```python
# Residue.py
# Functions to manipulate the 3D turtle relative to atoms in Residues.
# Based on the original
# work in the program Proteus.
# This implementation utilises the Vector class from the Biopython module:
# BIO.PDB
# Author: Eric G. Suchanek PhD
# Last Modification: 11/22/2022
#

import numpy  # type: ignore

from Bio.PDB import Vector

from proteusPy.turtle3D import Turtle3D

def build_residue(turtle: Turtle3D):
    '''
    build residue requires the turtle to be in orientation #2
    (at Ca, headed to C, with N on left), and returns coordinates
    for n, ca, cb, and c.

    NOTE: Position of Oxygen depends on psi, which may not be know
    Returns: <Vector> n, ca, cb, c
    '''

    assert turtle._orientation == 2, f'build_residue() requires Turtle3D to be in orientation #2'

    # canonical internal coordinates for backbone atoms with Turtle3D at Ca, heading towards C with N on the left
    # AKA Orientation #2
    # we set these as arrays since that's what the Turtle3D expects
    _n = numpy.array((-0.486, 1.366, 0.0), "d")
    _ca = numpy.array((0, 0, 0), "d")
    _cb = numpy.array((-0.523, -0.719, -1.245), "d")
    _c = numpy.array((1.53, 0.0, 0.0), "d")

    n = Vector(turtle.to_global(_n))
    ca = Vector(turtle.to_global(_ca))
    cb = Vector(turtle.to_global(_cb))
    c = Vector(turtle.to_global(_c))
    return n, ca, cb, c

def get_backbone_from_chain(chain, resnumb):
    '''
    Retrieve the backbone atom positions (N, Ca, C, O) for the given chain and residue number.
```



```python

    Arguments:
        chain: list of Residues in the model, eg: chain = model['A']
        resnumb: residue number
    Returns: <Vector> n, ca, c, o atomic coordinates
    '''
    residue = chain[resnumb]

    assert residue is not None, f'get_backbone_from_sidechain() requires valid residue number'

    # proximal residue
    n = residue['N'].get_vector()
    ca = residue['CA'].get_vector()
    c = residue['C'].get_vector()
    o = residue['O'].get_vector()

    return n, ca, c, o


def to_alpha(turtle: Turtle3D, phi):
    """
    Moves the Turtle3D from backbone nitrogen to alpha carbon.  Turtle
    begins at nitrogen, headed towards alpha carbon,
    with carbonyl carbon of previous residue on left side and ends in
    orientation #2 (at alpha carbon, headed towards carbonyl carbon, with
    nitrogen on left side).

    Arguments: turtle, the Turtle3D in correct orientation
               phi: backbone dihedral angle
    Returns: Position of the modeled Ca. <Vector>
    """
    turtle.move(1.45)
    turtle.roll(phi)
    turtle.yaw(110.0)
    return(Vector(turtle.getPosition()))

def to_carbonyl(turtle: Turtle3D, psi):
    """
    Moves turtle from alpha carbon to carbonyl carbon. Turtle begins in
    orientation #2 (at alpha carbon, headed towards carbonyl carbon, with
    nitrogen on left) and ends at carbonyl carbon, headed towards nitrogen of
    next residue, with alpha carbon of current residue on left side.

    Arguments: turtle, the Turtle3D in correct orientation
               psi: backbone dihedral angle
    Returns: Position of the modeled C atom. <Vector>
```

```
    """

    turtle.move(1.53)
    turtle.roll(psi)
    turtle.yaw(114.0)
    return(Vector(turtle.getPosition()))

def to_nitrogen(turtle: Turtle3D, omega):
    """
    Turtle begins at carbonyl carbon, headed towards nitrogen of
    second residue, with alpha carbon of first residue on left side.
    Turtle ends at nitrogen of second residue, headed towards alpha carbon
    of second residue, with carbonyl carbon of first residue on left side.
    Omega will almost always be +180 degrees for trans peptide bonds.

    Arguments: turtle, the Turtle3D in correct orientation
               omega: backbone dihedral angle (peptide bond angle)
    Returns: Position of the modeled C atom. <Vector>
    """

    turtle.move(1.32)
    turtle.roll(omega)
    turtle.yaw(123.0)
    return(Vector(turtle.getPosition()))

def add_oxygen(turtle: Turtle3D):
    '''
    Returns the position of the carbonyl oxygen assuming the Turtle3D
    begins at carbonyl carbon, headed towards nitrogen of
    second residue, with alpha carbon of first residue on left side.

    Arguments: turtle, the Turtle3D in correct orientation

    Returns: <Vector>Position of the modeled C atom.
    '''
    loc = numpy.array((-0.673, -1.029, 0), "d")

    return Vector(turtle.to_global(loc))

def to_oxygen(turtle: Turtle3D):
    '''
    Returns the position of the carbonyl oxygen assuming the Turtle3D
    begins at carbonyl carbon, headed towards nitrogen of
    second residue, with alpha carbon of first residue on left side.

    Arguments: turtle, the Turtle3D in correct orientation

    Returns: <Vector>Position of the modeled C atom.
```



```
    '''

    loc = numpy.array((-0.673, -1.029, 0.0), "d")

    # update the Turtle3D object
    turtle.to_global(loc)
    return

# End of file
```