Họ và tên: Nguyễn Phước Khải

MSSV: B2207531

1. ***Lab 1***

# 1. Hoán đổi 2 ký tự đầu tiên của 2 chuỗi và nối chúng lại

def cau1(str1, str2):

    lst1 = list(str1)

    lst2 = list(str2)

    temp = lst1[0:2]

    lst1[0:2] = lst2[0:2]

    lst2[0:2] = temp

    return ''.join(lst1)+ ' ' + ''.join(lst2)

print(cau1('abc', 'efg'))

Kết quả: efc abg

# 2. Loại bỏ ký tự tại vị trí chẵn trong chuỗi

def cau2(string):

    return string[1::2]

print(cau2('python'))

Kết quả: yhn

# 3. Đếm số lần xuất hiện của các từ trong câu

from collections import Counter

def cau3(string):

    lst = string.split(' ')

    counter = Counter(lst)

    return dict(counter)

print(cau3('no pain no gain'))

Kết quả: {'no': 2, 'pain': 1, 'gain': 1}

# 4. Mã hóa chuỗi bằng cách dịch trái các ký tự 3 bước

def cau4(ciphertext, shift = 3):

    deciphered\_text = ""

    for char in ciphertext:

        # Check if the character is an uppercase letter

        if char.isupper():

            deciphered\_text += chr((ord(char) - shift - 65) % 26 + 65)

        # Check if the character is a lowercase letter

        elif char.islower():

            deciphered\_text += chr((ord(char) - shift - 97) % 26 + 97)

        else:

            # If it's not a letter, don't change it

            deciphered\_text += char

    return deciphered\_text

print(cau4('def'))

Kết quả: abc

# 5. Kiểm tra chuỗi có hợp lệ không (có được sinh từ bộ ký tự cho trước)

def cau5(string, inputS):

    for c in string:

        if c not in inputS:

            return False

    return True

print(cau5('0011', {'0', '1', '2'}))

print(cau5('123', {'0', '1', '2'}))

Kết quả:

True

False

# 6. Chuyển chuỗi thành danh sách các từ

def cau6(string):

    return string.split(' ')

print(cau6('This is a list'))

Kết quả: ['This', 'is', 'a', 'list']

# 7. In ra ký tự không lặp lại đầu tiên trong chuỗi

def cau7(string):

    counter = Counter(string)

    for k, v in counter.items():

        if v == 1:

            return k

    return None

print(cau7('abcdef'))

print(cau7('abcabcdef'))

print(cau7('aabbbcc'))

Kết quả:

a

d

None

# 8. Loại bỏ khoảng trắng trong chuỗi

def cau8(string):

    return string.replace(' ', '')

print(cau8('a b c'))

Kết quả: abc

# 9. In ra từ được lặp lại đầu tiên trong chuỗi

def cau9(string):

    words = string.split()

    seen\_words = set()

    for word in words:

        if word in seen\_words:

            return word

        seen\_words.add(word)

    return None

print(cau9('ab ca bc ca ab bc'))

print(cau9('ab ca bc ab'))

Kết quả:

ca

ab

# 10. Tìm độ dài tối đa của chuỗi con gồm các số 0 liên tiếp

def cau10(binary\_string):

    # Tách chuỗi nhị phân dựa trên '1' để lấy các chuỗi con toàn là '0'

    zero\_groups = binary\_string.split('1')

    # Tìm chuỗi '0' có độ dài lớn nhất

    max\_zeros = max(len(group) for group in zero\_groups)

    return max\_zeros

print(cau10('1110001110000'))

Kết quả: 4

#BT nangcao 1: Kiểm tra 2 danh sách khác nhau có thể sinh ra cùng một chuỗi hay không?

def cau1NangCao(lstA, lstB):

    permA = permutations(lstA)

    permB = permutations(lstB)

    setA = {''.join(x) for x in permA}

    setB = {''.join(x) for x in permB}

    if setA & setB:

        return True

    else:

        return False

print(cau1NangCao(['110', '0011', '0110'], ['110110', '00', '110']))

print(cau1NangCao(['0011', '11', '1101'], ['101', '011', '110']))

print(cau1NangCao(['100', '0', '1'], ['1', '100', '0']))

Kết quả:

True

False

True

#BT nangcao 2

def cau2NangCao(genString):

    for c in genString:

        if c not in {'A', 'C', 'G', 'T'}:

            return False

    if len(genString) % 3 != 0:

        return False

    else:

        start\_codon = 'ATG'

        end\_codon = {'TAA', 'TAG', 'TGA'}

        condonList = split\_into\_chunks(genString, 3)

        if condonList[0] == start\_codon and condonList[-1] in end\_codon:

            for condon in condonList[1:-1]:

                    if len(set(condon)) == 1:

                        return False

            return True

        else:

            return False

print(cau2NangCao('ATGCCCTAG'))

print(cau2NangCao('ATGCGTTGA'))

Kết quả:

False

True

1. ***Lab 2***

#Câu 1: Chỉnh sửa chương trình cho phép nhận DFA từ file hoặc do người dùng nhập vào, và

#kiểm tra một chuỗi nhập từ bàn phím có thuộc DFA đã cho hay không?

# Xây dựng DFA

class DFA(object):

    def \_\_init\_\_(self, states, alphabet, transition\_function, start\_state, accept\_states):

        self.states = states

        self.alphabet = alphabet

        self.transition\_function = transition\_function

        self.start\_state = start\_state

        self.accept\_states = accept\_states

        self.current\_state = start\_state

    def transition\_to\_state\_with\_input(self, input\_value):

        if ((self.current\_state, input\_value) not in self.transition\_function.keys()):

            self.current\_state = None

            return

        self.current\_state = self.transition\_function[(self.current\_state, input\_value)]

        return

    def in\_accept\_state(self):

        return self.current\_state in self.accept\_states

    def go\_to\_initial\_state(self):

        self.current\_state = self.start\_state

        return

    def run\_with\_input\_list(self, input\_list):

        self.go\_to\_initial\_state()

        for inp in input\_list:

            self.transition\_to\_state\_with\_input(inp)

        return self.in\_accept\_state()

def dfa\_file(filename):

    with open(filename, 'r') as f:

        L = f.readlines()

        states = set(map(int, L[0].split(',')))

        alphabet = set(L[1].strip().split(','))

        start\_state = int(L[2])

        accept\_states = set(map(int, L[3].split(',')))

        tf = {}

        for line in L[4:]:

            part = line.strip().split(',')

            state\_from = int(part[0])

            char = part[1]

            state\_to = int(part[2])

            tf[(state\_from, char)] = state\_to

        return DFA(states, alphabet, tf, start\_state, accept\_states)

L = list('1011101')

dfa1 = dfa\_file(r"./DFA\_sample.txt")

dfa1.go\_to\_initial\_state()

print(dfa1.run\_with\_input\_list(L))

file DFA\_sample.txt

0,1,2

0,1

0

0

0,0,1

0,1,1

1,0,2

1,1,0

2,0,1

2,1,2

Kết quả: True

#Câu 2:Dựa trên DFA đã xây dựng, mở rộng xây dựng lớp NFA và kiểm tra 1 chuỗi có thuộc ngôn ngữ sinh bởi NFA đã cho

# Xây dựng NFA

class NFA():

    def \_\_init\_\_(self, states, alphabet, transition\_function, start\_state, accept\_states):

        self.states = states

        self.alphabet = alphabet

        self.transition\_function = transition\_function

        self.start\_state = start\_state

        self.accept\_states = accept\_states

        self.current\_state:list = self.start\_state

    def transition\_to\_state\_with\_input(self, input\_value):

        print("state: ", self.current\_state)

        temp\_state = []

        for c\_state in self.current\_state:

            if (c\_state, input\_value) in self.transition\_function.keys():

                state = self.transition\_function[(c\_state, input\_value)]

                if isinstance(state, (int)):

                    if state not in temp\_state:

                        temp\_state.append(state)

                else:

                    temp\_state += state

        if len(temp\_state) == 0:

            self.current\_state = []

        else:

            self.current\_state = temp\_state.copy()

        return

    def in\_accept\_state(self):

        for c\_state in self.current\_state:

            if c\_state in self.accept\_states:

                return True

        return False

    def go\_to\_initial\_state(self):

        self.current\_state = self.start\_state

        return

    def run\_with\_input\_list(self, input\_list):

        self.go\_to\_initial\_state()

        for inp in input\_list:

            self.transition\_to\_state\_with\_input(inp)

        return self.in\_accept\_state()

states = {0, 1, 2, 3 ,4}

alphabet = {'0', '1'}

start\_state = [0]

accept\_states = {2, 4}

tf = dict()

tf[(0, '0')] = [0, 3]

tf[(0, '1')] = [0, 1]

tf[(1, '1')] = 2

tf[(2, '0')] = 2

tf[(2, '1')] = 2

tf[(3, '0')] = 4

tf[(4, '0')] = 4

tf[(4, '1')] = 4

L = list('10111011')

nfa1 = NFA(states, alphabet, tf, start\_state, accept\_states)

print(nfa1.run\_with\_input\_list(L))

1. ***Lab 3***

import re

#file test.txt

this is a test with re

This line has exactly twenty char

this should meet the required minimum length?!

Do you agree?!

Some random words a, r, s, m, l appear here.

No commas or periods here just words

A mouse is in the house

aaaaabbbb test

ambition arises amidst amazing abundance

myemail@gmail.com

hello@domain.com, user@yahoo.com, support@outlook.com

#Cau A

#Cau A1: Các dòng bắt đầu bằng ‘t’ hoặc ‘h’, và có chứa ‘re’ (sử dụng phương thức re.match())

with (open('test.txt') as file):

    for line in file.readlines():

        line = line.strip()

        # 1)

        result = re.match(r'^[th].\*re', line)

        if result:

            print(line)

Kết quả:

this is a test with re

this should meet the required minimum length?!

#Cau A2: Các dòng có chiều dài tối thiểu 20 kí tự

with (open('test.txt') as file):

    for line in file.readlines():

        line = line.strip()

        # 2)

        result = re.match(r'.{20,}', line)

        if result:

            print(line)

Kết quả

this is a test with re

This line has exactly twenty char

this should meet the required minimum length?!

Some random words a, r, s, m, l appear here.

No commas or periods here just words

A mouse is in the house

ambition arises amidst amazing abundance

hello@domain.com, user@yahoo.com, support@outlook.com

#Cau A3: Các dòng kết thúc bởi cặp dấu ‘?!’

with (open('test.txt') as file):

    for line in file.readlines():

        line = line.strip()

        # 3)

        result = re.search(r'\?!$', line)

        if result:

            print(line)

Kết quả

this should meet the required minimum length?!

Do you agree?!

#Cau A4: Các dòng chứa những ký tự: a, r, s, m, l (không cần liên tục)

with (open('test.txt') as file):

    for line in file.readlines():

        line = line.strip()

        # 4)

        result = re.search(r'^(?=.\*a)(?=.\*r)(?=.\*s)(?=.\*m)(?=.\*l).\*', line)

        if result:

            print(line)

Kết quả

Some random words a, r, s, m, l appear here.

hello@domain.com, user@yahoo.com, support@outlook.com

#Cau A5: Nội dung file ko chứa các dấu ‘,’ và ‘.’

with (open('test.txt') as file):

    for line in file.readlines():

        line = line.strip()

        # 5)

        result = re.search(r'[,.]', line)

        if not result:

            print(line)

Kết quả:

this is a test with re

This line has exactly twenty char

this should meet the required minimum length?!

Do you agree?!

No commas or periods here just words

A mouse is in the house

aaaaabbbb test

ambition arises amidst amazing abundance

#Cau A6: Các dòng có chứa chữ “mouse”

with (open('test.txt') as file):

    for line in file.readlines():

        line = line.strip()

        # 6)

        result = re.search(r'mouse', line)

        if result:

            print(line)

Kết quả:

A mouse is in the house

#Cau A7: Các từ có số chữ ‘a’ bất kỳ và theo sau bởi ‘b’

with (open('test.txt') as file):

    for line in file.readlines():

        line = line.strip()

        # 7)

        result = re.search(r'a+b', line)

        if result:

            print(line)  
Kết quả:

aaaaabbbb test

ambition arises amidst amazing abundance

#Cau A8: Domain địa chỉ mail (ví dụ: abc@gmail.com, in gmail.com )

with (open('test.txt') as file):

    for line in file.readlines():

        line = line.strip()

        # 8)

        result = re.search(r'@([\w.-]+\.[a-zA-Z]{2,})\b', line)

        if result:

            print(line)

Kết quả

myemail@gmail.com

hello@domain.com, user@yahoo.com, support@outlook.com

#Cau A9: Nội dung giữa cặp tag <head> </head>

# 9)

test\_str = "<head>This is a sample text within head tags.</head>"

result = re.search(r'<head>(.\*?)</head>', test\_str)

print(result.group(1))

#Cau B

import re

def is\_valid\_sentence(sentence):

    # 1) Câu được bắt đầu bằng ký tự in hoa, theo sau bởi ký tự thường

    if not re.search(r"^[A-Z][a-z]", sentence):

        return False

    # 2) Kết thúc bằng dấu chấm hoặc sau một ký tự in hoa

    if not re.search(r"[A-Za-z]\.$|[A-Z]$", sentence):

        return False

    # 3) Các từ cách nhau bằng một khoảng trắng, không chấp nhận nhiều hơn 1 khoảng trắng liên tiếp

    if re.search(r"\s{2,}", sentence):

        return False

    # 4) Không tồn tại hai ký tự liên tiếp viết hoa

    if re.search(r"[A-Z]{2,}", sentence):

        return False

    return True

# Ví dụ sử dụng:

sentences = [

    "Hello world.",

    "bat dau bang ky tu in thuong.",

    "This is a Test sentence.",

    "Ket thuc la mot ky tu in hoA",

    "Incorrect  sentence due to  extra spaces.",

    "AnotherIncorrectSentence",

    "Ends with period.",

    "NO TWO uppercase letters"

]

for sentence in sentences:

    print(f"\"{sentence}\": {is\_valid\_sentence(sentence)}")

Kết quả:

"Hello world.": True

"bat dau bang ky tu in thuong.": False

"This is a Test sentence.": True

"Ket thuc la mot ky tu in hoA": True

"Incorrect sentence due to extra spaces.": False

"AnotherIncorrectSentence": False

"Ends with period.": True

"NO TWO uppercase letters": False

#Cau C

import re

def process\_text(text):

    words = text.split()

    # Các từ thỏa mãn từng điều kiện

    results = {

        'contains\_a\_and\_digit': [],

        'contains\_a\_followed\_by\_b': [],

        'starts\_with\_a\_and\_ends\_with\_b': [],

        'contains\_only\_a\_to\_z\_and\_underscore': [],

        'length\_is\_5': [],

        'contains\_h': [],

        'starts\_with\_digit': [],

        'contains\_underscore': [],

        'date\_format\_conversion': []

    }

    for word in words:

        # 1. Các từ có chứa các ký tự thường ‘a-z’ và số từ ‘0-9’

        if re.search(r'[a-z].\*\d|[0-9].\*[a-z]', word):

            results['contains\_a\_and\_digit'].append(word)

        # 2. Các từ có chứa ký tự ‘a’ theo sau bởi b (b xuất hiện ít nhất 0 lần)

        if re.search(r'a\*b', word):

            results['contains\_a\_followed\_by\_b'].append(word)

        # 3. Các từ bắt đầu bằng ‘a’, theo sau là ký tự bất kỳ và kết thúc bằng ‘b’

        if re.match(r'^a.\*b$', word):

            results['starts\_with\_a\_and\_ends\_with\_b'].append(word)

        # 4. Các từ chỉ chứa ký tự thường ‘a-z’ và ‘\_’

        if re.match(r'^[a-z\_]+$', word):

            results['contains\_only\_a\_to\_z\_and\_underscore'].append(word)

        # 5. Các từ có chiều dài là 5

        if re.match(r'\b\w{5}\b', word):

            results['length\_is\_5'].append(word)

        # 6. Các từ có chứa ký tự ‘h’

        if 'h' in word:

            results['contains\_h'].append(word)

        # 7. Các từ bắt đầu là số từ ‘0-9’

        if re.match(r'^\d', word):

            results['starts\_with\_digit'].append(word)

        # 8. Các từ có chứa dấu ‘\_’ và thay bằng khoảng trắng

        if '\_' in word:

            results['contains\_underscore'].append(word.replace('\_', ' '))

        # 9. Có chứa định dạng mm-dd-yy, và chuyển thành định dạng dd-mm-yy

        if re.match(r'\d{2}-\d{2}-\d{2}', word):

            new\_date = re.sub(r'(\d{2})-(\d{2})-(\d{2})', r'\2-\1-\3', word)

            results['date\_format\_conversion'].append(new\_date)

    return results

# Ví dụ sử dụng:

text = "abc123 abc bbb abb a\_z 12345 hello 11-06-24 a\_ b\_ h h\_ hello\_123"

result = process\_text(text)

# In kết quả

for key, value in result.items():

    print(f"{key}: {value}")

Kết quả:

contains\_a\_and\_digit: ['abc123', 'hello\_123']

contains\_a\_followed\_by\_b: ['abc123', 'abc', 'bbb', 'abb', 'b\_']

starts\_with\_a\_and\_ends\_with\_b: ['abb']

contains\_only\_a\_to\_z\_and\_underscore: ['abc', 'bbb', 'abb', 'a\_z', 'hello', 'a\_', 'b\_', 'h', 'h\_']

length\_is\_5: ['12345', 'hello']

contains\_h: ['hello', 'h', 'h\_', 'hello\_123']

starts\_with\_digit: ['12345', '11-06-24']

contains\_underscore: ['a z', 'a ', 'b ', 'h ', 'hello 123']

date\_format\_conversion: ['06-11-24']

#Cau D

#Cau D1: Hoán đổi 2 ký tự đầu tiên của 2 chuỗi và nối chúng lại

def hoandoi(str1, str2):

# Sử dụng re.sub để hoán đổi hai ký tự đầu tiên

return re.sub(r'^(..)', lambda m: str2[:2], str1) + " " + re.sub(r'^(..)', lambda m: str1[:2], str2)

print(hoandoi('abc', 'efg'))

#Cau D2: Loại bỏ ký tự tại vị trí chẵn trong chuỗi

def loaibokytuchan(s):

return re.sub(r'(.)', lambda m: m.group(1) if m.start() % 2 != 0 else '', s)

print(loaibokytuchan('python'))

#Cau D3: Đếm số lần xuất hiện của các từ trong câu

def demsotu(cau):

tu = re.findall(r'\b\w+\b', cau)

tu\_dem = {}

for word in tu:

tu\_dem[word] = tu\_dem.get(word, 0) + 1

return tu\_dem

print(demsotu('no pain no gain'))

#Cau D4: Mã hóa chuỗi bằng cách dịch trái các ký tự 3 bước

def dichchuyenCeasar(s):

return re.sub(r'[a-z]', lambda m: chr((ord(m.group(0)) - 97 - 3) % 26 + 97), s)

print(dichchuyenCeasar('def'))

#Cau D5: Kiểm tra chuỗi có hợp lệ không (có được sinh từ bộ ký tự cho trước)

def chuoihople(s, valid\_set):

valid\_characters = ''.join(valid\_set)

return bool(re.fullmatch(f'[{valid\_characters}]+', s))

print(chuoihople('0011', {'0', '1', '2'}))

print(chuoihople('123', {'0', '1', '2'}))

#Cau D6: Chuyển chuỗi thành danh sách các từ

def stringtolist(s):

return re.findall(r'\b\w+\b', s)

print(stringtolist('This is a list'))

#Cau D7: In ra ký tự không lặp lại đầu tiên trong chuỗi

from collections import Counter

def inkhonglap(s):

# Sử dụng biểu thức chính quy để tìm tất cả các ký tự

chars = re.findall(r'.', s)

# Đếm số lần xuất hiện của mỗi ký tự

char\_count = Counter(chars)

# Duyệt qua các ký tự và tìm ký tự không lặp lại đầu tiên

for char in chars:

if char\_count[char] == 1:

return char

return None

print(inkhonglap('abcdef'))

print(inkhonglap('abcabcdef'))

print(inkhonglap('aabbcc'))

#Cau D8: Loại bỏ khoảng trắng trong chuỗi

def loaibospace(s):

return s.replace(' ', '')

print(loaibospace('a b c'))

#Cau D9:

def inlap(s):

# Tìm tất cả các từ trong chuỗi

words = re.findall(r'\b\w+\b', s)

seen = set()

# Duyệt qua các từ và tìm từ lặp lại đầu tiên

for word in words:

if word in seen:

return word

seen.add(word)

return None

print(inlap('ab ca bc ca ab bc')) # Kết quả: 'ca'

print(inlap('ab ca bc ab')) # Kết quả: 'ab'

#Cau D10: Tìm độ dài tối đa của chuỗi con gồm các số 0 liên tiếp

def chuoiconso0(bin\_str):

zeros = re.findall(r'0+', bin\_str)

if zeros:

# Tính độ dài của chuỗi con dài nhất

return max(len(zeros\_str) for zeros\_str in zeros)

return 0

print(chuoiconso0('1110001110000'))