I: PySOC and related programs preparation:

1. Use the compiled binary file and Python script:
   unpack the distribution file:
   tar –zxvf ***pysoc.tar.gz***
   In pysoc, there are five subdirectories: *src, bin, input_template, examples, parameters* should be found
   check pysoc/bin subdirectory for ***pysoc.py, soc.py*** and ***soc_td***
   check pysoc/input_template subdirectory for ***init.py***

2. If you want to install PySOC by yourself:
   You will need:
   - The pysoc source code.
   - A FORTRAN 90 compiler installed, preferentially the Intel Fortran compiler.
   - Python 2.7 series installed

   Go to the subdirectory pysoc/src and uncomment or add own fortran compiler, flags, and linker in the *Makefile*;
   In the linux command line use *make* to compile the pysoc source code. The *soc_td* executable will be created in the current directory;

3. Modified MolSOC code
   MolSOC is originally developed by Dr. Sandro Chiodo. Here it was used to calculate the spin-orbit coupling integral on atomic basis, and was modified and adapted to PySOC interface and dftb+ fitted basis sets. The original and modified version is distributed together with PySOC source code with filename: ***molsoc.tar.gz***. The compiled binary file *molsoc0.1.exe* is stored in subdirectory *molsoc_modified/molsoc0.1/bin.*
4. Fitted dftb+ orbital parameters
   In td-dftb+, STOs are used. They are fitted to GTOs and could be recognized by Modified MolSOC. The parameters corresponding to **mio-1-1** (which should be downloaded through http://www.dftb.org/parameters/download/) in dftb can be found in the subdirectory pysoc/parameters/mio-1-1-fit within PySOC source code.
5. Third party quantum chemistry code
   Gaussian 09
   TD-DFTB+(available from Prof. Thomas Niehaus) and related tools subdirectory like tools/dptools

II: PySOC quick start:

1. Excited states electronic structure calculation:
   - ■ Gaussian 09

     Prepare *.com file for Gaussian input with the following suggested settings:

     %rwf=gaussian.rwf

     # td(50-50,nstates=5) wB97XD/TZVP 6D 10F nosymm GFInput

   The keywords, 50-50, nstates=5, mean 5 singlets and 5 triplest will be calculated.

   Note:

   a) gaussian.rwf, 6D, 10F, GFInput key words are necessary.

   b) when setting basis, check the max layer for each element which should be less or equal than f shell.(higher level like g shell is not available in the following SOC calculation now)

   - ■ TD-DFTB+(strongly suggest read the manual before use it)

     a) Prepare geometry file from .xyz file:

        xyz2gen *.xyz to *.gen

     Note: xyz2gen is a kind of geometry generation tool from dptools for td-dftb+ (see above) and should be installed and work.

     b) Prepare *dftb_in.hsd* for td-dftb+ input:

        Besides general settings the following key words should be added in the input.

        set parameters: HubbardDerivatives for related elements.

        set WriteTransitions = Yes

        set WriteTransitionDipole = Yes

        set WriteEigenvectors = Yes

        set WriteXplusY = Yes

        set WriteHS = No

     c) run tddftb+ calculation

     d) run tddftb+ once again(this step should be very fast, because it just read the parameterized matrix elements) in the same directory again with the following changes:

        set WriteHS = Yes

2. run PySOC to do spin-orbit coupling calculation:

   a) copy the *init.py* file from pysoc/input_template to the working directory which should include the successful outputs from the above electronic structure calculation

   b) edit *init.py* file to control the SOC calculation

   - ✓ set QM_code = 'gauss_tddft' or 'tddftb' corresponding to the above quantum chemistry code.
   - ✓ set # of excited singlets for SOC by n_s = [1, 2, 3,,,]. Of cause, the max number should be less than that in electronic structure calculation.

     NOTE: [1,2,3,4,5]=range(1,6), [1, 2, 3, 4, 5, 9, 10]=range(1,6)+[9,10]
   - ✓ set # of excited singlets for SOC by n_t = [1, 2, 3,,,]. Again, the max

number should be less than that in electronic structure calculation.

- ✓ do SOC between ground state(defauld singlet) and triplets or not by applying n_g = ['True'] or n_g = ['false']
- ✓ set MolSOC binary file path by molsoc_path like molsoc_path = '/fsnfs/users/bin/molsoc/molsoc0.1.exe'
- ✓ On case of Gaussian, set environmental variable for Gaussian 09 by g09root like g09root = '/usr/users'
- ✓ On case of td-dftb+:
  modify geom_xyz = ['*.xyz'], *.xyz is the geometry filename for your dftb+ calculation.
  set parameters directory for td-dftb+ which should be found in directory pysoc/parameters:
  dir_para_basis = '/fsnfs/users/ pysoc/parameters/mio-1-1'

c) Set the path location to *soc.py* and *soc_td* in *pysoc.py* by scrip_soc like scrip_soc ='/fsnfs/users/xinggao/work/gsh/thiothymine/gtsh/test_python/soc_tb/bin'. Include the system environmental path to *pysoc.py* in shell initialization file (.cshrc, .bashrc…)

d) run *pysoc.py* in working directory

e) check output file
The SOC elements should be found in output file: *soc_out.dat*, looks like:
sum_soc, <S0|Hso|T1,1,0,-1> (cm-1): 115.94270 81.98387 0.01644 81.98387
There are four numbers each line corresponding respectively to root sum square of the subshell number, the module length of the subshell with quantum num 1, 0, -1. The unit is cm^{-1}. For e.g. <S0|Hso|T1,1,0,-1> means SOC between ground state and first triplet with quantum number 1, 0, -1.