```
"""
This module is responsible for the program flow.
In addition it is in charge of the formatting of the user's text.
"""

import screen_module
import text_module


def normalise_letter(x):
    """
    normalisation of the given letter x if it is a special character
    the cases of French special characters and German "Umlaute" are shown
    :param: x is the given letter
    :return: chr
    """
    # checks if given character is part of the alphabet
    # if yes returns the letter, if not normalise it
    if ('a' <= x <= 'z') or ('A' <= x <= 'Z'):
        return x
    elif x in "àâ":
        return 'a'
    elif x in "æ":
        return "ae"
    elif x in "ç":
        return 'c'
    elif x in "èéêë":
        return 'e'
    elif x in "îï":
        return 'i'
    elif x in "ô":
        return 'o'
    elif x in "ùûü":
        return 'u'
    elif x in "ÿ":
        return 'y'
    elif x in "œ":
        return "oe"
    else:
        return "\nSome problem occurred."


```

```
43 def format_text(text):
44     """
45     formats the given text (deleting numbers, white spaces
   , etc.)
46     :param: text is the text entered by the user which has
    to be formatted to en-/decrypt it.
47     :return: string
48     """
49     # all characters of the given text are saved in a list
50     text_list = [x for x in text]
51     i = 0
52     while i < len(text_list):
53         # check if character is part of the alphabet
54         if text_list[i].isalpha():
55             # call normalise function to normalise special
    characters
56             text_list[i] = normalise_letter(text_list[i])
57         else:
58             # if character is not in the alphabet (such as
    digits) it is replaced by ''
59             text_list[i] = ''
60         i += 1
61     else:
62         # join the characters to a text to return it
63         text = ''.join(text_list)
64         # capitalise text
65         text = text.upper()
66         return text
67
68
69 # IMPORTANT ANNOTATION:
70 # at the moment the function "run()" is still under
   construction and NOT doing what it should do
71 def run():
72     """
73     starts the program
74     :return: None
75     """
76     # initialise variables with default values (language:
   English, chosen mode: encryption)
77     english = True
78     encryption = True
79
80     # set language
81     if screen_module.show_start_ask_language() == 'f':
```

```
 82         english = False
 83
 84     while True:
 85         # show main menu and set encryption variable
 86         if screen_module.show_main_menu(english) == 'd':
     # decryption
 87             encryption = False
 88         elif screen_module.show_main_menu(english) == 's'
    :  # settings
 89             if screen_module.show_language_settings() ==
    'f':  # change language to French
 90                 english = False
 91                 continue  # restart with the main menu (
    now in French)
 92         elif screen_module.show_main_menu(english) == 'q'
    :  # quit program
 93             screen_module.show_quit_message(english)
 94             break  # stop program
 95
 96         # set principle for en-/decryption
 97         principle = screen_module.show_principles(english
    , encryption)
 98         if principle == 'm':  # main menu
 99             continue  # go back to main menu
100         else:
101             # set key for en-/decryption
102             key = screen_module.show_ask_key(english,
    principle)
103             if key == 'm':  # main menu
104                 continue  # go back to main menu
105             else:
106                 # set text for en-/decryption
107                 text = screen_module.show_ask_text(
    english)
108                 if text == 'm':  # main menu
109                     continue  # go back to main menu
110                 else:
111                     # call en-/decryption function
112                     if principle == 'c':
113                         if encryption:
114                             text = text_module.
    encrypt_caesar(text, key)
115                         else:
116                             text = text_module.
    decrypt_caesar(text, key)
```

```
117                     elif principle == 'v':
118                         if encryption:
119                             text = text_module.
    encrypt_vigenere(text, key)
120                         else:
121                             text = text_module.
    decrypt_vigenere(text, key)
122                     elif principle == 'e':
123                         text = text_module.enigma(text,
    key)
124                     else:
125                         return "This should never happen"
126                     screen_module.show_treated_text(
    english, encryption, text)
127                     # program waits for the user's input
    and goes back to the main menu, no matter what the input
    was
128
129
130 # start program
131 run()
132
```