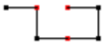
```
"""
Algorithms in computational Biology (INFO-F438)
Assignment 1 : Optimal Protein Folding in the HP Model
Aim : Gives the score of the best folding of a protein composed of hydrophobic (H)
and polar (P) amino acids residues. The best score maximizes the number of adjacent
but not covalently bounds hydrophobic amino acids residues
"""

__author__ = "Charlotte Nachtegael"
__date__ = "1 mars 2016"

import turtle


def dico_positions_H(protein):
    """
    :param protein: string with the residues of the protein
    :return: dictionary with the position of the H residues
    """
    dico = {}
    for i, character in enumerate(protein):
        if character == 'H':
            dico[i] = 0
    return dico


def fold(protein, dico_positions_H, length_to_fold, coordinates, best_score, best_coordinates):
    """
    :param protein: string with the residues of the protein
    :param dico_positions_H: dictionary with the position of the H residues
    :param length_to_fold: length of the protein that we still have to fold, used to keep track
of
    the progress
    :param coordinates: list with the coordinates of the residues already placed in the grid
    :param best_score: best score obtained when calling the function
    :param best_coordinates: list with the coordinates of the residues corresponding to the best
     score obtained when calling the function
    :return: the best score of the folding maximizing the number of hydrogen bonds, with the
     corresponding list of coordinates (the coordinates provided could not be the only one, as
     several other foldings could have the same best score)
    """

    # partial folding
    if length_to_fold > 0:
        next_positions = possible_positions(coordinates)
        # checking all the possible positions
        for position in next_positions:
            coordinates.append(position)
            optimistic_score = score(protein, coordinates, dico_positions_H)
            # pursue only if the node could lead to a score above the current best score
            if optimistic_score > best_score:
                best_score, best_coordinates = fold(protein, dico_positions_H, length_to_fold -
                                                    1, coordinates,
                                                    best_score, best_coordinates)
            # set the coordinates to the previous states to test the next possibility
            coordinates.remove(position)


    # folding completed
    else:
        final_score = score(protein, coordinates, dico_positions_H)
        if final_score > best_score:
            best_score = final_score
            best_coordinates = coordinates.copy()
    return best_score, best_coordinates
```

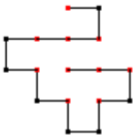
```python
def possible_positions(coordinates):
    """
    :param coordinates: list with the coordinates of the residues already placed in the grid
    :return: list with the possible positions for the next residue
    """
    positions = []
    previous_position_x, previous_position_y = coordinates[-1]
    if (previous_position_x - 1, previous_position_y) not in coordinates:
        positions.append((previous_position_x - 1, previous_position_y))
    if (previous_position_x + 1, previous_position_y) not in coordinates:
        positions.append((previous_position_x + 1, previous_position_y))
    if (previous_position_x, previous_position_y - 1) not in coordinates:
        positions.append((previous_position_x, previous_position_y - 1))
    if (previous_position_x, previous_position_y + 1) not in coordinates:
        positions.append((previous_position_x, previous_position_y + 1))

    return positions


def score(protein, coordinates, dico_positions_H):
    """
    :param protein: string of the residues of the protein
    :param coordinates: list with the coordinates of the residues already placed in the grid
    :param dico_positions_H: dictionary with the position of the H residues
    :return: the optimal (if partial folding) or the final (if complete folding) of the folding,
    based on the number of adjacent but not covalent H residues
    """
    length_fold = len(coordinates)
    length_prot = len(protein)
    number_bonds_for_H = dico_positions_H.copy() # to keep track of the number of bonds per H
    score = 0
    i = 0

    # If folding contains more than 3 residues, some hydrogen bonds can be already formed
    while length_fold > 3 and i < length_fold:
        if protein[i] == 'H':
            x_first_H, y_first_H = coordinates[i]

            # H bonds can only be formed between an H in an even position and an H in an odd
            # position (or vice versa).
            for j in range(i + 3, length_fold, 2):
                if protein[j] == 'H':
                    x_second_H, y_second_H = coordinates[j]
                    if abs(x_first_H - x_second_H) + abs(y_first_H - y_second_H) == 1:
                        score += 1
                        number_bonds_for_H[i] += 1
                        number_bonds_for_H[j] += 1
        i += 1




    # Calculus of the optimal score
    i = 0
    if length_fold < length_prot:
        while i < length_prot:
            if protein[i] == 'H':

                # H residues at extremities can have 3 hydrogen bonds, other H residues 2 maximum
                if (i == 0 and number_bonds_for_H[i] < 3) or \
                        (0 < i < length_prot - 1 and number_bonds_for_H[i] < 2):
```

```
                    # look only to the residues that are not yet placed in the grid
                    for j in range(i + 3, length_prot, 2):
                        if j > length_fold - 1 and protein[j] == 'H':
                            if j == length_prot - 1 and number_bonds_for_H[j] < 3:
                                score += 1
                                number_bonds_for_H[i] += 1
                                number_bonds_for_H[j] += 1
                            elif j != length_prot - 1 and number_bonds_for_H[j] < 2:
                                score += 1
                                number_bonds_for_H[i] += 1
                                number_bonds_for_H[j] += 1
            i += 1

    return score


def draw(protein, score, coordinates, t):
    """
    :param protein: string with the residues of the protein
    :param score: best score obtained for the folding of the protein
    :param coordinates: list of the coordinates corresponding to one of the best folding
    :param t: turtle
    :return: draw the folding and give the score
    """
    list_x = []
    list_y = []

    # increase the size of the drawing
    for i in range(len(coordinates)):
        x,y = coordinates[i]
        list_x.append(x*20)
        list_y.append(y*20)

    for i in range(len(coordinates)):
        t.setposition(list_x[i],list_y[i])
        if protein[i] == 'H':
            t.dot(4, 'red')
        else:
            t.dot(4, 'black')

    t.penup()
    t.ht()
    t.setpos(0,-150)
    t.write("For the protein" + protein, False, align="center",font=("Arial", 20, "normal"))
    t.write("Your score is " + str(score), False, align="center",font=("Arial", 20, "normal"))
    t.exitonclick()



if __name__ == '__main__':
    protein = str(input('Please enter your protein : '))
    # the first two points are always placed this way, so we study after the first two residues
    # use of a list for the coordinates because we need the data in order
    start = [(0,0), (1,0)]
    length_to_study = len(protein) - 2
    dico = dico_positions_H(protein)

    best_score, best_coordinates = fold(protein, dico, length_to_study, start, 0, start)
    draw(protein, best_score, best_coordinates, turtle)
```

For the protein HHHHHHH
Your score is 2



For the protein HPPHPHP
Your score is 2



For the protein HHHPHPPHPHPPHHHPH
Your score is 8