It underscores the iterative nature that underpins such conversions and emphasises the critical aspect of preserving the correct sequence of digits throughout the process. The incorporation of user interaction through input prompts contributes to the code's user-friendly nature, enhancing its accessibility. As for potential enhancements, it's important to note that the code operates under the assumption of valid user inputs. Incorporating input validation mechanisms could bolster its robustness. While the code's functionality is accurate, the inclusion of supplementary comments and explanations would likely enhance its comprehensibility for readers. Additionally, implementing error-handling mechanisms to address scenarios involving invalid inputs, such as non-numeric or non-binary characters, would further elevate the code's dependability and usability.

The code effectively demonstrates the conversion processes between decimal and binary number systems. By breaking down the conversions into step-by-step procedures, it provides a clear example of how numerical systems can be transformed using basic mathematical operations and programming constructs.



```python
## Student Name: Harsh Siddhapura
## Student ID: 1230169813
## Date: 08/20/2023

# Decimal to Binary Conversion

# Step 1: Get the decimal number as input from the user
decimal_num = int(input("Enter a decimal number: "))

# Step 2: Create an empty list to store the binary digits
binary_digits = []

# Step 3: Perform the decimal to binary conversion
while decimal_num > 0:
    remainder = decimal_num % 2
    binary_digits.append(remainder)
    decimal_num //= 2

# Step 4: Reverse the order of binary digits
binary_digits.reverse()

# Step 5: Convert the binary digits list to a string
binary_str = ''.join(str(digit) for digit in binary_digits)

# Step 6: Print the binary representation of the decimal number
print("Binary representation:", binary_str)


# Binary to Decimal Conversion

# Step 1: Get the binary number as input from the user
binary_str = input("Enter a binary number: ")

# Step 2: Calculate the decimal value of the binary number
decimal_num = 0
power = len(binary_str) - 1

for digit in binary_str:
    decimal_num += int(digit) * (2 ** power)
    power -= 1

# Step 3: Print the decimal value of the binary number
print ("Decimal representation:", decimal_num)
```

```
harshsiddhapura@Harshs-MacBook-Air Conversion % source .venv/bin/activate
(.venv) harshsiddhapura@Harshs-MacBook-Air Conversion % python3 conversions.py
Enter a decimal number: 5
Binary representation: 101
Enter a binary number: 100
Decimal representation: 4
(.venv) harshsiddhapura@Harshs-MacBook-Air Conversion % python3 conversions.py
Enter a decimal number: 9
Binary representation: 1001
Enter a binary number: 0011
Decimal representation: 3
(.venv) harshsiddhapura@Harshs-MacBook-Air Conversion %
```