Keyboard Analysis Programming Assignment

Objective

Develop a Python program that analyzes keyboard usage patterns for a given text input. The program should generate a heatmap visualization of key usage and calculate the total distance traveled by fingers while typing.

An example of this can be seen at https://www.patrick-wied.at/projects/heatmap-keyboard/

- As a key is used more and more often, the colour allocated for it tends towards the high end of the heatmap (red). Less used keys are towards the violet end.
- How the colours are blended to create a smooth heatmap is not obvious or fixed - this is left open to you.

Requirements

1. Input:
    - A keyboard layout specification (you may provide this as a data structure). The format of the structure for the common *QWERTY* layout is given as the first input.
    - A text string to analyze
2. Keyboard Layout:
    - The layout for *QWERTY* is given as a sample input. Your program should accept the layout as a parameter for the analysis.
3. Key Usage Analysis:
    - Count the frequency of each key press in the input text
    - Generate a heatmap visualization of key usage
    - Consider uppercase letters and special characters (use of Shift key)
4. Finger Travel Distance:
    - Assign a position (x, y coordinates) to each key on the keyboard
    - Calculate the distance between consecutive key presses
    - Sum up the total distance traveled by fingers
    - Include the travel to and from the Shift key for uppercase letters
5. Output:
    - A heatmap image showing key usage frequency

- The total finger travel distance for the input text

6. (Optional) Performance Analysis:
    - Compare the efficiency of different keyboard layouts for the same input text
    - Generate the output image layout for different layouts (this is not trivial since you need to modify the keyboard image - or you could generate a fresh image)
    - Allow layouts that are not in the standard footprint of a keyboard - for example split keyboards - this can be done by changing the coordinates in the layout.

Bonus Challenges

- Implement multiple keyboard layouts (e.g., QWERTY, Dvorak, Colemak) and compare their efficiency
- Create an animated visualization of the typing process
- Suggest an optimized keyboard layout based on the input text

Deliverables

1. Python script(s) implementing the required functionality
2. A brief report (single PDF file - not more than 3 pages) explaining your approach, any assumptions made, and your findings
3. Sample outputs (heatmap images, distance calculations) for at least two different input texts

The python script can be given any name, but there should be a README document that clearly states how to run it for a given layout. Layout files should follow the same format as shown for QWERTY.

Evaluation Criteria

- (5 marks) Correctness of the implementation: does the heatmap match the expected output for the given text, are the locations of the heat signatures correct etc.
- (2 marks) Code quality and organization - is the code documented, is it formatted properly, are functions clear and understandable. Note that unnecessarily verbose documentation will lose marks here - the idea is to be clear.
- (2 marks) Creativity in visualization and analysis: are the heatmaps fixed rectangles or do they blend into contours, is the finger travel captured in some way, how are Shift keys etc visualized etc.
- (1 mark) Depth of insights in the report

Note that creativity, insights etc. will be considered at a high level for the initial marks - so it is quite possible to get full marks here, but for the final grading these will play a role in differentiating grades.

Optimizing the layout is NOT part of this assignment - that will be another assignment.

Appendix - sample layout

```
QWERTY_LAYOUT = {
    'row1': {
        'keys': '`1234567890-=',
        'positions': [
            (0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0),
            (8, 0), (9, 0), (10, 0), (11, 0), (12, 0)
        ]
    },
    'row2': {
        'keys': 'qwertyuiop[]\\',
        'positions': [
            (0.5, 1), (1.5, 1), (2.5, 1), (3.5, 1), (4.5, 1), (5.5, 1),
            (6.5, 1), (7.5, 1), (8.5, 1), (9.5, 1), (10.5, 1), (11.5, 1),
↪   (12.5, 1)
        ]
    },
    'row3': {
        'keys': 'asdfghjkl;\'',
        'positions': [
            (0.75, 2), (1.75, 2), (2.75, 2), (3.75, 2), (4.75, 2), (5.75,
↪   2),
            (6.75, 2), (7.75, 2), (8.75, 2), (9.75, 2), (10.75, 2)
        ]
    },
    'row4': {
        'keys': 'zxcvbnm,./',
        'positions': [
            (1.25, 3), (2.25, 3), (3.25, 3), (4.25, 3), (5.25, 3), (6.25,
↪   3),
            (7.25, 3), (8.25, 3), (9.25, 3), (10.25, 3)
        ]
    },
    'special_keys': {
        'Shift_L': (0, 3),
        'Shift_R': (11.25, 3),
```

```
        'Space': (3.5, 4),
        'Backspace': (13, 0),
        'Tab': (0, 1),
        'CapsLock': (0, 2),
        'Enter': (12, 2)
    }
}

# Function to get key position
def get_key_position(key):
    for row in QWERTY_LAYOUT.values():
        if 'keys' in row and key in row['keys']:
            index = row['keys'].index(key)
            return row['positions'][index]
    return QWERTY_LAYOUT['special_keys'].get(key)

# Example usage
print(get_key_position('a'))  # Output: (0.75, 2)
print(get_key_position('Shift_L'))  # Output: (0, 3)
```