In [ ]:
```
from Bio import PDB as bio
import numpy as np
import pymol_helper as ph
import math
from scipy.spatial.distance import pdist, squareform
from collections import defaultdict
```

In [ ]:
```
protein = bio.MMCIFParser().get_structure('7kkk', '7kkk.cif')
atoms_list = list(protein.get_atoms())
```

```
/opt/miniconda3/envs/pymol/lib/python3.10/site-packages/Bio/PDB/StructureBui
lder.py:100: PDBConstructionWarning: WARNING: Chain A is discontinuous at li
ne 52224.
  warnings.warn(
/opt/miniconda3/envs/pymol/lib/python3.10/site-packages/Bio/PDB/StructureBui
lder.py:100: PDBConstructionWarning: WARNING: Chain C is discontinuous at li
ne 52504.
  warnings.warn(
/opt/miniconda3/envs/pymol/lib/python3.10/site-packages/Bio/PDB/StructureBui
lder.py:100: PDBConstructionWarning: WARNING: Chain E is discontinuous at li
ne 52784.
  warnings.warn(
```

We want to get all the SG atom coordinates subsetting to those within a distance threshold but I also want to know the adjacent carbon atoms.

In [6]:
```
# the following code looks up the cys and methionine residues and gets the g
# then pairs it with the beta carbon to keep as pairs to calculate dihedral
res_lookup = defaultdict(dict)
for atom in atoms_list:
    res = atom.get_parent()  # parent is the Residue
    resname = res.get_resname()
    resid = res.get_id()[1]  # tuple like (' ', 45, ' '), so take the residu
    atom_name = atom.get_name()

    res_lookup[(resname, resid)][atom_name] = atom

sulfur_pairs = []
for (resname, resid), atomdict in res_lookup.items():
    if resname == "CYS" and "SG" in atomdict and "CB" in atomdict:
        sulfur_pairs.append({
            "resname": resname,
            "resid": resid,
            "s_atom": atomdict["SG"],
            "c_atom": atomdict["CB"]
        })
```

```
for p in sulfur_pairs:
    print(f"{p['resname']} {p['resid']}: {p['s_atom'].get_name()} ↔ {p['c_at
```

```
CYS 131: SG ↔ CB
CYS 136: SG ↔ CB
CYS 166: SG ↔ CB
CYS 291: SG ↔ CB
CYS 301: SG ↔ CB
CYS 336: SG ↔ CB
CYS 361: SG ↔ CB
CYS 379: SG ↔ CB
CYS 391: SG ↔ CB
CYS 432: SG ↔ CB
CYS 480: SG ↔ CB
CYS 488: SG ↔ CB
CYS 525: SG ↔ CB
CYS 538: SG ↔ CB
CYS 590: SG ↔ CB
CYS 617: SG ↔ CB
CYS 649: SG ↔ CB
CYS 662: SG ↔ CB
CYS 671: SG ↔ CB
CYS 738: SG ↔ CB
CYS 743: SG ↔ CB
CYS 749: SG ↔ CB
CYS 760: SG ↔ CB
CYS 1032: SG ↔ CB
CYS 1043: SG ↔ CB
CYS 1082: SG ↔ CB
CYS 1126: SG ↔ CB
CYS 22: SG ↔ CB
CYS 96: SG ↔ CB
```

In [ ]:
```
coordsList = []
for a in sulfur_pairs:
    cys_id = a['resid']
    s = a['s_atom']
    s_coord = s.get_coord()
    coordsList.append(s_coord)

coords = np.vstack([np.asarray(a, dtype=float) for a in coordsList])
```

In [ ]:
```
# this codeblock takes in the coordinates of gamma sulfur atoms and calculat
# if the distance is within a threshold, then it keeps it as a unique pair a
threshold = 2.1

dist_condensed = pdist(coords)
dist_matrix = squareform(dist_condensed)

idx, jdx = np.triu_indices_from(dist_matrix, k = 1)
mask = dist_matrix[idx, jdx] <= threshold

results = []
for i, j, d in zip(idx[mask], jdx[mask], dist_matrix[idx[mask], jdx[mask]]):
    results.append({
        "atom1_index": i,
```

```
        "atom1_type": (sulfur_pairs[i])['s_atom'],
        "atom1_cb": sulfur_pairs[i]['c_atom'],
        "atom2_index": j,
        "atom2_type": sulfur_pairs[j]['s_atom'],
        "distance": d,
        "atom2_cb": sulfur_pairs[j]['c_atom']
        })


for r in results:
          print(f"{r['atom1_type']}({r['atom1_index']}) - {r['atom2_type']}(
```

```
<Atom SG>(0) - <Atom SG>(2) : 2.08 Å
<Atom SG>(3) - <Atom SG>(4) : 2.02 Å
<Atom SG>(5) - <Atom SG>(6) : 2.03 Å
<Atom SG>(7) - <Atom SG>(9) : 2.04 Å
<Atom SG>(8) - <Atom SG>(12) : 2.07 Å
<Atom SG>(10) - <Atom SG>(11) : 2.03 Å
<Atom SG>(13) - <Atom SG>(14) : 2.03 Å
<Atom SG>(15) - <Atom SG>(16) : 2.03 Å
<Atom SG>(17) - <Atom SG>(18) : 2.04 Å
<Atom SG>(19) - <Atom SG>(22) : 2.02 Å
<Atom SG>(20) - <Atom SG>(21) : 2.07 Å
<Atom SG>(23) - <Atom SG>(24) : 2.02 Å
<Atom SG>(25) - <Atom SG>(26) : 2.02 Å
```

In [ ]:

```
# using dihedral angle function provided, returns dihedral angle for each pa
dihedral_angles = []
for r in results:
    p0 = np.array(r['atom1_type'].get_coord())
    p1 = np.array(r['atom1_cb'].get_coord())
    p2 = np.array(r['atom2_type'].get_coord())
    p3 =np.array(r['atom2_cb'].get_coord())
    d_angle= abs(ph.new_dihedral(np.array([p0, p1, p2, p3])))
    dihedral_angles.append(d_angle)
    print(d_angle)
```

```
115.59984
103.36704
120.913414
88.907776
74.74362
92.2515
87.198586
78.32191
101.3381
70.27162
117.8208
101.73399
31.033663
```