## ◇ Part 1: Model Questions (Identity, Membership, Bitwise)

**Q1.** State the difference between `is` and == operators in Python. Give an example for each.

**Q2.** Predict the output:

```
x = [1, 2, 3]
y = [1, 2, 3]
print(x is y)
print(x == y)
```

**Q3.** Which operator would you use to check if None is assigned to a variable? Give an example.

**Q4.** Write an expression to test if `"a"` exists in the string `"apple"`.

**Q5.** Evaluate:

```
"py" in "python"
"on" not in "python"
```

**Q6.** Write the bitwise AND, OR, and XOR results for 12 and 6.

**Q7.** Convert `0b1010 & 0b1100` into decimal.

**Q8.** Predict the output:

```
a = 5
b = 5
print(a is b)
print(a == b)
```

**Q9.** Which operator would you use to check whether an element belongs to a list? Write an example.

**Q10.** Explain the result of:

```
x = 4
print(x << 1)
print(x >> 1)
```

## ◇ Part 2: Program Development Problem

# Problem Definition

You are asked to develop a **Python program** that can **encrypt** a message and then **decrypt** it back to its original form.

- The program must use **bitwise operators** for the encryption and decryption.
- The program should only allow **letters** in the message to be encrypted.
- You must demonstrate the use of **identity operators (`is`, `is not`)** and **membership operators (`in`, `not in`)** in appropriate places.

- The program must run **sequentially** (no functions or advanced structures like OOP are required).

# 🧭 Guidelines for Flow of the Program

1. **Input / Setup**
    a. Define a plain text message (e.g., `"HELLO"`).
    b. Choose a numeric key (small integer, e.g., `5`) to be used in the encryption/decryption process.
2. **Encryption Process**
    a. Go through each character of the message one by one.
    b. Check if the character **belongs to the set of alphabetic characters** (`isalpha()` or `"A" in ...`).
    c. If it is a valid character, convert it into its ASCII code (`ord()`), then apply **bitwise XOR (^) with the key** to transform it.
    d. Store the encrypted values in a list (this becomes your ciphertext).
3. **Decryption Process**
    a. Go through each value in the encrypted list.
    b. Reapply the same **bitwise XOR (^) with the same key** to get back the original ASCII value.
    c. Convert it back to a character (`chr()`).
    d. Append each decrypted character to form the original message again.
4. **Use of Identity Operators**
    a. Demonstrate by checking if any variable is `None` before using it (though in this program it is unlikely to be `None`, still a check like `if val is None:` can be included for exam purposes).
5. **Output**
    a. Display the encrypted list of numbers.
    b. Display the decrypted string, which should exactly match the original message.

## 🎯 Expected Output Example

If the original message is "HELLO" and the key is 5:

```
Encrypted: [77, 64, 73, 73, 74]
Decrypted: HELLO
```