Lab on Python Classes

# Problem 7.25 (“Explorations in Computing” by John S. Conery)

Define a class named Sequence that will represent DNA sequences. To create a new Sequence call the constructor with a string of letters that are either A, C, G, or T:

>>> s = Sequence(’GATTACA’)

The constructor should save the string in an instance variable named \_seq. Your class should define the \_\_repr\_\_ method so it returns the value of \_seq when a user wants to print a sequence:

>>> s

GATTACA

Your class should also have a method named mutate that will change a single letter in the sequence. For example, this call will change the letter at location 4 to a T:

>>> s.mutate(0,’A’)

>>> s

AATTACA

*Note:* before you start to write any code, experiment with Python’s slice operator. Since \_seq is a string, \_seq[i:j] represents the characters from locations i through j in \_seq. One way to implement mutate is to have it replace \_seq with a new string that has the first i letters in \_seq (*i.e.* \_seq[:i]) plus the new letter plus the rest of the letters in \_seq (*i.e.* \_seq[i+1:]). Include one or more of the following methods.

• Define a method named insert. A call to s.insert(i,t) should insert the letters in string t before location i, *e.g.*

>>> s = Sequence(’AAAA’)

>>> s.insert(2,’TT’)

>>> s

AATTAA

Define a method named delete. A call to s.delete(i,n) should remove the n letters starting at location i:

>>> s

AATTAA

>>> s.delete(3,2)

>>> s

AATA

• Define a method named subseq. A call to s.subseq(i,j) should create a new Sequence object using the letters in locations i through j-1 of s.

# Problem 7.26 (“Explorations in Computing” by John S. Conery)

Define a class named Point that will represent a point on a graph. To create a point pass the x and y co-ordinates to the constructor:

>>> p1 = Point(1,1)

>>> p2 = Point(4,5)

Include the following methods in your class (the examples refer to the two points p1 and p2 shown above):

• The \_\_repr\_\_ method should display the point in standard mathematical notation, *e.g.*

>>> p1

(1,1)

• A method named dist should compute the distance between two points, *e.g.*

>>> p1.dist(p2)

5.0

• A method named polar should return a pair of values corresponding to the polar coordinates of the point:

>>> p1.polar()

(1.4142135623730951, 0.7853981633974483)

The polar coordinates of a point (x, y) are a pair of numbers (r, *q*) where r = Sqrt (x2 + y2) and *q* = tan-1 y/x (Python’s math library has a function named atan that computes tan-1).

# Problem 7.27 (“Explorations in Computing” by John S. Conery)

The formula for polar coordinates in the previous problem is valid only if the x-coordinate of a point is greater than 0. Modify the polar method so it returns the correct value if x is negative or 0.