

Mainスクリプト

```

""" =====
                Main
=====
if __name__ == '__main__':

    """ Test-CTRL Sw -----
    SwTest01 = True           # very simple examples
    SwTest02 = False          # make sr.Rng() table
    SwTest03 = False          # E5A beam-line: min/max LET
    SwTest04 = False          # plot LET, Range, Straggling
    SwTest05 = False          # info srImWS, srImWB

    import pySRIMfit as srIm # load MySRIMwb.pickle
    sr = srIm.mySRIM()
    sr.info()

```

動作確認用スクリプト [t_pySRIMfit.py](#) をエディターで見てください。

- ① テスト・スクリプトが 5種類 (SwTest01～05) 用意してあります。
必要に応じて、これらの SwTest * = True にしてください。
勿論、全て = True でも OK です。
- ② 一番最初に、pySRIMfit モジュールをインポート します。
次に、 `sr = srим.mySRIM()` です。 ※1
その後は、 `sr.関数名()` の様に使えるようになります。 ※2
`sr.info()` で、ワークシートの一覧が表示されます。 ※3

※1: `sr.mySRIM()` で、
`pySRIMfit` 用のデータベース `MySRIMwb.pickle`
 が自動的に load されます。
 詳しくは、`mySRIMbase.py : class mySRIM()` 参照。

※2: **sr.関数名()** のヘルプ は、以下で表示できます。

```
> help(sr.info)
> dir(sr)
> help(sr)
```

関数名は、Excel版の SRIMfit に合わせてあります。

【注】pySRIMfit で使用している、
anaconda 3 標準以外のモジュール
 > conda install xlwings
 > pip install python-pptx
が予め必要かもしれません。

出力結果

[illegible]

- ③ ※3: `sr.info()` の出力
現在 load された `MySRIMwb.pickle` に含まれている
`Beam x Target` の組合せリストが表示されます。
○印が、該当あり ×印が、該当なし です。
データベースに過不足がある場合 は、
→ 巻末: 今まで使っていた Excel版 SRIMfit を、pySRIMfit で使いたい場合
を参照ください。

テスト・スクリプト

```

""" Test01 : HowToUse ; very simple examples
-----"""
if(SwTest01):
    print('==== Test-01 =====')
    wsn = 'srin40Ar_Si' # work sheet name
    print("sr.maxLETt('{')= {:.2f} [{}]; using sr.func()".format(
        wsn, sr.maxLETt(wsn), sr.LETunit(0) ) )
    srws = sr.getWS(wsn) # srinWS object
    print("ws.maxLETt()= {:.2f} ; using srinWS object".format(
        srws.maxLETt() ) )

```

出力結果

```

==== Test-01 =====
sr.maxLETt('srin40Ar_Si')= 18.67 [MeV/(mg/cm2)] ; using sr.func()
ws.maxLETt()= 18.67 ; using srinWS object

```

① class srinWB() を使った例 です。

最初に宣言した sr = srin.mySRIM() は、srinWB() クラスを返します。
 ですので、その後は、

sr.関数名(‘ワークシート名’, パラメータ)

で、WBクラスの method が使用できます。

WBクラスの関数一覧は、

> dir(sr) で表示されます。

② class srinWS() を使った例 です。

> srWS = sr.getWS(wsn)

で、WBクラスから、シート1枚:WSクラス を抽出します。

その後は、

srWS.関数名(パラメータ) ← ‘ワークシート名’ 不要

で、WSクラスの method が使用できます。

WSクラスの関数一覧は、

> dir(srWS) で表示されます。

Test01は、一番簡単な例題 です。

pySRIMfit には、2つの class があります。

class srinWB()

Excel版 SRIMfit のデータベースファイル

MySRIMwb.xlsx のシート1枚 に相当します。

class srinWS() ※1

MySRIMwb.xlsx のブック全体 に相当します。

※1: class srinWB() の attribution 定義 → see) srinWBbase.py

```

class srinWB( object ):
    """ class Attributes
    -----"""
    def _myclear(self):
        #print('@srinWB:Home dir=',os.path.dirname(__file__))
        self.WBpath = os.path.join(os.path.dirname(__file__), './_MySRIMwb')
        self.WBname = 'MySRIMwb.pickle' # defa. fname
        self.WSL = [] # [srWS]
        self.nWS = 0 # Len([srWS])
        self.WSnameL = [] # [srWS.wsName]
        self.WSnameLDB = [] # [dict{'Bm':*, 'Tg':*,...}]
        self.curWS = None # srWS
        self.curWSname = '' # srWS.wsName

    def __init__(self):
        """ Constructor
        -----"""
        self._myclear()

    def getWS(self, WSnm:str) -> object:
        """ Find srWS by WSname
        WSnm: str work-sheet name as 'srin'+Beam+'_'+Target
        <Ret> srinWS or None, if not found
        <Set> .curWS, .curWSname
        -----"""
        if(WSnm == self.curWSname):
            return( self.curWS )
        if(WSnm not in self.WSnameL):
            msg=("#@ getWS: WSnm=[{}] not found"
                ", then return value = float('NaN') or ''").format(WSnm)
            _Err(1,msg)
            return(None)
        ix = self.WSnameL.index(WSnm)
        self.curWS = self.WSL[ix]
        self.curWSname = WSnm
        return( self.curWS )

```

load されている WS名を確認するには、

> sr.WSnameL です。

srWB.getWS() は、WS名で sr.WSnameL[] を検索し、

srWB.curWS: 現在参照中の WS を切り替えます。

この方式は、Excel版 と同じ手法です。

テスト・スクリプト

```

""" Test02 : make sr.Rng() table
-----"""
if(SwTest02):
    print('==== Test-02 =====')
    ① wsnL=['sr12C_Si','sr20Ne_Si','sr40Ar_Si',
          'sr56Fe_Si','sr84Kr_Si']
        EL=[70,90,95,100,110,120,135,140,150,160]
        print('      Beam E =',end='')
        for E in EL:
            print('{:5.0f}'.format(E),end='')
        print(' [MeV/A]')
        for wsn in wsnL:
            print('{:<14}'.format(wsn),end='')
            for E in EL:
                ② print('{:5.1f}'.format(sr.Rng(wsn,E)/1000),end='')
            print(' [mm]')

```

出力結果

```

==== Test-02 =====
      Beam E =   70   90   95  100  110  120  135  140  150  160 [MeV/A]
sr12C_Si      :  7.6 11.8 13.0 14.2 16.8 19.6 24.1 25.7 28.9 32.4 [mm]
sr20Ne_Si     :  4.5  7.0  7.7  8.4 10.0 11.7 14.3 15.2 17.2 19.2 [mm]
sr40Ar_Si     :  2.8  4.3  4.7  5.1  6.0  7.0  8.6  9.1 10.2 11.5 [mm]
sr56Fe_Si     :  2.0  3.1  3.4  3.7  4.3  5.0  6.2  6.5  7.4  8.2 [mm]
sr84Kr_Si     :  1.6  2.5  2.7  3.0  3.5  4.0  4.9  5.2  5.8  6.5 [mm]

```

Test02は、Rangeテーブル表示 です。

- ① で、表示したい WS名と、Beam Energy のリストを一括定義しておいて、
- ② で、sr.Rng() を用いて、Range値を表示します。 ※1

※1: `sr12C_Si.Rng()` → `sr12C_Si.getWS().Rng()` calling sequence
 WB関数 `Rng(WSnm, E)` は、`getWS()` で `WSnm` を特定してから、
 WS関数 `Rng(E)` を呼び出し、
 WS 内の attribution `_E[]` `_Rng[]` ; Energy vs Range を参照し、
 指定した `E` 値での Range値を、
`numpy interp()` 関数で、内挿補間 して 返しています。

```

class sr12C_Si( object ):
    def Rng(self, WSnm:str, E:float) -> float:
        """ <Ret> Rng(E) [um] Range """
        ws = self.getWS(WSnm)
        return( float('NaN') if(ws is None) else ws.Rng(E) )

class sr12C_Si( object ):
    def Rng(self, E:float) -> float:
        """ <Ret> Rng(E) [um] Range """
        return( self._interpTbl('Rng',E, self._Rng) )

    def _interpTbl(self, fncH, E, Vtbl) -> float:
        """ Vtbl[] --> E : interpolation
        _E[] [MeV/A]
        Vtbl[] [um]
        <Ret> Vtbl[E] [um]
        -----"""
        if( E < self.minE ):
            v, ok = self.minE, False
        elif( E > self.maxE ):
            v, ok = self.maxE, False
        else:
            v, ok = np.interp(E, self._E, Vtbl), True
        if(not ok):
            msg = (self._msgH(fncH)+'Warn: E=[{:2e}] out of range.'
                  '(min,max)=[{:2e},{:2e}]'.format(E,self.minE,self.maxE)
                  _Err(1,msg)
            return( v )

```

pySRIMfit を使ってみる - (3) : SwTest03 = True の時

pyE5A: pySRIMfit

t_pySRIMfit.py

テスト・スクリプト

```

""" Test03 : E5A beam-line: min/max LET
-----"""
if(SwTest03):
    print('==== Test-03 =====')
    Beam, BeamE = '84Kr', 69.59 # [MeV/A]
    Pa, degC = 1010.*100., 23.7 # [Pa],[C] Air press. & temp.
    # Material thick [um or mm]
    Matr=[ ['Au', 45.8 ], # Beam Scattering foil
            ['Kapton', 51.2 ], # Vacuum Foil
            ['Al', 0.60], # IC1;Ion Chamber Alumi Foil
            ['Mylar', 14.54], # IC1 Mylar Foil
            ['Mylar', 10.2 ], # PL1;Plastic Scintillator Mylar Foil
            ['Al', 0.4 ], # PL1 Alumi on Mylar
            ['EJ212', 100.0 ], # PL1 Scinti (EJ-212)
            ['Air', 145.0 ], # [mm] Air1 gap; Kapton ~ EdegOut
            ['Air', 200.0 ] # [mm] Air2 gap; EdegOut ~ Sample
    ]
    # Sample Mold [um]
    Smp1=[ ['Al', 0.0 ], # mold#1
            ['Epoxy', 150.0 ], # mold#2
            ['Si', 5.0 ], # Device Layer#1
            ['Si', 10.0 ] # Device Layer#2
    ]
    Targ = Matr + Smp1
    E, wsnSi = BeamE, 'srin'+Beam+'_Si'
    print(' #, Matr , Thick, Eaftr LET, Range')
    print(' ,[um|mm], [MeV/A], , [um] ')
    print(' in Vacc ,{:6.2f}, {:6.2f}, {:5.2f}, {:6.1f}'.format(
        E, sr.LETt(wsnSi,E), sr.Rng(wsnSi,E)))
    for i,trg in enumerate(Targ):
        wsn = 'srin'+Beam+'_'+trg[0]
        if(sr.Gas(wsn)):
            if(sr.Gas(wsn)):
                E = sr.EnewGas(wsn,E,trg[1],Pa,degC)
            else:
                E = sr.Enew(wsn,E,trg[1])
            LET = sr.LETt(wsnSi,E)
            Rng = sr.Rng(wsnSi,E)
            msg = '{:2d}, {:<8}, {:6.2f}, {:6.2f}, {:5.2f}, {:6.1f}'.format(
                i,trg[0],trg[1],E,LET,Rng)
            print(msg)

```

Test03は、E5Aコース用の計算例 です。
例として、2301_Kr_技術報告書.pdf を計算してみます。

● Beam 通過物の厚さ

Beam	Kr用	
Mon系	μ m	
Au	45.8	φ 50
Kapton	51.2	
IC1.Al	0.60	
IC1.mylar	14.54	
PL.mylar	10.2	
PL.myAl	0.4	
PL.EJ212	100.0	
	mm	mm
Air1	145.0	Kap~Edeg出口
Air2	200.0	Edeg出口~試料
avr気温	23.7 °C	
avr気圧	1011.0 hPa	

ビームライン常設物	ref params	厚さ	
Au	45.8	45.8	μ m
Kapton	51.2	51.2	μ m
IC1.Al	0.60	0.60	μ m
IC1.mylar	14.54	14.54	μ m
PL.EJ212	100.0	100.0	μ m
PL.mylar	10.2	10.2	μ m
PL.Al(mylar)	0.4	0.4	μ m
Air1	145.0	145.0	mm
Air2	200.0	200.0	mm
AirT 気温	23.7	23.7	°C
AirP 気圧	1011.0	1011.0	hPa
ThkStd	0.9853		
Beam	84Kr	84Kr	A=84 Z=36
Ebm:加速器G測定値	69.590	69.59	ExpR:実測
δEbm補正 [%]	-0.660	0.000	/ExpR
Edeg出口まで			
	MeV/u	LET in Si	Rin Si μ m
in Vacc	69.59	9.51	1629
aft Au	63.296	10.16	1393
aft Kap	62.245	10.28	1356
aft IC1-Al	62.225	10.29	1355
aft IC1-Mylar	61.936	10.32	1345
aft PL-EJ212	60.370	10.50	1290
aft PL-Mylar	60.167	10.52	1283
aft PL-Al(Mylar)	60.154	10.52	1282
aft Air1	57.709	10.84	1202
照射物位置で			
aft Air2	54.259	11.30	1092

出力結果

```

===== Test-03 =====
# , Matr , Thick, Eaftr LET, Range
# ,[um|mm], [MeV/A], , [um]
in Vacc , 69.59, 9.51, 1629.0
0, Au , 45.80, 63.30, 10.16, 1393.1
1, Kapton , 51.20, 62.24, 10.28, 1356.0
2, Al , 0.60, 62.22, 10.29, 1355.3
3, Mylar , 14.54, 61.94, 10.32, 1345.1
4, Mylar , 10.20, 61.73, 10.34, 1338.0
5, Al , 0.40, 61.72, 10.34, 1337.5
6, EJ212 , 100.00, 60.15, 10.52, 1282.2
7, Air , 145.00, 57.71, 10.84, 1202.1
8, Air , 200.00, 54.26, 11.30, 1092.1
9, Al , 0.00, 54.26, 11.30, 1092.1
10, Epoxy , 150.00, 49.51, 12.05, 944.4
11, Si , 5.00, 49.34, 12.08, 939.4
12, Si , 10.00, 49.02, 12.13, 929.4

```

④ が、Excel版で計算した結果
⑤ が、python版で計算した結果
当然のことながら一致しています。

- ① に、技術報告書と同じパラメータを指定しておき、
- ② で、それぞれのビーム通過物を通過後の
E = sr.Enew() or sr.EnewGas() 気体の場合、
LET = sr.LETt(E)
Rng = sr.Rng(E) を計算してゆきます。

pySRIMfit を試してみる - (4) : SwTest04 = True の時

pyE5A: pySRIMfit

t_pySRIMfit.py

テスト・スクリプト

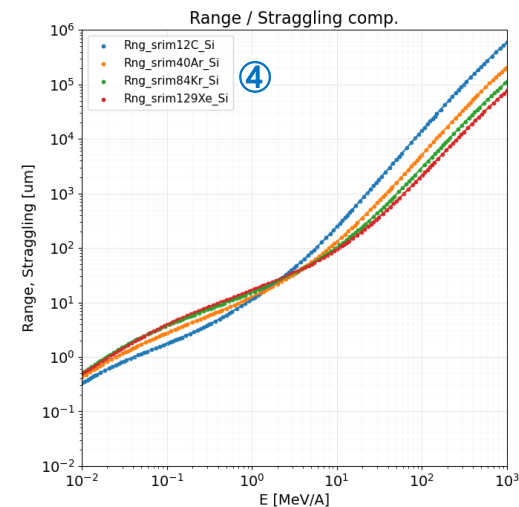
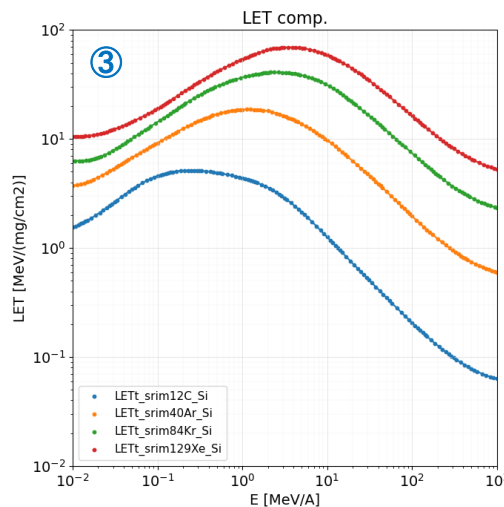
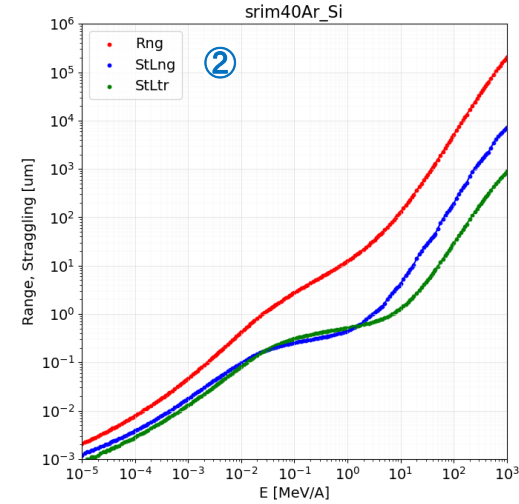
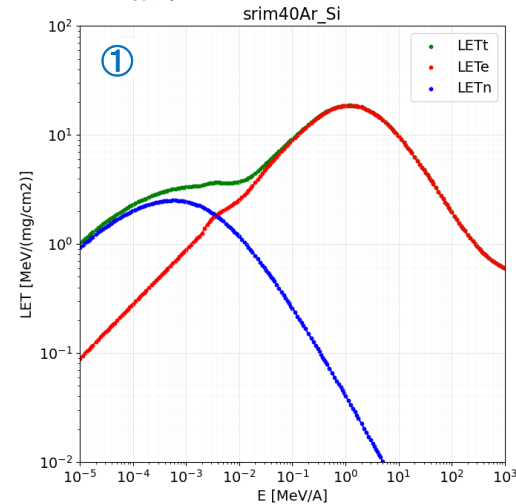
```
""" Test04 : plot LET, Range, Stragglng
----- """
if(SwTest04):
    print('==== Test-04 =====')
    wsn = 'srim40Ar_Si'
    ① sr.WSplot_LET(wsn, LETe=True, LETn=True, LETt=True,
        rngX=(1e-5, 1e3), rngY=(1e-2, 1e2))
    ② sr.WSplot_RngSt(wsn, Rng=True, StLng=True, StLtr=True,
        rngX=(1e-5, 1e3), rngY=(1e-3, 1e6))
    wsnL = ['srim12C_Si', 'srim40Ar_Si', 'srim84Kr_Si', 'srim129Xe_Si']
    ③ sr.WSplot_LETcmp(wsnL, LETe=False, LETn=False, LETt=True,
        rngX=(1e-2, 1e3), rngY=(1e-2, 1e2))
    ④ sr.WSplot_RngStcmp(wsnL, Rng=True, StLng=False, StLtr=False,
        rngX=(1e-2, 1e3), rngY=(1e-2, 1e6))
```

- ① `sr.WSplot_LET()` は、
指定WSの LETテーブルをプロットします。
Optin param の指定で、
LETnuclear, LETelectric, LETtotal の表示ON/OFF や
X軸 range, Y軸 range の調整が可能です。
- ② `sr.WSplot_RngSt()` は、
指定WSの Range, StLtr, StLng テーブルをプロットします。
- ③ `sr.WSplot_LETcmp()` と、
④ `sr.WSplot_RngStcmp()` は、
複数WSの テーブルを 比較プロットします。

WSテーブルの参照は、
Excel版では、いちいち MySRIMwb.xlsx を開いて見る
必要がありましたが、
Python 版では、object なので、表示が楽です。

Test04は、ワークシート内テーブルのプロット表示 です。

出力結果



テスト・スクリプト

```

""" Test05 : info srinWS, srinWB
-----"""
if(SwTest05):
    print('===== Test-05 =====')
    sr.info(mod=0) # srinWB simple info.
    sr.WBinfo() # srinWB info(Beam x Trg table)
    ① sr.WSinfo('srin84Kr_Si',modL=[-1]) # srinWS all info.
    wsnmL = sr.getWSnmL(Beam='Kr',swSort=1) # get WSnames
    ② sr.WBdump(wsnmL) # dump srWB selected wsnmL
    #sr.WBdump() # dump srWB all srWS to Power-Point file
    
```

② sr.WBdump() の出力結果 パワポ・ファイルに出力されます。

sr.WSinfo() と sr.WSplot() を用いて dump しています。



Test05は、srinWS や srinWB の情報表示 です。

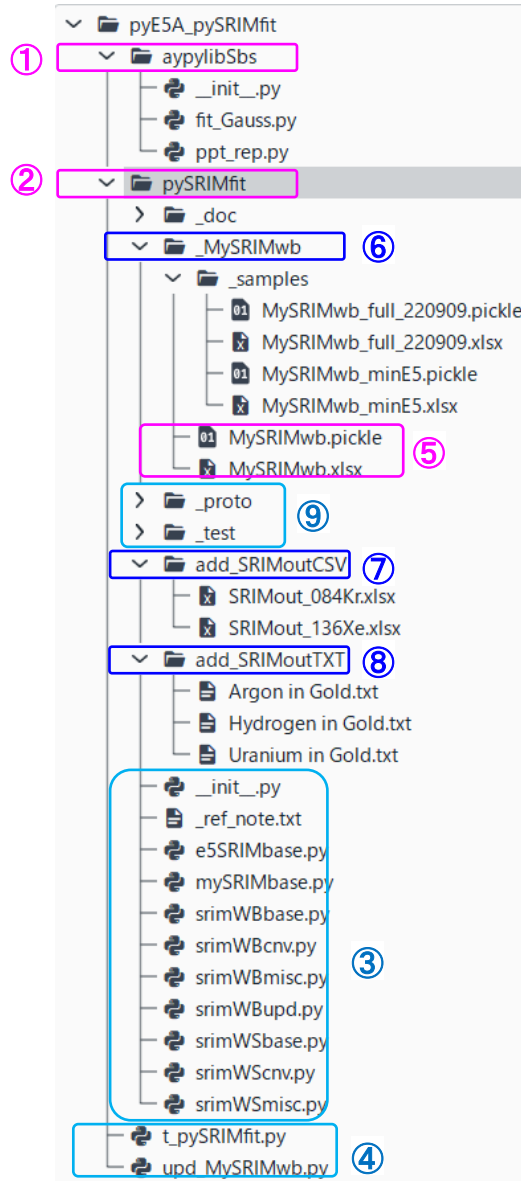
いろいろな情報を表示できます。
詳しくは、各関数の help() をご覧ください。

① sr.WSinfo() の使い方

この method で、WSの全ての dump が確認できます。

```

class srinWS( object ):
    def info(self, modL:list=[0], *,
             intrvE:int=1, show:bool=True ) -> list:
        """ Info. of this srinWS
        <optional> see) srinWSdump()
        modL: [int,...] dump mode list
        0: simple info. ; wsName,Beam,Target
        1: other info. ; TrgDens, BraggC, SRIMver, Corded
        2: target compo.; TrgCmp table
        3: conv. factor; LET unit conversion table
        4: table size ; {min|max} of E, LETe,n,t, Rng, StLng, StLtr
        5: table contents; E, LET{e|n|t}, Rng, StLng, StLtr
        6: table size ; LET{e|n|t}|L|H}
        -1: all of above
        <optional: kwd args>
        intrvE: int when mod=5, skip E interval
        show: bool show dump result as print()
        <Ret>
        nothing if show=True
        txL: [str,...] else dump result as list of text
        """
    
```



①、② の2つが **import** に必要なフォルダーです。

③ が、pySRIMfit module のスクリプト

④ は、pySRIMfit を用いた サンプルプログラム(前述) と ユーティリティ(後述)

⑤ が、自動 load される データベース **MySRIMwb.pickle** です。

Excel版 **MySRIMwb.xlsx** は、**.pickle** 版を生成する時に読み込んだ元ファイルです。

同フォルダー内の **_samples** フォルダーにあるのは、

MySRIMwb_full_220909 : 現行 SRIMfit HP から down load できる最新版(でもゴミ多い)

(*) **MySRIMwb_minE5** : E5Aコースに必要な最小セット(お掃除済版) です。

Default で ⑤に置いてあるのは、(*) です。必要に応じてファイル名を変更してお使いください。

名前	サイズ	def saveWS(self, path:str=None, fnm:str=None):
MySRIMwb_full_220909.pickle	7,133 KB	""" save srWB as .pickle ->r#101
MySRIMwb_full_220909.xlsx	18,743 KB	<optional>
MySRIMwb_minE5.pickle	4,075 KB	path: str load path (defa. self.WBpath)
MySRIMwb_minE5.xlsx	10,979 KB	fnm: str file name (defa. self.WBname)

```

"""
    _path = path if(path is not None) else self.WBpath
    _fnm = fnm if(fnm is not None) else self.WBname
    fn = os.path.join(_path,_fnm)
    with open(fn,'wb') as f:
        #pickle.dump(self.WSL,f,pickle.HIGHEST_PROTOCOL)
        pickle.dump(self.WSL,f,protocol=None)

```

.pickle は、**sr.WSL[]** に登録された **class srinWS()** object を **sr.saveWS()** method でシリアライズして保存したものです。**pickle.dump(protocol=None)** を使用しています。

そのファイルサイズは **.xlsx** に比べ約半分以下になります。

※ 但し、**class srinWS()** の構造を「将来変更する」と、**.pickle** の互換性が無くなります。

よって、元ファイル **.xlsx** も一緒に残しておいてください。

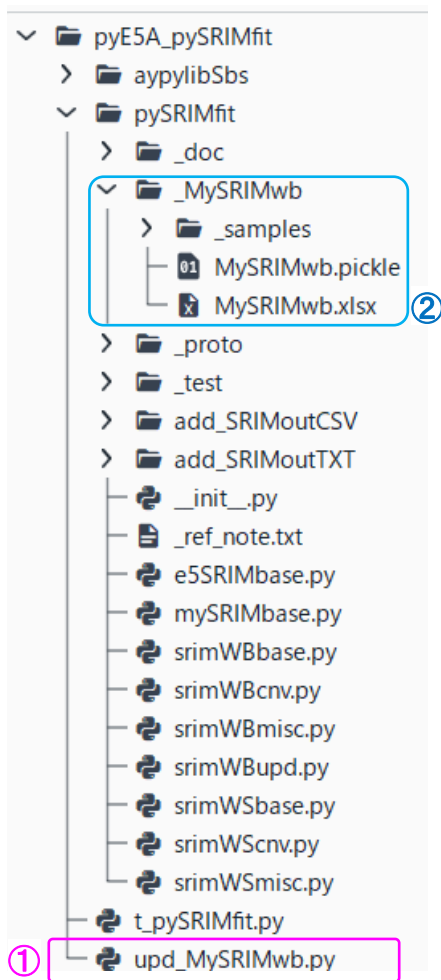
⑥、⑦、⑧ は、後述の ユーティリティ (**upd_MySRIMwb.py**) で使用します。

⑨ は、module 開発・デバッグ用のファイルです。

今まで使っていた Excel版 SRIMfit を、pySRIMfit で使いたい場合

pyE5A: pySRIMfit

upd_MySRIMwb.py



① upd_MySRIMwb.py を使います。

1: 元: MySRIMwb.xlsx を、予め

② ~¥pySRIMfit¥_MySRIMwb に 上書きコピー しておく
同時に ~¥pySRIMfit¥_MySRIMwb¥MySRIMwb.pickle を 削除 しておく

2: upd_MySRIMwb.py を実行する

3: Menu= 5) Save Book を選択

4: 2) Save ~tmp.xlsx as ~new.pickle & ~new.xlsx を選択。変換が始まります。

5: Menu= 9) exit で終了してから、

6: 変換出力された ファイルの名前変更: .pickleファイルのみでOKです
MySRIMwb_new.pickle --> MySRIMwb.pickle

7: 名前変更した .pickle ファイルを、~¥pySRIMfit¥_MySRIMwb に保存
以上で、元: MySRIMwb.xlsx と MySRIMwb.pickle が同期されます。

① upd_MySRIMwb.py の 他メニュー を使うと、

- ・ 内容表示: シート名一覧
- ・ SRIM-2013 で計算したファイルの追加
- ・ シートの削除、追加、並べ替え
- ・ 編集結果の保存

等の編集作業もできます。

詳しくは → upd_MySRIMwb.py のコメント文を参照。

取敢えず、ここまで。 今後の展望

現状の pySRIMfit には、Excel版の様な 主だったアプリ は、まだ用意されていませんが、Pythonist の皆さんなら、このベースモジュールで、様々な応用が考えられると思います。何か良いアプリができたなら、あ吉田にも紹介してください。

開発の本来の目的は、ビーム調整時に取得した SSDテレスコープの $\angle E-E$ スペクトルの解析用なのですが(これは鋭意やってみます)、折角ここまで出来たので、他にも使い道がいろいろありそうですね。

(あ吉田の開発予定)

- 当チーム HP に、html から python を呼び (pyscript かなあ〜) Web上 で SRIM 計算ができるようにしたい。
いちいち Excel や python を起動するのが面倒なので。
例えば、Test02 や Test03 などが、Web上で計算できると便利ですよね。
- `srimWB()` の method を全面的に改造して、argument と return に多態性を持たせたい。
例) `.Enew(wsnm:str, E:float, T:thick) -> float :` → `.Enew(wsnm:str, E:float or list, T:thick) -> float or list :`
- そうしてから、現状の Excel版アプリ を、VBAマクロ でなく xlwings 経由の pySRIMfit 呼び出しに改造してみる。
list 対応にしないと、xlwings 経由の overhead が重くて、Excel版より遅いため。
MS-Excelも、早く native で python対応 になってくれるとありがたいのですが…。
- TRIM の python版 pyTRIMfit も作れるかなあ〜。
誰か欲しいですか？ せめてグラフィック程度は、matplotlib で綺麗になりそう。

などなど。暇を見て作っておきます。

では、今後ともよろしくお願い申し上げます。