Formal Languages and Compilers

Lab5

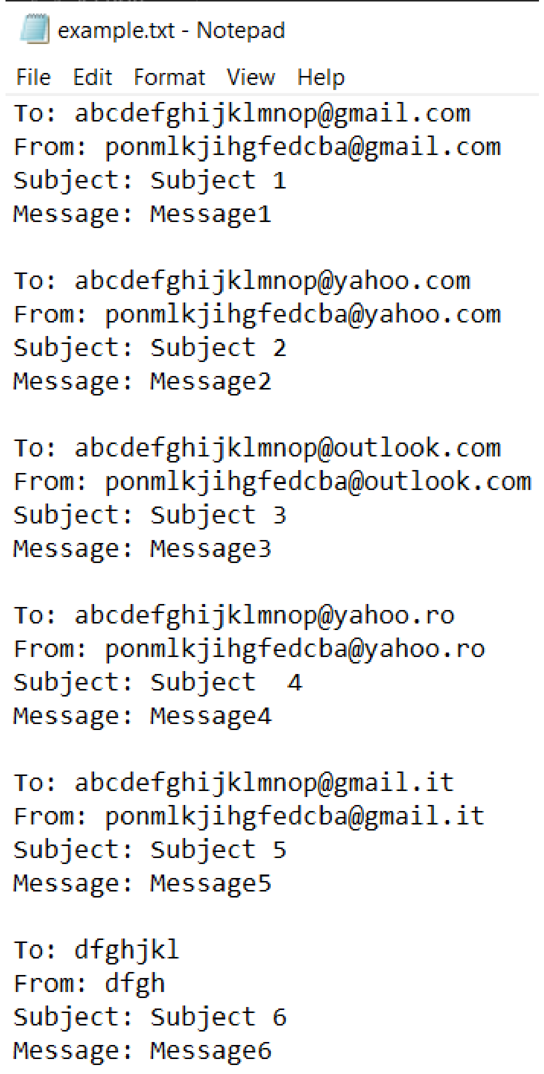
Regex in text files

- Regular expressions can be used to extract, change, analyze text from files
- Regular expressions can be used for web scrapping (data collection)
- NLP (Natural Language Processing) enables computers to understand natural language as humans do
  - Regular expressions can be used in the data preprocessing phase of NLP (preparing and "cleaning" text data for machines to be able to analyze it)

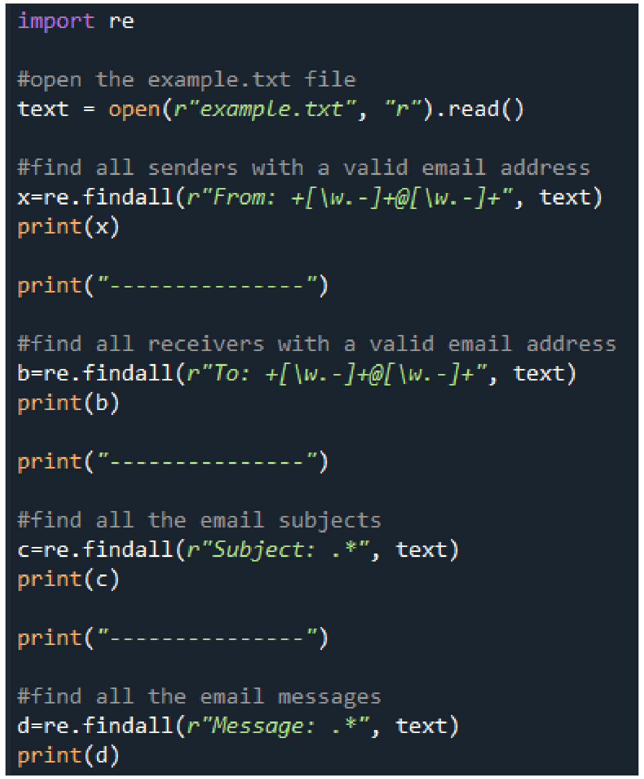
- pandas-library used for data analysis and manipulation

Examples

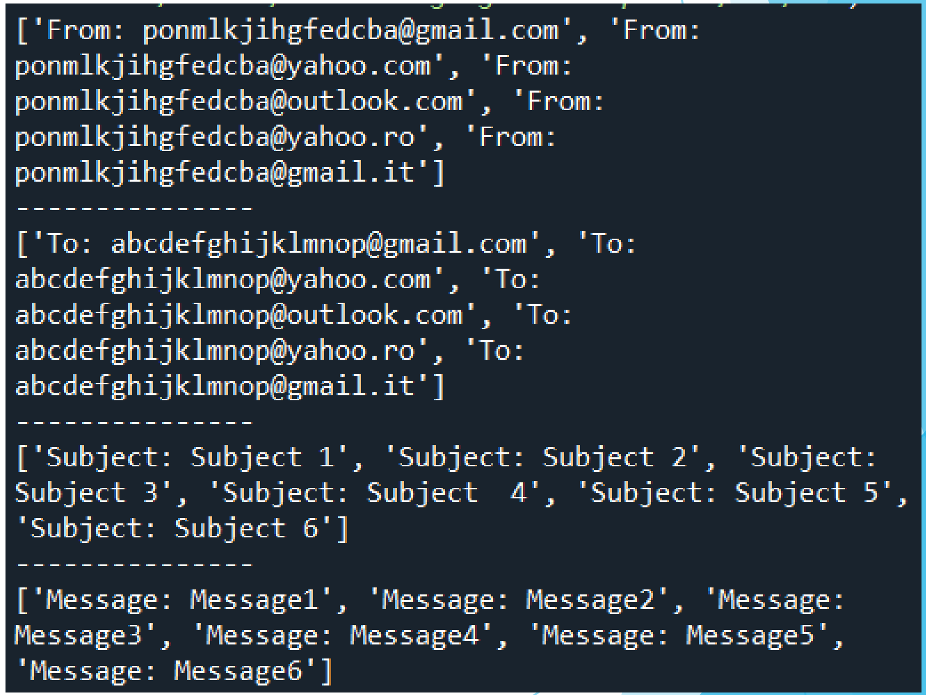
- Extract data from file



```
To: abcdefghijklmnop@gmail.com
From: ponmlkjihgfedcba@gmail.com
Subject: Subject 1
Message: Message1

To: abcdefghijklmnop@yahoo.com
From: ponmlkjihgfedcba@yahoo.com
Subject: Subject 2
Message: Message2

To: abcdefghijklmnop@outlook.com
From: ponmlkjihgfedcba@outlook.com
Subject: Subject 3
Message: Message3

To: abcdefghijklmnop@yahoo.ro
From: ponmlkjihgfedcba@yahoo.ro
Subject: Subject  4
Message: Message4

To: abcdefghijklmnop@gmail.it
From: ponmlkjihgfedcba@gmail.it
Subject: Subject 5
Message: Message5

To: dfghjkl
From: dfgh
Subject: Subject 6
Message: Message6
```



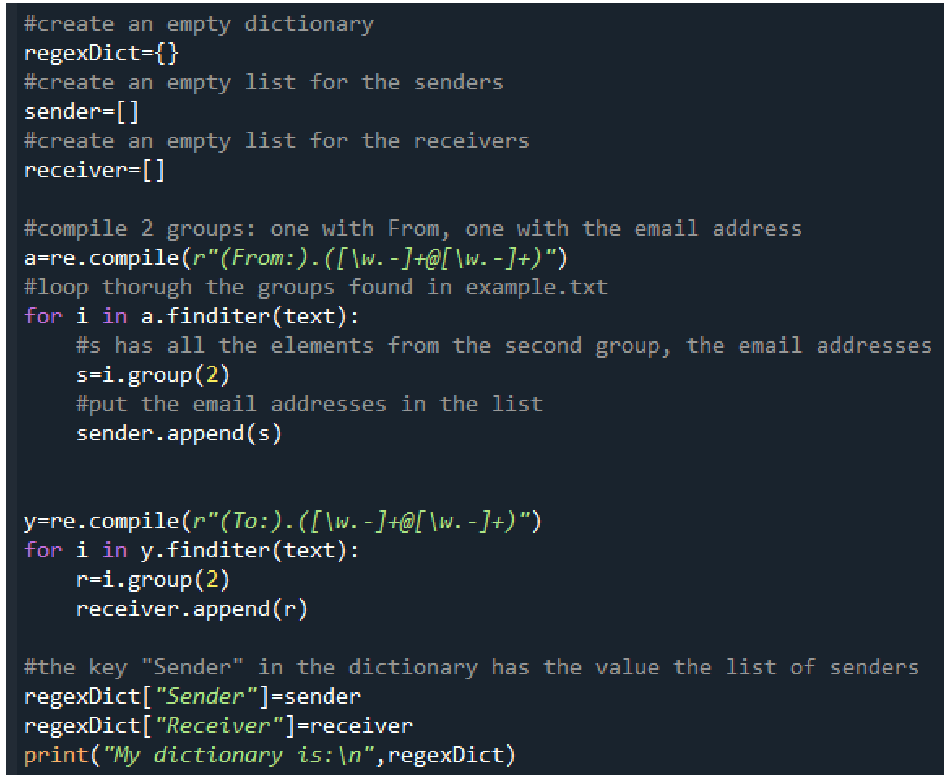
```python
import re

#open the example.txt file
text = open(r"example.txt", "r").read()

#find all senders with a valid email address
x=re.findall(r"From: +[\w.-]+@[\w.-]+", text)
print(x)

print("---------------")

#find all receivers with a valid email address
b=re.findall(r"To: +[\w.-]+@[\w.-]+", text)
print(b)

print("---------------")

#find all the email subjects
c=re.findall(r"Subject: .*", text)
print(c)

print("---------------")

#find all the email messages
d=re.findall(r"Message: .*", text)
print(d)
```



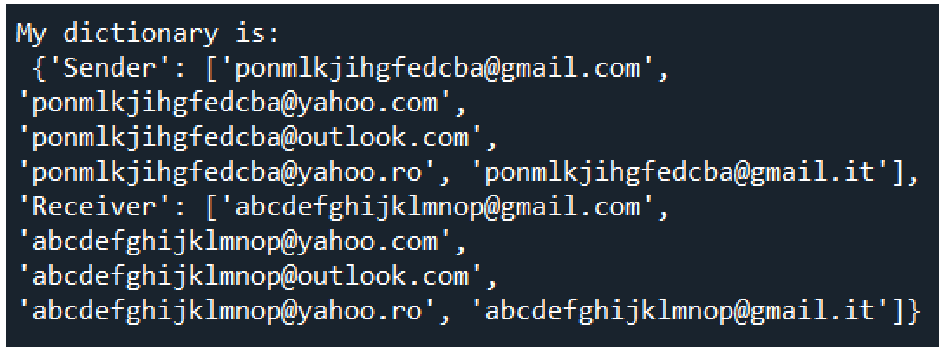
```
['From: ponmlkjihgfedcba@gmail.com', 'From:
ponmlkjihgfedcba@yahoo.com', 'From:
ponmlkjihgfedcba@outlook.com', 'From:
ponmlkjihgfedcba@yahoo.ro', 'From:
ponmlkjihgfedcba@gmail.it']
---------------
['To: abcdefghijklmnop@gmail.com', 'To:
abcdefghijklmnop@yahoo.com', 'To:
abcdefghijklmnop@outlook.com', 'To:
abcdefghijklmnop@yahoo.ro', 'To:
abcdefghijklmnop@gmail.it']
---------------
['Subject: Subject 1', 'Subject: Subject 2', 'Subject:
Subject 3', 'Subject: Subject  4', 'Subject: Subject 5',
'Subject: Subject 6']
---------------
['Message: Message1', 'Message: Message2', 'Message:
Message3', 'Message: Message4', 'Message: Message5',
'Message: Message6']
```

Examples

- Extract data from file



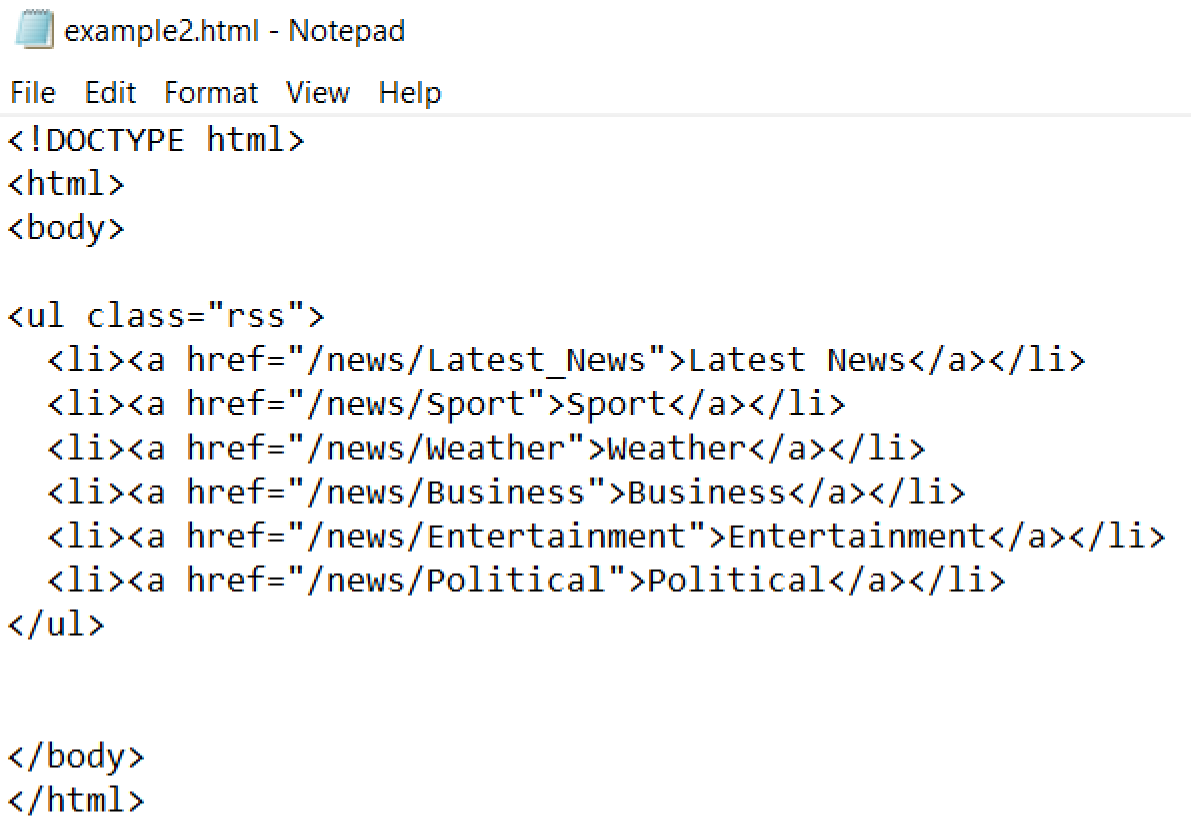
```python
#create an empty dictionary
regexDict={}
#create an empty list for the senders
sender=[]
#create an empty list for the receivers
receiver=[]

#compile 2 groups: one with From, one with the email address
a=re.compile(r"(From:).([\w.-]+@[\w.-]+)")
#loop thorugh the groups found in example.txt
for i in a.finditer(text):
    #s has all the elements from the second group, the email addresses
    s=i.group(2)
    #put the email addresses in the list
    sender.append(s)


y=re.compile(r"(To:).([\w.-]+@[\w.-]+)")
for i in y.finditer(text):
    r=i.group(2)
    receiver.append(r)

#the key "Sender" in the dictionary has the value the list of senders
regexDict["Sender"]=sender
regexDict["Receiver"]=receiver
print("My dictionary is:\n",regexDict)
```



```
My dictionary is:
 {'Sender': ['ponmlkjihgfedcba@gmail.com',
'ponmlkjihgfedcba@yahoo.com',
'ponmlkjihgfedcba@outlook.com',
'ponmlkjihgfedcba@yahoo.ro', 'ponmlkjihgfedcba@gmail.it'],
'Receiver': ['abcdefghijklmnop@gmail.com',
'abcdefghijklmnop@yahoo.com',
'abcdefghijklmnop@outlook.com',
'abcdefghijklmnop@yahoo.ro', 'abcdefghijklmnop@gmail.it']}
```

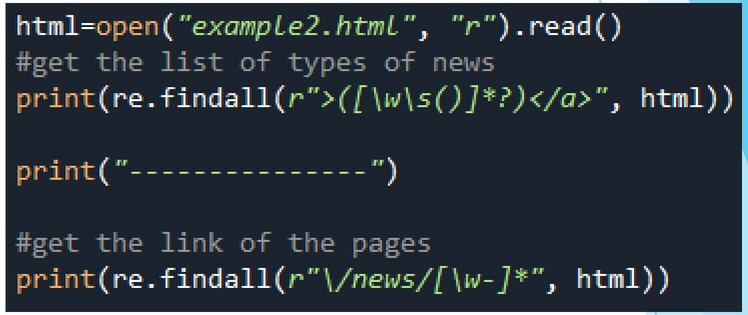
Examples

- Extract data from html files

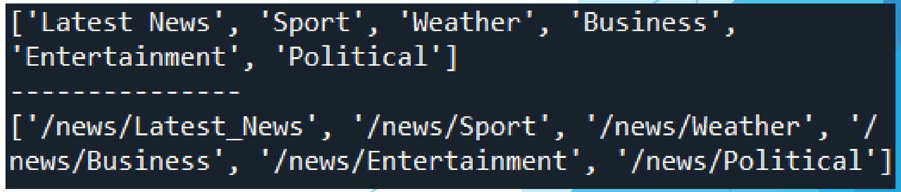

example2.html - Notepad

File Edit Format View Help

```html
<!DOCTYPE html>
<html>
<body>

<ul class="rss">
  <li><a href="/news/Latest_News">Latest News</a></li>
  <li><a href="/news/Sport">Sport</a></li>
  <li><a href="/news/Weather">Weather</a></li>
  <li><a href="/news/Business">Business</a></li>
  <li><a href="/news/Entertainment">Entertainment</a></li>
  <li><a href="/news/Political">Political</a></li>
</ul>


</body>
</html>
```



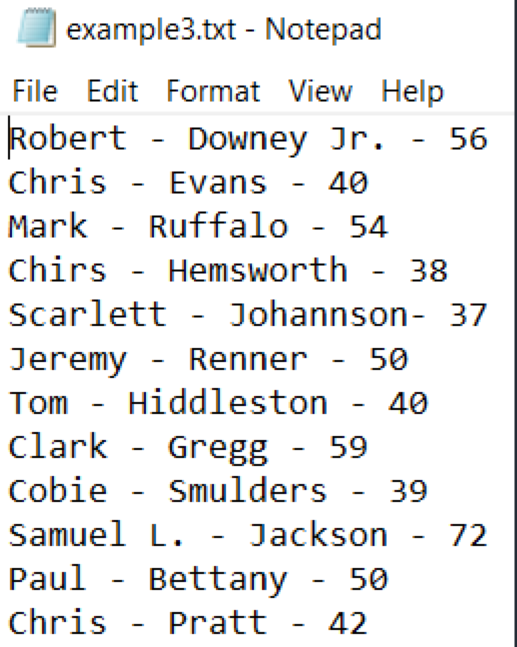
```python
html=open("example2.html", "r").read()
#get the list of types of news
print(re.findall(r">([\w\s()]*?)</a>", html))

print("---------------")

#get the link of the pages
print(re.findall(r"\/news/[\w-]*", html))
```



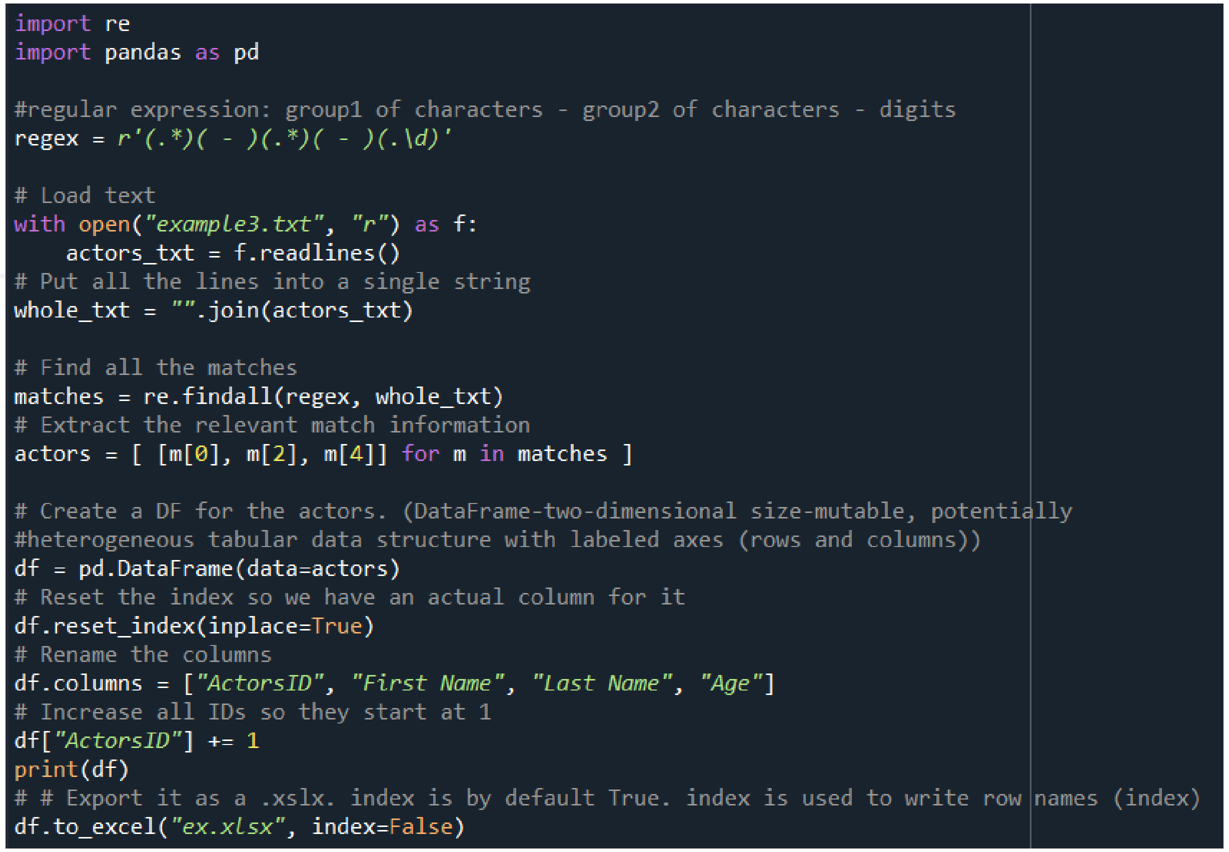
```
['Latest News', 'Sport', 'Weather', 'Business',
'Entertainment', 'Political']
---------------
['/news/Latest_News', '/news/Sport', '/news/Weather', '/
news/Business', '/news/Entertainment', '/news/Political']
```

Examples

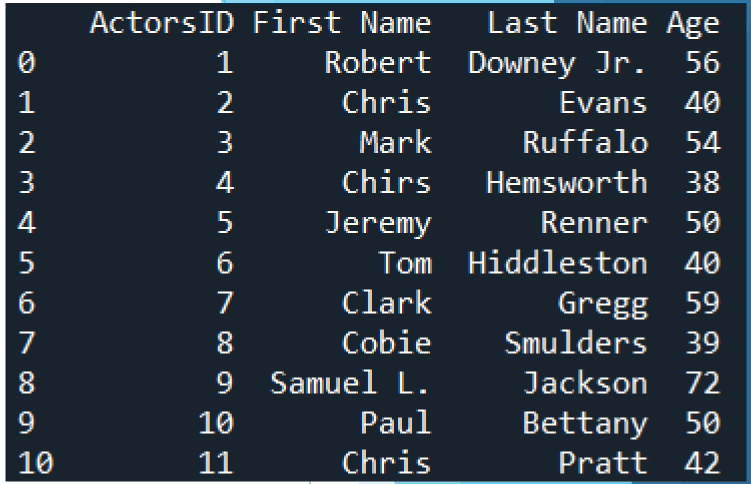
- pandas-create .xslx from .txt



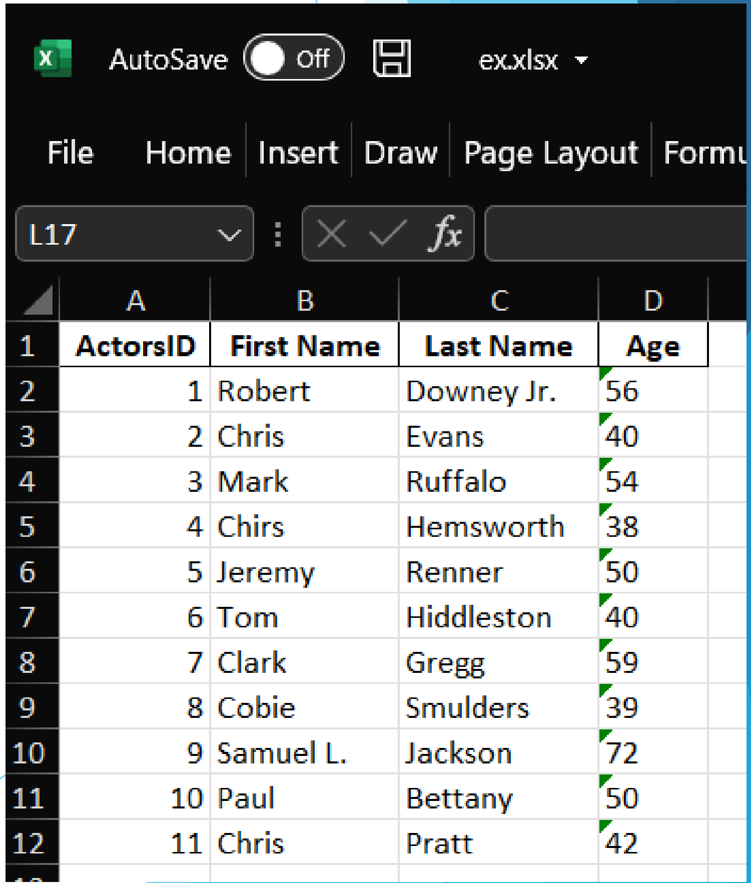
```
Robert - Downey Jr. - 56
Chris - Evans - 40
Mark - Ruffalo - 54
Chirs - Hemsworth - 38
Scarlett - Johannson- 37
Jeremy - Renner - 50
Tom - Hiddleston - 40
Clark - Gregg - 59
Cobie - Smulders - 39
Samuel L. - Jackson - 72
Paul - Bettany - 50
Chris - Pratt - 42
```



```python
import re
import pandas as pd

#regular expression: group1 of characters - group2 of characters - digits
regex = r'(.*)( - )(.*)( - )(.\d)'

# Load text
with open("example3.txt", "r") as f:
    actors_txt = f.readlines()
# Put all the lines into a single string
whole_txt = "".join(actors_txt)

# Find all the matches
matches = re.findall(regex, whole_txt)
# Extract the relevant match information
actors = [ [m[0], m[2], m[4]] for m in matches ]

# Create a DF for the actors. (DataFrame-two-dimensional size-mutable, potentially
#heterogeneous tabular data structure with labeled axes (rows and columns))
df = pd.DataFrame(data=actors)
# Reset the index so we have an actual column for it
df.reset_index(inplace=True)
# Rename the columns
df.columns = ["ActorsID", "First Name", "Last Name", "Age"]
# Increase all IDs so they start at 1
df["ActorsID"] += 1
print(df)
# # Export it as a .xslx. index is by default True. index is used to write row names (index)
df.to_excel("ex.xlsx", index=False)
```



```
    ActorsID First Name    Last Name Age
0          1     Robert  Downey Jr.  56
1          2      Chris        Evans  40
2          3       Mark      Ruffalo  54
3          4      Chirs    Hemsworth  38
4          5     Jeremy       Renner  50
5          6        Tom   Hiddleston  40
6          7      Clark        Gregg  59
7          8      Cobie     Smulders  39
8          9  Samuel L.      Jackson  72
9         10       Paul      Bettany  50
10        11      Chris        Pratt  42
```



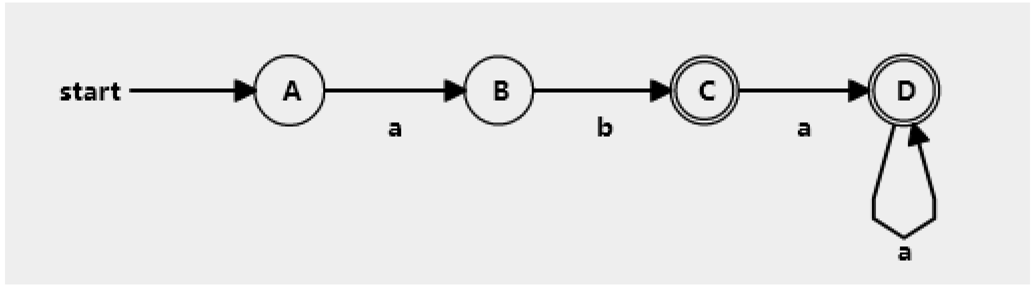
	A	B	C	D
1	ActorsID	First Name	Last Name	Age
2	1	Robert	Downey Jr.	56
3	2	Chris	Evans	40
4	3	Mark	Ruffalo	54
5	4	Chirs	Hemsworth	38
6	5	Jeremy	Renner	50
7	6	Tom	Hiddleston	40
8	7	Clark	Gregg	59
9	8	Cobie	Smulders	39
10	9	Samuel L.	Jackson	72
11	10	Paul	Bettany	50
12	11	Chris	Pratt	42

Finite automata

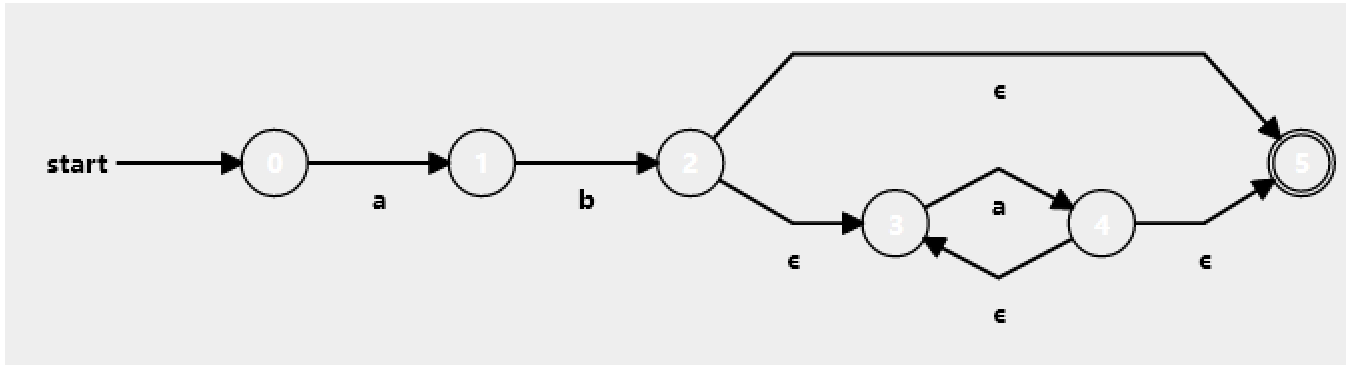
- It is the simplest machine to recognize patterns
- It takes the string of symbol as input and changes its state accordingly. When the desired symbol is found, then the transition occurs.
- At the time of transition, the automata can either move to the next state or stay in the same state.
- DFA (deterministic finite automata)-can exist only in one state at a time, cannot have null moves
- NDFA (non-deterministic finite automata) – can exist in multiple states at a time, can have null moves
- The language accepted by finite automata can be easily described by **regular expressions**

Example

- Regex: aba+
- DFA

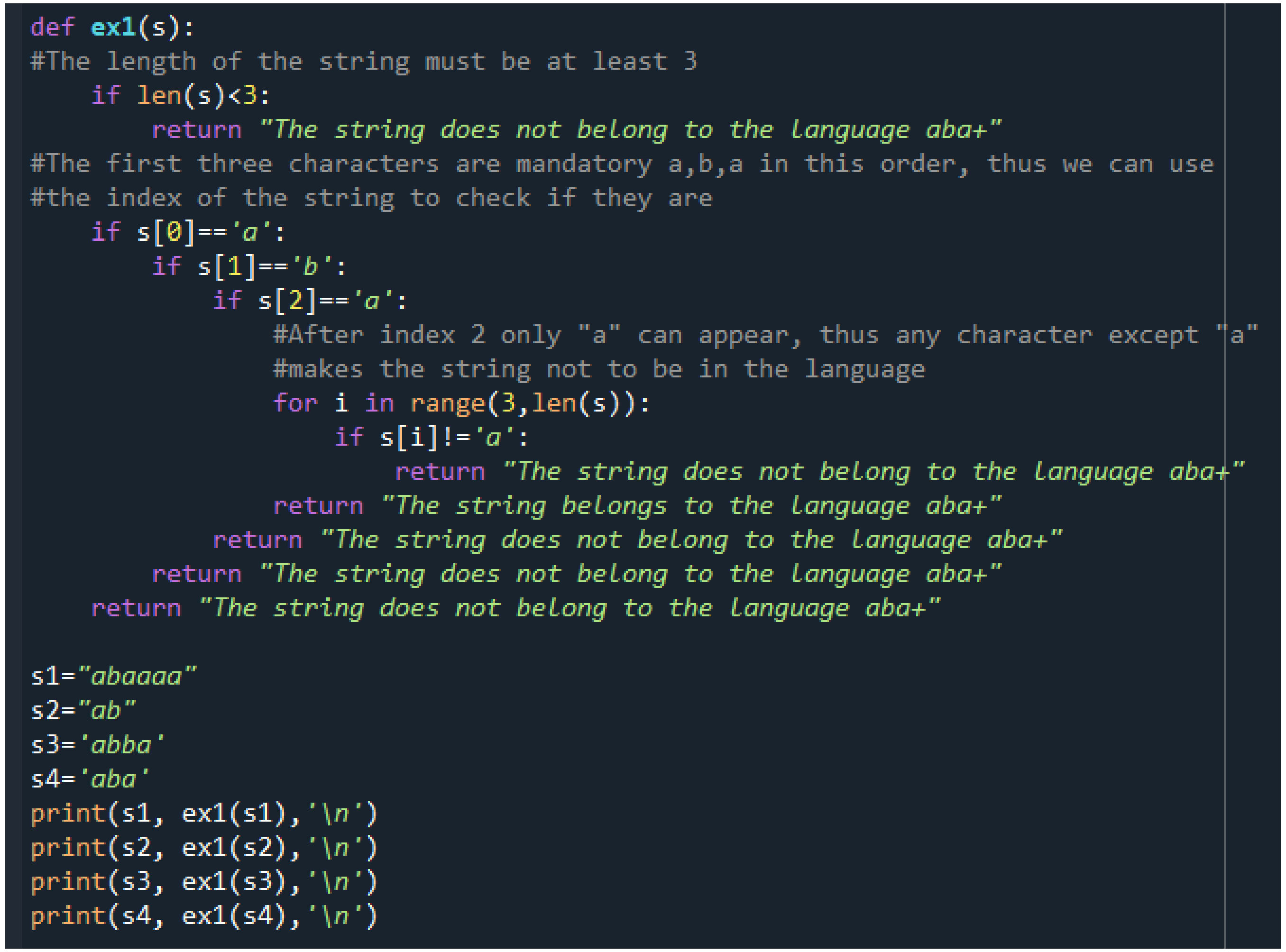


- NDFA

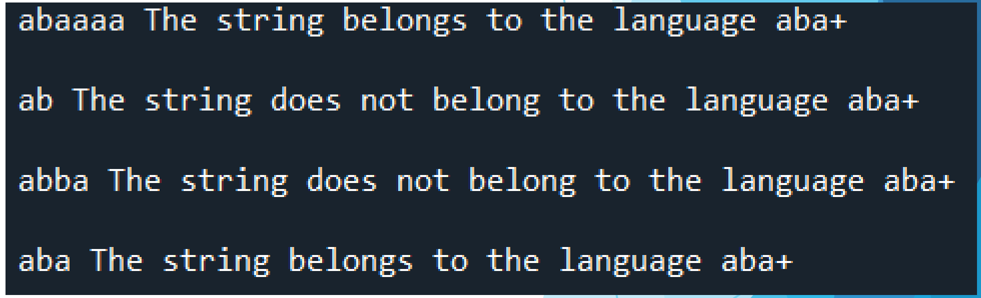


Example

- Using if statements



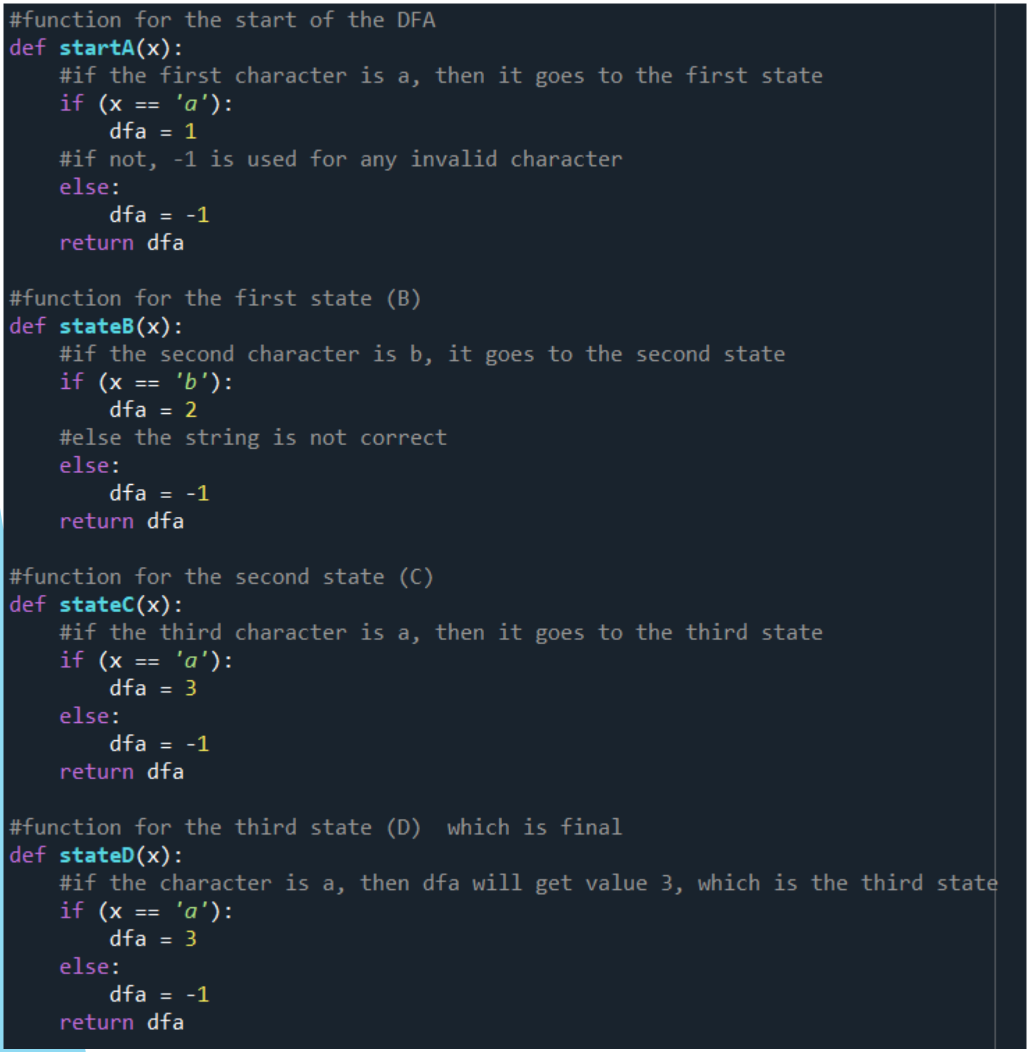
```
def ex1(s):
#The length of the string must be at least 3
    if len(s)<3:
        return "The string does not belong to the language aba+"
#The first three characters are mandatory a,b,a in this order, thus we can use
#the index of the string to check if they are
    if s[0]=='a':
        if s[1]=='b':
            if s[2]=='a':
                #After index 2 only "a" can appear, thus any character except "a"
                #makes the string not to be in the language
                for i in range(3,len(s)):
                    if s[i]!='a':
                        return "The string does not belong to the language aba+"
                return "The string belongs to the language aba+"
            return "The string does not belong to the language aba+"
        return "The string does not belong to the language aba+"
    return "The string does not belong to the language aba+"

s1="abaaaa"
s2="ab"
s3='abba'
s4='aba'
print(s1, ex1(s1),'\n')
print(s2, ex1(s2),'\n')
print(s3, ex1(s3),'\n')
print(s4, ex1(s4),'\n')
```



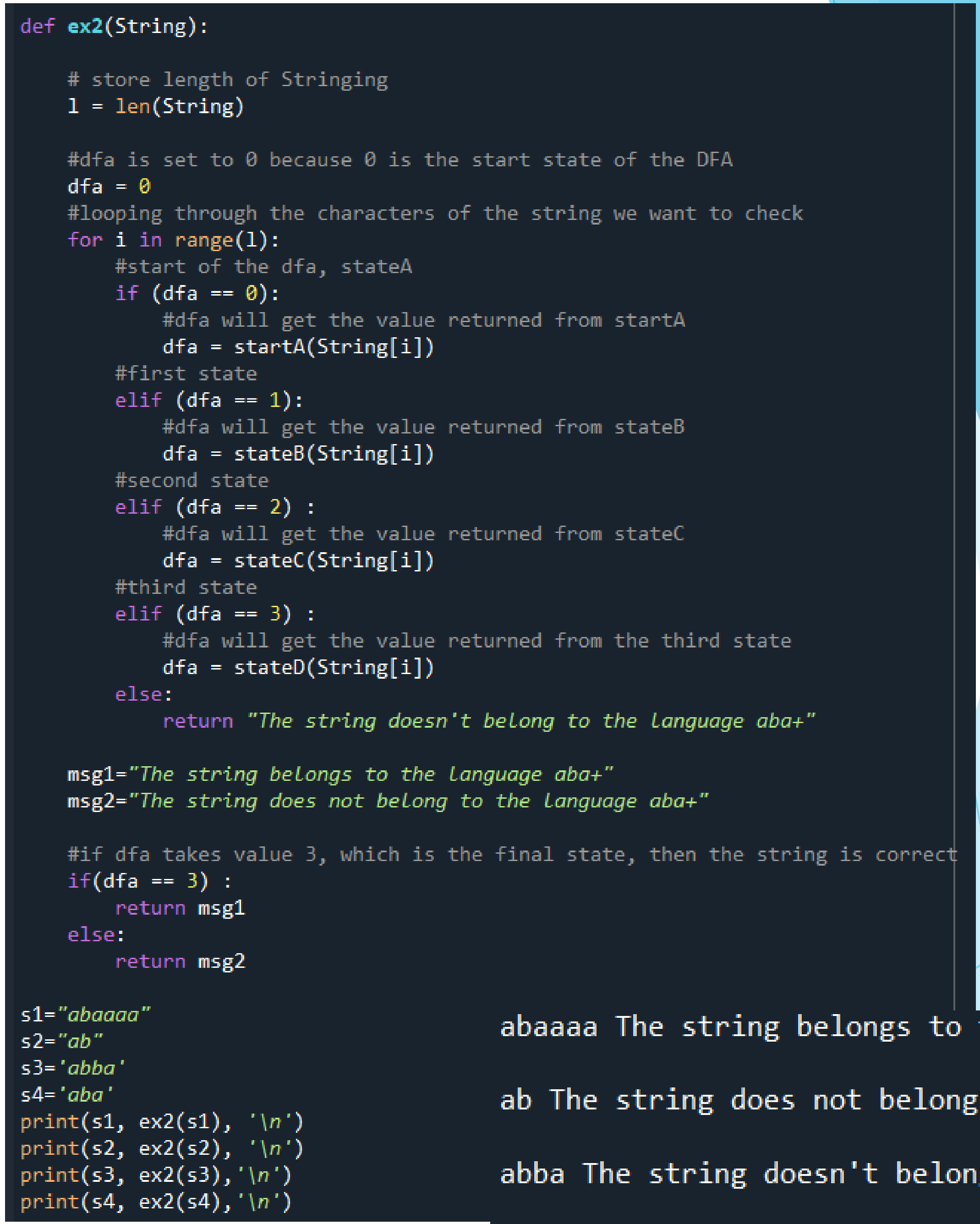
```
abaaaa The string belongs to the language aba+

ab The string does not belong to the language aba+

abba The string does not belong to the language aba+

aba The string belongs to the language aba+
```

Example

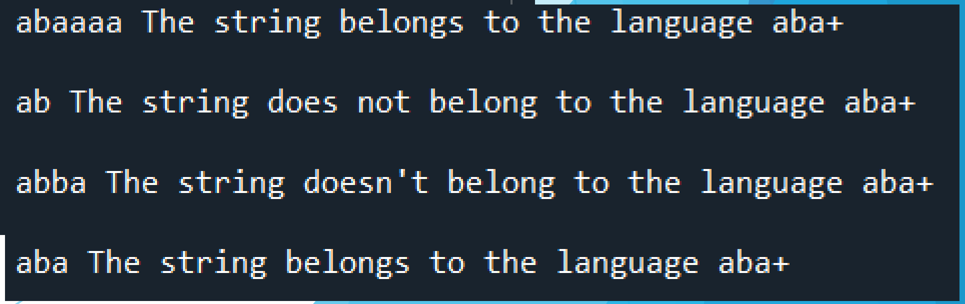
- Using functions for every state



```python
#function for the start of the DFA
def startA(x):
    #if the first character is a, then it goes to the first state
    if (x == 'a'):
        dfa = 1
    #if not, -1 is used for any invalid character
    else:
        dfa = -1
    return dfa

#function for the first state (B)
def stateB(x):
    #if the second character is b, it goes to the second state
    if (x == 'b'):
        dfa = 2
    #else the string is not correct
    else:
        dfa = -1
    return dfa

#function for the second state (C)
def stateC(x):
    #if the third character is a, then it goes to the third state
    if (x == 'a'):
        dfa = 3
    else:
        dfa = -1
    return dfa

#function for the third state (D)  which is final
def stateD(x):
    #if the character is a, then dfa will get value 3, which is the third state
    if (x == 'a'):
        dfa = 3
    else:
        dfa = -1
    return dfa
```



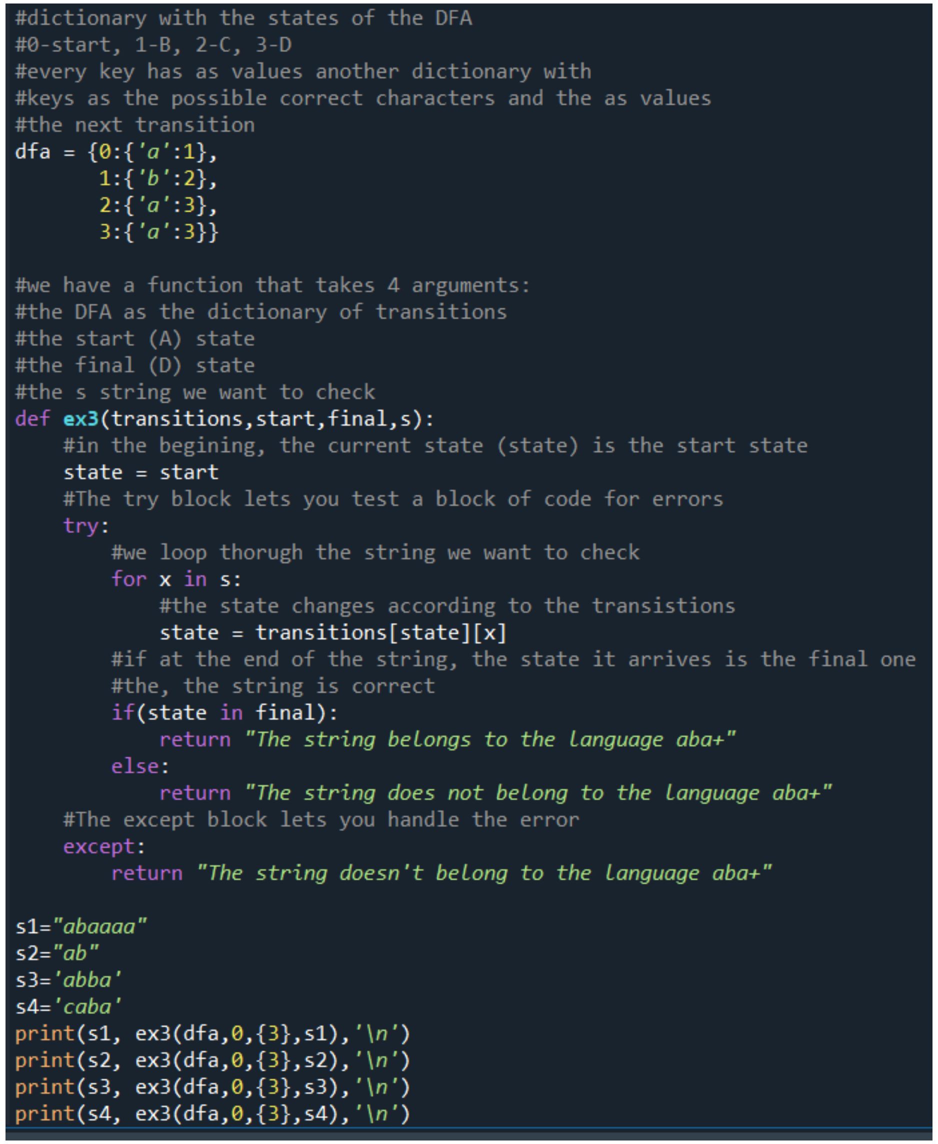
```python
def ex2(String):

    # store length of Stringing
    l = len(String)

    #dfa is set to 0 because 0 is the start state of the DFA
    dfa = 0
    #looping through the characters of the string we want to check
    for i in range(l):
        #start of the dfa, stateA
        if (dfa == 0):
            #dfa will get the value returned from startA
            dfa = startA(String[i])
        #first state
        elif (dfa == 1):
            #dfa will get the value returned from stateB
            dfa = stateB(String[i])
        #second state
        elif (dfa == 2) :
            #dfa will get the value returned from stateC
            dfa = stateC(String[i])
        #third state
        elif (dfa == 3) :
            #dfa will get the value returned from the third state
            dfa = stateD(String[i])
        else:
            return "The string doesn't belong to the language aba+"

    msg1="The string belongs to the language aba+"
    msg2="The string does not belong to the language aba+"

    #if dfa takes value 3, which is the final state, then the string is correct
    if(dfa == 3) :
        return msg1
    else:
        return msg2

s1="abaaaa"
s2="ab"
s3='abba'
s4='aba'
print(s1, ex2(s1), '\n')
print(s2, ex2(s2), '\n')
print(s3, ex2(s3),'\n')
print(s4, ex2(s4),'\n')
```



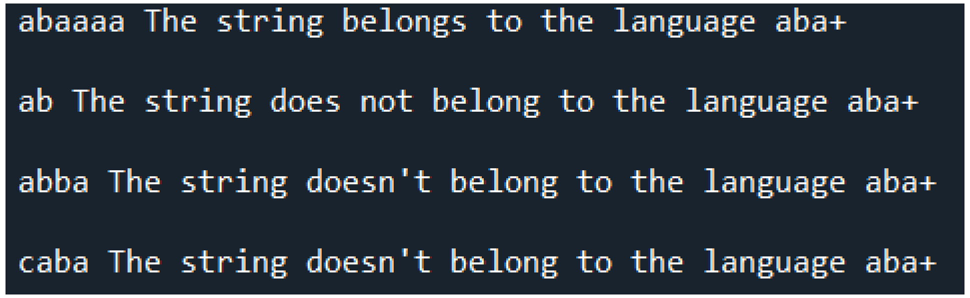
```
abaaaa The string belongs to the language aba+

ab The string does not belong to the language aba+

abba The string doesn't belong to the language aba+

aba The string belongs to the language aba+
```

Example

- Using a dictionary for states



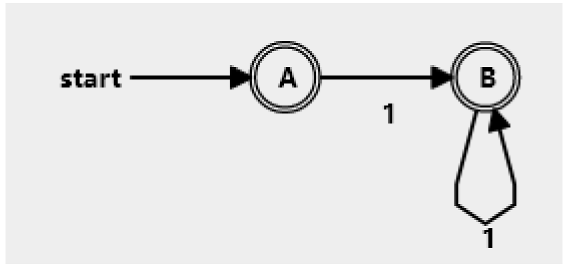
```python
#dictionary with the states of the DFA
#0-start, 1-B, 2-C, 3-D
#every key has as values another dictionary with
#keys as the possible correct characters and the as values
#the next transition
dfa = {0:{'a':1},
       1:{'b':2},
       2:{'a':3},
       3:{'a':3}}

#we have a function that takes 4 arguments:
#the DFA as the dictionary of transitions
#the start (A) state
#the final (D) state
#the s string we want to check
def ex3(transitions,start,final,s):
    #in the begining, the current state (state) is the start state
    state = start
    #The try block lets you test a block of code for errors
    try:
        #we loop thorugh the string we want to check
        for x in s:
            #the state changes according to the transistions
            state = transitions[state][x]
        #if at the end of the string, the state it arrives is the final one
        #the, the string is correct
        if(state in final):
            return "The string belongs to the language aba+"
        else:
            return "The string does not belong to the language aba+"
    #The except block lets you handle the error
    except:
        return "The string doesn't belong to the language aba+"

s1="abaaaa"
s2="ab"
s3='abba'
s4='caba'
print(s1, ex3(dfa,0,{3},s1),'\n')
print(s2, ex3(dfa,0,{3},s2),'\n')
print(s3, ex3(dfa,0,{3},s3),'\n')
print(s4, ex3(dfa,0,{3},s4),'\n')
```



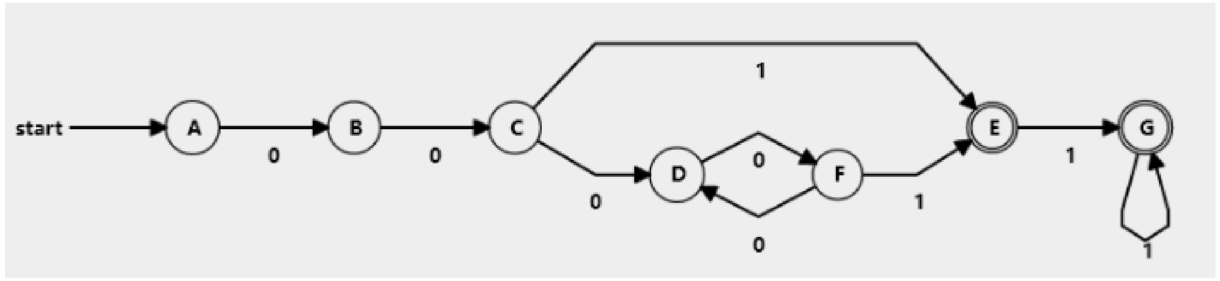
```
abaaaa The string belongs to the language aba+

ab The string does not belong to the language aba+

abba The string doesn't belong to the language aba+

caba The string doesn't belong to the language aba+
```

Exercises

1. Create a DFA in Python that checks if a word belongs to a language where the words are composed only by 1s (1+). Provide at least 2 examples



2. Create a DFA in Python that checks if a word belongs to a language where the words start with an even numbers of 0s followed by at least one 1 ((00)+1+). Provide at least 2 examples



3. Create a DFA in Python that checks if a word belongs to a language of words that start and end with the same letter. Provide at least 2 examples.

Homework

1. In HW5Ex1.html:
   a) Find and display the links (https://www.google.com/)
   b) Find and display the labels of the form (Last name, First name)
   c) Find and display the types of the attributes in the form (text, number, email)
   d) Find and display the values of the options in the form (PhD Student, Professor)
   e) Find and display all the ids in the form (lname, fname)
2. Create a regex to check if HTML tags are correctly written. Give at least 6 different examples (5 should be with different inconsistencies such as <p, p>, <id=“>, “<p>”, <p/>)
3. Create an excel file from HW5Ex2.txt (The columns should be LastName, FirstName, HiringDate, Salary)
4. Create a DFA in Python that checks if a word belongs to a language of words that either start or end with ab. (e.g. abaa,aabab are ok, aabba, aabb are not ok). Provide at least 2 examples.