syle 関数

2021 年 11 月 16 日

---

1 STYLE 関数

https://pandas.pydata.org/docs/reference/api/pandas.io.formats.style.Styler.applymap.html?highlight=style%20applym

[30]:
```
import pandas as pd
import numpy as np
```

1.0.1 サンプル DF 作成

[53]:
```
np.random.seed(0)

df = pd.DataFrame(np.random.randint(-2000, 5000, (4,4)), columns=list('ABCD'))
df.iloc[1,2] = -10.34567
df.iloc[3,1] = 23456.2345
df.iloc[3,3] = np.nan
df
```

[53]:

	A	B	C	D
0	732	607.0	-347.0	1264.0
1	2931	2859.0	-10.3	-967.0
2	2373	3874.0	3924.0	4743.0
3	4744	23456.2	4458.0	NaN

1.1 NaN の背景色に色付け

- 参考 HP

[56]:
```
display(
    df.style.highlight_null(),
    df.style.highlight_null(null_color='yellow')
```

```
)
```

```
<pandas.io.formats.style.Styler at 0x7fe377ac19d0>
```

```
<pandas.io.formats.style.Styler at 0x7fe377ac1f70>
```

1.2 style.applymap(関数)

- CSS を返す関数を作成し、df.style.applymap(関数) の引数とする
- カスタマイズする場合は、CSS の記法を調べる

1.2.1 文字色を変更

[34]:
```
def less_than_zero(val):
    color = 'red' if val < 0 else 'black'
    return 'color: {}'.format(color)
```

[35]:
```
less_than_zero(10)
# この式はエラー処理がなされてないため、str だとエラーになる
```

[35]:
```
'color: black'
```

- 文字列に変数の値を展開する際には、3 つのやり方があります。もしかしたら私が知らないだけで他にもあるかもしれません。
- sprintf スタイル: '%s, %s' % ('Hello', 'World')
- 拡張 sprintf スタイル: '%(a)s, %(b)s' % dict(a='Hello', b='World')
- format メソッド利用: '{0}, {1}'.format('Hello', 'World')

[36]:
```
df.style.applymap(less_than_zero)
```

[36]:
```
<pandas.io.formats.style.Styler at 0x7fe377a6e070>
```

[37]:
```
def less_than_zero_2(v, color):
    return f"color: {color};" if v < 0 else None


display(
    df.style.applymap(less_than_zero_2, color='blue'),
    df.style.applymap(less_than_zero_2, color='green', subset='C'),
    df.style.applymap(less_than_zero_2, color='purple', subset=['A','C'])
)
```

```
<pandas.io.formats.style.Styler at 0x7fe37651fac0>
```

```
<pandas.io.formats.style.Styler at 0x7fe37651ff70>

<pandas.io.formats.style.Styler at 0x7fe373b486d0>
```

1.2.2 文字を太字にする

[38]:
```
def font_bold(val):
    font = 'bold' if val < 0 else ""
    return 'font-weight:{}'.format(font)
```

[39]:
```
df.style.applymap(font_bold)
```

[39]:
```
<pandas.io.formats.style.Styler at 0x7fe377a6e2e0>
```

1.2.3 背景色を変える

[40]:
```
def change_bgcolor(val, color):
    return f'background-color:{color}' if val > 4000 else ''
```

[41]:
```
df.style.applymap(change_bgcolor, color='yellow')
```

[41]:
```
<pandas.io.formats.style.Styler at 0x7fe377a6e4f0>
```

1.3 複数の色付け

- メソッドチェーンで複数条件も可能

[42]:
```
df.style.applymap(less_than_zero).highlight_null()
```

[42]:
```
<pandas.io.formats.style.Styler at 0x7fe377a6e9d0>
```

[43]:
```
df.style.applymap(less_than_zero).highlight_null().applymap(font_bold)
```

[43]:
```
<pandas.io.formats.style.Styler at 0x7fe377a3f5b0>
```

2 3 桁区切り

- 嵌った箇所。結論、以下

■df.style.format('{:,}')

[44]:
```
df = pd.DataFrame(np.random.randint(-2000, 5000, (4,4)), columns=list('ABCD'))
df.iloc[1,2] = -10.34567
```

```
df.iloc[3,1] = 23456.2345
df
```

[44]:

	A	B	C	D
0	599	135.0	222.0	897
1	-299	-1463.0	-10.3	4216
2	4921	4036.0	163.0	3072
3	2851	23456.2	-129.0	496

[45]:
```
# これがベスト
df.style.format('{:,}')
```

[45]: <pandas.io.formats.style.Styler at 0x7fe377a79c70>

[46]:
```
# メソッドチェーンでもいける
df.style.format('{:,}').applymap(less_than_zero).highlight_null()
```

[46]: <pandas.io.formats.style.Styler at 0x7fe377a6ec70>

2.0.1 注意

- 以下の関数は単体で使うには問題ないが、他条件とメソッドチェーンができない

[47]:
```
def thousands(val):
    try:
        return "{:,}".format(val)
    except ValueError as e:
        return val
```

■style を抜くと効く、style 付与すると NG

[48]:
```
display(
    df.applymap(thousands), #  こっちは効くが
    df.style.applymap(thousands) # 桁区切りが適用されてない
)


# 以下だとエラーになる
#     df.applymap(thousands).applymap(less_than_zero).highlight_null() #  で他条件とメソッドチェーン不可
```

	A	B	C	D
0	599	135.0	222.0	897

```
1   -299     -1,463.0  -10.34567  4,216
2  4,921      4,036.0      163.0  3,072
3  2,851  23,456.2345     -129.0    496
```

```
<pandas.io.formats.style.Styler at 0x7fe377a6e610>
```

---

3 以下、nkmk より。

- https://note.nkmk.me/python-pandas-option-display/
- https://note.nkmk.me/python-pandas-option-setting/

[5]:
```
import pandas as pd
import numpy as np
import pprint
```

4 pandas の表示設定変更

4.1 バージョン確認

- バージョンが違うと設定のデフォルト値などが違う場合あり

[6]:
```
print(pd.__version__)
```

```
1.2.4
```

4.2 小数点以下の桁数：display.precison

- デフォルト：6
- 小数点以下の桁数を制御する

[7]:
```
s_decimal = pd.Series([123.456, 12.3456, 1.23456, 0.123456, 0.0123456, 0.00123456])
```

[8]:
```
print(s_decimal)
```

```
0    123.456000
1     12.345600
2      1.234560
3      0.123456
4      0.012346
```

```
5      0.001235
dtype: float64
```

- 設定値により変わる

[12]:
```
pd.options.display.precision = 4


print(s_decimal)
```

```
0    123.4560
1     12.3456
2      1.2346
3      0.1235
4      0.0123
5      0.0012
dtype: float64
```

[13]:
```
pd.options.display.precision = 2


print(s_decimal)
```

```
0    1.23e+02
1    1.23e+01
2    1.23e+00
3    1.23e-01
4    1.23e-02
5    1.23e-03
dtype: float64
```

4.3 有効数字（有効桁数）：display.float_format

- 小数点や整数部も含む有効数字（有効桁数）を指定する場合
- デフォルト：None
- ‘.[桁数]f’ ・・・ 小数点以下の桁数
- ‘.[桁数]g’ ・・・ 全体の桁数（有効数字）、を指定

[15]:
```
pd.options.display.float_format = '{:.2f}'.format
print(s_decimal)
```

```
0   123.46
1    12.35
```

```
2     1.23
3     0.12
4     0.01
5     0.00
dtype: float64
```

[16]:
```
pd.options.display.float_format = '{:.4g}'.format
print(s_decimal)
```

```
0      123.5
1      12.35
2      1.235
3     0.1235
4    0.01235
5   0.001235
dtype: float64
```

4.4 四捨五入についての注意

- display.precison、display.float_format では値が丸められるが、四捨五入ではなく偶数への丸めとなる
- 例）0.5 は 0 に丸められる

4.4.1 偶数への丸め（round to even）・・・わかりにくい！とばす

- 偶数への丸めは、端数が 0.5 より小さいなら**切り捨て**
- 端数が 0.5 より大きいならは**切り上げ**
- **端数がちょうど 0.5 なら切り捨てと切り上げのうち「結果が偶数」となる方へ丸める**
- (つまり偶数 +0.5 なら切り捨て、奇数 +0.5 なら切り上げとなる)

[23]:
```
df_decimal = pd.DataFrame({'s': ['0.4', '0.5', '0.6', '1.4', '1.5', '1.6', '2.5',␣
↪'2.6','3.6', '4.5'],
                           'f': [0.4, 0.5, 0.6, 1.4, 1.5, 1.6, 2.5, 2.6, 3.6, 4.5]})

pd.options.display.float_format = '{:.0f}'.format

print(df_decimal)
```

```
     s  f
0  0.4  0
1  0.5  0
2  0.6  1
```

```
3  1.4  1
4  1.5  2
5  1.6  2
6  2.5  2
7  2.6  3
8  3.6  4
9  4.5  4
```

4.4.2 小数点以下で丸める場合は、値によって偶数への丸めになったり、奇数への丸めになったりする

[28]:
```
df_decimal2 = pd.DataFrame({'s': ['0.04', '0.05', '0.06', '0.14', '0.15', '0.16',␣
↪'0.24', '0.25', '0.26', '0.34', '0.35', '0.36'],
                            'f': [0.04, 0.05, 0.06, 0.14, 0.15, 0.16, 0.24, 0.25, 0.
↪26, 0.34, 0.35, 0.36]})

pd.options.display.float_format = '{:.1f}'.format

print(df_decimal2)
```

```
      s   f
0   0.04 0.0
1   0.05 0.1
2   0.06 0.1
3   0.14 0.1
4   0.15 0.1
5   0.16 0.2
6   0.24 0.2
7   0.25 0.2
8   0.26 0.3
9   0.34 0.3
10  0.35 0.3
11  0.36 0.4
```