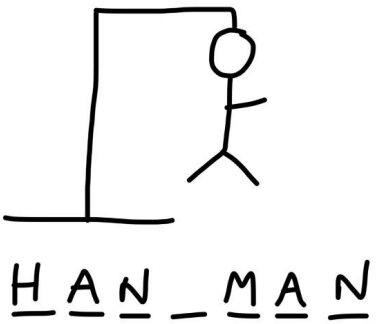
Lab 2 : Hangman



In this assignment, you will recreate the classic "Hangman" game.

Rules of the game

- The computer picks a random word, and displays it with all letters replaced with `_`.
- The game has 10 turns maximum.
- At each turn:
    - The player is asked for a letter.
    - If the letter was already tried before, ask for another letter
    - Otherwise, display the "hint". For each letter in the word:
        - If the letter was tried by the player, display it (uppercase)
        - Otherwise, display `_`
    - Display the number of turns remaining
- The game stops when:
    - all the letters are revealed
    - OR
    - the maximum number of tries has been reached.

Step 1

- Write the code to make the game work as required.

Step 2: clean the code / think like a programmer

You may have created a fully functional program, but is it good? Remember: good code uses reusable functions, and is well organized.

One way to write good code is to think about your objectives before you start writing actual code. The flow of the program can be represented as follows:

```
TURNS_LEFT <-- NUMBER 10
MY_WORD <-- PICK_A_RANDOM_WORD()
MY_LETTERS <-- EMPTY LIST []
WHILE TURNS_LEFT > 0:
  LETTER <-- ASK_LETTER_INPUT()
  IF LETTER_ALREADY_TRIED:
    KEEP_ASKING_AGAIN()

  APPEND LETTER TO MY_LETTERS
  OUTPUT <-- REVEAL_LETTERS(MY_WORD, MY_LETTERS)
```

```
    IF ALL_LETTERS_FOUND(MY_WORD, MY_LETTERS):
      PRINT "YOU WON!"
      QUIT

    PRINT(OUTPUT)
```

Work inside out

It's now time to turn the pseudocode into Python code. You are going to implement each "complex action" that is described in the pseudocode, starting from the "deepest" actions:

- `PICK_A_RANDOM_WORD`
- `REVEAL_LETTERS`
- `ALL_LETTERS_FOUND`

Create the Python functions for these "actions":

- `def pick_random_word()` -> reads from the text file, picks a random word, and returns it (string).
- `def reveal_letters(word, letters)` -> returns a string, showing letters in `word` that are in `letters`, or `_`
- `def all_letters_found(word, letters)` -> returns `True` if `word` can be made from `letters`, `False` otherwise

The `reveal_letters` and `all_letters_found` have unit tests in `test_hangman.py`. Use them! You can use the starter code in `hangman.py` too.

> ⚠️ Remember: all these functions **RETURN** values (and not `print` them).

Write the main function

When you have all the basic "logic" for the hangman game in functions, you can now write the `def main()` function that will connect everything together. Follow the pseudocode above.

Step 3: polish your program

- Add comments and docstrings where necessary.
- Fix any issues related to case (uppercase / lowercase input and words).
- Do additional checks on user input (reject numbers, empty strings, more than one letter at a time, etc).
- Add the possibility to "guess" the word by typing it in full instead of providing a single letter.