# Color Sorting Algorithm

By: Uzziel Kyle Ynciong

## The Program:

### Classes

- Color
- ColorsArray

### Functions

- ColorsArray.sort_colors()
- min_max()
- rand_colors_generator()

## Color

```python
class Color:
    def __init__(self, color: str) -> None:
        self.name = color
        self.visual = {
            'red': '❤️ ',
            'white': '⚪ ',
            'blue': '🔷 ',
            'green': '🍀 ',
        }[color]

    def __str__(self) -> str:
        return self.name

    def __repr__(self) -> str:
        return self.visual
```

## ColorsArray

```
class ColorsArray:
    def __init__(self, length: int) -> None:
        self.length = min_max(length=length)
                      # A function that keeps a min and max number
        self.colors = rand_colors_generator(num=self.length)
                      # A function that generates a list of
                      # random colors on a given length

    def sort_colors(self) -> list:
        ...
```

## min_max()

```python
def min_max(length: int, min: int = 2, max: int = 10) -> int:
    if length < min:
        length = min

    elif length > max:
        length = max

    return length
```

## rand_colors_generator()

```python
def rand_colors_generator(num: int, min: int = 2) -> list:
    colors = [Color('red'), Color('white'), Color('blue'), Color('green')]

    if num <= len(colors):
        if num < min:
            num = min

        shuffle(colors)
        random_colors = [color for color in colors[:num]]

        return random_colors


    random_colors = [color for color in colors]
    random_colors.extend([colors[randint(0, len(colors)-1)] for i in range(0, num-len(colors))])

    shuffle(random_colors)

    return random_colors
```

## ColorsArray - sorting self method

```python
class ColorsArray:
    ...

    def sort_colors(self) -> list:
        ...
```

## Nested For Loop:

```python
def sort_colors(self) -> list:
        colors_copy = self.colors.copy()

        sorting = list(map(
            str.strip,
            input('*Enter the order of colors(separated by commas): ').split(',')
            ))

        sorted_colors = []
        for color_to_match in sorting:
            grouped_colors = []
            for color in colors_copy:
                if color_to_match == color.name:
                    grouped_colors.append(color)

            sorted_colors.extend(grouped_colors)

        return sorted_colors
```

## Absurd List Comprehension:

```
def sort_colors(self) -> list:
        colors_copy = self.colors.copy()

        sorting = list(map(
            str.strip,
            input('*Enter the order of colors(separated by commas): ').split(',')
            ))

        sorted_colors = [color for color_to_match in sorting for color in colors_copy if color_to_match == color.name]

        return sorted_colors
```

## For Loop + List Comprehension:

```
def sort_colors(self) -> list:
        colors_copy = self.colors.copy()

        sorting = list(map(
            str.strip,
            input('*Enter the order of colors(separated by commas): ').split(',')
            ))

        sorted_colors = []
        for color_to_match in sorting:
            grouped_colors = [color for color in colors_copy if color_to_match == color.name]

            sorted_colors.extend(grouped_colors)

        return sorted_colors
```

## Creating a copy of the list

```
        colors_copy = self.colors.copy()
```

```
# Output
    colors_copy = [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]
```

## Asking for input - order of colors

```
        sorting = list(map(
            str.strip,
            input('*Enter the order of colors(separated by commas): ').split(',')
            ))
```

```
# Input
    *Enter the order of colors(separated by commas): red, white,blue, green

# Output
    sorting = ['red', 'white', 'blue', 'green']  # A list of color ordering
```

## Creating an empty list to store grouped colors by order of sorting

```
        sorted_colors = []
```

```python
        for color_to_match in sorting:
            grouped_colors = [color for color in colors_copy if color_to_match == color.name]
```

```
sorting = ['red', 'white', 'blue', 'green']
sorting[0] = 'red'

colors_copy = [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]
```

```
Searching for 'red'

index 0 - [] || NO NEW ADDITION
index 1 - [❤️ ] || ADDED
index 2 - [❤️ , ❤️ ] || ADDED
index 3 - [❤️ , ❤️ ] || NO NEW ADDITION
index 4 - [❤️ , ❤️ ] || NO NEW ADDITION
index 5 - [❤️ , ❤️ ] || NO NEW ADDITION
index 6 - [❤️ , ❤️ ] || NO NEW ADDITION
index 7 - [❤️ , ❤️ ] || NO NEW ADDITION

grouped_colors = [❤️ , ❤️ ]
```

## Adding grouped color to sorted_colors

```
sorted_colors.extend(grouped_colors)
```

```
# Output
sorted_colors = [❤️ , ❤️ ]
```

```
    for color_to_match in sorting:
        grouped_colors = [color for color in colors_copy if color_to_match == color.name]
```

```
sorting = ['red', 'white', 'blue', 'green']
sorting[1] = 'white'

colors_copy = [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]
```

```
Searching for 'white'

index 0 - [⚪ ] || ADDED
index 1 - [⚪ ] || NO NEW ADDITION
index 2 - [⚪ ] || NO NEW ADDITION
index 3 - [⚪ , ⚪ ] || ADDED
index 4 - [⚪ , ⚪ ] || NO NEW ADDITION
index 5 - [⚪ , ⚪ ] || NO NEW ADDITION
index 6 - [⚪ , ⚪ ] || NO NEW ADDITION
index 7 - [⚪ , ⚪ ] || NO NEW ADDITION

grouped_colors = [⚪ , ⚪ ]
```

## Adding grouped color to sorted_colors

```
sorted_colors.extend(grouped_colors)
```

```
# Output
sorted_colors = [❤️ , ❤️ , ⚪ , ⚪ ]
```

```
        for color_to_match in sorting:
            grouped_colors = [color for color in colors_copy if color_to_match == color.name]
```

```
sorting = ['red', 'white', 'blue', 'green']
sorting[2] = 'blue'

colors_copy = [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]
```

```
Searching for 'blue'

index 0 - [] || NO NEW ADDITION
index 1 - [] || NO NEW ADDITION
index 2 - [] || NO NEW ADDITION
index 3 - [] || NO NEW ADDITION
index 4 - [🔷 ] || ADDED
index 5 - [🔷 ] || NO NEW ADDITION
index 6 - [🔷 ] || NO NEW ADDITION
index 7 - [🔷 , 🔷 ] || ADDED

grouped_colors = [🔷 , 🔷 ]
```

## Adding grouped color to sorted_colors

```
sorted_colors.extend(grouped_colors)
```

```
# Output
sorted_colors = [❤️ , ❤️ , ⚪ , ⚪ , 🔷 , 🔷 ]
```

```
        for color_to_match in sorting:
            grouped_colors = [color for color in colors_copy if color_to_match == color.name]
```

```
sorting = ['red', 'white', 'blue', 'green']
sorting[3] = 'green'

colors_copy = [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]
```

```
Searching for 'green'

index 0 - [] || NO NEW ADDITION
index 1 - [] || NO NEW ADDITION
index 2 - [] || NO NEW ADDITION
index 3 - [] || NO NEW ADDITION
index 4 - [] || NO NEW ADDITION
index 5 - [🍀 ] || ADDED
index 6 - [🍀 , 🍀 ] || ADDED
index 7 - [🍀 , 🍀 ] || NO NEW ADDITION
grouped_colors = [🍀 , 🍀 ]
```

## Adding grouped color to sorted_colors

```
sorted_colors.extend(grouped_colors)
```

```
# Output
sorted_colors = [❤️ , ❤️ , ⚪ , ⚪ , 🔷 , 🔷 , 🍀 , 🍀 ]
```

## Returns a list of sorted colors

```
def sort_colors() -> list:
    ...
        ...
            return sorted_colors
```

## Final Output in Terminal

```
*Enter number of colors: 8
Unsorted: [⚪ , ❤️ , ❤️ , ⚪ , 🔷 , 🍀 , 🍀 , 🔷 ]

*Enter the sorting of colors(separated by commas): red, white, blue, green
Sorted: [❤️ , ❤️ , ⚪ , ⚪ , 🔷 , 🔷 , 🍀 , 🍀 ]
```

## Running Time: O(m*n)

m = length of `sorting` - number of different colors to sort

n = length of `colors_copy` - number of colors of a given list