Lada Ivanna Matveeva AIS ID 99198

Report Lab 2

When the program starts, the **initialize()** function is called in which the alphabet is initialized in shifted order for encryption.

Parameters:
K - The value of K is individual for every student as it was given at the start of the semester. The value of my K is 2.

```
# Initializes dictionaries for shifted order of encoding message
def initialize(K):  1 usage
    shifted_order = {}

    for i in range(26):
        letter = chr(ord('a') + i)

        shifted_value = (i + 1 + K) % 26
        if shifted_value == 0:
            shifted_value = 26
        shifted_order[letter] = shifted_value

    return shifted_order
```

The program has two execution modes:

1. Input of an decrypted string.
2. Input of an encrypted message with a delimiter ",".

Input of an decrypted string

The entered string is transformed into lowercase for further use.

```
    user_input = user_input.lower()
```

The **encoding()** function is called.
Parameters:
text - the text entered by the user
letters - dictionary with the order of encryption alphabet

```
def encoding(text, letters):
    message = []
    for i in text:
        direct_number = letters[i]
        message.append(direct_number)
    return message
```

In the function, a loop finds the corresponding ordinal number for each letter in the text. The function returns an array of the encoded message.

Then, we write out the message.

```
print("[{:.3f}ms] {} message:\t\t{} {}".format( *args: duration(start_time, end_time) * 1000, function_message,
                                               input_message, output))
```

Input of an encrypted string

When inputting a message in the form of an encoded message, the delimiter **","** is removed from the string, and the message is saved into an array.

```
user_input_array = [int(num) for num in user_input.split(',')]
```

Then, the **decoding()** function is called.

Parameters:
text - the text entered by the user
letters - dictionary with the order of the alphabet

```
def decoding(text, letters):
    decoded_message = "".join(find_key_by_value(letters, num) for num in text)
    return decoded_message
```

In this function, the function **find_key_by_value()** is called.
Parameters:
dictionary - dictionary with the direct order of encryption alphabet
search_value - number from the encoded message

```
def find_key_by_value(dictionary, search_value):
    for key, value in dictionary.items():
        if value == search_value:
            return key
    return None
```

Using a loop, we find the letter corresponding to the number.

Then, we write out the message.

```
print("[{:.3f}ms] {} message:\t\t{} {}".format( *args: duration(start_time, end_time) * 1000, function_message,
                                                input_message, output))
```

Execution time of the program

The execution time of the program is calculated as follows:
Before executing the program, we save the start time and at the end, we save the end time.

```
start_time_direct = time.perf_counter()
```

```
end_time_direct = time.perf_counter()
```

Then, in function **duration()** we calculate the difference between the end and the start.

```
def duration(start_time, end_time):
    return end_time - start_time
```

Example of use

When the program starts, an example of what data can be entered will first be listed.

```
Example:
      decoded message: matveeva
      encoded message: 15,3,22,24,7,7,24,3
```

Then we can write our data:

Decoded message. When the program finishes, we can see the shifted method of alphabet encoding listed. At the beginning of the line, the execution time will be written. Then, we see the message and its encoding.

```
Please enter surname: matveeva
[0.008ms] Shifted message:      matveeva [15, 3, 22, 24, 7, 7, 24, 3]
```

Encoded message:

```
Please enter surname: 15,3,22,24,7,7,24,3
[0.029ms] Shifted message:      [15, 3, 22, 24, 7, 7, 24, 3] matveeva
```