APSSDC

Andhra Pradesh State Skill Development Corporation

Skill AP
APSSDC

(https://apssdc.in)

Day12 Python Programming

Day Objectives

- File Handling Contd..
- Regular Expressions

File Modes

1. r
2. w
3. a
4. r+
5. w+
6. a+
7. wb
8. rb
9. ab
10. wb+
11. wa+
12. wr+

In [1]:
```python
with open('../regex.txt', 'r') as f:
    print(f)
    #data = f.read()
    #print(data)

    print(f.readlines())
```

```
<_io.TextIOWrapper name='../regex.txt' mode='r' encoding='cp1252'>
['Regular expression\n', 'From Wikipedia, the free encyclopedia\n', 'Jump t
o navigationJump to search\n', '"Regex" redirects here. For the comic book,
see Re:Gex.\n', '\n', 'The match results of the pattern\n', '(?<=\\.) {2,}
(?=[A-Z]).\n', 'At least two spaces are matched, but only if they occur dir
ectly after a period (.) and before an uppercase letter.\n', '\n', 'Stephen
Cole Kleene, who helped invent the concept\n', '\n', 'A blacklist on Wikipe
dia which uses regular expressions to identify bad titles\n', 'A regular ex
pression (shortened as regex or regexp;[1] also referred to as rational exp
ression[2][3]) is a sequence of characters that define a search pattern. Us
ually such patterns are used by string-searching algorithms for "find" or
"find and replace" operations on strings, or for input validation. It is a
technique developed in theoretical computer science and formal language the
ory.\n', '\n', 'The concept arose in the 1950s when the American mathematic
ian Stephen Cole Kleene formalized the description of a regular language. T
he concept came into common use with Unix text-processing utilities. Differ
ent syntaxes for writing regular expressions have existed since the 1980s,
one being the POSIX standard and another, widely used, being the Perl synta
x.\n', '\n', 'Regular expressions are used in search engines, search and re
place dialogs of word processors and text editors, in text processing utili
ties such as sed and AWK and in lexical analysis. Many programming language
s provide regex capabilities either built-in or via libraries.\n', 'New dat
a appended using r+ mode in the last line']
```

a+, w+, r+

In [2]:
```python
with open('../regex.txt', 'r+') as f:
    data = '\nNew data appended using r+ mode in the last line'
    f.write(data)
    data = f.readlines()[-1]
    print(data)
```

```
New data appended using r+ mode in the last line
```

Copying data from one file to other files

- Open the Original File in `r` mode
- open create the req copy file in `w`
- do some operation
- Close the opened files

In [3]:
```python
with open('../regex.txt', 'r') as f:
    with open('copy1.txt', 'w') as cf1:
        with open('copy2.txt', 'w') as cf2:
            read_data = f.read()
            cf1.write(read_data)
            cf2.write(read_data)
```

Copying data from one file to 10 different files

In [4]:
```python
with open('../regex.txt', 'r') as f:
    read_data = f.read()
    for num in range(1,11):
        with open('fileCopy{}.txt'.format(num), 'w') as cf:
            cf.write(read_data)
```

In [5]:
```python
with open('../regex.txt', 'r') as f:
    read_data = f.read()
    for num in range(1,11):
        with open('fileCopy{}.doc'.format(num), 'wb') as cf:
            cf.write(read_data.encode())
```

Creating a file with rollNumbers from 18X41A1201 to 18X41A1299

In [6]:
```python
# File Name RollNumber.txt

with open('rollNumber.txt', 'w') as f:
    s = '18X41A12'
    s1 = s + '0'
    for i in range(1, 100):
        if i < 10:
            rollNum = s1 + str(i)
        else:
            rollNum = s + str(i)
        f.write(rollNum + '\n')
```

Regular Expression

It is a sequence of characters that define a search pattern.

- `re` is the package avaliable in python for implementing the Regular EXpressions.
- Our Data Should avaliable in String format

Syntax

method(pattern, string)

In [7]:
```python
import re
```

Search

```
In [8]: s = 'APSSDC Python Online Training for SRKIT Students'
```

```
In [9]: print(re.search('AP', s))
```

```
<re.Match object; span=(0, 2), match='AP'>
```

```
In [10]: print(re.search('in', s))
```

```
<re.Match object; span=(17, 19), match='in'>
```

```
In [11]: print(re.search('apssdc', s))
```

```
None
```

```
In [12]: print(re.search('AC', s))
```

```
None
```

```
In [13]: roll = '18X41A1299'

         re.search('18X41A12', roll)
```

```
Out[13]: <re.Match object; span=(0, 8), match='18X41A12'>
```

Match()

```
In [14]: re.match('APSSDC', s)
```

```
Out[14]: <re.Match object; span=(0, 6), match='APSSDC'>
```

```
In [15]: print(re.match('18', s))
```

```
None
```

```
In [16]: print(re.match('PSSDC', s))
```

```
None
```

findall()

```
In [17]: re.findall('in', s)
```

```
Out[17]: ['in', 'in', 'in']
```

In [18]:
```
print(re.findall('ap', s))
```
```
[]
```

sub() - Used to replace the data in the main String

Syntax:

re.sub(pattern, newString, OriginalString)

In [19]:
```
re.sub('in', 'IN', s)
```
Out[19]:
```
'APSSDC Python OnlINe TraININg for SRKIT Students'
```

In [20]:
```
s
```
Out[20]:
```
'APSSDC Python Online Training for SRKIT Students'
```

In [21]:
```
re.sub('SRKIt', 'IN', s)
```
Out[21]:
```
'APSSDC Python Online Training for SRKIT Students'
```

Split

In [22]:
```
s2 = """A regular expression (shortened as regex or regexp;[1] also refe
```

In [23]:
```
s2
```
Out[23]:
```
'A regular expression (shortened as regex or regexp;[1] also referred to as
rational expression[2][3]) is a sequence of characters that define a search
pattern. Usually such patterns are used by string-searching algorithms for
"find" or "find and replace55" operations on strings, or for input validati
on. It is a techn7ique developed in theoretical computer s8cience and forma
l 5language theory.'
```

In [24]:
```
re.split('\. ', s2)
```
Out[24]:
```
['A regular expression (shortened as regex or regexp;[1] also referred to a
s rational expression[2][3]) is a sequence of characters that define a sear
ch pattern',
 'Usually such patterns are used by string-searching algorithms for "find"
or "find and replace55" operations on strings, or for input validation',
 'It is a techn7ique developed in theoretical computer s8cience and formal
5language theory.']
```

Special characters for creating `re` patterns

1. `^` - Checking the string is started with that pattern
2. `$` - Checking the string is ended with that pattern
3. `*` - any characters
4. `\d` - for matching the digits
5. `\w` - for matching alphabets
6. `\s` - for matching space

In [25]:
```python
re.findall('\d', s2)
```

Out[25]:
```
['1', '2', '3', '5', '5', '7', '8', '5']
```

In [26]:
```python
print(re.findall('\w', s2))
```

```
['A', 'r', 'e', 'g', 'u', 'l', 'a', 'r', 'e', 'x', 'p', 'r', 'e', 's', 's',
'i', 'o', 'n', 's', 'h', 'o', 'r', 't', 'e', 'n', 'e', 'd', 'a', 's', 'r',
'e', 'g', 'e', 'x', 'o', 'r', 'r', 'e', 'g', 'e', 'x', 'p', '1', 'a', 'l',
's', 'o', 'r', 'e', 'f', 'e', 'r', 'r', 'e', 'd', 't', 'o', 'a', 's', 'r',
'a', 't', 'i', 'o', 'n', 'a', 'l', 'e', 'x', 'p', 'r', 'e', 's', 's', 'i',
'o', 'n', '2', '3', 'i', 's', 'a', 's', 'e', 'q', 'u', 'e', 'n', 'c', 'e',
'o', 'f', 'c', 'h', 'a', 'r', 'a', 'c', 't', 'e', 'r', 's', 't', 'h', 'a',
't', 'd', 'e', 'f', 'i', 'n', 'e', 'a', 's', 'e', 'a', 'r', 'c', 'h', 'p',
'a', 't', 't', 'e', 'r', 'n', 'U', 's', 'u', 'a', 'l', 'l', 'y', 's', 'u',
'c', 'h', 'p', 'a', 't', 't', 'e', 'r', 'n', 's', 'a', 'r', 'e', 'u', 's',
'e', 'd', 'b', 'y', 's', 't', 'r', 'i', 'n', 'g', 's', 'e', 'a', 'r', 'c',
'h', 'i', 'n', 'g', 'a', 'l', 'g', 'o', 'r', 'i', 't', 'h', 'm', 's', 'f',
'o', 'r', 'f', 'i', 'n', 'd', 'o', 'r', 'f', 'i', 'n', 'd', 'a', 'n', 'd',
'r', 'e', 'p', 'l', 'a', 'c', 'e', '5', '5', 'o', 'p', 'e', 'r', 'a', 't',
'i', 'o', 'n', 's', 'o', 'n', 's', 't', 'r', 'i', 'n', 'g', 's', 'o', 'r',
'f', 'o', 'r', 'i', 'n', 'p', 'u', 't', 'v', 'a', 'l', 'i', 'd', 'a', 't',
'i', 'o', 'n', 'I', 't', 'i', 's', 'a', 't', 'e', 'c', 'h', 'n', '7', 'i',
'q', 'u', 'e', 'd', 'e', 'v', 'e', 'l', 'o', 'p', 'e', 'd', 'i', 'n', 't',
'h', 'e', 'o', 'r', 'e', 't', 'i', 'c', 'a', 'l', 'c', 'o', 'm', 'p', 'u',
't', 'e', 'r', 's', '8', 'c', 'i', 'e', 'n', 'c', 'e', 'a', 'n', 'd', 'f',
'o', 'r', 'm', 'a', 'l', '5', 'l', 'a', 'n', 'g', 'u', 'a', 'g', 'e', 't',
'h', 'e', 'o', 'r', 'y']
```

In [27]:
```python
print(re.findall('\s', s2))
```

```
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
```

In [28]:
```python
print(re.findall('^r', s))
```

```
[]
```

In [29]:
```python
print(re.findall('\D', s2))
```

```
['A', ' ', 'r', 'e', 'g', 'u', 'l', 'a', 'r', ' ', 'e', 'x', 'p', 'r', 'e',
's', 's', 'i', 'o', 'n', ' ', '(', 's', 'h', 'o', 'r', 't', 'e', 'n', 'e',
'd', ' ', 'a', 's', ' ', 'r', 'e', 'g', 'e', 'x', ' ', 'o', 'r', ' ', 'r',
'e', 'g', 'e', 'x', 'p', ';', '[', ']', ' ', 'a', 'l', 's', 'o', ' ', 'r',
'e', 'f', 'e', 'r', 'r', 'e', 'd', ' ', 't', 'o', ' ', 'a', 's', ' ', 'r',
'a', 't', 'i', 'o', 'n', 'a', 'l', ' ', 'e', 'x', 'p', 'r', 'e', 's', 's',
'i', 'o', 'n', '[', ']', '[', ']', ')', ' ', 'i', 's', ' ', 'a', ' ', 's',
'e', 'q', 'u', 'e', 'n', 'c', 'e', ' ', 'o', 'f', ' ', 'c', 'h', 'a', 'r',
'a', 'c', 't', 'e', 'r', 's', ' ', 't', 'h', 'a', 't', ' ', 'd', 'e', 'f',
'i', 'n', 'e', ' ', 'a', ' ', 's', 'e', 'a', 'r', 'c', 'h', ' ', 'p', 'a',
't', 't', 'e', 'r', 'n', '.', ' ', 'U', 's', 'u', 'a', 'l', 'l', 'y', ' ',
's', 'u', 'c', 'h', ' ', 'p', 'a', 't', 't', 'e', 'r', 'n', 's', ' ', 'a',
'r', 'e', ' ', 'u', 's', 'e', 'd', ' ', 'b', 'y', ' ', 's', 't', 'r', 'i',
'n', 'g', '-', 's', 'e', 'a', 'r', 'c', 'h', 'i', 'n', 'g', ' ', 'a', 'l',
'g', 'o', 'r', 'i', 't', 'h', 'm', 's', ' ', 'f', 'o', 'r', ' ', '"', 'f',
'i', 'n', 'd', '"', ' ', 'o', 'r', ' ', '"', 'f', 'i', 'n', 'd', ' ', 'a',
'n', 'd', ' ', 'r', 'e', 'p', 'l', 'a', 'c', 'e', '"', ' ', 'o', 'p', 'e',
'r', 'a', 't', 'i', 'o', 'n', 's', ' ', 'o', 'n', ' ', 's', 't', 'r', 'i',
'n', 'g', 's', ',', ' ', 'o', 'r', ' ', 'f', 'o', 'r', ' ', 'i', 'n', 'p',
'u', 't', ' ', 'v', 'a', 'l', 'i', 'd', 'a', 't', 'i', 'o', 'n', '.', ' ',
'I', 't', ' ', 'i', 's', ' ', 'a', ' ', 't', 'e', 'c', 'h', 'n', 'i', 'q',
'u', 'e', ' ', 'd', 'e', 'v', 'e', 'l', 'o', 'p', 'e', 'd', ' ', 'i', 'n',
' ', 't', 'h', 'e', 'o', 'r', 'e', 't', 'i', 'c', 'a', 'l', ' ', 'c', 'o',
'm', 'p', 'u', 't', 'e', 'r', ' ', 's', 'c', 'i', 'e', 'n', 'c', 'e', ' ',
'a', 'n', 'd', ' ', 'f', 'o', 'r', 'm', 'a', 'l', ' ', 'l', 'a', 'n', 'g',
'u', 'a', 'g', 'e', ' ', 't', 'h', 'e', 'o', 'r', 'y', '.']
```

In [30]:
```python
print(re.sub('\s', '-', s2))
```

```
A-regular-expression-(shortened-as-regex-or-regexp;[1]-also-referred-to-as-
rational-expression[2][3])-is-a-sequence-of-characters-that-define-a-search
-pattern.-Usually-such-patterns-are-used-by-string-searching-algorithms-for
-"find"-or-"find-and-replace55"-operations-on-strings,-or-for-input-validat
ion.-It-is-a-techn7ique-developed-in-theoretical-computer-s8cience-and-form
al-5language-theory.
```