BÀI THỰC HÀNH SỐ 2: AUTOMATA HỮU HẠN

Mục tiêu: hướng dẫn xây dựng **Automata hữu hạn đơn định (DFA)** và **Automata hữu hạn không đơn định (NFA)**

1. Kiến thức:

- ***Mở file:***

f = open('text.txt', 'r')

f = open('./python/text.txt', 'r')

'r'	Read (mặc định)	Đọc file ở chế độ read-only
'w'	Write	Mở file và chỉ cho phép ghi đè nên file hiện tại (overwrite)
'a'	Append	Mở file chỉ cho thêm mới vào cuối fil (append)
'r+'	Read+Write	Mở file cho phép đọc và ghi.
'x'	Create	Tạo file mới

- ***Đọc file:***

```
f.read()        #đọc hết file
f.readlines()   #đọc hết file
f.readline()    #đọc 1 dòng
```

- ***Ví dụ:*** *nội dung file text.txt*

```
Hi all,
How are you?
What are you doing?
```

- ***Đọc, hiển thị tất cả các dòng trong file***

```
try:
        f = open('text.txt','r')
        print(f.read())
except Exception as e:
        print(e)
finally:
        f.close()
        print("File closed!")
# Kết quả:
Hi all,
How are you?
```

What are you doing?
File closed!

- ***Đọc, hiển thị tất cả các dòng trong file đưa vào danh sách, đọc cả ký tự xuống dòng:***

```
f = open('text.txt','r')
print(f.readlines())
f.close()
print("File closed!")
# Kết quả:
['Hi all,\n', 'How are you?\n', 'What are you doing?']
File closed!
```

- ***Đọc, hiển thị tất cả các dòng trong file đưa vào danh sách, đọc cả ký tự xuống dòng (***f.readlines()***, in dòng thứ 2*** *(*print(L[1]))

```
f = open('text.txt','r')
L =f.readlines()
print(L[1])  # in dòng thứ 2 trong file
f.close()
# Kết quả:
How are you?
```

- ***Đọc, hiển thị 1 dòng đầu tiên***

```
f = open('text.txt','r')
print(f.readline())
f.close()
print("File closed!")
# Kết quả:
Hi all,

File closed!
```

- ***Đọc, hiển thị các dòng trong file, bỏ qua dòng đầu tiên***

```
f = open('text.txt','r')
lines = f.readlines()[1:] #bỏ qua dòng đầu tiên
print(lines)
f.close()
```

- ***Đọc tất cả các dòng trong file đưa vào danh sách, đọc cả ký tự xuống dòng; mỗi dòng là 1 danh sách, mỗi phần tử trong danh sách là một ký tự.***

```
from itertools import islice
f = open('text.txt','r')
L =f.readlines()
```

```
f.close()
print(L)
for i in range(len(L)):
        line = list(islice(L[i],len(L[i])))
        print(line)
#kết quả:
['Hi all,\n', 'How are you?\n', 'What are you doing?']
['H', 'i', ' ', 'a', 'l', 'l', ',', '\n']
['H', 'o', 'w', ' ', 'a', 'r', 'e', ' ', 'y', 'o', 'u', '?', '\n']
['W', 'h', 'a', 't', ' ', 'a', 'r', 'e', ' ', 'y', 'o', 'u', ' ', 'd', 'o', 'i', 'n', 'g', '?']
```

- ***Tìm hiểu thêm phương thức islice***

```
from itertools import islice
line = list(islice("Hello World",len(L[i])))
print(line)
#kết quả:
['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd']
str1 = "Hello World"
line = list(islice(str1, 2, len(str1)-2, 2))
print(line)
#kết quả:
['l', 'o', 'W', 'r']
```

2. Hướng dẫn viết các hàm

☛ **Bài toán: xây dựng Automata hữu hạn đơn định (DFA) và kiểm tra 1 chuỗi có thuộc ngôn ngữ sinh bởi DFA đã cho**

Lớp DFA có các thuộc tính: trạng thái states, bộ chữ cái nhập alphabet, hàm chuyển trạng thái transition_function, trạng thái bắt đầu start_state, trạng thái kết thúc accept_state, trạng thái hiện tại current_state với hàm khởi tạo như sau:

```
Def __init__ (self, states, alphabet, transition_function, start_state, accept_states, current_state):
        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
        return
```

Khi đọc vào một ký tự, trạng thái hiện tại thay đổi qua phương thức: "transition_to_state_with_input":

```
def transition_to_state_with_input(self, input_value):
```

```
        #nếu không tồn tại đường đi từ trạng thái hiện tại trên input_value
        if ((self.current_state, input_value) not in self.transition_function.keys()):
            self.current_state = None
            return
        #tồn tại đường đi từ trạng thái hiện tại trên input_value
        self.current_state = self.transition_function[(self.current_state, input_value)]
        return
```

Kiểm tra trạng thái hiện tại có phải là trạng thái kết thúc:

```
def in_accept_state(self):
        return self.current_state in accept_states
```

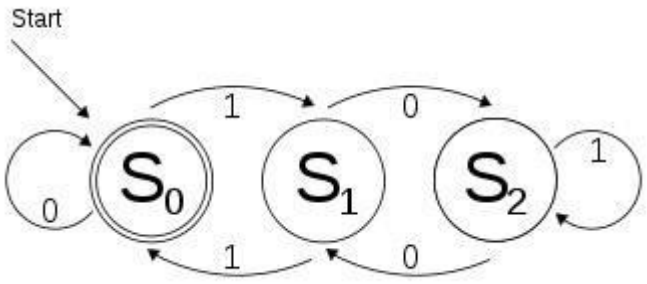
Phương thức quay về trạng thái bắt đầu:

```
def go_to_initial_state(self):
        self.current_state = self.start_state
        return
```

Để kiểm tra một chuỗi có thuộc ngôn ngữ sinh bởi automat đã cho, sử dụng phương thức:

```
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)
            continue
        return self.in_accept_state()
```

- **Cho DFA sau:**



- **Kiểm tra các chuỗi 1011101, 10111011 có thuộc ngôn ngữ sinh bởi DFA đã cho?**
- **Hướng dẫn**
  - Xây dựng lớp DFA với current_state = None và phương thức khởi tạo __init__như trên
  - Khởi tạo tập trạng thái states, bộ chữ cái nhập alphabet, trạng thái bắt đầu, và tập trạng thái kết thúc

states = {0, 1, 2}

alphabet = {'0', '1'}

start_state = 0

accept_states = {0}

- Xây dựng hàm chuyển trạng thái có kiểu dữ liệu tự điển tf = dict()

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

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

tf[(1,'0')] = 2

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

tf[(2,'0')] = 1

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

- Kiểm tra chuỗi nhập là một danh sách có dạng: list('10111011')

☛ **Chương trình:**

```
# xay dung DFA, NFA
class DFA(object):
    def __init__(self, states, alphabet, transition_function, start_state, accept_states, current_state):
        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
        return
    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 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)
            continue
        return self.in_accept_state()
```

```
states = {0, 1, 2}
alphabet = {'0', '1'}
start_state = 0
accept_states = {0}
tf = dict()
tf[(0,'0')] = 0
tf[(0,'1')] = 1
tf[(1,'0')] = 2
tf[(1,'1')] = 0
tf[(2,'0')] = 1
tf[(2,'1')] = 2
L = list('10111011')
current_state = None
dfa1 = DFA(states, alphabet, tf, start_state, accept_states, current_state)
print(dfa1.go_to_initial_state())
print(dfa1.run_with_input_list(L))
```

3. Bài tập

- ***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?***
- ***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***

Gợi ý: xét sự khác biệt của *current_state* giữa DFA và NFA