This is to carefully document my Exact Denationalization python code
I have three files for this purpose. ED_ necessary_functions.py , Input_for_ED.py and bonds_and_coord.py

ED_necessary_functions.py

produce_states(nsite, nup, ndn, U):
gives the allowed binary strings from the 4^N available states (4 because emp up dn or double)
allowed only if even ones = nup and odd ones = ndn and (no of double = 0 if U = ∞)

fix_bonds(bonds):
takes the bonds list starting from site 1 to site N and gives bake the bonds list starting from site 0 to site 2N-1 in the basis 0up 0dn 1up 1dn 2up 2dn .....

swap_bits(num,N,n,m,U):
takes a state (num) and any 2 sites n m and checks if the particle can hop and gives back the new state with the correct fermi sign if possible

CdagC(num,N,n,m,U):
very similar to swap_bits
I use it to make operators

Input_for_ED.py (main file)

Provide the parameters of the lattice (Nx, Ny, Nup, Ndn, U = (zero or inf))

From bonds_and_coord.py lat_bonds gives the bonds for a **triangular lattice** of **rectangular geometry** and **cylindrical** boundary conditions

fix_bonds fixes the bonds to work in the basis 0up 0dn 1up 1dn 2up 2dn .....

produce_states gives the physical states allowed in the form of binary strings depending on Nsite, Nup, Ndn and U

Then make the many-body Hubbard Hamiltonian matrix with dimensions (states x states)
PS: It is only the hopping term either free(U=0) or restricted(U=∞).

swap_bits is used to hop a particle if allowed and keep track of the fermi sign to change the Hamiltonian matrix accordingly

Now the easy part: solve the Hamiltonian

Measuring observables on the ground state:

To measure $C^\dagger_{i\uparrow}C_{j\uparrow}$ ($C^\dagger_{i\downarrow}C_{j\downarrow}$) we loop over even (odd) numbers and use the function CdagC (similar to swab_bits) to create the operator

Find the expectation value $\langle GS|C^\dagger_{i\sigma}C_{j\sigma}|GS\rangle$