Author: Dávid Kromka

Task 1- Encryption

Algorithm description

There are three functions in main.py, `taskA`, `taskB`, and `taskC`, that perform different operations on an input string and return results along with their execution times. Let's describe each of these functions:

Task 1a

taskA(input_str):

- This function takes an input string `input_str` as its parameter.
- It records the start time using `time.perf_counter()`.
- It converts the input string to uppercase using `input_str.upper()`.
- It initializes three empty strings: `result_order`, `result_reverse`, and an empty list `result_reverse_array`.
- It iterates through each character in the input string.
- If the character is a letter (determined using `char.isalpha()`), it calculates two numbers:
- `number_in_order`: The position of the letter in the alphabet (1 for 'A', 2 for 'B', etc.).
- `number_in_reverse`: The position of the letter when counting backward from 'Z' (26 for 'A', 25 for 'B', etc.).
- It appends these numbers to `result_order` and `result_reverse` as strings.
- It also appends `number_in_reverse - 1` to `result_reverse_array`.
- If the character is not a letter, it appends it as is to `result_order`, `result_reverse`, and appends it to `result_reverse_array`.
- It records the end time and calculates the execution time in milliseconds.
- It returns four values: `result_order`, `result_reverse`, and the result of converting `result_reverse_array` back to letters using the `numbers_to_letters` function, along with the execution time.

Task 1b

taskB(input_str):

- This function takes an input string `input_str` as its parameter.
- It performs similar operations to `taskA` with some differences:
- It calculates `number_in_reverse` for each letter.
- It calculates `half_number` based on whether `number_in_order` is less than or equal to 13.
- It appends `half_number` to the `b_result_array`.
- It records the end time and calculates the execution time in milliseconds.
- It returns the result of converting `b_result_array` back to letters using the `numbers_to_letters` function, along with the execution time and b_result_array.

Task 1c

taskC(input_str, k):

- This function takes two parameters: `input_str` (an input string) and `k` (an integer).
- It converts the input string to uppercase.

- It initializes an empty list `c_result_array`.
- It iterates through each character in the input string:
- If the character is a letter, it calculates a new number based on the letter's position in the alphabet and the value of `k`.
- It appends this new number to the `c_result_array`.
- If `k + ord(char) - ord('A') + 1` exceeds 26, it wraps around to the beginning of the alphabet.
- It records the end time and calculates the execution time in milliseconds.
- It returns the result of converting `c_result_array` back to letters using the `numbers_to_letters` function, along with the execution time and c_result_array.

numbers_to_letters(result_reverse_array):

- This is a helper function that takes a list of integers `result_reverse_array`, where each integer corresponds to a letter's position in the alphabet.
- It converts each integer back to the corresponding uppercase letter and joins them together to form a string.
- It returns the resulting string.

Task 2- Decryption

Task A Decrypt Algorithm:

1. **Input:** Accepts a string containing space-separated numbers.
2. **Processing:**
    - Splits the input string into a list of numbers.
    - Decrypts each number by subtracting it from 26.
    - Stores the decrypted numbers in an array.
3. **Output:** Converts the decrypted numbers back to letters using a function **numbers_to_letters**.
4. **Execution Time:** Records the time taken for decryption in milliseconds.

Task B Decrypt Algorithm:

1. **Input:** Accepts a string containing space-separated numbers.
2. **Processing:**
    - Splits the input string into a list of numbers.
    - Decrypts each number differently based on its value:
        - If the number is less than or equal to 13, subtracts it from 13.
        - If the number is greater than 13, subtracts it from 13 and adds 26.
    - Stores the decrypted numbers in an array.
3. **Output:** Converts the decrypted numbers back to letters using a function **numbers_to_letters**.
4. **Execution Time:** Records the time taken for decryption in milliseconds.

Task C Decrypt Algorithm:

1. **Input:** Accepts a string containing space-separated numbers and an integer k.
2. **Processing:**
    - Splits the input string into a list of numbers.
    - Decrypts each number based on the value of k:
        - If the result of subtracting k from the number is greater than 0, subtracts k and 1 from the number.
        - If the result is less than or equal to 0, adds 25 to the number and then subtracts k.
    - Stores the decrypted numbers in an array.
3. **Output:** Converts the decrypted numbers back to letters using a function **numbers_to_letters**.
4. **Execution Time:** Records the time taken for decryption in milliseconds.

Test for task 1 and task 2

Input string is my surname.

Enter a string to encrypt or space separated numbers to decrypt: kromka

False

Task A:

Result in order: 11 18 15 13 11 1

Result in reverse order: 16 9 12 14 16 26

Letters from reverse order: PILNPZ

Execution Time: 0.0124 milliseconds

Task B:

Result: CVYACM

Numbers result: 3 22 25 1 3 13

Execution Time: 0.0036 milliseconds

Task C:

Result with k = 11: VCZXVL

Numbers result with k = 11: 22 3 26 24 22 12

Execution Time : 0.0029 milliseconds

Enter a string to encrypt or space separated numbers to decrypt: 22 3 26 24 22 12

True

Task A decryption: EXACEO

Execution Time : 0.017 milliseconds

Task B decryption: RKNPRB

Execution Time : 0.0052 milliseconds

Task C decryption: KROMKA

Execution Time : 0.0047 milliseconds

Enter a string to encrypt or space separated numbers to decrypt:

Note: Execution time is real time, no processor time, because it was too small value to record.