Harry Potter Typing Trainer Version 4: Diffindo

Finally, it's time to make your typing trainer real.

In this version, your program will time the user's input and calculate the score based on the time and correctness.

We will use the following formula to estimate the target typing speed:

`Target typing time (TTT) = number of characters * 0.3 seconds`

If the user types the spell correctly, the score is:

- If the user's typing time is faster or equal to TTT, the score will be **10**.
- If the user's typing time is faster or equal to (TTT * 1.5) but slower than TTT, the score will be **6**.
- If the user's typing time is faster or equal to (TTT * 2) but slower than (TTT * 1.5), the score will be **3**.
- If the user's typing time is slower than (TTT * 2), the score will be **1**.

Otherwise, the score will be **-5**.

What to do

We provided the complete `get_user_input` function for you. Now, it takes time into account. Carefully study the code of this function and make sure you understand how it works.

This function return a tuple of **two** values:
a) The spell that the user typed
b) The time it took the user to type the spell (rounded to two digits after the decimal point)

1. Implement the new `get_target_time` function to calculate the target time (TTT) using the formula above.

2. Finally, implement the `calculate_points` function to calculate the score using the approach above. The score can be a negative number.

Use the following template. All functions defined in the template **must be present and implemented** in your code (you may **not** omit functions). You **cannot** change the header of the functions given in the template or omit these functions. You **may** add extra functions if needed. **In this version you will add or modify code in the `main` function in order to satisfy the requirements of this version**.

You **may** add extra functions if needed.

```python
import time
# All functions implemented in Version 3 code should be placed here EXCEPT for
# the function get_user_input
# The get_user_input function of version 3 is replaced by the new
# get_user_input function that is provided with this version
def get_user_input(spell: str) -> (str, float):
    """
    Gets input from the user
    Returns the input and the time it took the user to type the input
    """
    start = time.time()
    print(f"Type the following spell: {spell}")
    user_input = input().lower()
    user_time = round(time.time() - start, 2)
    print(f"Result: {user_time} seconds (goal: {get_target_time(spell)} seconds).")
    return user_input, user_time

def get_target_time(spell: str) -> float:
    """
    Returns the target time to type the spell.
    """
    # TODO: Implement this function

def calculate_points(spell: str, user_input: str, user_time: float) -> int:
    """
    Calculates the points that the user gets.
    spell: The spell that the user is typing.
    user_input: The input that the user typed.
    user_time: The time that the user took to type the input.
    """
    # TODO: Implement this function

def main() -> None:
    """
    Main program.
    """
    spells = read_spells('spells.txt')
    display_header()
    display_instructions()
    # Game loop (call play_again())
    # TODO: Move the score calculation logic from main() to calculate_points()
main()
```

Hints

- The `get_user_input` function returns a tuple of two objects. If a function returns two objects, you can specify 2 identifiers on the left side of the assignment statement to refer to the two objects. Like so:
  `user_input, user_time = get_user_input(spell)`

Program name

Save your program as `spells4.py`.

Demo

https://asciinema.org/a/vROCi70Y0GlrjFL2rFhLeYEIp

Testing

To make sure your program works correctly, you should test it.

Try playing a few rounds of the game. Make sure that:

- You can play the game multiple times
- You can quit the game after any number of rounds
- Each correct answer adds 10, 6, 3, or 1 points to the score, depending on the user's typing time *(if the target speed feels too hard for you, feel free to make the game easier for testing purposes!)*
- Each incorrect answer subtracts 5 points from the score

Submitting

Submit `spells4.py` via eClass.

You may submit either all versions you complete, or only the final version.