```
import string

import collections


# List of the most common English letters for frequency substitution

english_common = ['E', 'T', 'A']


# --- Read ciphertext from file ---

with open("ciphertext.txt", "r") as file:

    ciphertext = file.read().strip()  # Remove any leading/trailing whitespace

print("Ciphertext:", ciphertext)


# --- Count letter frequency ---

def count_letter_frequency(ciphertext):

    """

    Counts the frequency of each letter (A-Z) in the ciphertext.

    Non-letter characters are ignored.

    Returns a dictionary with letters as keys and counts as values.

    """

    # Keep only alphabetic characters

    ciphertext = ''.join([c for c in ciphertext if c in string.ascii_letters])

    # Count frequency of each letter (case-insensitive)

    frequency = collections.Counter(ciphertext.upper())

    # Make sure every letter A-Z is represented, even if frequency is 0

    full_freq = {letter: frequency.get(letter, 0) for letter in string.ascii_uppercase}

    return full_freq


# --- User substitution (protecting certain letters) ---

def user_replace_letter(ciphertext):
```

```
"""
Allows the user to manually replace letters in the ciphertext.
Protects letters 'E', 'T', 'A' from being replaced.
"""
ciphertext = ciphertext.upper()
protected_letters = ['E', 'T', 'A']  # Letters that cannot be changed
print(f"Current ciphertext: {ciphertext}")

while True:
    # Ask user for the letter they want to replace
    old_letter = input("Enter the letter you want to replace (or 'exit' to finish): ").upper()
    if old_letter == 'EXIT':
        break
    if old_letter in protected_letters:
        print(f"Letter '{old_letter}' is protected and cannot be replaced. Please try again.")
        continue
    if old_letter not in ciphertext:
        print(f"Letter '{old_letter}' not found in the ciphertext. Please try again.")
        continue

    # Ask for the new letter
    new_letter = input(f"Enter the new letter to replace '{old_letter}': ").upper()
    if len(new_letter) != 1 or new_letter not in string.ascii_letters:
        print("Please enter a single valid letter.")
        continue

    # Replace all occurrences of old_letter with new_letter
    ciphertext = ciphertext.replace(old_letter, new_letter)
    return ciphertext


# --- Atbash cipher mapping ---

def atbash_c_
```