```
#!/usr/bin/python
__author__ = "morganlnance"

"""
HW2 Question 4 (Workshop #2 Exercise 4)

This program calculates the propensities of the 20
amino acids in either a helix, sheet, or loop. Phi
and psi values defining these secondary structures
are set values taken from an NMR website. The program
prints the propensities to the screen.

Usage:   ./ss_propensities.py <.pdb file>
Example: ./ss_propensities.py 1m40.pdb
"""


###########
# IMPORTS #
###########
import sys
from pyrosetta import init, \
    pose_from_file


#############
# ARGUMENTS #
#############
try:
    user_input = sys.argv[1]
except IndexError:
    print "\nPlease give me a .pdb file.\n"
    sys.exit()


###############
# AMINO ACIDS #
###############
# one-letter name to three-letter name
aa_dict = { "A": "ALA", "R": "ARG", "N": "ASN", "D": "ASP", "C":
"CYS",
            "Q": "GLN", "E": "GLU", "G": "GLY", "H": "HIS", "I":
"ILE",
            "L": "LEU", "K": "LYS", "M": "MET", "F": "PHE", "P":
"PRO",
            "S": "SER", "T": "THR", "W": "TRP", "Y": "TYR", "V":
"VAL" }

# ugly dictionary instantiation
# residue name to count of helix/sheet/loop in pose
```

```
aa_helix_propensities = { "ALA": 0, "ARG": 0, "ASN": 0, "ASP": 0,
"CYS": 0,
                          "GLN": 0, "GLU": 0, "GLY": 0, "HIS": 0,
"ILE": 0,
                          "LEU": 0, "LYS": 0, "MET": 0, "PHE": 0,
"PRO": 0,
                          "SER": 0, "THR": 0, "TRP": 0, "TYR": 0,
"VAL": 0 }
aa_sheet_propensities = { "ALA": 0, "ARG": 0, "ASN": 0, "ASP": 0,
"CYS": 0,
                          "GLN": 0, "GLU": 0, "GLY": 0, "HIS": 0,
"ILE": 0,
                          "LEU": 0, "LYS": 0, "MET": 0, "PHE": 0,
"PRO": 0,
                          "SER": 0, "THR": 0, "TRP": 0, "TYR": 0,
"VAL": 0 }
aa_loop_propensities = { "ALA": 0, "ARG": 0, "ASN": 0, "ASP": 0,
"CYS": 0,
                          "GLN": 0, "GLU": 0, "GLY": 0, "HIS": 0,
"ILE": 0,
                          "LEU": 0, "LYS": 0, "MET": 0, "PHE": 0,
"PRO": 0,
                          "SER": 0, "THR": 0, "TRP": 0, "TYR": 0,
"VAL": 0 }


###############
# PHI AND PSI #
###############
# http://nmr.chem.uu.nl/~adrien/course/molmod/analysis2.html
# phi and psi for helix
helix_phi_min = -140
helix_phi_max = -30
helix_psi_min = -80
helix_psi_max = 50

# phi and psi for sheet
sheet_phi_min = -180
sheet_phi_max = -40
sheet_psi_min = 50
sheet_psi_max = 180


########
# MAIN #
########
# initialize pyrosetta and the pose
init("-mute all -ignore_unrecognized_res")
pose = pose_from_file(user_input)
```

```
###########################
# GET SECONDARY STRUCTURE #
###########################
sec_struct = ""
for ii in range(1, pose.size()+1):
    # if it's an amino acid
    if pose.residue(ii).is_protein():
        # HELIX
        # phi and psi range for helices
        if pose.phi(ii) < helix_phi_max \
                and pose.phi(ii) > helix_phi_min \
                and pose.psi(ii) < helix_psi_max \
                and pose.psi(ii) > helix_psi_min:
            sec_struct += "H"
            # increase counter in helix dictionary
            aa_helix_propensities[aa_dict[
                    pose.residue(ii).name1()]] += 1
        # phi and psi range for sheets
        elif pose.phi(ii) < sheet_phi_max \
                and pose.phi(ii) > sheet_phi_min \
                and pose.psi(ii) < sheet_psi_max \
                and pose.psi(ii) > sheet_psi_min:
            sec_struct += "E"
            # increase counter in sheet dictionary
            aa_sheet_propensities[aa_dict[
                    pose.residue(ii).name1()]] += 1
        # if not a helix or sheet, it's a loop
        else:
            sec_struct += "L"
            # increase counter in loop dictionary
            aa_loop_propensities[aa_dict[
                    pose.residue(ii).name1()]] += 1


################
# PROPENSITIES #
################
# number of residues that are in a helix, sheet, or loop
num_helix = float(sum(aa_helix_propensities.values()))
num_sheet = float(sum(aa_sheet_propensities.values()))
num_loop = float(sum(aa_loop_propensities.values()))
print "     HELIX\tSHEET\tLOOP"
for aa in aa_dict.itervalues():
    # helix
    helix_propensity = round(float(aa_helix_propensities[aa]) /
num_helix, 3)
    # sheet
    sheet_propensity = round(float(aa_sheet_propensities[aa]) /
num_sheet, 3)
```

```
    # loop
    loop_propensity = round(float(aa_loop_propensities[aa]) /
num_loop, 3)
    # print to screen
    print "%s: %s\t%s\t%s" %(aa, helix_propensity,
                             sheet_propensity, loop_propensity)
```