# Evidence 1

class Stack:

def \_\_init\_\_(self): # Create()

self.\_\_items = []

self.\_\_top = 0

def Push(self, item):

self.\_\_items.append(item)

self.\_\_top += 1

def Pop(self):

if not self.IsEmpty():

self.\_\_top -= 1

return self.\_\_items.pop()

else:

return None # stack is empty, nothing to pop

def Peep(self):

if not self.IsEmpty():

return self.\_\_items[self.\_\_top - 1]

else:

return None # stack is empty, nothing to pop

def IsEmpty(self):

return (self.\_\_top == 0)

# Evidence 2

def CheckNested(NestedBrackets):

NestStack = Stack()

SeqValid = True

BracketL = { "(": 0, "[": 1, "{": 2 }

BracketR = { ")": 0, "]": 1, "}": 2 }

for character in NestedBrackets:

if character in "([{":

NestStack.Push(character)

else:

if NestStack.IsEmpty():

SeqValid = False

break

elif (BracketR[character] == BracketL[NestStack.Peep()]):

NestStack.Pop()

else:

SeqValid = False

break

if not NestStack.IsEmpty():

SeqValid = False

return SeqValid # true if valid, false if not

# Evidence 3

DataFile = open("DATA.txt", "r")

ErrorsFile = open("ERRORS.txt", "w")

for line in DataFile:

if line[-1] == "\n":

line = line[:-1]

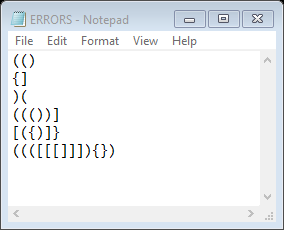
if CheckNested(line) == False: # not properly nested

ErrorsFile.write("{0}\n".format(line))

DataFile.close()

ErrorsFile.close()

# Evidence 4



# Evidence 5

def CheckWellformed(NestedBrackets):

NestStack = Stack()

Expected = None # the expected bracket if there is an error

BracketL = { "(": 0, "[": 1, "{": 2 }

BracketR = { ")": 0, "]": 1, "}": 2 }

BracketLList = [ "(" , "[" , "{" ]

BracketRList = [ ")" , "]" , "}" ]

for character in NestedBrackets:

if character in "([{":

NestStack.Push(character)

else:

if NestStack.IsEmpty():

Expected = BracketLList[BracketR[character]] # its corresponding opposite

break

elif (BracketR[character] == BracketL[NestStack.Peep()]):

NestStack.Pop()

else:

Expected = BracketLList[BracketR[character]] # its corresponding opposite

break

if (not NestStack.IsEmpty() and Expected == None):

Expected = BracketRList[BracketL[NestStack.Peep()]]

if Expected != None:

return "{0}: Expecting '{1}'".format(NestedBrackets, Expected)

else:

return None # means it's properly nested

# task 5

DataFile = open("DATA.txt", "r")

ErrorsFile = open("ERRORS.txt", "w")

for line in DataFile:

if line[-1] == "\n":

line = line[:-1]

ErrorMessage = CheckWellformed(line)

if ErrorMessage != None:

ErrorsFile.write("{0}\n".format(ErrorMessage))

DataFile.close()

ErrorsFile.close()

# Evidence 6

