**10/29/15**

"""

Donato, Brandon

bdonato1@binghamton.edu

CS 110 - B57

Jia Yang

Assignment8Ex1

"""

# Imports --------------------------------------------------------------------

import random

# Functions -------------------------------------------------------------------

# Creates list of random integers

# returns the list of random integers (list)

def listGenerator():

newList = []

for a in range(random.randint(1,25)):

newList.append(random.randint(1,25))

return newList

# Check that the starting and limiting indicies are positive integers

# param startIndex (str) - The beginning index

# param limitIndex (str) - The limiting index

# invoke str.isdigit()

# returns True when valid, False otherwise (bool)

def integerValidation(startIndex,limitIndex):

return startIndex.isdigit() and limitIndex.isdigit()

# Checks index inputs are valid inputs within the range of the list's

# index

# param startIndex (str) - The beginning index

# param limitIndex (str) - The limiting index

# param aList (list) - The inital list

# invoke len()

# invoke int()

# invoke integerValidation()

# returns True when valid, False otherwise (bool)

def indexValidation(startIndex,limitIndex,aList):

return (integerValidation(startIndex,limitIndex)) and\

(int(limitIndex) > int(startIndex)) and\

(int(limitIndex) <= (len(aList) - 1))

**Page 1 of 11**

# Converts index inputs from (str) to (int)

# param indexStr (str) - The index string

# invoke int()

# returns indexInt (int) - The index integer

def indexConversion(indexStr):

indexInt = int(indexStr)

return indexInt

# Checks the mySlice() output against Python's slice operation

# param theSlice (list) - The slice computed from mySlice()

# param pythonsSlice (list) - The slice computed from Python's slice

# operator

# returns True when valid, False otherwise (bool)

def mySliceValidation(theSlice, pythonsSlice):

return theSlice == pythonsSlice

# Creates a sublist when given a list, starting index and limiting index

# param aList (list) - The given list

# param startIndex (int) - The starting index

# param limitIndex (int) - The limiting index

# returns the new sublist (list)

def mySlice(aList,startIndex,limitIndex):

newList = []

for i in range(startIndex, limitIndex):

newList.append(aList[i])

##print(newList)

return newList

"""

Donato, Brandon

bdonato1@binghamton.edu

CS 110 - B57

Jia Yang

Assignment8Ex1

"""

"""

Analysis:

Write a program that returns a slice when given a list, a starting index

and a limit index. Compare the results with Python's slice operation.

**Page 2 of 11**

Output to monitor:

theSlice (list) - Slice created by the program

pythonsSlice (list) - Slice created using Python operator

Input from keyboard:

startIndex (int) - Starting slice index

limitIndex (int) - Limiting slice index

Tasks allocated to functions:

All functions are located in listFunctionsModule

integerValidation() - Simple predicate function

indexValidation() - Simple predicate function

indexConversion() - Converts from (str) to (int)

mySliceValidation() - Simple predicate function

mySlice()

"""

# Imports --------------------------------------------------------------------

import listFunctionsModule

# Main ------------------------------------------------------------------------

# Generates the initial list that will be sliced, the starting slice index, and

# the limiting slice index

# Generates a slice using mySlice() and compares it to Python's slice

# operation

def main():

# Generates and prints the initial list, the list's length and the program's

# purpose

aList = listFunctionsModule.listGenerator()

print("The list is: ", aList, "and its length is", len(aList),"\n")

print("This program will create a slice of a list without using the\

slice operator \n")

# Priming read and repeat

startIndex = input("Enter the starting index for the slice\nOr press\

<Enter> to quit: ")

while startIndex:

**Page 3 of 11**

# Gets the limiting index, validates inputted data and converts as

# necessary

limitIndex = input("Enter the limiting index for the slice: ")

while not listFunctionsModule.\

indexValidation(startIndex,limitIndex,aList):

print("One or both of your indices are invalid")

startIndex = input("Enter the starting index for the slice: ")

limitIndex = input("Enter the limiting index for the slice: ")

startIndex = listFunctionsModule.indexConversion(startIndex)

limitIndex = listFunctionsModule.indexConversion(limitIndex)

# Prints out the slice generated by mySlice() and the slice generated

# by Python

theSlice = listFunctionsModule.\

mySlice(aList,startIndex,limitIndex)

print("My slice is: %-s" % theSlice)

pythonsSlice = aList[startIndex:limitIndex]

print("Python's slice is: %-s " % pythonsSlice)

# Compares the slices generated to see whether they are equal or not

# and prints whether mySlice() makes successful slices or not

if listFunctionsModule.\

mySliceValidation(theSlice,pythonsSlice):

print("Success! The two slices are equal.\n")

else:

print("The slices weren't equal. Please report this problem.\n")

# Continuation read

startIndex = input("Enter the starting index for the slice\nOr press\

<Enter> to quit: ")

main()

'''

Donato, Brandon

bdonato1@binghamton.edu

CS 110 - B57

Jia Yang

Assignment8Ex2

'''

**Page 4 of 11**

"""

Analysis:

Create a program that outputs a L-System through a certain amount of

iterations and uses the turtle module in Python to draw the L-System.

Output to window:

The L-System is draw after a predetermined number of iterations

Tasks allocated to functions:

createLSystem() - This makes the actual L-System through so many iterations

processedString() - This function creates the single iteration of the

L-System

applyRules() - This function modifies the string copy to obey the given

rules

drawLSystem() - This function actually draws the L-System

"""

# Imports --------------------------------------------------------------------

import turtle

# Constants ------------------------------------------------------------------

ITERATIONS = 5

ANGLE = 25.7

TURTLE\_SIZE = 5

# Functions ------------------------------------------------------------------

# Iterates the L-System through a given number of times

# param iterations (int) - The number of iterations the program is executed

# param beginning (str) - The beginning string of the L-System

# invokes range()

# invokes processedString()

# returns endStr (str) - The final L-System after being iterated through

def createLSystem(iterations, beginning):

startStr = beginning

endStr = ""

for i in range(iterations):

endStr = processedString(startStr)

startStr = endStr

return endStr

**Page 5 of 11**

# Given a string, copies the string, applies the rules of the L-System

# and outputs the new string

# param oldStr (str) - The initial string to be copied and modified

# invokes in()

# invokes applyRules()

# returns newStr (str) - The modified initial string which the rules have

# been applied to

def processedString(oldStr):

newStr = ""

for character in oldStr:

newStr += applyRules(character)

return newStr

# Based upon a character, modifies the character in a new string based upon

# given rules

# param character (str) - A single character of a string

# returns newStr (str) - The modified character in a copy that's been put

# through the rules

def applyRules(character):

newStr = ""

if character == 'H':

newStr = 'HFX[+H][-H]'

elif character == 'X':

newStr ='X[-FFF][+FFF]FX'

else:

newStr = character

return newStr

# Draws the L-System given the inputted L-System

# param turtle (turtle) - The turtle that draws the system

# param instructions (str) - The L-System that is to be drawn

# param angle (float) - The angle that the turtle will turn

# param distance (int) - The distance the turtle goes before processing the

# next step

# invokes turtle.heading()

# invokes turtle.forward()

# invokes turtle.right()

# invokes turtle.left()

# invokes turtle.xcor()

# invokes turtle.ycor()

# invokes turtle.setheading()

# invokes turtle.setposition()

# invokes list.pop()

# invokes list.append()

**Page 6 of 11**

def drawLSystem(turtle, instructions, angle, distance):

infoList = []

for cmd in instructions:

if cmd == 'F':

turtle.forward(distance)

elif cmd == '+':

turtle.right(angle)

elif cmd == '-':

turtle.left(angle)

elif cmd == '[':

infoList.append([turtle.heading(),turtle.xcor(),turtle.ycor()])

elif cmd ==']':

newInfo = infoList.pop()

turtle.setheading(newInfo[0])

turtle.setposition(newInfo[1],newInfo[2])

# Main -----------------------------------------------------------------------

# Generates the list of instructions that will be used in drawing the L-System

# as well as sets up of the turtle and window to draw on.

def main():

# Generates instructions

inst = createLSystem(ITERATIONS, "H")

print(inst)

# Creates the turtle and window and sets the turtle in place before drawing

turt = turtle.Turtle()

wn = turtle.Screen()

turt.speed(0)

turt.penup()

turt.goto(-200,0)

turt.pendown()

# Draws the actual L-System

drawLSystem(turt, inst, ANGLE, TURTLE\_SIZE)

main()

**Page 7 of 11**

'''

Donato, Brandon

bdonato1@binghamton.edu

CS 110 - B57

Jia Yang

Assignment8Ex3

'''

"""

Analysis:

Create a program that outputs a L-System through a certain amount of

iterations and uses the turtle module in Python to draw the L-System.

Output to window:

The L-System is draw after a predetermined number of iterations

Tasks allocated to functions:

createLSystem() - This makes the actual L-System through so many iterations

processedString() - This function creates the single iteration of the

L-System

applyRules() - This function modifies the string copy to obey the given

rules

drawLSystem() - This function actually draws the L-System

"""

# Imports --------------------------------------------------------------------

import turtle

# Constants ------------------------------------------------------------------

ITERATIONS = 5

ANGLE = 25

TURTLE\_SIZE = 5

# Functions ------------------------------------------------------------------

**Page 8 of 11**

# Iterates the L-System through a given number of times

# param iterations (int) - The number of iterations the program is executed

# param beginning (str) - The beginning string of the L-System

# invokes range()

# invokes processedString()

# returns endStr (str) - The final L-System after being iterated through

def createLSystem(iterations, beginning):

startStr = beginning

endStr = ""

for i in range(iterations):

endStr = processedString(startStr)

startStr = endStr

return endStr

# Given a string, copies the string, applies the rules of the L-System

# and outputs the new string

# param oldStr (str) - The initial string to be copied and modified

# invokes in()

# invokes applyRules()

# returns newStr (str) - The modified initial string which the rules have

# been applied to

def processedString(oldStr):

newStr = ""

for character in oldStr:

newStr += applyRules(character)

return newStr

# Based upon a character, modifies the character in a new string based upon

# given rules

# param character (str) - A single character of a string

# returns newStr (str) - The modified character in a copy that's been put

# through the rules

def applyRules(character):

newStr = ""

if character == 'F':

newStr = 'F[-F]F[+F]F'

else:

newStr = character

return newStr

**Page 9 of 11**

# Draws the L-System given the inputted L-System

# param turtle (turtle) - The turtle that draws the system

# param instructions (str) - The L-System that is to be drawn

# param angle (int) - The angle that the turtle will turn

# param distance (int) - The distance the turtle goes before processing the

# next step

# invokes turtle.heading()

# invokes turtle.forward()

# invokes turtle.right()

# invokes turtle.left()

# invokes turtle.xcor()

# invokes turtle.ycor()

# invokes turtle.setheading()

# invokes turtle.setposition()

# invokes list.pop()

# invokes list.append()

def drawLSystem(turtle, instructions, angle, distance):

infoList = []

for cmd in instructions:

if cmd == 'F':

turtle.forward(distance)

elif cmd == '+':

turtle.right(angle)

elif cmd == '-':

turtle.left(angle)

elif cmd == '[':

infoList.append([turtle.heading(),turtle.xcor(),turtle.ycor()])

elif cmd ==']':

newInfo = infoList.pop()

turtle.setheading(newInfo[0])

turtle.setposition(newInfo[1],newInfo[2])

# Main -----------------------------------------------------------------------

# Generates the list of instructions that will be used in drawing the L-System

# as well as sets up of the turtle and window to draw on.

def main():

# Generates instructions

inst = createLSystem(ITERATIONS, "F")

print(inst)

**Page 10 of 11**

# Creates the turtle and window and sets the turtle in place before drawing

turt = turtle.Turtle()

wn = turtle.Screen()

turt.speed(0)

turt.penup()

turt.goto(-200,0)

turt.pendown()

# Draws the actual L-System

drawLSystem(turt, inst, ANGLE, TURTLE\_SIZE)

main()

**Page 11 of 11**