Assignment 4

Lodha Soumya Sachin

EE23B140

1 Keyboard Layout

```
keyboard_layout = {
    'top_row': {
        'keys': [("Esc", 1), ("F1", 1), ("F2", 1), ("F3", 1), ("F4", 1), ("F5",
            1), ("F6", 1), ("F7", 1), ("F8", 1), ("F9", 1), ("F10", 1), ("F11",
            1), ("F12", 1), ("prt sc", 2)],
        'x': [0,1,2,3,4,5,6,7,8,9,10,11,12,13],
        'y': 5,
        'frequency': [0]*14
    },
    'num_row': {
        'keys': [("`", 1), ("1", 1), ("2", 1), ("3", 1), ("4", 1), ("5", 1), ("6"
            , 1), ("7", 1), ("8", 1), ("9", 1), ("0", 1), ("-", 1), ("=", 1), ("
            Backspace", 2)],
        'x': [0,1,2,3,4,5,6,7,8,9,10,11,12,13],
        'y': 4,
        'frequency': [0]*14
    },
    'qwerty_row': {
        'keys': [("Tab", 1.5), ("Q", 1), ("W", 1), ("E", 1), ("R", 1), ("T", 1),
            ("Y", 1), ("U", 1), ("I", 1), ("O", 1), ("P", 1), ("[", 1), ("]", 1),
             ("\\", 1.5)],
        'x': [0,1.5,2.5,3.5,4.5,5.5,6.5,7.5,8.5,9.5,10.5,11.5,12.5,13.5],
        'y': 3,
        'frequency': [0]*14
    },
    'asdf_row': {
        'keys': [("Caps", 1.75), ("A", 1), ("S", 1), ("D", 1), ("F", 1), ("G", 1)
            , ("H", 1), ("J", 1), ("K", 1), ("L", 1), (";", 1), ("'", 1), ("Enter
            ", 2.25)],
        'x': [0,1.75,2.75,3.75,4.75,5.75,6.75,7.75,8.75,9.75,10.75,11.75,12.75],
        'y': 2,
        'frequency': [0]*13
    },
    'zxcv_row': {
        'keys': [("Shift", 2.25), ("Z", 1), ("X", 1), ("C", 1), ("V", 1), ("B",
            1), ("N", 1), ("M", 1), (",", 1), (".", 1), ("/", 1), ("Shift", 2.75)
            ],
        'x': [0,2.25,3.25,4.25,5.25,6.25,7.25,8.25,9.25,10.25,11.25,12.25],
        'y': 1,
        'frequency': [0]*12
```

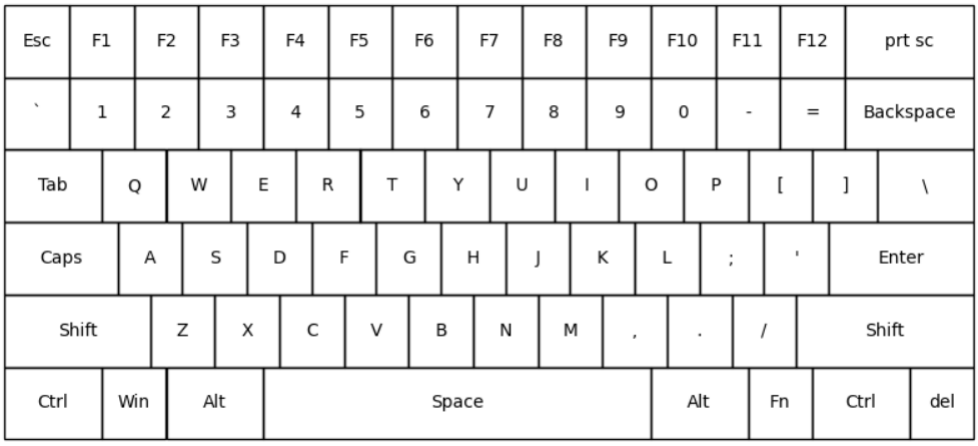
```
    },
    'bottom_row': {
        'keys': [("Ctrl", 1.5), ("Win", 1), ("Alt", 1.5), ("Space", 6), ("Alt",
            1.5), ("Fn", 1), ("Ctrl", 1.5), ("del", 1)],
        'x': [0,1.5,2.5,4.0,10.0,11.5,12.5,13],
        'y': 0,
        'frequency': [0]*8
    }
}
```

Define a dictionary which contains 4 dictionaries within itself :
1. keys : this is a list of tuples which contain the names of the keys in a row and their width.
2. x : this is a list of all the x coordinates of the keys,
3. y : this is the y coordinate of the row.
4. frequency : this is initialized to zero initially, it gives the frequency of each key.

WHILE RUNNING THE JUPYTER NOTEBOOK RUN THE KEY BOARD LAYOUT CELL EVERYTIME BEFORE CHANGING INPUT TO INITIALIZE ALL FREQUENCIES TO ZERO OTHERWISE IT WILL STORE FREQUENCY FROM PREVIOUS INPUTS AND GIVE WRONG OUTPUT.

We use the 'patches' tool in matplotlib to create rectangles for the keys. Key dimensions are height = 1 unit width mostly 1 unit but larger for special keys like shift, space, tab , etc.

This is what the input layout will look like -



2 Input String

```
user_input = input("Enter something: ")
user_input = user_input.replace(" ", "")
#remove spaces in the input to ensure proper distance calculation
```

We are **not considering travel from the space key**. So after taking in the input string we remove the spaces in it.

3 Distance Travelled Calculation

```
 for i in range(1, n):
       if string[i].isupper():
           # Find the closer shift key (left or right)
           dist_to_shift_left = ((x[i - 1] - shift_x_left)**2 + (y[i - 1] -
                shift_y)**2)**0.5
           dist_to_shift_right = ((x[i - 1] - shift_x_right)**2 + (y[i - 1] -
                shift_y)**2)**0.5

           if dist_to_shift_left < dist_to_shift_right:
               shift_x = shift_x_left
           else:
               shift_x = shift_x_right
           #distance for uppercase letters will be the sum of distances from
                shift key
           total_distance += key_dist(x[i-1], y[i-1], shift_x, shift_y) +
                key_dist(shift_x, shift_y, x[i], y[i])
       else:
           #just take distance between consecutive letters if lowercase
           total_distance += key_dist(x[i-1], y[i-1], x[i], y[i])
```

1. For distance travelled, we only consider **one finger** being used.
2. For lowercase letters, just take the distance between their coordinates.
3. But if the letters are capitalised, we have to add up distance between previous key and shift key + shift key and the capitalised letter.

4 Frequency calculation

```
def get_frequency(key):
    for row in keyboard_layout.values():
        #iterate over all the keys in a row to find out the proper match
        for index, (label, width) in enumerate(row['keys']):
            if label == key:
                #increment the frequency value for the particular input key
                row['frequency'][index] += 1
                return

for key in user_input.upper(): #use .upper as dictionary has elemnts in upeercase
    get_frequency(key)
```

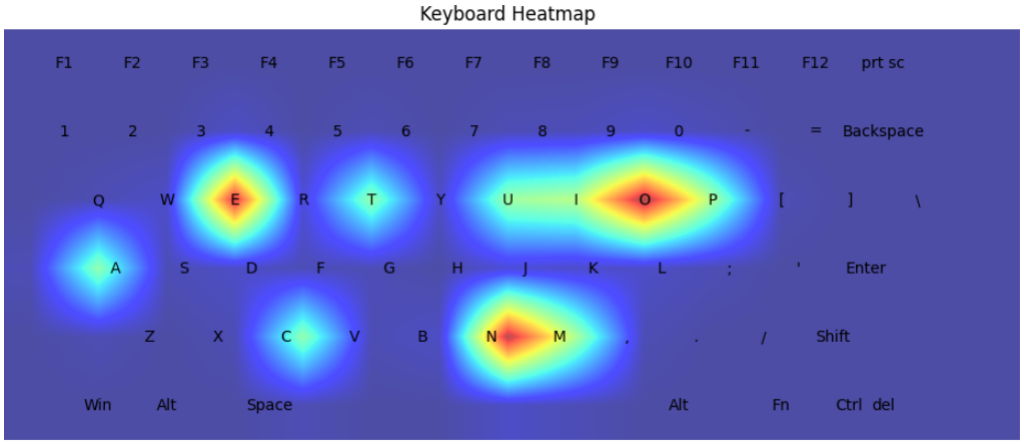
5 Heatmap Generation

1. We use the gaussian filter feature from scipy library in python to generate a heatmap.
2. Fill the heatmap with frequency values from the get frequency function.
3. Apply Gaussian filter and use the 'jet' colormap which goes from blue to red as frequency increases.
4. Print the keyboard on top of the heatmap.

EXAMPLE INPUT - 'once upon a time'

```
print(calculate_distance(user_input))
```

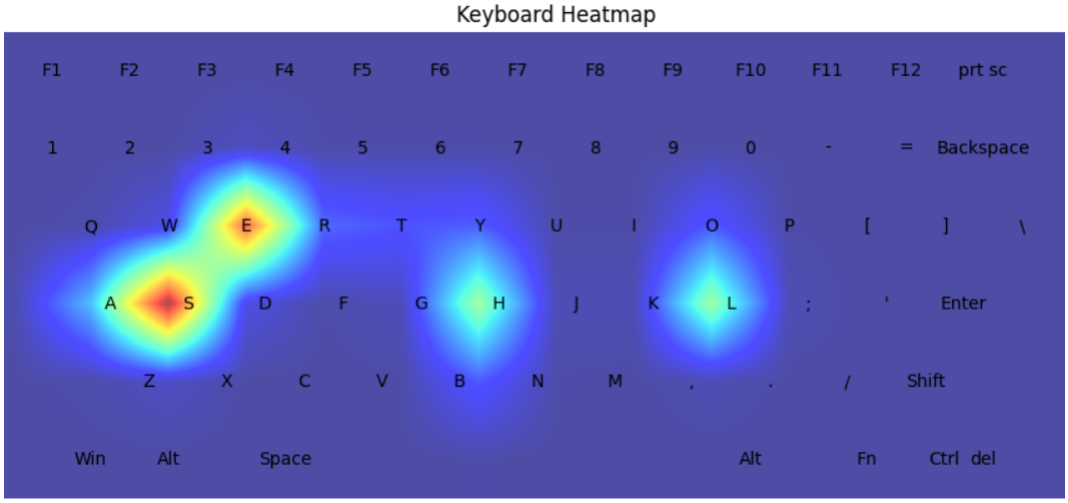
```
38.79745831263672
```



EXAMPLE INPUT - 'She sells seashells by the seashore.'

```
print(calculate_distance(user_input))
```

```
78.32379607562605
```


Keyboard Heatmap
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 prt sc
1 2 3 4 5 6 7 8 9 0 - = Backspace
Q W E R T Y U I O P [ ] \
A S D F G H J K L ; ' Enter
Z X C V B N M , . / Shift
Win Alt Space Alt Fn Ctrl del