Pythonのライブラリxlrdを使うと、Excelファイル（`.xls`, `.xlsx`）の読み込み、xlwtを使うとExcelファイル（`.xls`）の書き込みができる。それぞれの使い方を説明する。

PythonでExcelファイルを扱うライブラリの違いや使い分けなどは以下の記事を参照。

- **関連記事:** PythonでExcelファイルを扱うライブラリの比較

数値や文字列のデータを読み込んで分析するのであればpandas、既存のExcelファイルの書式を保ったまま編集・追記する場合はopenpyxlがおすすめ。

- **関連記事:** pandasでExcelファイル（xlsx, xls）の読み込み（read_excel）
- **関連記事:** PythonでExcelファイル（xlsx）を読み書きするopenpyxlの使い方

ここでは以下の内容について説明する。

- xlrd, xlwtのインストール
- xlrdの使い方（`xls`, `xlsx`ファイルの読み込み）
- xlwtの使い方（`xls`ファイルの書き込み）

xlrd, xlwtのインストール

どちらも`pip`でインストールできる（環境によっては`pip3`）。

```
$ pip install xlrd
$ pip install xlwt
```

xlrdの使い方（xls, xlsxファイルの読み込み）

xlrdは`xls`, `xlsx`ファイルの読み込みを行うライブラリ。GitHubのレポジトリとドキュメントは以下のリンクから。

- python-excel/xlrd: Library for developers to extract data from Microsoft Excel (tm) spreadsheet files

- xlrd documentation — xlrd 1.1.0 documentation

xlrdでは以下のクラスが定義されている。

- `Book`クラス: ワークブック全体
- `Sheet`クラス: 一つのシート
- `Cell`クラス: 一つのセル

ここでは以下の`xlsx`ファイルを例にセルの値を取得する方法を説明する。

- sample.xlsx

`sheet1`と`sheet2`の二つのワークシートを持つ。`sheet1`の中身は以下の通り。

```
    A   B   C
one 11  12  13
two 21  22  23
three   31  32  33
```

xlrdをインポート。結果を見やすくするためにpprintモジュールもインポートしている。

- **関連記事:** Pythonのpprintの使い方（リストや辞書を整形して出力）

```
import xlrd
import pprint
```

source: xlrd_example.py

`xlrd.open_workbook()`にExcelファイルのパスを指定して`Book`オブジェクトを取得。
`sheet_names()`メソッドでシート名一覧のリストを取得できる。

```
wb = xlrd.open_workbook('data/src/sample.xlsx')

print(type(wb))
# <class 'xlrd.book.Book'>

print(wb.sheet_names())
# ['sheet1', 'sheet2']
```

source: xlrd_example.py

`Book`オブジェクトのメソッド`sheets()`で`Sheet`オブジェクトのリストを取得。

```
sheets = wb.sheets()

print(type(sheets))
# <class 'list'>

print(type(sheets[0]))
# <class 'xlrd.sheet.Sheet'>
```

source: xlrd_example.py

`sheet_by_index()`にシート番号、または、`sheet_by_name()`にシート名を指定して特定のシートのみを取得することも可能。

```
sheet = wb.sheet_by_name('sheet1')

print(type(sheet))
# <class 'xlrd.sheet.Sheet'>
```

source: xlrd_example.py

`Sheet`オブジェクトのメソッド`cell()`に0始まりの行番号・列番号を指定して`Cell`オブジェクトを取得。`Cell`オブジェクトの属性`value`でそのセルの値を取得できる。

```
cell = sheet.cell(1, 2)

print(cell)
# number:12.0

print(type(cell))
# <class 'xlrd.sheet.Cell'>

print(cell.value)
# 12.0
```

source: xlrd_example.py

`Sheet`オブジェクトのメソッド`cell_value()`でセルの値を直接取得することも可能。

```
print(sheet.cell_value(1, 2))
# 12.0
```

source: xlrd_example.py

`Sheet`オブジェクトのメソッド`col()`は列番号を指定してその列の`Cell`オブジェクトのリストを返す。

```
col = sheet.col(1)

print(col)
# [text:'A', number:11.0, number:21.0, number:31.0]

print(type(col[0]))
# <class 'xlrd.sheet.Cell'>
```

source: xlrd_example.py

`col_values()`でセルの値のリストを直接取得することも可能。

```
col_values = sheet.col_values(1)

print(col_values)
# ['A', 11.0, 21.0, 31.0]
```

source: xlrd_example.py

行番号を指定して行の`Cell`オブジェクトのリスト、または、セルの値のリストを取得する`row()`, `row_values()`メソッドもある。

```
print(sheet.row_values(1))
# ['one', 11.0, 12.0, 13.0]
```

source: xlrd_example.py

任意の範囲のセルの値を2次元配列として取得

リスト内包表記で各行の値を`row_values()`メソッドで取得すると、セルの値を2次元配列（リストのリスト）として取得できる。

- **関連記事:** Pythonリスト内包表記の使い方

```
pprint.pprint([sheet.row_values(x) for x in range(4)])
# [['', 'A', 'B', 'C'],
#  ['one', 11.0, 12.0, 13.0],
```

```
#  ['two', 21.0, 22.0, 23.0],
#  ['three', 31.0, 32.0, 33.0]]
```

source: xlrd_example.py

`row_values()` メソッドには取得する列の始まりと終わりを引数で指定できる。0始まりの行番号・列番号で範囲を指定して2次元配列（リストのリスト）として取得する関数は以下のように定義できる。

```
def get_list_2d(sheet, start_row, end_row, start_col, end_col):
    return [sheet.row_values(row, start_col, end_col + 1) for row in range(start_row, end_row +

l_2d = get_list_2d(sheet, 2, 3, 1, 2)
print(l_2d)
# [[21.0, 22.0], [31.0, 32.0]]
```

source: xlrd_example.py

ワークシートの行数は `Sheet` オブジェクトの属性 `nrows` で取得できるので、ワークシート全体の値を2次元配列（リストのリスト）として取得する関数は以下のように定義できる。

```
print(sheet.nrows)
# 4

def get_list_2d_all(sheet):
    return [sheet.row_values(row) for row in range(sheet.nrows)]

l_2d_all = get_list_2d_all(sheet)
pprint.pprint(l_2d_all)
# [['', 'A', 'B', 'C'],
#  ['one', 11.0, 12.0, 13.0],
#  ['two', 21.0, 22.0, 23.0],
#  ['three', 31.0, 32.0, 33.0]]

print(l_2d_all[1][0])
# one
```

source: xlrd_example.py

xlwtの使い方（xlsファイルの書き込み）

xlwtは `xls` ファイルの書き込みを行うライブラリ。 `xlsx` ファイルはサポートされていない。GitHubのレポジトリとドキュメントは以下のリンクから。

- python-excel/xlwt: Library to create spreadsheet files compatible with MS Excel 97/2000/XP/2003 XLS files, on any platform.
- xlwt documentation — xlwt 1.3.0 documentation

`Workbook`オブジェクトを作成し`add_sheet()`メソッドでワークシートを追加、`Worksheet`オブジェクトの`write()`メソッドでセルに値を入力していくイメージ。

`xlwt.Workbook()`で`Workbook`オブジェクトを作成。

```
import xlwt

wb = xlwt.Workbook()

print(type(wb))
# <class 'xlwt.Workbook.Workbook'>
```

source: xlwt_example.py

`add_sheet()`メソッドでワークシートを追加。`Worksheet`オブジェクトが返る。既存のシート名を指定するとエラーになるので注意。

```
sheet = wb.add_sheet('sheet1')

print(type(sheet))
# <class 'xlwt.Worksheet.Worksheet'>
```

source: xlwt_example.py

`Worksheet`オブジェクトの`write()`メソッドで値を入力。第一引数が行番号、第二引数が列番号、第三引数が入力する値。行番号、列番号は0始まり。

```
sheet.write(0, 0, 'A')
sheet.write(0, 1, 'B')
sheet.write(1, 0, 10)
sheet.write(1, 1, 20)
```

source: xlwt_example.py

既に入力した位置のセルを指定するとエラー。

```
# sheet.write(0, 0, 'A')
# Exception: Attempt to overwrite cell: sheetname='sheet1' rowx=0 colx=0
```

source: xlwt_example.py

`Worksheet`オブジェクトの`save()`メソッドにパスを指定するとファイルとして保存される。既存のファイルのパスを指定すると上書きされる（元のファイルのデータは削除される）ので注意。

```
wb.save('data/dst/xlwt_sample.xls')
```

source: xlwt_example.py

2次元配列を任意の位置に書き込み

以下のような関数を定義すると、2次元配列（リストのリスト）を書き込むことができる。引数`start_row`, `start_col`に2次元配列が書き込まれる行番号と列番号を0始まりで指定する。

```
sheet2 = wb.add_sheet('sheet2')

def write_list_1d(sheet, l, start_row, start_col):
    for i, val in enumerate(l):
        sheet.write(start_row, start_col + i, val)

def write_list_2d(sheet, l_2d, start_row, start_col):
    for i, l in enumerate(l_2d):
        write_list_1d(sheet, l, start_row + i, start_col)

l_2d = [['A', 'B', 'C'], [1, 2, 3]]
write_list_2d(sheet2, l_2d, 1, 2)

wb.save('data/dst/xlwt_sample.xls')
```