Load the BINANA module

In [1]:
```
import sys
sys.path.append("../")
import binana
```

Load optional visualization module for this demo

In [2]:
```
import py3Dmol
```

Load in a receptor and ligand

In [3]:
```
ligand, receptor = binana.load_ligand_receptor.from_files("ligand.pdbqt", "
```

In [4]:
```
ligand, receptor
```

Out[4]:
```
(<binana._structure.mol.Mol at 0x7ff3706b0908>,
 <binana._structure.mol.Mol at 0x7ff3706b0a90>)
```

Get information about the hydrogen bonds (example)

In [5]:
```
hbond_inf = binana.interactions.get_hydrogen_bonds(ligand, receptor)
```

In [6]:
```
# Counting/characterizing the acceptors and donors (counts)
hbond_inf["counts"]
```

Out[6]:
```
{'HDONOR_LIGAND_SIDECHAIN_OTHER': 2, 'HDONOR_RECEPTOR_SIDECHAIN_OTHER':
1}
```

In [7]:
```
# List the atoms involved in each hydrogen bond
for hbond_label in hbond_inf["labels"]:
    print(hbond_label)
```

```
('A:CHT(1):N1(14)', 'A:CHT(1):H1(16)', 'A:ASP(157):OD2(285)', 'LIGAND')
('A:CHT(1):O6(22)', 'A:ASN(156):2HD2(276)', 'A:ASN(156):ND2(274)', 'RECEP
TOR')
('A:CHT(1):O6(22)', 'A:CHT(1):HO6(23)', 'A:ASP(157):OD1(284)', 'LIGAND')
```

Get information about the cation-pi interactions (example)

In [8]:
```
cation_pi_inf = binana.interactions.get_cation_pi(ligand, receptor)
```

In [9]:
```
# Counting/characterizing the acceptors and donors (counts)
cation_pi_inf["counts"]
```

Out[9]:
```
{'PI-CATION_LIGAND-CHARGED_OTHER': 2,
 'PI-CATION_LIGAND-CHARGED_BETA': 2,
 'PI-CATION_RECEPTOR-CHARGED_OTHER': 1}
```

In [10]:
```
# List the atoms involved in each cation-pi interaction
for cation_pi_label in cation_pi_inf["labels"]:
    print(cation_pi_label)
```

```
('[A:CHT(1):N1(2) / A:CHT(1):C5(1) / A:CHT(1):C6(3) / A:CHT(1):C6(4) / A:
CHT(1):C7(9)]', '[A:TRP(43):CG(28) / A:TRP(43):CD1(29) / A:TRP(43):NE1(3
1) / A:TRP(43):CE2(32) / A:TRP(43):CD2(30)]')
('[A:CHT(1):N1(2) / A:CHT(1):C5(1) / A:CHT(1):C6(3) / A:CHT(1):C6(4) / A:
CHT(1):C7(9)]', '[A:TRP(43):CE2(32) / A:TRP(43):CD2(30) / A:TRP(43):CE3(3
3) / A:TRP(43):CZ3(35) / A:TRP(43):CH2(36) / A:TRP(43):CZ2(34)]')
('[A:CHT(1):N1(2) / A:CHT(1):C5(1) / A:CHT(1):C6(3) / A:CHT(1):C6(4) / A:
CHT(1):C7(9)]', '[A:TRP(205):CG(468) / A:TRP(205):CD1(469) / A:TRP(205):N
E1(471) / A:TRP(205):CE2(472) / A:TRP(205):CD2(470)]')
('[A:CHT(1):N1(2) / A:CHT(1):C5(1) / A:CHT(1):C6(3) / A:CHT(1):C6(4) / A:
CHT(1):C7(9)]', '[A:TRP(205):CE2(472) / A:TRP(205):CD2(470) / A:TRP(205):
CE3(473) / A:TRP(205):CZ3(475) / A:TRP(205):CH2(476) / A:TRP(205):CZ2(47
4)]')
('[A:CHT(1):C2(17) / A:CHT(1):O1(18) / A:CHT(1):C5(19) / A:CHT(1):C4(20)
/ A:CHT(1):C3(21)]', '[A:LYS(94):NZ(144) / A:LYS(94):HZ1(146) / A:LYS(9
4):HZ2(147) / A:LYS(94):HZ3(148)]')
```

Other interactions are also available

In [11]:
```
print("Available functions for detecting interactions:")
for f in dir(binana.interactions):
    if "get_" in f:
        print("    " + f)
```

```
Available functions for detecting interactions:
    get_active_site_flexibility
    get_all_interactions
    get_cation_pi
    get_close
    get_closest
    get_electrostatic_energies
    get_hydrogen_bonds
    get_hydrophobics
    get_ligand_atom_types
    get_pi_pi
    get_salt_bridges
```

Get and display PDB-formatted text

In [12]:
```
pdb_txt = binana.output.pdb_file.write(
    ligand, receptor,
    hydrogen_bonds=hbond_inf,
    cat_pi=cation_pi_inf,
    as_str=True
)
print(
    "\n".join(
        [
            l
            for l
            in pdb_txt.split("\n")
            if l.startswith("REMARK")
        ]
    )
)
```

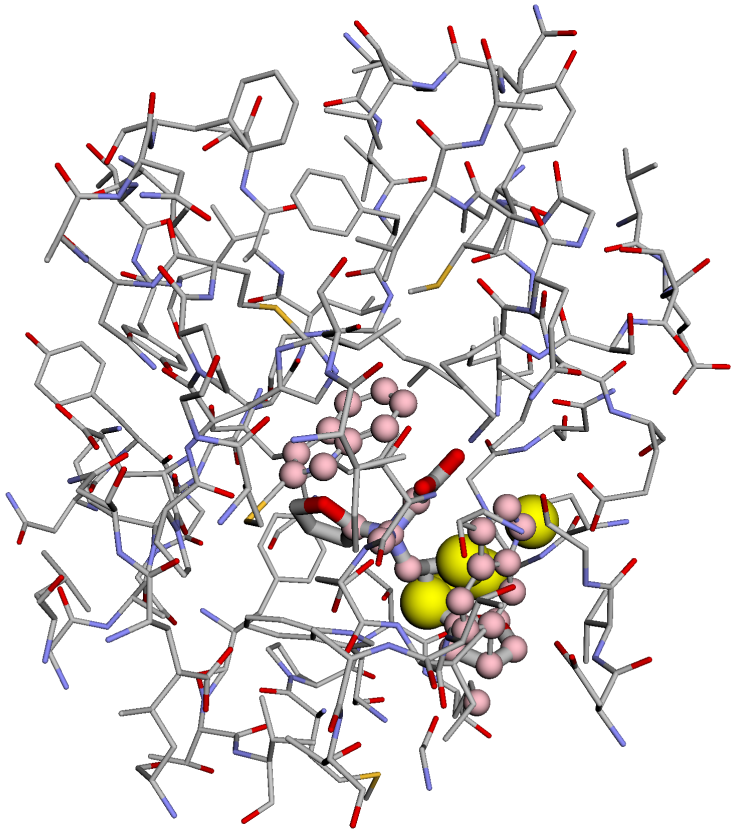
```
REMARK
REMARK The residue named "CCN" contains the closest contacts between the
REMARK protein and receptor. "CON" indicates close contacts. "ALP", "BE
T", and
REMARK "OTH" indicate receptor contacts whose respective protein residues
have
REMARK the alpha-helix, beta-sheet, or "other" secondary structure. "BAC"
and
REMARK "SID" indicate receptor contacts that are part of the protein back
bone
REMARK and sidechain, respectively. "HYD" indicates hydrophobic contacts
REMARK between the protein and ligand. "HBN" indicates hydrogen bonds. "S
AL"
REMARK indicates salt bridges. "PIS" indicates pi-pi stacking interaction
s,
REMARK "PIT" indicates T-stacking interactions, and "PIC" indicates catio
n-pi
REMARK interactions. Protein residue names are unchanged, but the ligand
REMARK residue is now named "LIG".
REMARK
```

In [13]:
```
view = py3Dmol.view(data="",linked=False)

view.addModel(pdb_txt)

view.setStyle({'stick':{'radius':0.1}})
view.setStyle({"resn": "LIG"}, {'stick':{'radius':0.3}})
view.setStyle({"resn": "HBN"}, {'sphere':{'radius':1, "color": "yellow"}})
view.setStyle({"resn": "PIC"}, {'sphere':{'radius':0.5, "color": "pink"}})

view.zoomTo()
```



Out[13]:
```
<py3Dmol.view at 0x7ff33020a978>
```

Get the interactions as a dictionary for easier big-data analysis

In [14]:
```
data = binana.output.dictionary.collect(
    hydrogen_bonds=hbond_inf,
    cat_pi=cation_pi_inf
)

print("Keys:")
print(list(data.keys()))

print("")
print("Hydrogen-bond data (example):")
print(data["hydrogenBonds"])
```

```
Keys:
['hydrogenBonds', 'cationPiInteractions']

Hydrogen-bond data (example):
[{'ligandAtoms': [{'chain': 'A', 'resID': 1, 'resName': 'CHT', 'atomNam
e': 'N1', 'atomIndex': 14}, {'chain': 'A', 'resID': 1, 'resName': 'CHT',
'atomName': 'H1', 'atomIndex': 16}], 'receptorAtoms': [{'chain': 'A', 're
sID': 157, 'resName': 'ASP', 'atomName': 'OD2', 'atomIndex': 285}]}, {'li
gandAtoms': [{'chain': 'A', 'resID': 1, 'resName': 'CHT', 'atomName': 'O
6', 'atomIndex': 22}], 'receptorAtoms': [{'chain': 'A', 'resID': 156, 're
sName': 'ASN', 'atomName': 'ND2', 'atomIndex': 274}, {'chain': 'A', 'resI
D': 156, 'resName': 'ASN', 'atomName': '2HD2', 'atomIndex': 276}]}, {'lig
andAtoms': [{'chain': 'A', 'resID': 1, 'resName': 'CHT', 'atomName': 'O
6', 'atomIndex': 22}, {'chain': 'A', 'resID': 1, 'resName': 'CHT', 'atomN
ame': 'HO6', 'atomIndex': 23}], 'receptorAtoms': [{'chain': 'A', 'resID':
157, 'resName': 'ASP', 'atomName': 'OD1', 'atomIndex': 284}]}]
```

Get all the interactions at once

In [15]:
```
all_inf = binana.interactions.get_all_interactions(ligand, receptor)
```

In [16]:
```
all_inf.keys()
```

Out[16]:
```
dict_keys(['closest', 'close', 'electrostatic_energies', 'active_site_fle
xibility', 'hydrophobics', 'hydrogen_bonds', 'ligand_atom_types', 'pi_p
i', 'cat_pi', 'salt_bridges', 'ligand_rotatable_bonds'])
```

Get and display PDB-formatted text containing all interactions

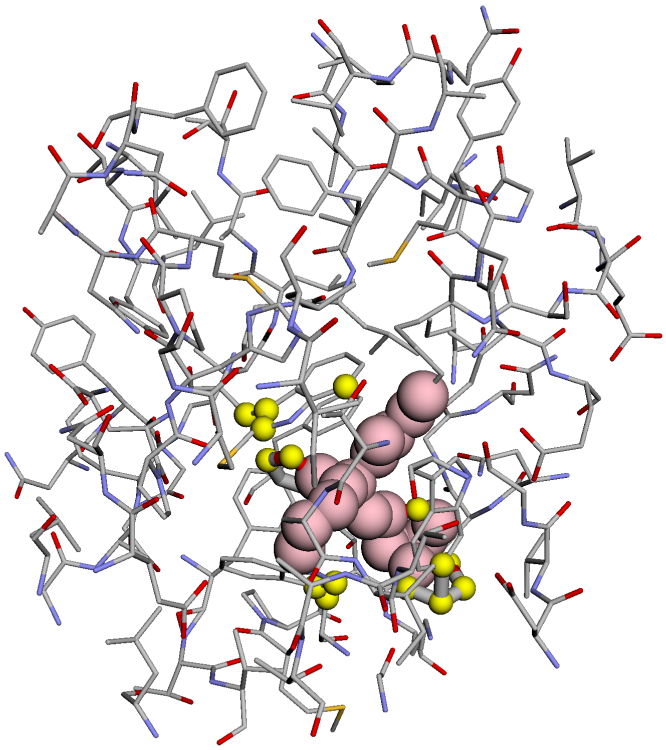
In [17]:
```
pdb_txt = binana.output.pdb_file.write_all(
    ligand, receptor,
    all_inf,
    None,
    as_str=True
)
```

In [18]:
```
view = py3Dmol.view(data="",linked=False)

view.addModel(pdb_txt)

view.setStyle({'stick':{'radius':0.1}})
view.setStyle({"resn": "LIG"}, {'stick':{'radius':0.3}})
view.setStyle({"resn": "CCN"}, {'sphere':{'radius':0.5, "color": "yellow"}}
view.setStyle({"resn": "SAL"}, {'sphere':{'radius':1, "color": "pink"}})

view.zoomTo()
```



Out[18]:
```
<py3Dmol.view at 0x7ff37077c9b0>
```

Get all interactions as a single dictionary

In [19]:
```
all_data = binana.output.dictionary.collect_all(all_inf)

print(all_data.keys())
print()
print("Hydrogen bonds (example):")
print()
print(all_data["hydrogenBonds"])
```

```
dict_keys(['closestContacts', 'closeContacts', 'hydrophobicContacts', 'hy
drogenBonds', 'piPiStackingInteractions', 'tStackingInteractions', 'catio
nPiInteractions', 'saltBridges', 'activeSiteFlexibility', 'electrostaticE
nergies', 'ligandAtomTypes', 'ligandRotatableBonds'])

Hydrogen bonds (example):

[{'ligandAtoms': [{'chain': 'A', 'resID': 1, 'resName': 'LIG', 'atomNam
e': 'N1', 'atomIndex': 14}, {'chain': 'A', 'resID': 1, 'resName': 'LIG',
'atomName': 'H1', 'atomIndex': 16}], 'receptorAtoms': [{'chain': 'A', 're
sID': 157, 'resName': 'ASP', 'atomName': 'OD2', 'atomIndex': 285}]}, {'li
gandAtoms': [{'chain': 'A', 'resID': 1, 'resName': 'LIG', 'atomName': 'O
6', 'atomIndex': 22}], 'receptorAtoms': [{'chain': 'A', 'resID': 156, 're
sName': 'ASN', 'atomName': 'ND2', 'atomIndex': 274}, {'chain': 'A', 'resI
D': 156, 'resName': 'ASN', 'atomName': '2HD2', 'atomIndex': 276}]}, {'lig
andAtoms': [{'chain': 'A', 'resID': 1, 'resName': 'LIG', 'atomName': 'O
6', 'atomIndex': 22}, {'chain': 'A', 'resID': 1, 'resName': 'LIG', 'atomN
ame': 'HO6', 'atomIndex': 23}], 'receptorAtoms': [{'chain': 'A', 'resID':
157, 'resName': 'ASP', 'atomName': 'OD1', 'atomIndex': 284}]}]
```

In [ ]: