

CSV読み込みの補足

- 日本語が含まれているCSVファイルは、文字コードがUTF-8ではなくShift JIS (あるいは EUC-JP) になっていることがあります。
- UTF-8以外のファイルはJupyterLabのEditorで開くことができません。また、`read_csv()` で読み込もうとするとエラーになります。
- この場合、`read_csv()` の`encoding`オプションで文字コードを正しく指定すると読み込むことができます。

```
df = pd.read_csv(csv_in, sep=',', skiprows=N, header=M,  
                 encoding='shift-jis')
```

※ EUC-JPの場合は、`encoding='euc-jp'`

UTF-8以外のファイルを開くには

- VSCode を用いると、UTF-8以外のファイルの中身を確認することができます。
- cs3-02フォルダの下、以下のファイルの内容をVSCodeで確認してみましょう。エクスプローラなどで cs3-02 フォルダを開き、それぞれのファイルを次のページ以降の手順で VSCode で開きます。
 - sample-euc-spaces.csv
 - sample-sjis-semicolon.csv
 - sample-sjis-tab.csv
- まず、sample-sjis-semicolon.csv をVSCodeで開いてみます。

sample-sjis-semicolon.csv をVSCodeで開く

```

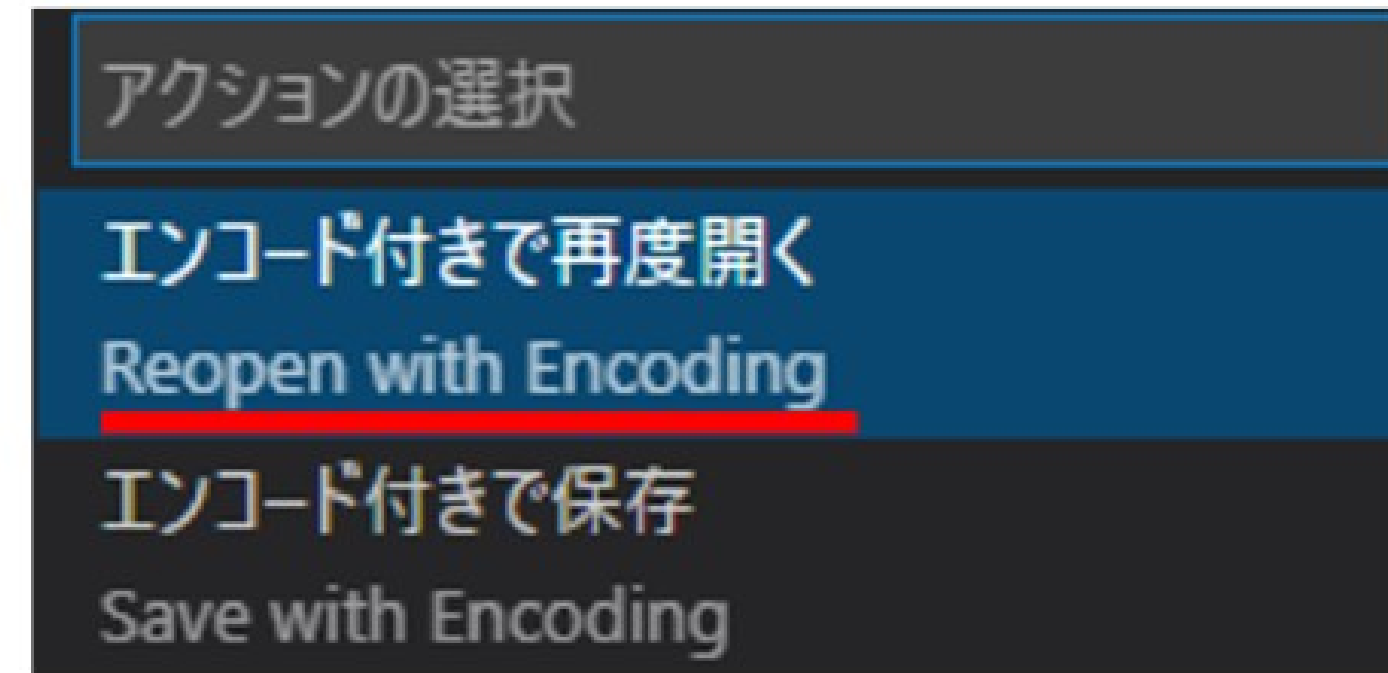
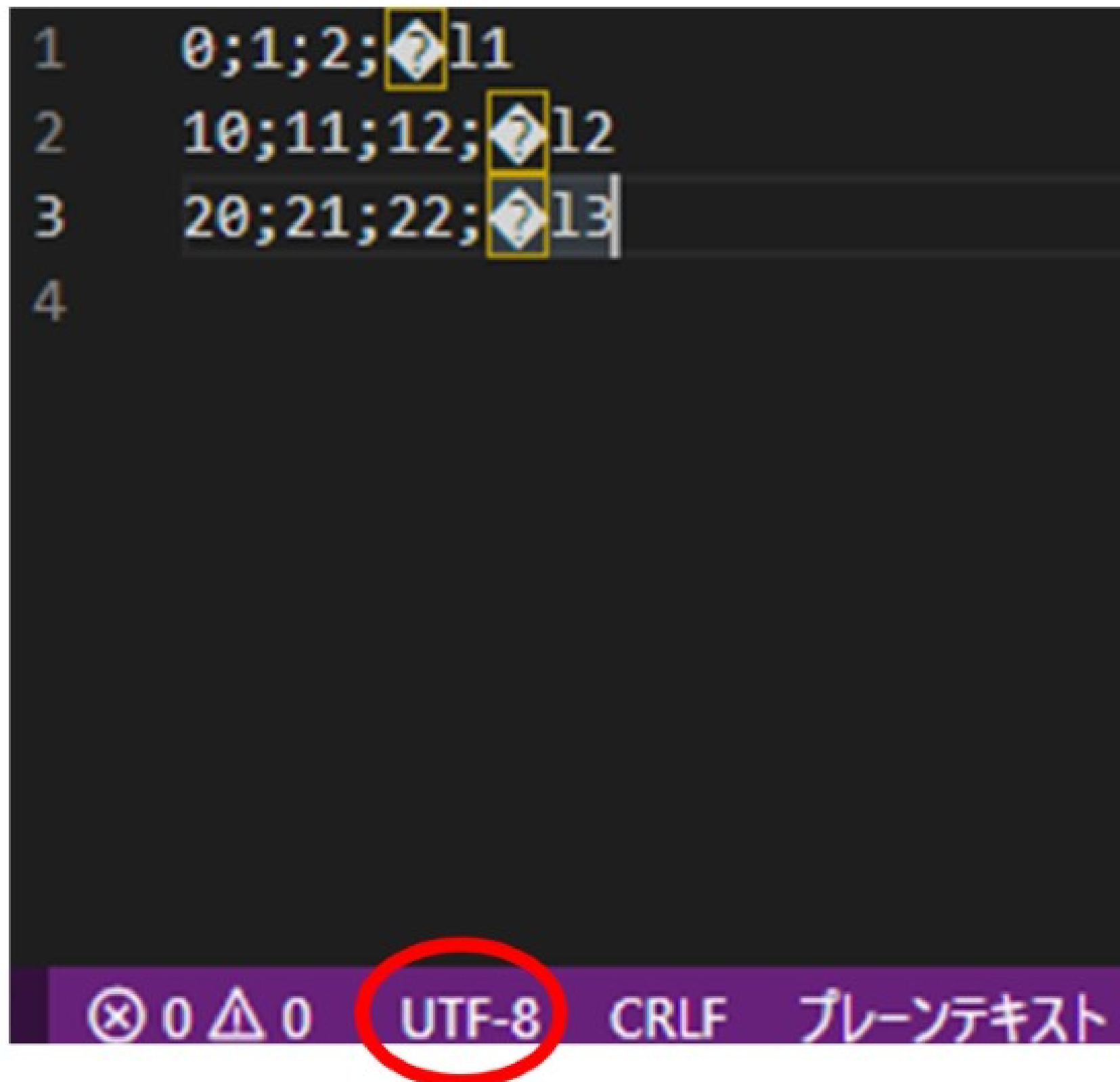
1  0;1;2;11
2  10;11;12;12
3  20;21;22;13
4

```

⊗ 0 △ 0 UTF-8 CRLF プレーンテキスト

中身が UTF-8 でないため、
漢字が文字化けしている

現在の表示用の文字コードは
UTF-8



「エンコード付きで再度開く」
をクリック



「Japanese (Shift JIS)」 「Japanese (EUC-JP)」
のうち文字化けしない方を選択


```
1 0;1;2;値1
2 10;11;12;値2
3 20;21;22;値3
4
```

このファイルは Shift JIS の方で
字化けが解消したので、Shift JIS
だとわかる。

以下の3点をよく確認する。

- 読み飛ばすべき先頭の実データ行
はなし。
- 列ラベルの行もなし。
- 区切り字は「;」。

⊗ 0 △ 0 **Shift JIS** CRLF プレーンテキスト

(再掲) CSVファイルの読み込み

変数 = pd.read_csv('CSVファイル名', sep=',', skiprows=**N**, header=**M**)

- 第1引数に直接 CSVファイル名を指定することももちろんできます。
- 区切り字がTab文字の場合は sep='¥t'、(可変個数の)空白文字の場合は sep='¥s+' とします。また、sep= を省略した場合は区切り字はコンマとみなされます。
- 先頭の読み飛ばすべきデータ以外の行がない場合は、skiprows=**0** とします。
- 列ラベルの行がない場合は、header=**None** とします。
- 漢字コードがutf8以外 (たとえば shift-jis) の場合は、オプション引数 encoding='shift-jis' を追加します。

(Markdown) ##### Check read_csv options

```
csv_in = 'sample-sjis-semicolon.csv'
df_check = pd.read_csv(csv_in, sep=';', skiprows=0, header=None,
                        encoding='shift-jis')

print(df_check.shape)
display(df_check.head())
df_check.columns = ['c0', 'c1', 'c2', 'c3']
display(df_check.head())
```

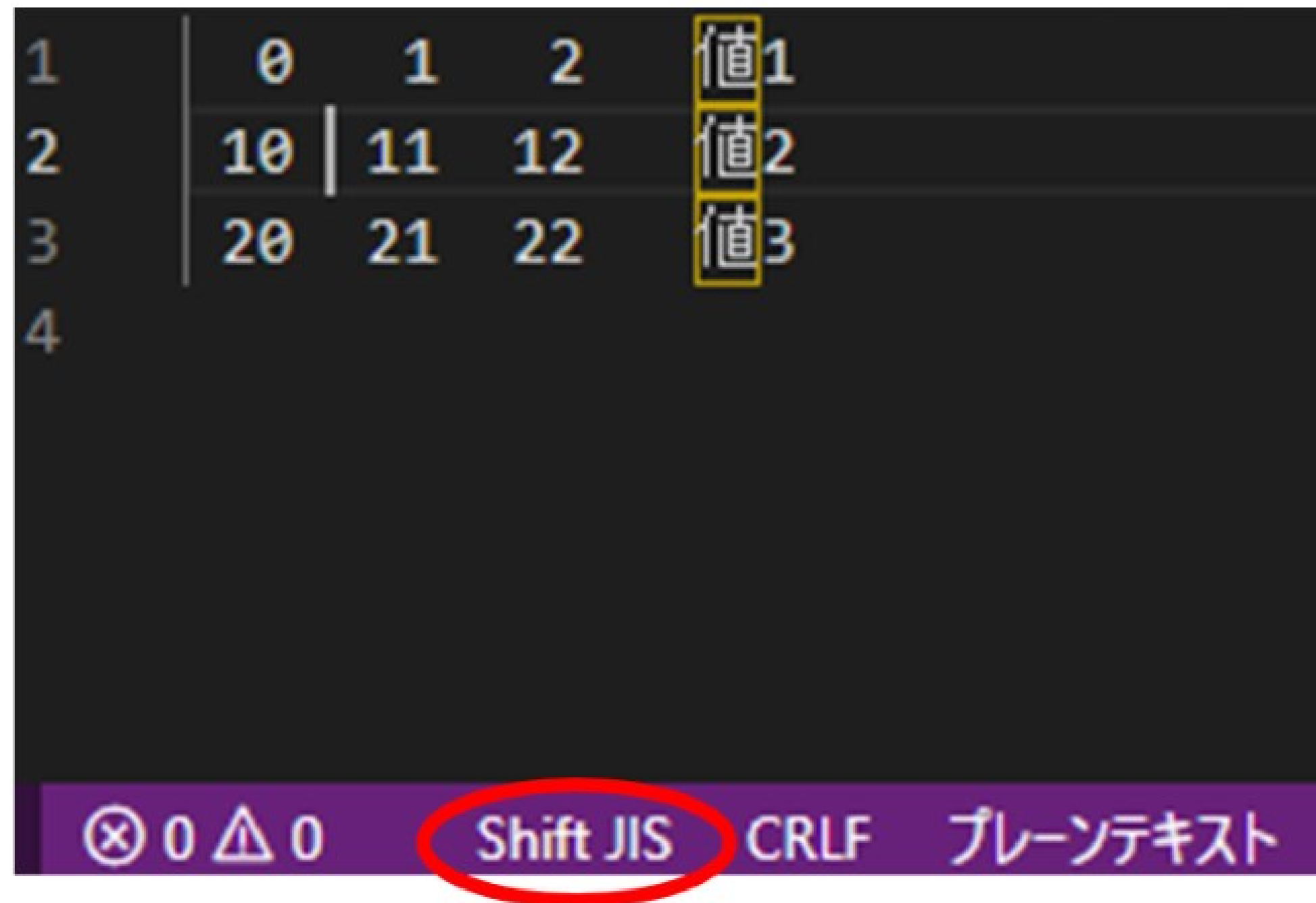
正しく、3行4列のデータ
として読み込めている。
columns はあとで付加。

(3, 4)				
	0	1	2	3
0	0	1	2	値1
1	10	11	12	値2
2	20	21	22	値3

	c0	c1	c2	c3
0	0	1	2	値1
1	10	11	12	値2
2	20	21	22	値3

同様にして、sample-sjis-spaces.csv の文字コードを同定。

VSCoDe で開いた結果、Shift JIS だとわかる。



また、ファイル内でカーソルを移動させて確認すると、データは可変個数の空白文字で区切られていることがわかる。

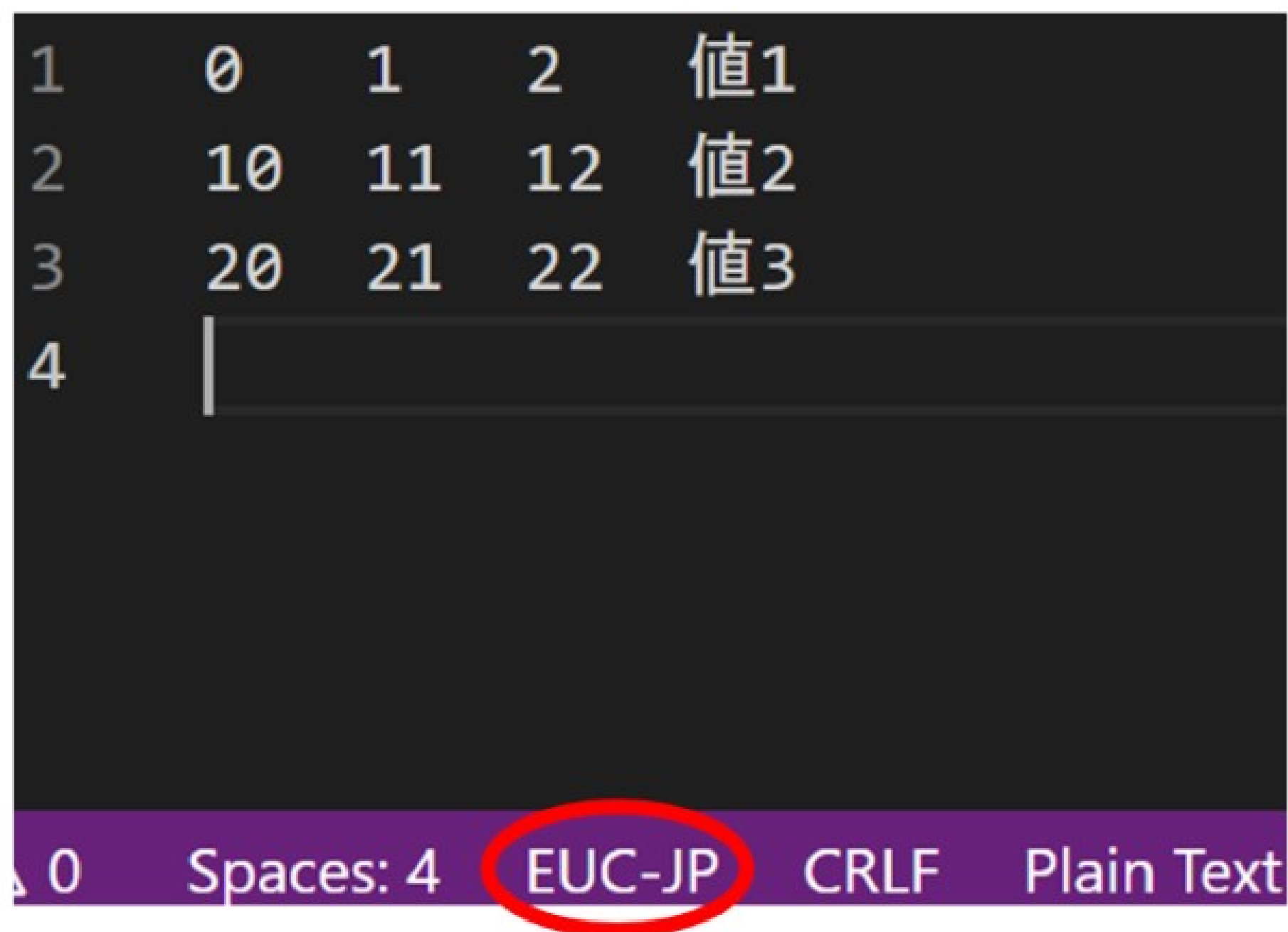

```
csv_in = 'sample-sjis-spaces.csv'
df_check = pd.read_csv(csv_in, sep='¥s+', skiprows=0, header=None,
                        encoding='shift-jis')
df_check.columns = ['c0', 'c1', 'c2', 'c3']
print(df_check.shape)
display(df_check.head())
```

正しく、3行4列のデータ
として読み込めている。
columns はあとで付加。

(3, 4)				
	c0	c1	c2	c3
0	0	1	2	値1
1	10	11	12	値2
2	20	21	22	値3

同様にして、sample-euc-tab.csv の文字コードを同定。

VSCode で開いた結果、EUC-JP だとわかる。



The screenshot shows a VS Code editor window with a CSV file. The file content is as follows:

	0	1	2	値
1				値1
2	10	11	12	値2
3	20	21	22	値3
4				

The status bar at the bottom indicates the file is 0 bytes, contains 4 spaces, and is encoded in EUC-JP (highlighted with a red circle), using CRLF line endings, and is Plain Text.

また、ファイル内でカーソルを移動させて確認すると、データは1文字のタブ文字で区切られていることがわかる。

Tab文字について

- 伸び縮みする(可変長の)空白文字。
- 表組の列を縦に整列させたいときなどに用いられる。

0	→	1	→	2	→	値1
10	→	11	→	12	→	値2
20	→	21	→	22	→	値3

赤矢印1つが Tab文字1つ。

Tab文字が区切り字になっている場合は、`pd.read_csv()` では `sep='¥t'` を指定する。なお、Tab文字区切りの場合は、ファイル名の拡張子が `.csv` の代わりに `.tsv` となっている場合もある。

```
csv_in = 'sample-euc-tab.csv'
df_check = pd.read_csv(csv_in, sep='¥t', skiprows=0, header=None,
                        encoding='euc-jp')
df_check.columns = ['c0', 'c1', 'c2', 'c3']
print(df_check.shape)
display(df_check.head())
```

正しく、3行4列のデータ
として読み込めている。
columns はあとで付加。

	c0	c1	c2	c3
0	0	1	2	値1
1	10	11	12	値2
2	20	21	22	値3

注意: DataFrame や ndarray などのコピーについて

- DataFrame, Series, ndarray は Python の list と同様に「ミュータブル」なので、単なる代入は「別名」がつくだけで実体は同じになってしまう。
- このため、list と同じく「複製」するための copy メソッドが用意されている。

例

```
nd = np.array([0, 1, 2])  
nd2 = nd.copy()
```

```
ser = pd.Series([0, 1, 2])  
ser2 = ser.copy()
```

```
df = pd.DataFrame([[0, 1, 2], [3, 4, 5]])  
df2 = df.copy()
```