

```
1 # Residue.py
2 # Functions to manipulate the 3D turtle relative to atoms in Residues.
... Based on the original
3 # work in the program Proteus.
4 # This implementation utilises the Vector class from the Biopython module:
... BIO.PDB
5 # Author: Eric G. Suchanek PhD
6 # Last Modification: 11/22/2022
7 #
8
9 import numpy # type: ignore
10
11 from Bio.PDB import Vector
12
13 from proteusPy.turtle3D import Turtle3D
14
15 def build_residue(turtle: Turtle3D):
16     """
17     build residue requires the turtle to be in orientation #2
18     (at Ca, headed to C, with N on left), and returns coordinates
19     for n, ca, cb, and c.
20
21     NOTE: Position of Oxygen depends on psi, which may not be know
22     Returns: <Vector> n, ca, cb, c
23     """
24
25     assert turtle._orientation == 2, f'build_residue() requires Turtle3D to
... be in orientation #2'
26
27     # canonical internal coordinates for backbone atoms with Turtle3D at
... Ca, heading towards C with N on the left
28     # AKA Orientation #2
29     # we set these as arrays since that's what the Turtle3D expects
30     _n = numpy.array((-0.486, 1.366, 0.0), "d")
31     _ca = numpy.array((0, 0, 0), "d")
32     _cb = numpy.array((-0.523, -0.719, -1.245), "d")
33     _c = numpy.array((1.53, 0.0, 0.0), "d")
34
35     n = Vector(turtle.to_global(_n))
36     ca = Vector(turtle.to_global(_ca))
37     cb = Vector(turtle.to_global(_cb))
38     c = Vector(turtle.to_global(_c))
39     return n, ca, cb, c
40
41 def get_backbone_from_chain(chain, resnumb):
42     """
43     Retrieve the backbone atom positions (N, Ca, C, O) for the given chain
... and residue number.
```

```
44
45 Arguments:
46     chain: list of Residues in the model, eg: chain = model['A']
47     resnumb: residue number
48 Returns: <Vector> n, ca, c, o atomic coordinates
49     """
50     residue = chain[resnumb]
51
52     assert residue is not None, f'get_backbone_from_sidechain() requires
... valid residue number'
53
54     # proximal residue
55     n = residue['N'].get_vector()
56     ca = residue['CA'].get_vector()
57     c = residue['C'].get_vector()
58     o = residue['O'].get_vector()
59
60     return n, ca, c, o
61
62
63 def to_alpha(turtle: Turtle3D, phi):
64     """
65     Moves the Turtle3D from backbone nitrogen to alpha carbon. Turtle
66     begins at nitrogen, headed towards alpha carbon,
67     with carbonyl carbon of previous residue on left side and ends in
68     orientation #2 (at alpha carbon, headed towards carbonyl carbon, with
69     nitrogen on left side).
70
71     Arguments: turtle, the Turtle3D in correct orientation
72                phi: backbone dihedral angle
73     Returns: Position of the modeled Ca. <Vector>
74     """
75     turtle.move(1.45)
76     turtle.roll(phi)
77     turtle.yaw(110.0)
78     return(Vector(turtle.getPosition()))
79
80 def to_carbonyl(turtle: Turtle3D, psi):
81     """
82     Moves turtle from alpha carbon to carbonyl carbon. Turtle begins in
83     orientation #2 (at alpha carbon, headed towards carbonyl carbon, with
84     nitrogen on left) and ends at carbonyl carbon, headed towards nitrogen
... of
85     next residue, with alpha carbon of current residue on left side.
86
87     Arguments: turtle, the Turtle3D in correct orientation
88                psi: backbone dihedral angle
89     Returns: Position of the modeled C atom. <Vector>
```

```
90     """
91
92     turtle.move(1.53)
93     turtle.roll(psi)
94     turtle.yaw(114.0)
95     return(Vector(turtle.getPosition()))
96
97 def to_nitrogen(turtle: Turtle3D, omega):
98     """
99     Turtle begins at carbonyl carbon, headed towards nitrogen of
100     second residue, with alpha carbon of first residue on left side.
101     Turtle ends at nitrogen of second residue, headed towards alpha carbon
102     of second residue, with carbonyl carbon of first residue on left side.
103     Omega will almost always be +180 degrees for trans peptide bonds.
104
105     Arguments: turtle, the Turtle3D in correct orientation
106                omega: backbone dihedral angle (peptide bond angle)
107     Returns: Position of the modeled C atom. <Vector>
108     """
109
110     turtle.move(1.32)
111     turtle.roll(omega)
112     turtle.yaw(123.0)
113     return(Vector(turtle.getPosition()))
114
115 def add_oxygen(turtle: Turtle3D):
116     """
117     Returns the position of the carbonyl oxygen assuming the Turtle3D
118     begins at carbonyl carbon, headed towards nitrogen of
119     second residue, with alpha carbon of first residue on left side.
120
121     Arguments: turtle, the Turtle3D in correct orientation
122
123     Returns: <Vector>Position of the modeled C atom.
124     """
125     loc = numpy.array((-0.673, -1.029, 0), "d")
126
127     return Vector(turtle.to_global(loc))
128
129 def to_oxygen(turtle: Turtle3D):
130     """
131     Returns the position of the carbonyl oxygen assuming the Turtle3D
132     begins at carbonyl carbon, headed towards nitrogen of
133     second residue, with alpha carbon of first residue on left side.
134
135     Arguments: turtle, the Turtle3D in correct orientation
136
137     Returns: <Vector>Position of the modeled C atom.
```

```
138     """
139
140     loc = numpy.array((-0.673, -1.029, 0.0), "d")
141
142     # update the Turtle3D object
143     turtle.to_global(loc)
144     return
145
146 # End of file
147
```