CSE Project Report: Command-Line Typing Test

Aryaman Singh Chauhan
Registration Number: 25BCY10217

November 23, 2025

Abstract

This report details the design, implementation, and evaluation of a simple command-line based typing speed test application developed in Python. The application measures a user's typing speed in words per minute (WPM) by comparing their input against a randomly selected target phrase and calculating the time taken. It highlights fundamental programming concepts such as function definition, conditional logic, and string manipulation in a practical context.

Contents

1 Introduction

The objective of this project was to create a functional and reliable command-line utility for measuring typing speed. The application, named 'Command-Line Typing Test,' serves as an interactive tool demonstrating basic control flow and string manipulation techniques in Python. The core functionality involves randomly selecting a text prompt, timing the user's input, and calculating the WPM based on accuracy and time taken.

2 System Architecture and Functional Analysis

The system follows a lightweight, single-file architecture. The logic is cleanly separated into two main Python functions.

2.1 System Components

- `calculate_wpm(time_taken, textlength)`: Handles the mathematical calculation of the WPM score.
- `typing_test()` (Controller/View): Manages the entire user interaction flow, including text selection, timing, validation, and result presentation.

2.2 Words Per Minute (WPM) Calculation

The WPM is calculated using the industry-standard methodology where an average word is approximated to be 5 characters.

$$\text{WPM} = \frac{(\text{Total Characters}/5)}{\text{Time Taken in Seconds}} \times 60$$

2.3 Input Validation and Error Feedback

The comparison logic uses `u_input.strip() == targettext.strip()` to ensure leading/trailing whitespace is ignored. In case of mismatch, a detailed word-by-word analysis is performed by splitting the target and user input strings into lists of words. This provides crucial diagnostic feedback, showing which words were mistyped, missed, or added unnecessarily.

3 Implementation Details

The application relies solely on the built-in `random` and `time` modules, ensuring high portability across Python 3 environments. The use of `time.time()` provides high-resolution timing for accurate WPM calculation.

4 Source Code Listing

The following is the complete source code for the `cseprojectaraman.py` utility, which includes the WPM utility function and the main test function.

```
import random
import time


```

```python
def calculate_wpm(time_taken, textlength):
    """
    Calculates Words Per Minute (WPM) based on time taken and the length of the text.
    Assumes 5 characters per word.
    """
    if time_taken <= 0:
        return 0
    # Calculate words typed based on 5 characters per word
    wordstyped = textlength / 5
    # WPM = (words / time in seconds) * 60 seconds/minute
    wpm = (wordstyped / time_taken) * 60
    return round(wpm)

def typing_test():
    """
    Runs the main typing test logic: selects text, measures time,
    compares input, calculates WPM, and displays results/errors.
    """
    sentences = [
        "The quick brown fox jumps over the lazy dog.",
        "Programming in Python is fun and easy to learn.",
        "Learning to code opens many new doors.",
        "You say this after someone fails, and you hope they do better next time.",
        "This ship is too big to pass through the canal.",
        "She didn't drink, but she didn't want people to realize that, so she ordered a ginger ale at the bar.",
        "She wanted her grandmother's old bike, but the tires were worn out and she couldn't find a shop that could replace them.",
        "The tree was so old that it was considered a historic monument.",
        "Tom is used to driving a pickup truck, but he's never driven a monster truck.",
        "She broke the pot two hours after she bought it.",
        "The alligator's teeth were so scary that I ran back to the car as fast as I could.",
        "I saw some cool vintage chairs in a thrift store, but I really need to stop buying things.",
        "Hello world is the classic starting program.",
        "The green trees out the window swayed in the wind.",
        "I'm just trying to stop you from making it worse than it already is, but you don't care at all.",
        "The old man was wearing a suit and carrying a cane and he looked very official and kind of scary.",
        "There is a lot of snow on the ground, and I can't leave the house. I can't even open the front door.",
        "I wish I could remember where I put the keys, but I cannot.",
        "The cat was chasing a mouse in the corner of the room, and I couldn't stop it.",
        "I had a lot of fun writing this list of sentences.",
    ]
    targettext = random.choice(sentences)

    # Display UI
    print("\n\n" + "*" * 75)
    print("Welcome to the Command-Line Typing Test")
    print("**" * 75)
    print("Type the following phrase exactly as it appears:")
    print("-" * 75)
    print(f"\n\n>>>>> {targettext}<<<< \n\n")
    print("-" * 75)

    # Start Timer and get input
    start_time = time.time()
    u_input = input("Type the above phrase here : ")
    end_time = time.time()

    total_time = end_time - start_time

    # Compare result and display output
    if u_input.strip() == targettext.strip():
        wpm = calculate_wpm(total_time, len(targettext))
        print("\n" + "*" * 50)
        print("CONGRATULATIONS! Your answer is correct.")
        print(f"Your typing speed is: {wpm} words per minute (WPM)")
        print(f"Time taken: {total_time:.1f} seconds")
        print("#" * 50)
    else:
        print("\n" + "!" * 50)
        print("SORRY! Your answer is incorrect.")
        print("Please try again for accurate WPM.")
        print(f"Time taken: {total_time:.1f} seconds")
        print(f"Provided Entry:{targettext}")
        print(f"Your Entry:{u_input}")

        # Word-by-word error analysis
        targetwords=targettext.strip().split()
        u_words=u_input.strip().split()

        print("\n--- Error Analysis ---")

        # Check for incorrect words
        for i in range(min(len(targetwords), len(u_words))):
            if targetwords[i] != u_words[i]:
                print(f"Word {i+1} mismatch:")
                print(f"  Expected: '{targetwords[i]}'")
                print(f"  Got:      '{u_words[i]}'")

        # Check for missing/extra words
        if len(targetwords) > len(u_words):
             print(f"Missing {len(targetwords) - len(u_words)} words.")
        elif len(u_words) > len(targetwords):
             print(f"Extra {len(u_words) - len(targetwords)} words were typed.")
        print("----------------------")


if __name__ == "__main__":
    typing_test()
```

Listing 1: Source Code for the Typing Test Utility

5 Conclusion

The Command-Line Typing Test successfully meets the project requirements, offering a clear and functional utility for measuring typing performance. The project demonstrates proficiency in Python function definition, user interaction, and time-based calculations, providing a robust tool for assessing keyboarding skills.