同様に、「item_price」についても確認してみましょう。

図2-4：データの揺れ（商品金額）

```
In [4]: uriage_data["item_price"].head()

Out[4]: 0     100.0
        1       NaN
        2       NaN
        3    2600.0
        4       NaN
        Name: item_price, dtype: float64
```

欠損値「NaN」がデータとして確認できます。このような状態をデータ欠損といい、欠損値をどのように補完するかが今後のデータ分析に影響します。

このように集計対象のデータに揺れや欠損値が存在していると正しい集計が得られません。

試しに、このまま集計を行ってみましょう。

ノック13：データに揺れがあるまま集計してみよう

データの揺れがどれくらい集計に影響するかを確認する事で、いかにデータの整合性が重要か分かると思います。

まずは「売上履歴」から商品ごとの月売上合計を集計してみましょう。

```
uriage_data["purchase_date"] = pd.to_datetime(uriage_data["purchase_dat
e"])
uriage_data["purchase_month"] = uriage_data["purchase_date"].dt.strftime("
%Y%m")
res = uriage_data.pivot_table(index="purchase_month", columns="item_name",
aggfunc="size", fill_value=0)
res
```


図2-5：データ補正前の集計結果（商品毎）

ノック１３：データに揺れがあるまま集計しよう

```
In [6]: uriage_data["purchase_date"] = pd.to_datetime(uriage_data["purchase_date"])
        uriage_data["purchase_month"] = uriage_data["purchase_date"].dt.strftime("%Y%m")
        res = uriage_data.pivot_table(index="purchase_month", columns="item_name", aggfunc="size", fill_value=0)
        res
```

Out[6]:

item_name	商品W	商品n	商品E	商品M	商品P	商品S	商品W	商品X	商品O	商品Q	…	商品k	商品l	商品o	商品p	商品r	商品s	商品t	商品v	商品x	商品y
purchase_month																					
201901	0	1	0	0	0	0	0	0	0	0	…	1	1	1	0	0	0	0	0	0	0
201902	0	0	0	0	0	0	0	1	0	0	…	0	0	0	0	0	1	1	1	0	0
201903	0	0	1	1	1	0	0	0	0	0	…	0	0	0	0	0	0	0	0	0	0
201904	1	0	0	0	0	0	0	0	0	1	…	0	0	0	0	0	1	0	0	0	0
201905	0	0	0	0	0	1	0	0	0	0	…	0	1	0	0	0	0	0	0	0	1
201906	0	0	0	0	0	0	1	0	0	0	…	0	0	0	1	0	0	0	0	1	0
201907	0	0	0	0	0	0	0	0	1	0	…	0	0	1	0	2	0	0	0	0	0

7 rows × 99 columns

１～２行目は日付型の定義と、日付を年月の形に変換を行っています。これは**第１章**でも実施していますので復習しておきましょう。集計単位に合わせて日付を変換する処理は実際かなり行われます。

３行目で縦軸に購入年月、横軸に商品として件数を集計しています。

４行目は画面上に集計結果を表示しています。

データの揺れを補正せずに集計をしてみました。結果の表を見ると、一部省略されていますが、「商品S」や「商品ｓ」等、本来同じ商品が別の商品として集計されている事が確認できます。

また、表の最後の「7 rows × 99 columns」に注目してください。

今回横軸（columns）は「商品」として集計しました。つまり、データの揺れがあるため、本来26個の商品が99商品に増えてしまっている事が分かります。

同様に横軸に「item_price」を設定し集計してみましょう。

```
res = uriage_data.pivot_table(index="purchase_month", columns="item_name",
values="item_price", aggfunc="sum", fill_value=0)
res
```