```python
import numpy as np
import pycubelib.plotting_functions as pf
import pycubelib.general_functions as gf

pi = np.pi
pi2 = pi * 2


def renormalize(aa, alimit, blimit, btype=None):

    atype = type(aa)
    aa = aa.astype(float)
    if alimit:
        aa = np.clip(aa, alimit[0], alimit[1])
    else:
        alimit = (np.min(aa), np.max(aa))
    amin, amax = alimit
    bmin, bmax = blimit
    bb = bmin + (bmax - bmin) * (aa - amin) / (amax - amin)
    if btype:
        bb = bb.astype(btype)
    else:
        bb = bb.astype(atype)

    return bb


def get_roi(aa, roi):
    rx0, ry0, rx, ry = roi
    return aa


def stitch(zz12n_in, zz1n, lam12, lam1n):
    yy = np.round(zz12n_in / lam1n) * lam1n
    zz = yy + zz1n
    dzz = zz - zz12n_in
    ezz = (np.abs(dzz) > (0.5 * lam1n)) * lam1n * np.sign(dzz)
    zz12n_out = np.mod(zz - ezz + lam12/2, lam12) - lam12/2

    if 1:
        iy = 900
        ylimit = (-1.5 * lam12/2, 1.5 * lam12/2)
        graphs = []
        graphs += [(zz12n_in[iy, :], (0, 0), 'zz12n_in', ylimit)]
        graphs += [(zz1n[iy, :], (0, 1), 'zz1n', ylimit)]
        graphs += [(yy[iy, :], (0, 2), 'yy', ylimit)]
        graphs += [(zz[iy, :], (0, 3), 'zz', ylimit)]
        graphs += [(dzz[iy, :], (0, 4), 'dzz', ylimit)]
        graphs += [(ezz[iy, :], (0, 5), 'ezz', ylimit)]
        graphs += [(zz12n_out[iy, :], (0, 6), 'zz12n_out', ylimit)]
        pf.graph_many(graphs, 'stitch', (1, 7), sxy=(.25, .25), pause=1)

    return zz12n_out


def calib_lam1n(zz12n_in, zz1n, lam12, lam1n, roi):
    nbin = 100
    dlam = lam12 / nbin
    ix, iy, rx, ry = roi
    ep1n = zz1n * pi2 / lam1n
    lam12limit = (-lam12/2, lam12/2)
```

```
    print(f'>>> lam1n_in = {lam1n:.1f}')
    while True:
        ans = float(input('> lam1n = '))
        if ans:
            lam1n = ans
        else:
            break

        zz1n_ = ep1n * lam1n / pi2
        dzz = zz12n_in - zz1n_
        dzz1n = np.mod(dzz + lam1n/2, lam1n) - lam1n/2
        # pf.plotAAB(dzz, capA=f'dzz: lam1n = {lam1n:.1f}', sxy=(.35, .35), pause=1)

        # histo, uu = np.histogram(dzz, bins=nbin, range=(-lam12/2, lam12/2))
        histo, uu = np.histogram(dzz1n, bins=nbin, range=(-lam1n/2, lam1n/2))

        graphs = [(zz12n_in[iy, :], (0, 0), f'zz12n: lam12 = {lam12:.1f}', lam12limit),
                  (zz1n_[iy, :], (0, 1), f'zz1n_: lam1n = {lam1n:.1f}', lam12limit),
                  (dzz[iy, :], (0, 2), f'dzz', lam12limit),
                  (dzz1n[iy, :], (0, 3), f'dzz1n', (-lam1n/2, lam1n/2))]
        pf.graph_many(graphs, 'calib', col_row=(1, 4), sxy=(.35, .3), pause=1)
        # pf.graphB(histo, caption='histogram', xpars=(-lam12/2, lam12/nbin), sxy=(7.5, .3),
line='-+', pause=1)
        pf.graphB(histo, caption='histogram', xpars=(-lam1n/2, lam1n/nbin), sxy=(7.5, .3),
line='-+', pause=1)

        zz_12n = stitch(zz12n_in, zz1n, lam12, lam1n)
        pf.plotAAB(zz_12n, capA=f'ZZ12n: lam_1n = {lam1n:.1f}', roi=roi, sxy=(.35, .35),
pause=1)

        # xx = np.arange(nbin) * lam12 / nbin - lam12/2
        # pf.plt.figure('histogram')
        # pf.plt.plot(xx, histo, '-+')
        # pf.plt.pause(1)

    return lam1n


def stitch_x(zz12n_in, zz1n, lam12, lam1n):
    yy = np.round(zz12n_in / lam1n) * lam1n
    zz = np.mod(yy + zz1n + lam12/2, lam12) - lam12/2
    dzz = np.mod(zz - zz12n_in + lam12/2, lam12) - lam12/2
    ezz = (np.abs(dzz) > (0.5 * lam1n)) * lam1n * np.sign(dzz)
    zz12n_out = np.mod(zz - ezz + lam12/2, lam12) - lam12/2
    return zz12n_out


if __name__ == '__main__':

    pass
```