Lab 7

# Exercise 1

from stack import Stack

'''Does the conversion bit'''

def baseConverter(decNumber,base):

digits = "0123456789ABCDEF"

remstack = Stack()

while decNumber > 0:

rem = decNumber % base

remstack.push(rem)

decNumber = decNumber // base

newString = ""

while not remstack.isEmpty():

newString = newString + digits[remstack.pop()]

return newString

'''Main function of the program'''

def main():

numlist = [15, 30, 267, 32344]

for i in range(len((numlist))): #To save my time and energy, and just made the test for each number to be a list tested for each item in the list for the base needed.

print("Number: " + str(numlist[i]))

print("Number in base three is " + baseConverter(numlist[i], 3))

print("Number in base seven is " + baseConverter(numlist[i], 7))

print("Number in base sixteen is " + baseConverter(numlist[i], 16))

if \_\_name\_\_ == "\_\_main\_\_":

main()

Outputs (since I did them all in a loop, there does not need to be four test cases):

Number: 15

Number in base three is 120

Number in base seven is 21

Number in base sixteen is F

Number: 30

Number in base three is 1010

Number in base seven is 42

Number in base sixteen is 1E

Number: 267

Number in base three is 100220

Number in base seven is 531

Number in base sixteen is 10B

Number: 32344

Number in base three is 1122100221

Number in base seven is 163204

Number in base sixteen is 7E58

# Exercise 2

from stack import Stack

'''The main function of the program'''

def main():

'''This code is copy and pasted 4 times to open and check each file I used. Again, to save my time.'''

f = open("page1.html", "r")

string = f.read()

if brackChecker(string): #Checks first to make sure the brackets work, before moving onto the more complex function. Prints results.

print(wordChecker(string))

else:

print(brackChecker(string))

f = open("page2.html", "r")

string = f.read()

if brackChecker(string): #Checks first to make sure the brackets work, before moving onto the more complex function. Prints results.

print(wordChecker(string))

else:

print(brackChecker(string))

f = open("page3.html", "r")

string = f.read()

if brackChecker(string): #Checks first to make sure the brackets work, before moving onto the more complex function. Prints results.

print(wordChecker(string))

else:

print(brackChecker(string))

f = open("page4.html", "r")

string = f.read()

if brackChecker(string): #Checks first to make sure the brackets work, before moving onto the more complex function. Prints results.

print(wordChecker(string))

else:

print(brackChecker(string))

'''Checks to makes sure the brackets (for lack of a better term) are correct.'''

def brackChecker(symbolString):

s = Stack()

balanced = True

index = 0

while index < len(symbolString) and balanced:

symbol = symbolString[index]

if symbol == "<":

s.push(symbol)

elif symbol == ">":

if s.isEmpty():

balanced = False

else:

s.pop()

index = index + 1

if balanced and s.isEmpty():

return True

else:

return False

'''I found my algorithm was easier to work with than using the import re.

It does stuff similar to the first comparison, except it adds the words inside the brackets, without the <>.

Then makes sure every single word starts with a / for comparison, then it compares.'''

def wordChecker(string):

'''Gets the variables'''

i = 0

wordstart = False

word = ""

wordlist = []

comparelist = []

wordcheck = True

'''Gets all the words enclosed in <> in a list without the <>'''

while i < len(string):

symbol = string[i]

if symbol == ">" and wordstart:

wordlist.append(word)

wordstart = False

word = ""

elif wordstart:

word = word + symbol

elif symbol == "<":

wordstart = True

i = i + 1

'''Makes each word start with /'''

for x in range(len(wordlist)):

word = wordlist[x]

if word[0] != "/":

word = "/" + word

comparelist.append(word)

comparelist.sort() #Sorting makes comparison easier.

'''Compares each item. Then, if a match is found, they be banished to the shadow realm.'''

leng = len(comparelist)

while True:

if 0 >= leng:

break

if comparelist[0] == comparelist[1]:

comparelist.pop(0)

comparelist.pop(0)

leng = leng - 2

else:

wordcheck = False

break

'''Makes sure the list is empty, and returns results.'''

if comparelist == []:

return wordcheck

else:

wordcheck = False

return wordcheck

if \_\_name\_\_ == "\_\_main\_\_":

main()

Files used:

page1.html:

<html>

<head>Example</head>

<h1> some text </h1>

</html>

page2.html:

<hml>

<hed>Example</hd>

<h1> some text </h1>

</tml>

page3.html:

html

<head>Example</head>

<h1> some text </h1

</html

page4.html:

<html></html>

<head>Computer Science is suppossed to be my major</head>

<h1> With that being said, </h1>

<b>this will be most unfortunate if I do not get an A on this assignment</b>

Outputs (once again, I did all of my outputs in one go, so there is no need for four test cases.

True

False

False

True