# Checking Phone Number Without Regex

def isPhoneNumber(num):

if len(num) != 12:

return False

for i in range(0, 3):

if not num[i].isdecimal():

return False

if num[3] != '-':

return False

for i in range(4, 7):

if not num[i].isdecimal():

return False

if num[7] != '-':

return False

for i in range(8, 12):

if not num[i].isdecimal():

return False

return True

message = 'Call me at 415-555-1011 tomorrow. 415-555-9999 is my office number.'

for i in range(len(message)):

phone\_number = message[i:i+12]

if isPhoneNumber(phone\_number):

print('Phone number found:', phone\_number)

print('Done')

# Phone Number and Email Address Extractor

import re, pyperclip

txt = str(pyperclip.paste())

phone\_emails = []

*# Phone number*

phone\_regex = re.compile(r'''(

(\(\d{3}\)|\d{3}) # **first** group - area code

([-.\s]) # **second** group

(\d{3}) # **third** group - 3 digit numbers

([-.\s]) # **fourth** group

(\d{4}) # **fifth** group - 4 digit numbers

)''', re.VERBOSE)

phone\_matches = phone\_regex.findall(txt)

for res in phone\_matches:

phone\_emails.append('-'.join([res[1], res[3], res[5]]))

*# Email address*

email\_regex = re.compile(r'(([a-zA-Z0-9.-]+)@([a-zA-Z0-9]+)\.(com|edu|ca))')

email\_matches = email\_regex.findall(txt)

for email in email\_matches:

phone\_emails.append(email[0])

***# Copy to Clipboard***

# REM: Only str, int, float, and bool values can be copied to the clipboard, not list

pyperclip.copy('\n'.join(phone\_emails))

print('Matched Phone Numbers and Email Adress:')

print('\n'.join(phone\_emails))

# Strong Password Detection

import re

def password\_validity(password):

p1 = re.compile(r'[A-Z]+')

p2 = re.compile(r'[a-z]+')

p3 = re.compile(r'\d+')

p4 = re.compile(r'.{8,}')

if not p1.search(password):

print('Password must contain at least one uppercase letter.')

elif not p2.search(password):

print('Password must contain at least one lowercase letter')

elif not p3.search(password):

print('Password must contain at least one digit')

elif not p4.search(password):

print('Password must have at least 8 chars')

else:

print('Valid Password')

password = 'abcdefgh123'

password\_validity(password)

# Regex Version of Strip

import re

def regex\_strip(string, rm\_str=None):

if rm\_str == None:

return string.strip()

else:

string = string.strip()

string\_regex = re.compile(rm\_str)

return string\_regex.sub('', string)

if \_\_name\_\_ == '\_\_main\_\_':

string = input('Enter a string: ')

rm\_str = input('Enter the str you want to remove from the string or just clik enter: (Optional)')

string = regex\_strip(string, rm\_str)

print (string)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

file\_size = 0

for filename in os.listdir('C:\\Windows\\System32'):

file\_size = file\_size + os.path.getsize(os.path.join('C:\\Windows\System32', filename))

print(file\_size) # 2159434078

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Generating Random Quiz Files

import random

# The quiz data. Keys are states and values are their capitals.

capitals = {'Alabama': 'Montgomery', 'Alaska': 'Juneau', 'Arizona': 'Phoenix', 'Arkansas': 'Little Rock', 'California': 'Sacramento', 'Colorado': 'Denver', 'Connecticut': 'Hartford', 'Delaware': 'Dover', 'Florida': 'Tallahassee', 'Georgia': 'Atlanta', 'Hawaii': 'Honolulu', 'Idaho': 'Boise', 'Illinois': 'Springfield', 'Indiana': 'Indianapolis', 'Iowa': 'Des Moines', 'Kansas': 'Topeka', 'Kentucky': 'Frankfort', 'Louisiana': 'Baton Rouge', 'Maine': 'Augusta', 'Maryland': 'Annapolis', 'Massachusetts': 'Boston', 'Michigan': 'Lansing', 'Minnesota': 'Saint Paul', 'Mississippi': 'Jackson', 'Missouri': 'Jefferson City', 'Montana': 'Helena', 'Nebraska': 'Lincoln', 'Nevada': 'Carson City', 'New Hampshire': 'Concord', 'New Jersey': 'Trenton', 'New Mexico': 'Santa Fe', 'New York': 'Albany', 'North Carolina': 'Raleigh', 'North Dakota': 'Bismarck', 'Ohio': 'Columbus', 'Oklahoma': 'Oklahoma City', 'Oregon': 'Salem', 'Pennsylvania': 'Harrisburg', 'Rhode Island': 'Providence', 'South Carolina': 'Columbia', 'South Dakota': 'Pierre', 'Tennessee': 'Nashville', 'Texas': 'Austin', 'Utah': 'Salt Lake City', 'Vermont': 'Montpelier', 'Virginia': 'Richmond', 'Washington': 'Olympia', 'West Virginia': 'Charleston', 'Wisconsin': 'Madison', 'Wyoming': 'Cheyenne'}

# Generate 35 quiz files.

for quizNum in range(35):

# Create the quiz and answer key files.

quizFile = open('quiz\_files/capitalsquiz%s.txt' % (quizNum + 1), 'w')

answerKeyFile = open('answer\_files/capitalsquiz\_answers%s.txt' %

(quizNum + 1), 'w')

# Write out the header for the quiz.

quizFile.write('Name:\n\nDate:\n\nPeriod:\n\n')

quizFile.write((' ' \* 20) + 'State Capitals Quiz (Form %s)' % (quizNum + 1))

quizFile.write('\n\n')

# Shuffle the order of the states.

states = list(capitals.keys()) # get all states in a list

random.shuffle(states) # randomize the order of the states

# Loop through all 50 states, making a question for each.

for num in range(50):

# Get right and wrong answers.

correctAnswer = capitals[states[num]]

# returns the capital of a state

wrongAnswers = list(capitals.values())

# get a complete list of capitals

del wrongAnswers[wrongAnswers.index(correctAnswer)]

# remove the right answer

wrongAnswers = random.sample(wrongAnswers, 3)

# pick 3 random capitals of states

answerOptions = wrongAnswers + [correctAnswer]

random.shuffle(answerOptions) # randomize the order of the answers

# Write the question and answer options to the quiz file.

quizFile.write('%s. What is the capital of %s?\n' % (num + 1,

states[num]))

for i in range(4):

quizFile.write(' %s. %s\n' % ('ABCD'[i], answerOptions[i]))

quizFile.write('\n')

# Write out the answer key to a file.

answerKeyFile.write('%s. %s\n' % (num + 1,

'ABCD'[answerOptions.index(correctAnswer)]))

quizFile.close()

answerKeyFile.close()

# Regex Search

# -----------------------| Regex Search, 1st approach |------------------------|

import re, os

regex = re.compile(r'Panda', re.I)

files = os.chdir('C:\py\others\Chapter 8')

results = os.listdir()

for res in results:

files\_found = os.path.splitext(res)

if files\_found[1] == '.txt':

with open(files\_found[0]+files\_found[1], 'r') as file:

content = file.read()

res = regex.findall(content)

for r in res:

print(files\_found[0] +' file contains', res[0])

# -----------------------| Regex Search, 2nd approach|-------------------------|

"""Search for a regex pattern in all txt files in a directory."""

import os, re

files = os.listdir(os.path.join(os.getcwd()))

txt\_regex = re.compile(r'.txt')

docs = []

for file in files:

if txt\_regex.search(file) is not None:

docs.append(file)

user\_val = input('What are you looking for?\n')

regex = re.compile(r'{}'.format(user\_val))

for doc in docs:

file = open(doc)

content = file.read()

results = regex.findall(content)

results = ''.join(results)

print(doc , ' contains', results)

# Multiclipboard

import pyperclip, sys, shelve

shelve\_mod = open('mcb')

if len(sys.argv) == 3 and sys.argv[1] == 'save':

shelve\_mod[sys.argv[2]] = pyperclip.paste()

elif len(sys.argv) == 2:

if sys.argv[2] == 'list':

pyperclip.copy(list(shelve\_mod.keys()))

else:

pyperclip.copy(list(shelve\_mod[sys.argv[2]]))

# Mad Libs

import os, re

filename = input('What is the name of the file?\n')

file = open('{}/{}.txt'.format(os.getcwd(), filename))

string = file.read()

regex = re.compile(r'(NOUN|ADJECTIVE|VERB)')

matches = regex.findall(string)

for match in matches:

val = input('Enter a/an {}:\n'.format(match))

string = string.replace(match, val, 1)

file.close()

file = open('{}/{}.txt'.format(os.getcwd(), filename), 'w')

string = file.write(string)

file.close()

# Renaming Files with American-Style Date to European-Style Dates

import re, os, shutil

usa\_date\_for = re.compile(r'''^(.\*?)

((0|1)?\d)-

((0|1|2|3)?\d)-

((19|20)\d{2})

(.\*?)$

''', re.VERBOSE)

for usa\_file\_name in os.listdir('.'):

matched = usa\_date\_for.search(usa\_file\_name)

if matched is None:

continue

before\_date = matched.group(1)

monthpart = matched.group(2)

daypart = matched.group(4)

yearpart = matched.group(6) # group(6) not group(5)

after\_date = matched.group(8)

curr\_dir = os.path.abspath('.')

eur\_format = f'{before\_date}\_{daypart}-{monthpart}-{yearpart}\_{after\_date}'

oldfile = os.path.join(curr\_dir, usa\_file\_name)

newfiles= os.path.join(curr\_dir, eur\_format)

shutil.move(oldfile, newfiles)

# Raising Exceptions

def boxPrint(symbol, width, height):

if len(symbol) != 1:

raise Exception('Symbol must be a single character string.')

if width <= 2:

raise Exception('Width must be greater than 2.')

if height <= 2:

raise Exception('Height must be greater than 2.')

print(symbol \* width)

for i in range(height - 2):

print(symbol + (' ' \* (width - 2)) + symbol)

print(symbol \* width)

for sym, w, h in (('\*', 4, 4), ('O', 20, 5), ('x', 1, 3), ('ZZ', 3, 3)):

try:

boxPrint(sym, w, h)

except Exception as err:

print('An exception happened: ' + str(err))

# \*\*\*\*

# \* \*

# \* \*

# \*\*\*\*

# OOOOOOOOOOOOOOOOOOOO

# O O

# O O

# O O

# OOOOOOOOOOOOOOOOOOOO

# An exception happened: Width must be greater than 2.

# An exception happened: Symbol must be a single character string.