1行目のstr.upper()で商品名の小文字を大文字に変換し統一しています。
2、3行目のstr.replace()で商品名の半角・全角スペースを除去しています。
4行目でデータitem_name順にソートし画面上に表示しています。
表をスクロールしてデータを見る限り、正しく補正されたように見えますが、必ず結果を検証する事を忘れてはいけません。

```
print(pd.unique(uriage_data["item_name"]))
print(len(pd.unique(uriage_data["item_name"])))
```

■図2-9：商品名の補正結果検証

```
In [10]: print(pd.unique(uriage_data["item_name"]))
         print(len(pd.unique(uriage_data["item_name"])))

         ['商品A' '商品S' '商品Z' '商品V' '商品O' '商品U' '商品L' '商品C' '商品I' '商品R' '商品X' '商品G'
          '商品P' '商品Q' '商品Y' '商品N' '商品W' '商品E' '商品K' '商品B' '商品F' '商品D' '商品M' '商品H'
          '商品T' '商品J']
         26
```

先ほどと同じく、unique()関数で商品名の一覧とその数を取得します。
結果、A～Zの商品に統一され、件数も26件となり、商品名におけるデータの揺れが解消した事が確認できました。

ノック15：金額欠損値の補完をしよう

続いて、金額の欠損値の補完をしてみましょう。
欠損値によっては現場にヒアリングを行い補完するか、スタッフに欠損値を埋めてもらう等の対応が必要になる事がありますが、今回のケースでは商品単価が集計中に変動していない事から、プログラムで欠損値を補完する事ができます。

まずはデータ全体から、欠損値が含まれているか確認する所から始めます。

```
uriage_data.isnull().any(axis=0)
```


■図2-10：欠損値チェック結果

```
In [11]: uriage_data.isnull().any(axis=0)

Out[11]: purchase_date     False
         item_name         False
         item_price         True
         customer_name     False
         purchase_month    False
         dtype: bool
```

売上履歴データに対して、isnull()関数を用いる事で欠損値の有無を確認する事ができます。
今回の結果を見ると、「item_price　True」となっていて、金額項目に欠損値が含まれている事が確認できます。

早速、金額の欠損値を補完していきましょう。
今回のケースは集計期間中に商品単価の変動はないという前提条件がありますので、欠損値は他の同じ商品の単価から補完ができる事が分かります。

少し複雑な処理になりますが、欠損値の補完を行ってみましょう。

```
flg_is_null = uriage_data["item_price"].isnull()
for trg in list(uriage_data.loc[flg_is_null, "item_name"].unique()):
    price = uriage_data.loc[(~flg_is_null) & (uriage_data["item_name"] ==
trg), "item_price"].max()
    uriage_data["item_price"].loc[(flg_is_null) & (uriage_data["item_name"
]==trg)] = price
uriage_data.head()
```