Assignment 1 Report

Andy Hameed and hameea1

January 31, 2018

In assignment 1, part 1 we were asked to create a program that produced a curve and sequence ADT in python. The instructions were intentionally made ambiguous to yield different interpretations by students. This report will summarize the issues with ambiguity in a program specification and some questions on the methods for each ADT class.

1 Testing of the Original Program

Most of the test cases in my testSeq module used assertions to match the output of a method call with the expected output of the method. For example, the add method either appends an element to the end of the list or inserts an element in the sequence between to other elements, so I checked for these two cases and asserted the resulting sequence after an element was added or appended. For tests that were supposed to fail, I used a exceptional statements in my original code that would print out an error message. In hindsight, this is a futile way of catching errors and asserting test cases because these messages are very specific and likely not taken into consideration or foreseen by someone else.

I followed a form similar to the one provided in the assignment solution test module and counted the number of passing tests and overall number of tests, with a print statement displaying a pass or fail for each method and the ratio of passing tests to total tests. When testing my code, all tests passed. However, this result, as expected, does not make the two modules robust because there are cases that are not covered and unspecified by the assignment instructions. The testing was based on a source input file that I created with my own data, which elicits distorted test results.

2 Results of Testing Partner's Code

Results of Testing Partners Code When testing my partners code, only 5/10 tests passed through the test module. The passing tests were:

- size
- indexInSeq
- linVal
- quadVal
- npolyVal

3 Discussion of Test Results

3.1 Problems with Original Code

My original code did not have many issues but my testSeq module was not reliable. Each test for the `SeqT` class relied on the previous test based on the order of elements in the sequence after a given test call. This gave off inaccurate testing results

3.2 Problems with Partner's Code

When naming the instance variable in the constructor, my partner used self.seq meanwhile I used self.state. This made the test module inaccurate in testing my partners code because it was specifically tailored for the instance variable that I created. As a result, methods in CurveT were affected by this as well since the SeqT is imported into CurveT and used.

3.3 Problems with Assignment Specification

The assignment specification were made ambiguous in many ways. An example of this is seen in the instructions for quadratic interpolation, where x_0 is not specified in relation to x itself. I assumed it to be a point to the left of x_1 but there could have been many interpretations of this. I also found the specification for the add method to be ambiguous as well. At first I interpreted "adding" an element at index i means that the element comes after index i, but later on I decided to assume the value is being placed to be at index i and the rest of the sequence is pushed to the right instead.

4 Answers

1. For each of the methods in each of the classes, please classify it as a constructor, accessor or mutator.

Constuctors	Accessors	Mutators
CurveT	indexInSeq	set
SeqT	size	rm
	get	add
	linVal	
	quadVal	
	npolyVal	

2. What are the advantages and disadvantages of using an external library like `numpy`?

   numpy has built in functions like polyval and polyfit that evaluate and implement complex mathematical procedures. This abstraction allows developers to focus on the main implementation of their methods. Another advantage is that external libraries improves reliability since these methods have been taken consideration by the developers of the external library. Overall, external libraries improve productivity. On the other hand, external libraries like numpy could restrict implementation and customization. When functions have already been built and are abstracted from the developer, optimizing algorithms, customizing outputs and other challenges arise because the programmer may not have access to/permission to change the implementation of library functions.

3. The `SeqT` class overlaps with the functionality provided by Python's in-built list type. What are the differences between `SeqT` and Python's list type? What benefits does Python's list type provide over the `SeqT` class?

   The sequence class is designed to represent and resemble a sequence. In general, using the list type is more intuitive when it comes to using methods as opposed to the SeqT class which has methods that are difficult to interpret (Ex. indexToSeq). Although SeqT uses a list as its instance variable, it has functions that are not built into the list python module. Take for example indexInSeq which returns theposition where a value would be placed into a given sequence. This function is unique to the class and The SeqT uses customized exception statements to dodge error messages which is another advantage over Pythons list type. However, the list type has a multitude of built-in functions and methods which provides users with more control over the manipulation of lists. Pythons list type and the SeqT class both have their place in the implementation with their own advantages.

4. What complications would be added to your code if the assumption that $x_i < x_{i+1}$ no longer applied?

   An unsorted set of data presents several challenges. In order to implement methods in the curve class especially, a sorting algorithm would be needed before indexInSeq is used or a searching algorithm to find the two numbers in the sequence for linear and quadratic interpolation.

5. Will `linVal(x)` equal `npolyVal(n, x)` for the same `x` value? Why or why not?

   No, because the npolyVal method takes the argument n representing the degree of the polynomial that best fits the whole data at that point. So for a call of npolVal(1, x) we have a line of best fit being used to represent the whole data. We obtain our y value from a method of regression. Meanwhile the linVal method draws a line between precisely two points to represent the data, the point before and after x.

E Code for SeqADT.py

```
## @file SeqADT.py
# @title SeqT
# @author Andy Hameed
# @date 01/22/2017

## @brief This class represents a sequence
class SeqT:

## @brief Constructor for SeqT creates an empty list
    def __init__(self):
        self.state = []

## @brief add Method: Adds an element to the sequence or appends the element to the end of sequence
# @param v is the element being added
# @param i is the index placement of the element in the sequence
    def add(self,i, v):
        # append if index is equal to number of elements
        if ( i >= len(self.state)):
            self.state.append(v)
        #inserts element v at index i
        else:
            self.state.insert(i,v)

## @brief rm Method: Removes an element from the sequence
# @param i is the index of the element to be removed from the sequence
    def rm(self,i):
        try:
            del self.state[i]
        #Exception: index is out of range
        except:
            print("Out of range! Please choose another index")

## @brief Accepts two parameters to replace an element in the sequence with a different element
# @param v is the replacement element
# @param i is the index of the element to be replaced
    def set(self,i, v):
        try:
            self.state[i] = v
        except:
            print("Pick an index in range!")

## @brief Gets the value of an element at a given index
# @param i is the index of the element required
# @return the value of element at index i
    def get(self,i):
        try:
            return self.state[i]
        except:
            return "out of range"
## @brief Gives the number of elements in the sequence (size)
# @return the size of sequence
    def size(self):
        return len(self.state)

## @brief finds the ordering of an element with respect to sequence members
# @details by determing two adjacent elements that surround the number
# @param v is the number being fitted into sequence
# @return the lower bound (left) element out of the two elements
    def indexInSeq(self,v):
        i = 0
        while (True):
            if ((self.state[i] <= v) and (v <= self.state[i+1])):
                return i
            i += 1
```

F Code for CurveADT.py

```
## @file CurveADT.py
# @title CurveT
# @author Andy Hameed
# @date 01/22/2017

#Assume values of x increasing

from SeqADT import *
import numpy as np
import math

## @brief This class represents a curve in the Cartesian x-y plane
class CurveT:
    ## @brief The Constructor method for CurveT creats two sequences
    # @details of x and y values gathered from imput s
    # @param s is a string containing the name of the file to be read
    def __init__(self, s):
        self.X = SeqT()
        self.Y = SeqT()
        infile = open(s, "r")

        for line in infile:
            temp = line.strip("\n")
            temp = temp.split(", ")

            self.X.add(self.Y.size(), int(temp[0])) #append each x coordinate
            self.Y.add(self.X.size(), int(temp[1])) #append y coordinate

        infile.close()

    ## @brief The linVal method implements linear interpolation
    # @details on x given the sequence of x and y values
    # @param x is the point corresponding to the y value being calculated
    # @return the estimated value y for a given x
    def linVal(self, x):

        i = self.X.indexInSeq(x)
        num = self.Y.get(i+1) - self.Y.get(i)
        denom = self.X.get(i+1) - self.X.get(i)
        y = (num/denom)*(x - self.X.get(i)) + self.Y.get(i)

        return y

    ## @brief The quadVal method implements Quadratic Interpolation
    # @details on x given the sequence of x and y values
    # @param x is the point corresponding to the y value being calculated
    # @return the estimated value y for a given x
    def quadVal(self,x):
        i = self.X.indexInSeq(x)

        num1 = self.Y.get(i+1) - self.Y.get(i-1)
        denom1 = self.X.get(i+1) - self.X.get(i-1)
        lin = (num1/denom1)*(x - self.X.get(i)) + self.Y.get(i)

        num = self.Y.get(i+1) - 2*(self.Y.get(i)) + self.Y.get(i-1)
        denom = 2*(self.X.get(i+1) - self.X.get(i))**2

        y = (num/float(denom))*(x - self.X.get(i))**2 + lin

        return y

    ## @brief The npolyVal method implements regression(Best fitting)
    # @details to find the y value for a given point x
    # @param x is the point corresponding to the y value being calculated
    # @return the estimated value y for a given x using numpy functions
    def npolyVal(self,n ,x):
        curve = np.polyfit(self.X.state, self.Y.state, n)

        value = np.polyval(curve, x)
        return value
```

G Code for testSeqs.py

```
## @file testSeqs.py
# @title testSeqs
# @author Andy Hameed
# @date 01/22/2017

from SeqADT import *
from CurveADT import *

##SeqADT

## @brief Tests the add Method of the SeqT class
# @details Checking if values at each index correspond to the added values
#       The SeqT add method
def test_add():
        global tested, tests
        tests+=1
        try:
                #Test 1: Inserting at the end of sequence (appending)
                test_seq.add(0,2)
                test_seq.add(1,3)
                test_seq.add(2,4)
                test_seq.add(3,6)

                #Test 2: Inserting between elements
                test_seq.add(2,5)

                assert test_seq.state[0] == 2
                assert test_seq.state[1] == 3
                assert test_seq.state[2] == 5
                assert test_seq.state[3] == 4
                assert test_seq.state[4] == 6
                tested+=1
                print("add test: PASS")
        except:
                print("add test: FAIL")

## @brief Tests the rm Method of the SeqT class
# @details Checking if values have been removed based on index
#       The SeqT rm method
def test_rm():
        global tested, tests
        tests+=1

        try:
                #Test 1: Removing element in range
                test_seq.rm(1)
                assert test_seq.state[1] != 3

                #Test 2: Removing element out of range
                #Error caught
                test_seq.rm(9)

                tested+=1
                print("rm test: PASS")
        except:
                print("rm test: FAIL")


## @brief Tests the set Method of the SeqT class
# @details Checking if indexed value has been replaced by input element
#       The SeqT set method
#[2,5,4,8]
def test_set():
        global tested, tests
        tests+=1

        try:

                #Test 1: Set element within range
                test_seq.set(3,8)
                assert test_seq.state[3] == 8

                #Test 2: Set element out of range
                #Error caught
                test_seq.set(10,6)
```

```
                tested+=1
                print("Set test: PASS")
        except:
                print("Set test: FAIL")


## @brief Tests the get Method of the SeqT class
# @details Checking if the value returned corresponds to element in sequence
#       The SeqT get method
def test_get():
        global tested, tests
        tests+=1

        try:
                #Test 1: Get element within range
                assert test_seq.get(2) == 4

                #Test 2: Get element out of range
                #Error caught
                assert test_seq.get(10) == "out of range"

                tested+=1
                print("get test: PASS")
        except:
                print("get test: FAIL")


## @brief Tests the size Method of the SeqT class
# @details Checking if the value returned corresponds to size of sequence
#       The SeqT size method
def test_size():
        global tested, tests
        tests+=1

        try:
                assert test_seq.size() == 4

                tested+=1
                print("size test: PASS")
        except:
                print("size test: FAIL")


## @brief Tests the indexInSeq Method of the SeqT class
# @details Checking if the value returned corresponds to size of sequence
#       The SeqT size method
def test_indexInSeq():
        global tested, tests
        tests+=1

        try:
                assert test_seq.indexInSeq(7) == 2

                tested+=1
                print("indexInSeq test: PASS")
        except:
                print("indexInSeq test: FAIL")

#***CurveADT***

## @brief Tests the Constructor of the CurveT class
# @details Testing based on the dataTest.txt file
#       The CurveADT Constructor __init__
def test_CurveT():
        global tested, tests
        tests+=1

        try:
                assert new.X.get(4) == 13
                assert new.Y.get(5) == 20

                tested+=1
                print("CurveT test: PASS")
        except:
                print("CurveT test: FAIL")


## @brief Tests the linVal Method of the CurveT class
# @details Checking if the value returned corresponds to
# @details value obtained using an online linear interpolation calculator
```

```
# @details https://www.ajdesigner.com/phpinterpolation/linear_interpolation_equation.php
#       The CurveADT linVal Method
def test_linVal():
        global tested, tests
        tests+=1
        Tolerance = 1e-9

        try:
                assert ((new.linVal(15) - 16) < Tolerance)

                tested+=1
                print("linVal test: PASS")
        except:
                print("linVal test: FAIL")

## @brief Tests the quadVal Method of the CurveT class
# @details Checking if the value returned corresponds to
# @details value obtained using an online quadratic interpolation calculator
# @details https://www.johndcook.com/quadratic_interpolator.html
#       The CurveADT quadVal Method
def test_quadVal():
        global tested, tests
        tests+=1
        Tolerance = 1e-9

        try:
                assert ((new.quadVal(15) - 14.666666666666666) < Tolerance)

                tested+=1
                print("quadVal test: PASS")
        except:
                print("quadVal test: FAIL")

## @brief Tests the npolyVal Method of the CurveT class
# @details Checking if the value returned corresponds to
# @details value obtained from online regression calculator
# @details https://www.graphpad.com/quickcalcs/linear1/
#       The CurveADT npolyVal Method
def test_npolyVal():
        global tested, tests
        tests+=1
        Tolerance = 1e-9

        try:
                assert ((new.npolyVal(3,8) - 24.190458598099138) < Tolerance)

                tested+=1
                print("npolyVal test: PASS")
        except:
                print("npolyVal test: FAIL")



tested=0
tests=0

new = CurveT("dataTest.txt")
test_seq = SeqT()

test_add()
test_rm()
test_set()
test_get()
test_size()
test_indexInSeq()

test_CurveT()
test_linVal()
test_quadVal()
test_npolyVal()


print('\n', tested,' of', tests,' tests passed.')
```

H Code for Partner's SeqADT.py

```
## @file SeqADT.py
#  @author Guiye Wu
#  @brief Provides the seqT ADT class for representing curve's data points
#  @date 1/21/2018

## @brief An ADT that modify a sequence for curve's data points
class SeqT:

    ## @brief seqT constructor
    #  @details Initializes an empty sequence
    def __init__(self):
        self.state = []

    ## @brief Add a real number into the sequence at specific position
    #  @details The real number can not add to the position that is not in the
    #           range of the sequence, the range of the sequence is from 0 to the
    #           sequence size.
    #  @param i The index in the sequence that the value is added at
    #  @param v The real number that should be added
    #  @exception If the index is greater than the length of the sequence or index
    #             is less than 0, ValueError raise.
    def add(self,i,v):
        if ((i > len(self.state)) or (i < 0)):
            raise ValueError("Index is out of the range")
        else:
            self.state.insert(i,v)

    ## @brief Remove a value in the sequence
    #  @param i The index of the value that should be remove
    def rm(self,i):
        ival = self.state[i]
        self.state.remove(ival)

    ## @brief Modify the sequence at specific index to have a value
    #  @param i The index in the sequence that the value is setted at
    #  @param v The real number that should be setted
    def set(self,i,v):
        self.state[i] = v

    ## @brief Get the value at specific positon
    #  @param i The index of the value that should be gotten
    #  @return The value at the specific position of the sequence
    def get(self,i):
        return self.state[i]

    ## @brief Get the size of the sequence
    #  @return The size of the sequence
    def size(self):
        return len(self.state)

    ## @brief Take an input of a real number and get the index from a specific range
    #  @details The function take the real number, finds if the any number in the
    #           sequence that is less than or equal to the real number and also the
    #           next value of the sequence is greater than or equal to the real
    #           number. If the value in the sequence exist, then output the index of
    #           the value.
    #  @param v The real number that is used to get the index
    #  @return Returns the index of the real number if it is in the range
    #  @exception ValueError Raise the error if the real number is not in the range
    def indexInseq(self,v):
        i = 0
        for j in range(len(self.state)-1):
            if ((self.state[j] <= v) and (v <= self.state[j+1])):
                return i
            else:
                i += 1
        if (i == (len(self.state)-1)):
            raise ValueError("value is not in range")
```

I Code for Partner's CurveADT.py

```
## @file CurveADT.py
#  @author Guiye Wu
#  @brief Provides the CurveT ADT class for finding a point by interpolation
#  @date 1/21/2018

import numpy
from SeqADT import *

## @brief An ADT that determine a point by interpolation and regression
#  @details The point is determined by curve's data points in Cartesian space
#           x and y. The first method is to use linear interpolation and the
#           second method is to use quadratic interpolation and the last one
#           is regression by using the function polyfit in numpy.
class CurveT:

    ## @brief CurveT constructor
    #  @details Initializes a data of x values and y values by using ADT of SeqT.
    #           The values are get from a txt file, the txt file has two column
    #           of values, the first column represent the data of x value, and
    #           the second column is y values. Each row of each number sparate by
    #           a comma and a space, and the end of the row is a newline.
    #  @param s The txt file name
    def __init__(self,s):
        f = open(s,"r")
        self.x = SeqT()
        self.y = SeqT()
        i = 0
        for line in f:
            lst = line.split(", ")
            self.x.add(i,float(lst[0]))
            self.y.add(i,float(lst[1]))
            i += 1
        f.close()

    ## @brief Use linear interpolation to find the y value
    #  @details Linear interpolation uses the formula
    #           'y = (y2-y1)/(x2-x1)*(x-x1)+y1'
    #           to find y by input x, (x1, y1) is the point on the curve to the
    #           left of x and (x2, y2) is the point on the curve to the right.
    #  @param x The input value x that is used to find value y
    #  @return The output value y by linear interpolation
    def linVal(self,x):
        index = self.x.indexInSeq(x)
        x1 = self.x.get(index)
        x2 = self.x.get(index+1)
        y1 = self.y.get(index)
        y2 = self.y.get(index+1)
        return (y2-y1)/(x2-x1)*(x-x1)+y1

    ## @brief Use quadratic interpolation to find the y value
    #  @details Quadratic interpolation uses the formula
    #           'y1+(y2-y0)/(x2-x0)*(x-x1)+(y2-2*y1+y0)/(2*(x2-x1)^2)*(x-x1)^2'
    #           to find y by input value x, this method is similar to linear
    #           interpolation, and (x0, y0) is the point on the curve to the left
    #           of (x1, y1).
    #  @param x The input value x that is used to find value y
    #  @return The output value y by quadratic interpolation
    def quadVal(self,x):
        index = self.x.indexInSeq(x)
        x0 = self.x.get(index-1)
        x1 = self.x.get(index)
        x2 = self.x.get(index+1)
        y0 = self.y.get(index-1)
        y1 = self.y.get(index)
        y2 = self.y.get(index+1)
        return y1+(y2-y0)/(x2-x0)*(x-x1)+(y2-2*y1+y0)/(2*(x2-x1)**2)*(x-x1)**2

    ## @brief Use regression to find the y value
    #  @details Regression is to find a polynomial that minimizes the square of
    #           error between the data and the fitted polynomial, it is done by
    #           using numpy.polyfit that is function from numpy directly. This
    #           function will need the degree of the polynomial as input, and
    #           also an array of x data and an array of y data, then function
    #           return an array of polynomial coefficient. Using the coeficient
    #           to be the input x coefficient to find the y value.
    #  @param n Polynomial coefficient
```

```
#  @param x The input x value
#  @return The output value y by regression
def npolyVal(self,n,x):
    xarr = []
    yarr = []
    for i in range(self.x.size()):
        xarr.append(self.x.get(i))
        yarr.append(self.y.get(i))
    coef = numpy.polyfit(xarr,yarr,n)
    y = 0
    for j in range(n+1):
        y += coef[j]*(x**(-j+n))
    return y
```

J Makefile

```
PY = python3
DOXY = doxygen
DOXYCFG = doxConfig

RMDIR = rm -rf

.PHONY: test doc clean

test:
	$(PY) src/testSeqs.py

doc:
	$(DOXY) $(DOXYCFG)
	cd latex && $(MAKE)

clean:
	@- $(RMDIR) html
	@- $(RMDIR) latex
```