1.Problem: Caesar Cipher encoding and decoding

```
def caesar_cipher(text, shift, encode=True):
    result = []
    shift = shift % 26
    if not encode:
        shift = -shift

    for char in text:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            shifted = (ord(char) - base + shift) % 26 + base
            result.append(chr(shifted))
        else:
            result.append(char)
    return ''.join(result)
```

2. Problem: Convert number into a comma separated Indian currency format

```
def format_indian_currency(amount):
    amount_str = f"{amount:.10f}".rstrip('0').rstrip('.')
    if '.' in amount_str:
        int_part, dec_part = amount_str.split('.')
    else:
        int_part, dec_part = amount_str, ''

    if len(int_part) > 3:
        prefix = int_part[:-3]
        suffix = int_part[-3:]
        grouped = ''
        while len(prefix) > 2:
            grouped = ',' + prefix[-2:] + grouped
            prefix = prefix[:-2]
        grouped = prefix + grouped
        int_part = grouped + ',' + suffix

    return int_part + ('.' + dec_part if dec_part else '')
```

3.Problem: Combining two lists

```
def combine_lists(list1, list2):
    combined = sorted(list1 + list2, key=lambda x: x["positions"][0])
    result = []

    for item in combined:
        if not result:
            result.append(item)
        else:
            last = result[-1]
```

```
            l1, r1 = last["positions"]
            l2, r2 = item["positions"]

            # Overlap condition: more than half of item fits within last
            overlap = max(0, min(r1, r2) - max(l1, l2))
            length = r2 - l2
            if overlap > length / 2:
                last["values"].extend(item["values"])
            else:
                result.append(item)

    return result
4.Problem: Minimizing Loss
def minimize_loss(prices):
    indexed_prices = list(enumerate(prices))
    sorted_prices = sorted(indexed_prices, key=lambda x: x[1])

    min_loss = float('inf')
    buy_year, sell_year = -1, -1

    for i in range(len(sorted_prices) - 1, 0, -1):
        i1, p1 = sorted_prices[i]
        i2, p2 = sorted_prices[i - 1]

        if i2 < i1:  # Ensure buy happens before sell
            loss = p1 - p2
            if 0 < loss < min_loss:
                min_loss = loss
                buy_year = i2 + 1  # +1 for 1-based indexing
                sell_year = i1 + 1

    return {"buy_year": buy_year, "sell_year": sell_year, "loss": min_loss}
```