typing_test Python program: les docs

by George Osborn

Initialisation method

First of all inside the class we have an initialisation function passed self because this method will define objects for use in other areas of the class. All the methods in the class are passed self as it is essential they can all access the variables and reassign values to them et cetera.

Inside here a dictionary is placed called self.tests where the typing tests are stored under some keys and at line 17 there is the same structure, called self.tests_list, but the tests (dictionary values) are split into a dictionary of list objects where each item is a word.

From lines 24-28 we create the main tkinter window and configure it: with an icon, altering the size, title and background fill.

Lines 31-33 we create a Label that is inside this window at coordinates x/y, altering the font and showing the title for the program and is placed on screen.

Lines 36-39 show us creating a variable where its value is set to ‘Choose difficulty’ in order to hold the user’s hand :), this will be assigned a value upon user input from the OptionMenu choose_test for later use and is placed on screen.

Lines 42-43 are where I create and place a button to call the typing test (run_test()) under its command parameter which links to our method and is how the actual test will be started and all other methods run from this button's actuation.

Lines 46-48 are where i create and place a large box for text entry (the user's answer), to be validated later on, by default the state is set to DISABLED so the user can not type in it unless the test has been run successfully via the Button (self.btn_run_test) and OptionMenu (choose_test).

Lines 51-56 are where we will present the length of time left in their test, the tk IntVar (self.length_of_time_text) is set to self.length_of_time here as otherwise it only changes to 60 after the btn_run_test is actuated and we want users to know how much time they will have beforehand.
Then this tk integer variable is used as the Label's text so it can present length of time left to the user, and the timer method will help decrement the value until 0 every 1 second (from 60), this Label object- self.lab_show_time_left is placed on screen.

Lines 59-63 are where i will present each of the user’s results and it features a sort of ‘memory’ whereby the tk string variable- self.results_text uses game_over method (we will get to later) and adds the player’s current score onto itself every time the game ends, to be presented in our

Label object self.lab_results which is placed on screen.
All of these lovely objects are looped over so they are always on screen, line 65.

Running the test method

For the first four lines there are variables being created. self.typed_correct and self.index_pos: these are both assigned the values 0 and are altered and used in the other methods but are run in here because they will definitely be needed since the test has been ran. The former will be reassigned a value when the user has typed a word correctly and the latter when validating if each word was entered correctly, both in the game over method.
In line 71 we have made the user's chosen difficulty into a variable, stored in memory for quick retrieval and elegance.
self.delete_first_y is used later on in order to ensure that the window is not 'reset' on its first usage.

In line 74 there is an if statement, the logic ensures the user has chosen a difficulty. Now, allow me to disregard the laws of chronology for one second...
Line 90-91 shows what happens if one has not satisfied line 74 (a difficulty has not been chosen)- a messagebox will appear (in a separate window) with info about how the user has misused the program and does nothing else.

Back to line 75, here the logic in the if states that the following commands are only to run if the run_test method has been run before. It utilises one of the vars in the dunder init method which is pre-set False, self.delete_first_y is set to True whenever line 74 is satisfied. So the if self.delete_first_y is True only runs on the second running of the program.
The commands that run when both ifs are satisfied in numerical order are: forgetting of the placement of the text to be typed Label, deletion of the contents of the answer entry Text object-text_enter_typing and destruction of the secondary window that shows the user's score. These objects are altered/shown after a full use of the program and need to be reset/removed each time it is used.

Line 79 is set to True for reasons mentioned above.

Below on line 81 one can see that the Text box is configured to NORMAL state, this enables the box to be typed into, whilst the run_test method is run and both ifs satisfied.
From line 82-85 i create a tk string variable that is immediately set to the test difficulty that was chosen via OptionMenu/drop-down list chosen_test at line 41. This tk string variable is set to the value (string) attributed to the key chosen upon the button's actuation (self.btn_run_test) and is immediately used to make and place a Message object to show the user what they are to type.

In line 87-89 we are creating a lovely Daemon thread as i had error messages using a standard threading package thread and Daemon worked. At line 87 we create the Thread object that also calls my elegant timer method.

Line 89 will call this self.new_thread object

The Timer method

At line 96 we create a variable that will retrieve the current time from the os and only take the seconds value as it is all we will need :)

At line 98 there is a lovely while loop which only runs when the length of time left or self.length_of_time is greater than 0 or in other words, when there are still seconds to be counted down.
Whilst this is still looping, the program will sleep for a second, so once a second has passed it can perform lines 100-102:

1. self.length_of_time or how long is left, which should be set to 60, will decrement by 1.
2. The tk integer variable that is the contents of our Label (lab_show_time_left) showing the length of time left is set to the above variable.
3. The Label showing the length of time left is configured with the tk integer variable above, so that the change can be reflected on this pre-placed Label on screen.

When the while loop is not satisfied meaning there is no more time left, it will call our game_over method with all its luxuries for the user :). Line 104.

This method used to be more complex with an if, elif and else clause and four datetime retrieval objects. As you can see it is far more elegant now, thanks to a small epiphany after a lot of painstaking arduity.

Game over method

On line 108 to 110 the variable answer_entered's creation can be seen where it compels Python to retrieve the user's answer typed- self.text_enter_typing in its entirety. Below this another variable is made that takes the aforementioned variable and splits it into a list, where each constituent is a word.
After this a variable (test_chosen_list) is made that is assigned the chosen text to be typed list, where each item is a word of course.

Next up are lines 112-114 the for loop here helps count the words in the test chosen. We create a variable that is incremented by 1 for each word in the test_chosen_list value. Therefore, counting each word in the list.

Lines 117-120 are not too dissimilar: it validates how many words were typed correctly, to give us the words per minute/ their score. It loops over a list of the user's answer entered and each time will check whether the two list values at [index_pos] position in the lists (answer_entered_list and test_chosen_list) are identical, if this parameter is satisfied then it

adds a 1 onto the self.typed_correct variable.
Irrespective of this it adds a 1 onto index_pos so that next time the loop runs it checks whether the next word/ value in the lists are identical.

I chose to loop over answer_entered_list instead of test_chosen_list (they are interchangeable) since it is likely there will be less items in the list to loop over, as the user is more likely to type less words than the test, thereby making our lovely program more efficient. Thumbs up.

Lines 123-127
Makes a second window and changes its title, size, icon of window and the colour fill of the window so both windows match. This is to be used to display the user's score of their current attempt/ game.

Lines 130-139
In line 130 there is a variable- self.results_text_updated set to 0 that is for use in the if statement.
Last of all In line 139, each time the, to be mentioned if statement is satisfied this variable is incremented by 1. So, with the if statement whilst self.results_text_updated is less than or equal to 13, it is satisfied. This number was chosen for aesthetics or… so it does not cover other Widgets or carry on forever and ever, and ever.
While satisfied we create an object at line 133 set to itself + the number of words typed correct and the difficulty of the test chosen. To be shown to the user.
Next we shall set the all results Label's text StringVar to the variable just mentioned,
now i configure the all results Label (self.lab_results) so that it updates the object and updates the Label placed on screen via the .place() method.
This all results Label can be added to 13 times, to display the previous results in a sort of 'memory' capacity, as it keeps adding onto itself in self.results_text_update.

Next we are going to add to this new second window with line 142-145, we are adding their current test's result as a Label. Using variable typed_correct_text which concatenates the number typed correct and some strings.
This is used as the text for our Label- lbl_typed_correct which has the font amended and is placed on screen this and its placement in self.score mean we can show the user their score in this second window that pops up.

As well as this, there is an exit button in lines 147 and 148 here a Button is created and placed on this second window, its command is set to self.score.destroy, this will destroy our second window used to display the user's current result.
It makes interacting with the program easier for the user.

At line 151 we call this class.