https://pandas.pydata.org/pandas-docs/stable/user_guide/style.html

Pandas Data Frame Styling

• *Styler.applymap (for each element)*

color negative numbers red and positive numbers black

```
def color_negative_red(val):
    """
    Takes a scalar and returns a string with
    the css property `'color: red'` for negative
    strings, black otherwise.
    """
    color = 'red' if val < 0 else 'black'
    return 'color: %s' % color
```

➔ df.style.applymap(color_negative_red)

• *styler.apply (apply for each "column" or "row"; by default it's column (axis=0))*
Styler.apply(func, axis=1) for rowwise styles
Styler.apply(func, axis=0) for columnwise styles
Styler.apply(func, axis=None) for tablewise styles

```
def highlight_max(s):
    '''
    highlight the maximum in a Series yellow.
    '''
    is_max = s == s.max()
    return ['background-color: yellow' if v else '' for v in is_max]
```

➔ df.style.apply(highlight_max)

• Chain previous two examples into one

```
df.style.\
    applymap(color_negative_red).\
    apply(highlight_max)
```

• Apply the function only for specific columns or rows

```
# Apply function only for columns B,C, and D
df.style.apply(highlight_max, subset=['B', 'C', 'D'])

# Apply function only for cells / grids where columns B and D & row 2 and row 5 converge
df.style.applymap(color_negative_red,
                  subset=pd.IndexSlice[2:5, ['B', 'D']])
```

• Control "formatting" of cell / grid values

```
# Round of all values by 2nd decimal digit
df.style.format("{:.2%}")

# Apply different formatting for different columns
df.style.format({'B': "{:0<4.0f}", 'D': '{:+.2f}'})


df.style.format({"B": lambda x: "±{:.2f}".format(abs(x))})
```

B
±1.33
±1.07
±1.63
±0.96
±1.45
±1.34
±0.12
±0.35
±1.69
±0.13

 # Format "missing values (null)"
df.style.format("{:.2%}", na_rep="-")
Can link with other functions too
df.style.highlight_max().format(None, na_rep="-")

• (built in) null highlighter
df.style.highlight_null(null_color='red')

	A	B	C	D	E
0	1.000000	1.329212	nan	-0.316280	-0.990810
1	2.000000	-1.070816	-1.438713	0.564417	0.295722
2	3.000000	-1.626404	0.219565	0.678805	1.889273
3	4.000000	0.961538	0.104011	nan	0.850229

• (Built in) null / missing value ➔ value formatter

```
(df.style
    .set_na_rep("FAIL")
    .format(None, na_rep="PASS", subset=["D"])
    .highlight_null("yellow"))
```

	A	B	C	D	E
0	1.000000	1.329212	FAIL	-0.316280	-0.990810
1	2.000000	-1.070816	-1.438713	0.564417	0.295722
2	3.000000	-1.626404	0.219565	0.678805	1.889273
3	4.000000	0.961538	0.104011	PASS	0.850229

• (built in) highlight_max and min
df.style.highlight_max(axis=0)
df.style.highlight_min( )

• sns heatmap styling

 # e.g.1
import seaborn as sns
cm = sns.light_palette("green", as_cmap=True)
df.style.background_gradient(cmap=cm)

 # e.g.2
df.style.background_gradient(cmap='viridis')

 # e.g.3
df.style.background_gradient(cmap='viridis', low=.5, high=0).highlight_null('red'))

• Bar Charts in DataFrame

```
# e.g.1
df.style.bar(subset=['A', 'B'], color='#d65f5f')

# e.g.2
df.style.bar(subset=['A', 'B'], align='mid', color=['#d65f5f', '#5fba7d'])
```

• share style

```
df2 = -df
style1 = df.style.applymap(color_negative_red)
style1

style2 = df2.style
style2.use(style1.export())
```

• Hiding certain elements

```
df.style.hide_index()
df.style.hide_columns(['C','D'])
```

• To Excel

```
df.style.\
        applymap(color_negative_red).\
        apply(highlight_max).\
        to_excel('styled.xlsx', engine='openpyxl')
```