Data Selection Process

1. Calculate height of every range gate, “beam_height”, in metres above sea level

```
beam_height[(360*j)+i,:] = radh/1000 + np.sin(np.deg2rad(el[j]))*rg + np.sqrt(rg**2 +
    (6371*4/3.0)**2) - (6371*4/3.0);
```

2. Extract melting layer information from ZDR scans

```
mlh, zdr_bias = extract_ml_zdr(time, ml_zdr)
```

3. Where the beam is above the melting layer, set values to nan

```
zind = beam_height > mlh
zdr[zind==True] = np.nan
```

4. Remove first 2km of data because PhiDP can be quite noisy (previously I had this set to first 3 range gates only, and then the second method set it to NaN)

```
n=13
uzh = uzh[:,n:]
zdr = zdr[:,n:]
```

5. Create a list of azimuths and elevations that you want to exclude from the processing e.g. due to known areas of blockage.

```
exclusions = [((0,90.1),(20,160)),((0,90.1),(201,207)),((0,0.51),(185,201.5))]
exclude_radials = np.any([np.all([rad.elevation['data']>=ele[0],
                                  rad.elevation['data']<ele[1],
                                  rad.azimuth['data']>=azi[0],
                                  rad.azimuth['data']<azi[1]],axis=0) for ele, azi in
exclusions],axis=0)
az_index = np.where(~exclude_radials)[0]
```

6. Loop through the rays that aren’t excluded, extracting each ray in turn

```
for i in az_index:
   uzh_f = uzh[i,:]
```

7. Adjust ZDR for the bias found from the vertical scans (retrieved in step 2)

```
zdr_f = zdr[i,:] - zdr_bias
```

8. Reset phidp ray to start at 0

```
phidp_att = phidp_f - phidp_f[0]
```

9. Apply an attenuation correction (can also be run without a correction)

```
PA = np.maximum.accumulate(phidp_att)
uzh_att = uzh_f + 0.28*PA
zdr_att = zdr_f + 0.04*PA
```

10. Find index of minimum value of PhiDP

```
phi1_valid = (np.nonzero(np.isfinite(phidp_att)))[0]
if phi1_valid.size != 0:
phi1 = phi1_valid[0]
```

11. Find indices where PhiDP has increased by between 4 and 6 degrees.

```
ind = np.where(np.logical_and(phidp_att > 4, phidp_att < 6))[0]
```

12. Find the index, ***ib***, of the maximum value of PhiDP between 4 and 6

```
ib = np.where(phidp_f==max(phidp_f[ind]))[0][0]
```

13. Assign the maximum value of PhiDP at the end of the path

```
pdpmax = phidp_att[ib]
```

14. Calculate the length of the path and exclude paths less than 15km

```
path_len = rg[ib]
```

15. Discard rays with any ZDR > ZDRmax (2dB)

```
ind = zdr_att > ZDRmax
```

16. Discard rays where 3 or more values of RhoHV are less than 0.98

```
ind = np.logical_and(rhohv_f > 0.0, rhohv_f < 0.98)
```

If we get as far as this point, then we have a good ray!

We take the observed value of PhiDP at the end of the ray (at ib) and compare it with the calculated value estimated from ZDR and Z. Any offset is an indication of the bias in Z.

phiobs = pdpmax
phiest = f(Z, ZDR)
bias = (phiest - phiobs) / phiobs

SCRIPT AND OUTPUT SUMMARY

STEP 1

- RUN ***process_volume_scans_time_series.py***
  - CALLS ***process_volume_scans_day.py***
    - CALLS ***calibrate_day_att*** in ***calib_functions.py***

Produces the following files:
phiobs_all_att_YYYYMMDD.npy - observed phidp
phiest_all_att_YYYYMMDD.npy - calculated phidp
startphi_all_YYYYMMDD.npy - calculated values for initial phidp at the start of each ray (used to track how it changes over time)

STEP 2
Use a python notebook or plot_Zcalib.py to plot the results

Calculating initial phidp of each ray

In the *calibrate_day_att* function there is additional code for calculating and saving the starting phidp in each of the selected good rays.

The code takes the first 20 values (after the initial 2km has been removed) and calculates the median value.

startphi[i] = np.nanmedian(uphidp_f[phi1:phi1+10])

These values are saved for each file in the same way phiobs and phiest are saved.

A python script is used to plot up the results
plot_initial_phase.py