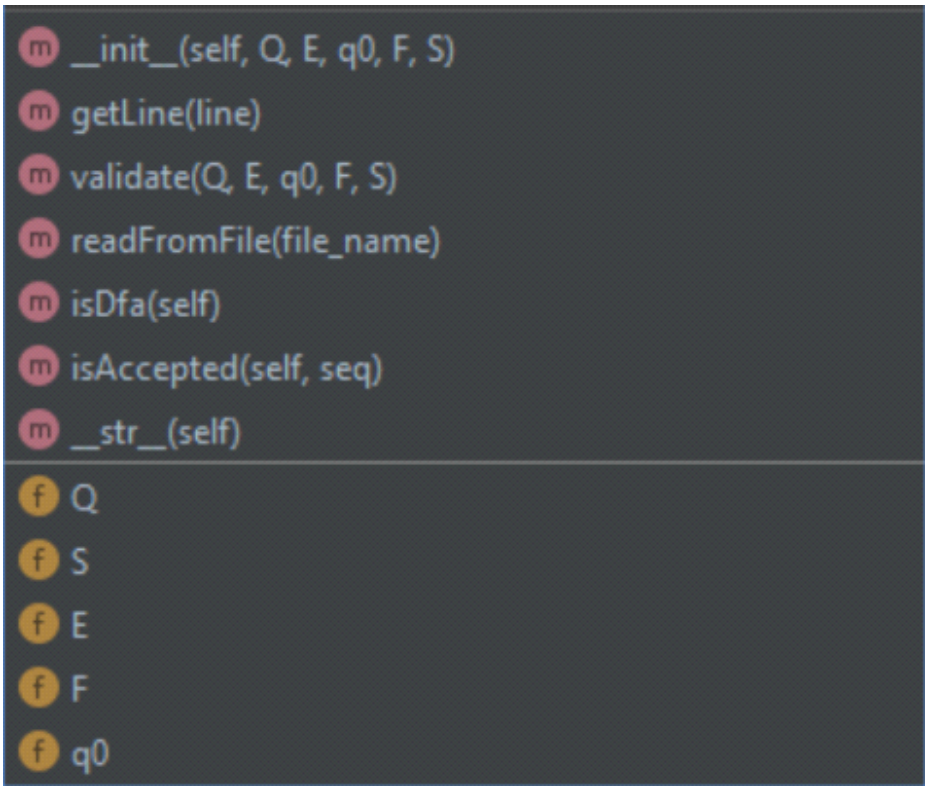
FA

Șorecău Adrian-Vasile – 937/1

https://github.com/cs-ubbcluj-ro/lab-work-computer-science-2024-sorecauadrian/tree/main/2-Finite-Automata

The Finite Automaton is structured as a class having:
self.Q = Q # a finite set of states
self.E = E # a finite set of input symbols called the alphabet
self.q0 = q0 # an initial or start state
self.F = F # a set of accept or final states
self.S = S # a set of state transitions represented as a HashMap



Checking DFA:

DFA refers to Deterministic Finite Automaton. A Finite Automata(FA) is said to be deterministic, if corresponding to an input

symbol, there is single resultant state i.e., there is only one transition. This means going through the dictionary keys and checking if any list has more than one element.

```
def isDfa(self):
    for k in self.S.keys():
        if len(self.S[k]) > 1:
            return False
    return True
```

Checking accepted sequence:

Iterating through symbols from the sequence and checking if it can be reached by following the transitions. Also, it must end in a final state.

```
def isAccepted(self, sequence):
    if self.isDfa():
        current = self.q0
        for symbol in sequence:
            if (current, symbol) in self.S.keys():
                current = self.S[(current, symbol)][0]
            else:
                return False
        return current in self.F
    return False
```

fa.in should be written like this:

```
<file>            ::= <states> <alphabet> <start_state> <final_states>
<transitions>

<states>          ::= "Q = " <state_list> "\n"
<alphabet>        ::= "E = " <symbol_list> "\n"
<start_state>     ::= "q0 = " <state> "\n"
<final_states>    ::= "F = " <state_list> "\n"
<transitions>     ::= "S = " "\n" <transition_list>

<state_list>      ::= <state> | <state> " " <state_list>
<symbol_list>     ::= <symbol> | <symbol> " " <symbol_list>
<transition_list>::= <transition> "\n" | <transition> "\n" <transition_list>

<transition>      ::= "\t(" <state> "," <symbol> ") -> " <state>

<state>           ::= <identifier>
<symbol>          ::= <character>
<identifier>      ::= <letter> | <letter> <identifier>
<character>       ::= <letter> | <digit>
<letter>          ::= "a" | "b" | ... | "z" | "A" | "B" | "C" | ... | "Z"
```

```
<digit>          ::= "0" | "1" | ... |"8" | "9"
```