author = '8175858, Braun'

Analysis:

The program is developed according to the EVA design pattern.

Output: Output is done in the console.

Description of the program:

The program does not require any further libraries.
The program is written and tested in Python 3.9.6. A Python interpreter must be installed.
The program can be started with the command "python3 ue07_happy_strings.py" in the terminal.

is_happy_string(s:str) checks if a string is happy or not. It returns a True or False. It uses a dictionary to keep track of the number of occurences of a character in the given string. If a character's counter is not even, it is not a happy string. If all characters are even, it is a happy string.

find_happy_pairs_helper(s: str, i: int, happy_pairs: set) is the recursive helper function for find_happy_pairs(s: str). It calls is_happy_string() to checks if the string is a happy string. If it is, it adds it to the happy_pairs set. If it is not, it calls itself and adds 1 to the index i which is stop index for the string slicing. It does this for every character in the string. Base Case is when i is equal to the length of the string for which it checks if the whole string s is a happy string.

find_happy_pairs(s: str) is the function that calls find_happy_pairs_helper(s: str, i: int, happy_pairs: set).
It goes through the string and uses slicing to slice of the first i characters the string, then gives that to the helper function. It does this for every character in the string. It returns the length happy_pairs set.

The happy_pairs set is used to keep track of the happy pairs. It is a set because it does not allow duplicates. The index i is used to slice the string.

Testing:

Tests done in the main function.

find_happy_pairs('20230322') => 4
find_happy_pairs('3190394') => 0
find_happy_pairs('1234556') => 1