

---

# 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.

## Keyboard typing distance

The keyboard layout is represented as a 2D coordinate system, with each key having a specific position (x, y) and a designated starting position. The x-coordinate increases from left to right, and the y-coordinate increases from bottom to top. The home row keys serve as starting positions for all other keys.

For example, in the QWERTY layout:

- The keys dictionary stores the layout information:
  - The 'f' key is at position (4.75, 2) and starts from 'f' itself (home position).
  - The 'y' key is at position (6.5, 3) and starts from 'j' (the index finger of the right hand on the home row).
- The characters dictionary shows how a character may be typed:
  - Normal characters like a just require the key itself to be pressed.
  - Capital letters or special characters may require pressing more than one key: the list of these keys is given as a *tuple* (could have been a list as well) - in that case, the distance traveled is the sum of individual distances to the keys. For example:
    - \* 'a': ('a',) indicates that only one letter needs to be pressed for a
    - \* 'Y': ('Shift\_L', 'y'): Both the left-shift and y keys need to be pressed.

To calculate the distance a finger travels to press a key:

1. Identify the starting position (home key) and the target key position.

- 
2. Calculate the Euclidean distance between these two points.

For instance, to type 'y':

- Start position (j): (7.75, 2)
- End position (y): (6.5, 3)
- Distance =  $\sqrt{(7.75 - 6.5)^2 + (2 - 3)^2} \approx 1.60$  units

For characters requiring the Shift key, calculate the distance to Shift, then to the character key, and sum these distances. For example, to type 'T':

1. Distance from ; to Shift\_R:  $\sqrt{(10.75 - 12.5)^2 + (2 - 1)^2} = 2.01$  units
2. Distance from f to t:  $\sqrt{(4.75 - 5.5)^2 + (2 - 3)^2} = 1.25$  units
3. Total distance:  $2.01 + 1.25 = 3.26$  units

This model assumes fingers return to the home position after each keystroke, providing a consistent basis for comparing different keyboard layouts. **Note that the actual distance is double of this since the finger also needs to return, but here only the one-way distance is required.**

## Requirements

1. Input:
  - A keyboard layout specification as described above. The full specification for QWERTY is also attached.
  - A text string to analyze
2. 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)
3. Finger Travel Distance:
  - Calculate the distance traveled for typing each character
  - Sum up the total distance traveled by fingers
  - Include the travel to and from the Shift key for uppercase letters
4. Output:
  - A heatmap image showing key usage frequency
  - The total finger travel distance for the input text
5. (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.