

# コンピュータリテラシ発展 ～Pythonを学ぶ～

## 第6回：Excel作業を自動化しよう(2)

情報学部 情報学科 情報メディア専攻

清水 哲也 ( [shimizu@info.shonan-it.ac.jp](mailto:shimizu@info.shonan-it.ac.jp) )

# 今回の授業内容

# 今回の授業内容

- 前回の課題解説
- Excelファイルを編集する
- Excelのレイアウトを編集する
- 課題

## 前回の課題解説

# 前回の課題解説

- 前回の課題の解答例を示します
- 解答例について質問があればご連絡ください

## 解答例

[https://colab.research.google.com/drive/1uG8eKYq9ryLH-ght91ojlO4XK-z96mbJ?  
usp=sharing](https://colab.research.google.com/drive/1uG8eKYq9ryLH-ght91ojlO4XK-z96mbJ?usp=sharing)

# Excelファイルを編集する

# Excelファイルを新規作成する

- Excelファイル（ワークブック）を新規作成します

```
import openpyxl as op

wb = op.Workbook()
wb.save('filename.xlsx')
```

- `workbook` の引数を空にすると新しいワークブックを読み込みます
- `save()` メソッドで保存します
- 保存作為を指定したい場合はパスを含めて指定する必要があります

# Excelファイルを新規作成する

- Excelファイル（ワークブック）を新規作成します（パスを含む場合）

```
import openpyxl as op
```

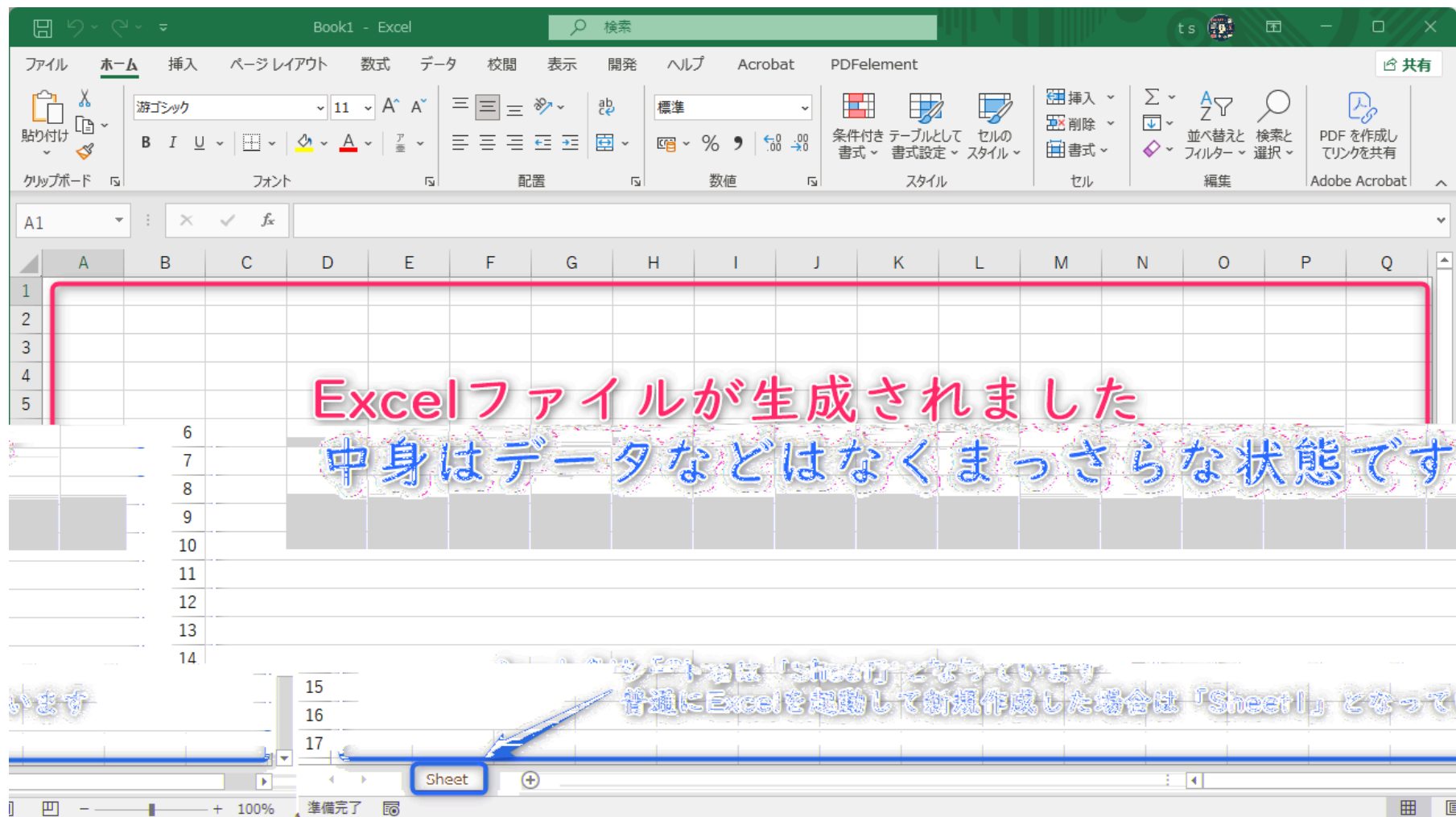
```
wb = op.Workbook()
```

```
wb.save('/content/drive/MyDrive/???/filename.xlsx')
```

- `workbook` の引数を空にすると新しいワークブックを読み込みます
- `save()` メソッドで保存します
- 保存作為を指定したい場合はパスを含めて指定する必要があります



# 新規作成したExcelファイルの確認



# Excelシートを追加/削除します

- 新しいシートを追加します

```
Workbookオブジェクト.create_sheet()
```

- 挿入位置とシート名を指定してシートを追加します

```
Workbookオブジェクト.create_sheet(index = 数字, title = 'シート名')
```

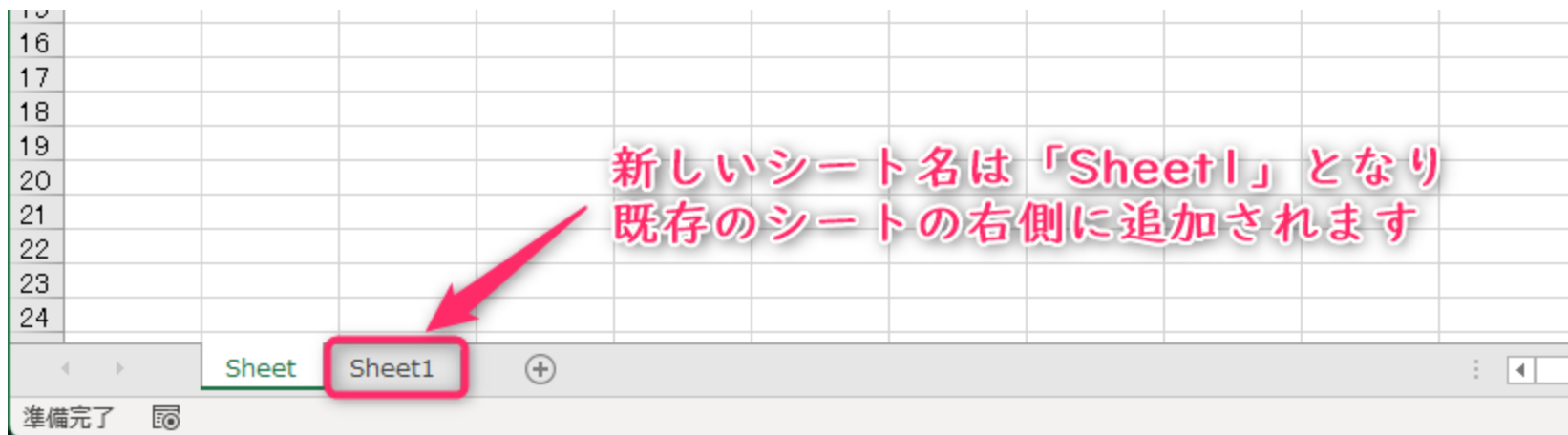
- Excelシートを削除します

```
Workbookオブジェクト.remove(Worksheetオブジェクト)
```

# Excelシートを追加/削除します

- 新しいシートを追加します

```
wb = op.Workbook()  
  
wb.create_sheet()  
print(wb.sheetnames)  
wb.save('/content/drive/MyDrive/???/create_sheet.xlsx')
```

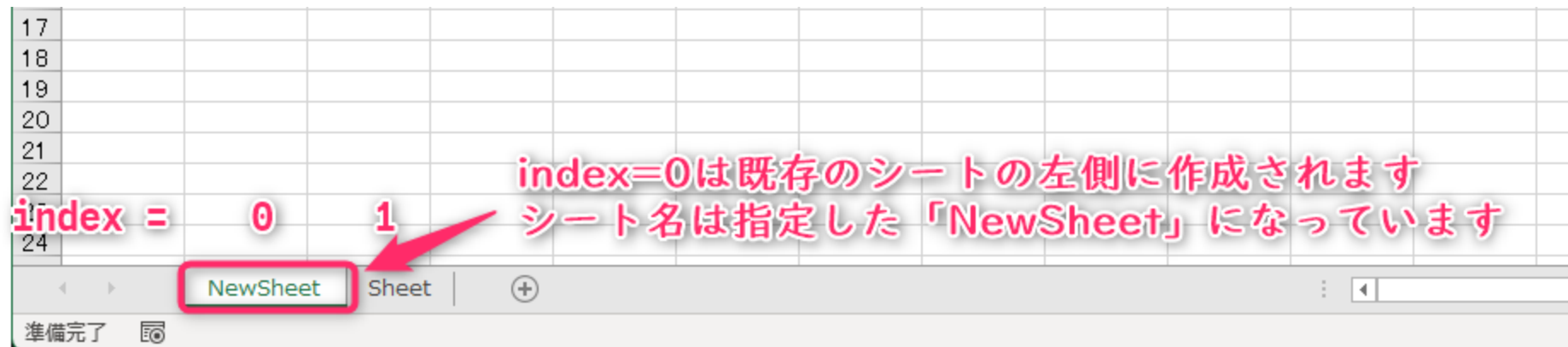


# Excelシートを追加/削除します

- 挿入位置とシート名を指定して新しいシートを追加します

```
wb = op.Workbook()

wb.create_sheet(index = 0, title = 'NewSheet')
print(wb.sheetnames)
wb.save('/content/drive/MyDrive/???/create_sheet.xlsx')
```



# Excelシートを追加/削除します

- シートを追加して既存のシートを削除します

```
wb = op.Workbook()

wb.create_sheet()
print(wb.sheetnames)

wb.remove(wb['Sheet'])
print(wb.sheetnames)
```

## セルの値を編集します

- 指定したセルを編集します
  - 文字列を入力します : `Worksheetオブジェクト[セル] = '文字列'`
  - 数値を入力します : `Worksheetオブジェクト[セル] = 数値`
  - 数式を入力します : `Worksheetオブジェクト[セル] = '=数式'`

## セルの値を編集します（結果）

	A	B	C	D	E
1					
2		文字列			
3		10			
4		10			
5		20			
6		30			

Diagram illustrating the result of editing cell values in a spreadsheet. The cells are highlighted with red boxes, and arrows point to the corresponding data type labels:

- Cell B2 (Text): 文字列 (Text)
- Cell B3 (Number): 数値 (Number)
- Cell B4 (Formula): 数式 (Formula)

The spreadsheet interface shows the following data:

	A	B	C	D	E
1					
2		文字列			
3		10			
4		10			
5		20			
6		30			

# フォントを設定します

- `Font()` 関数：セルのフォント設定をする関数です
- `Font()` 関数を使用するには `op.styles.fonts.Font` と書く必要があります
- 書くのが大変なので関数を指定してインポートしておきます

```
import openpyxl as op
from openpyxl.styles.fonts import Font
```

- `Font` 関数の基本的な使い方です

```
Font(キーワード引数1=値, キーワード引数2=値・・・)
```

詳細：<https://openpyxl.readthedocs.io/en/stable/styles.html>



# フォントを設定します

- Font 関数の基本的な使い方です

```
Font(キーワード引数1=値, キーワード引数2=値 . . . )
```

- 例：フォントサイズを18pt, 太文字にする設定です

```
Font(size=18, bold=True)
```

- 例：フォントサイズを24pt, 斜体にする設定です

```
Font(size=24, italic=True)
```

## Font() 関数の代表的な引数

引数名	データ型	解説
name	文字列型	フォント名を指定します（例： <code>name='メイリオ'</code> ）
size	整数型	フォントsizeを変更します（例： <code>size=18</code> ）
bold	ブール型	<code>True</code> で太文字になります（例： <code>bold=True</code> ）
italic	ブール型	<code>True</code> でイタリック(斜体)になります（例： <code>italic=True</code> ）
color	文字列型	カラーコードで文字の色を指定します（例： <code>color='FF0000'</code> ）
strike	ブール型	<code>True</code> で打ち消し線が引けます（例： <code>strike=True</code> ）

## フォントを設定します

```
wb = op.Workbook()
sheet = wb.active

sheet['B2'] = '18pt bold'
sheet['B2'].font = Font(size=18, bold=True)

sheet['B4'] = '24pt 下線'
sheet['B4'].font = Font(size=24, underline='single')

wb.save('/content/drive/MyDrive/???/font.xlsx')
```

## フォントを設定します（結果）

	A	B	C	D	E	
2		<b>18pt bold</b>				
3						
4		<u><b>24pt 下線</b></u>				
5						
6						
7						

## Excelのレイアウトを編集

# Excelの行高と列幅を設定する

- Excelの行高と列幅を設定する方法
  - 行高：行番号を指定して高さの数値（ポイント）を入力します

```
Worksheetオブジェクト.row_dimensions[行番号].height = 高さの数値
```

- 列幅：列番号を指定して幅の数値（文字数）を入力します

```
Worksheetオブジェクト.column_dimensions[列番号].width = 幅の数値
```

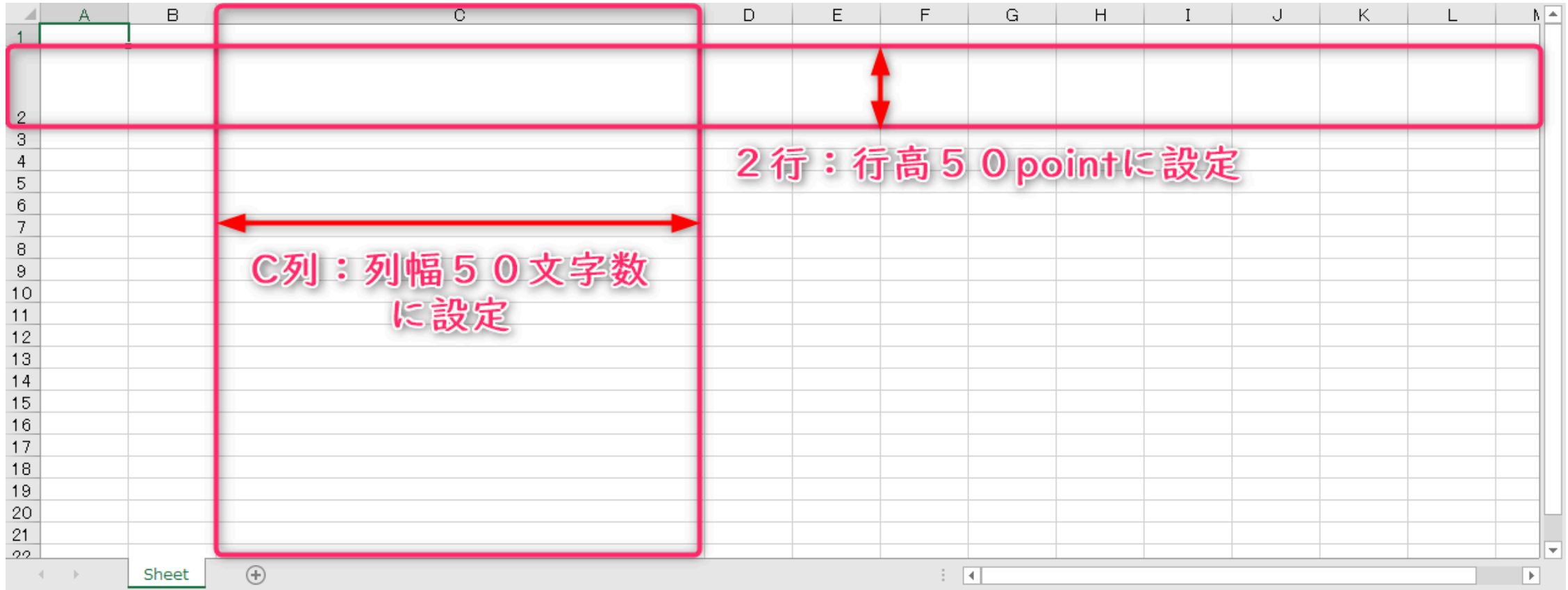
行高と列幅で単位が異なるので注意が必要です

# Excelの行高と列幅を設定する

- Excelの行高と列幅を設定する例です
  - 行高の設定：2行目を「50」に設定します
  - 列幅の設定：C列目を「50」に設定します

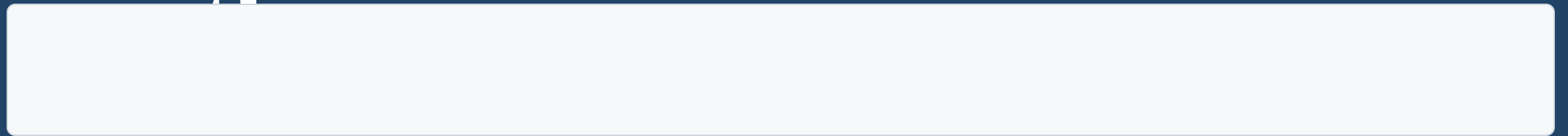
```
wb = op.Workbook()  
sheet = wb.active  
  
sheet.row_dimensions[2].height = 50  
sheet.column_dimensions['C'].width = 50  
wb.save('/content/drive/MyDrive/???/row_column.xlsx')
```

# Excelの行高と列幅を設定する（結果）





Excelの行と列を非表示にする



# Excelの行・列を非表示にする

- 2行目とB,D列を非表示設定にする例です

# Excelの行・列を非表示にする

- 2行目とB,D列を非表示設定にする例です

	A	B	C	D	E	F	G
1	1		3		5		
2	2		6		10		
3	4		12		20		
4	5		15		25		
5							
6							
7							
8							
9							

3行目が非表示

B列とD列が非表示

# Excelの行と列を固定表示にする

- Excelの行と列を固定表示します
  - 固定する行や列の下もしくは右，右下のセルを指定します
  - 例：1行目のみ固定する場合：指定するセルは「A2」
  - 例：B列のみ固定する場合：指定するセルは「C1」
  - 例：2行目とC列まで固定する場合：指定するセルは「D3」

```
Worksheetオブジェクト.freeze_panes = セル
```

# Excelの行と列を固定表示にする

- 2行目までを固定にします

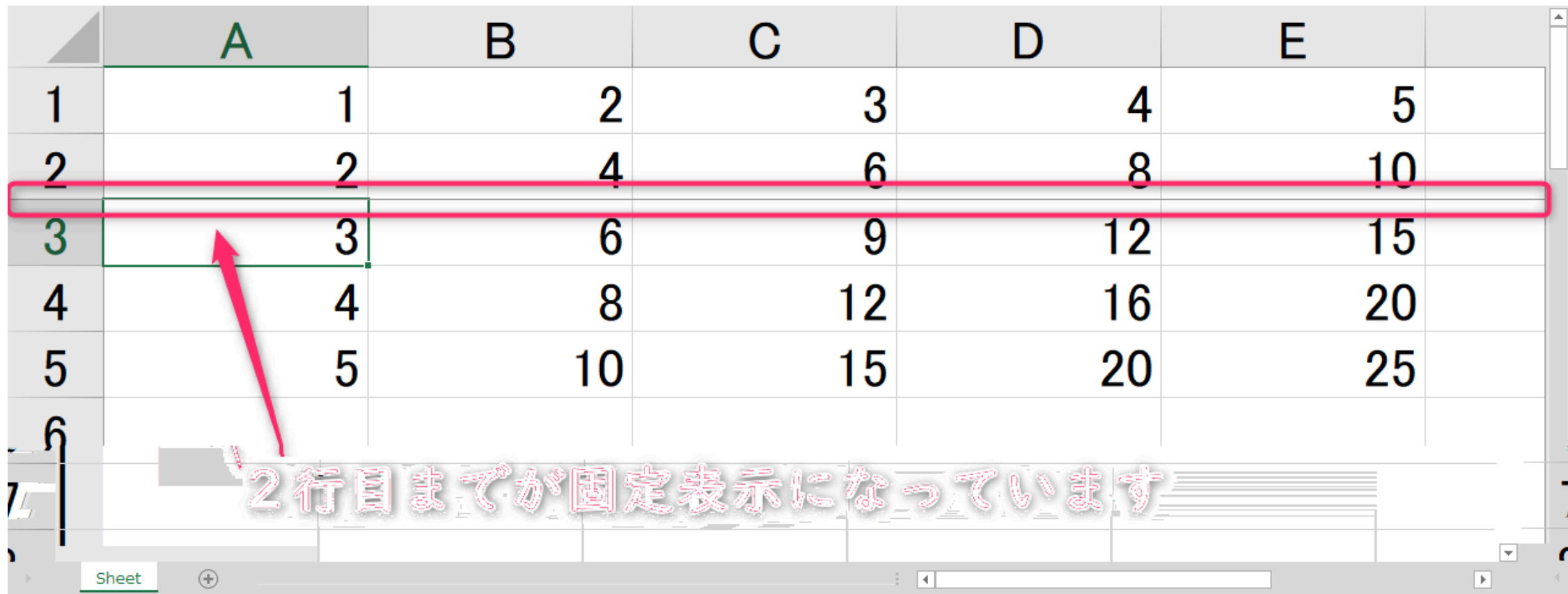
```
wb = op.Workbook()
sheet = wb.active

#セルA1～E5に1～25の数字を入れる
for i in range(1,6):
    for j in range(1,6):
        sheet.cell(j,i).value = i * j

sheet.freeze_panes = 'A3'          # 2行目までを固定:セルA3を指定

wb.save('/content/drive/MyDrive/???/freeze-panes.xlsx')
```

## Excelの行と列を固定表示にする(結果)



The image shows an Excel spreadsheet with columns A through E and rows 1 through 6. The data in the spreadsheet is as follows:

	A	B	C	D	E
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20
5	5	10	15	20	25
6					

A red box highlights row 2, and a red arrow points to it from a text box at the bottom that reads: 2行目までが固定表示になっています (Rows 1 and 2 are fixed).

# 課題

# 課題

- Moodleにある「SCfCL-6th-prac.ipynb」ファイルをダウンロードしてColabにアップロードしてください
- 課題が完了したら「File」>「Download」>「Download .ipynb」で「.ipynb」形式でダウンロードしてください
- ダウンロードした **.ipynb**ファイル と作成した **Excelファイル3つ** をMoodleに提出してください
- 提出期限は **5月30日(木) 20時まで** です