REVISION TEST JAN 2022

PYTHON PROGRAMMING

Name: G. Anmaiah
Roll No: 2089116618
Branch: CSM (AI & ML)

Date: 10-01-2022

Questions

a) Define Regular Expression
b) Write any five Regular Expression special symbol with their description
c) Explain five methods of the re module through proper examples.

2. a) List out the functions of Thread object in threading module.
b) Write regular Expression for DOB of pattern: dd/mm/yyyy
c) Explain Producer-Consumer problem. Write a python program to implement the Producer-Consumer problem

3. a) Write Regular expression for mail id of pattern : abcd@xyz.com
b) Write short notes on Global Interpreter Lock.
c) Define Thread. Write a python program to calculate square and cube of a given number using Multithreading.

Scanned with CamScanner

Solutions

① Regular Expression

a) 1) A Regular Expression is a string that contains special symbols and characters to extract and find information from the given data.

2) It helps to search information, match, find and split information.

3) In python there is a special module for regular expressed. That is 're' module.

① ⓑ There are many special symbols related to Regular Expression module. Some of them are

1. ^
2. $
3. .
4. *
5. +

1. ^

→ The symbol which is used to match at the begining of the line.

Ex: ^Hello

→ It will match 'Hello' at the start of the string.

Scanned with CamScanner

2. $

→ The symbol uses to matches at the end of line

Ex: Vignan$

→ It will match 'Vignan' at the end of the string

3. .

→ Matches any single character which is given before .

→ It cannot matches the newline character it is exception.

Ex: R0. → It matches with Roy, Row etc

4. *

This symbol is used to matches 0 (or) more occurances of regular expression

Ex: [a-z]*

[A-Z]*

[0-9]*

→ It will matches the zero (or) more of lower, upper, digit characters.

5. +

This symbol is used to match 1 (or) more occurance of regular expression

ex: [a-z]+, [A-Z]+, [0-9]+

→ It will match one (or) more occurances of lower, upper, digits characters.

Scanned with CamScanner

(1)
(c) Sol:- There are some methods of re module. It is used to search, find information etc.....
Some of them are.

1. match()
2. search()
3. findall()
4. split()
5. sub()

1. Match()

→ The match() function matches a pattern to a string with optional flags.

Syntax:

```
re.match(pattern, string, flags)
```

→ The function tries to match the pattern with a string.

→ The flag field is optional.

→ If re.match() function finds a match, it returns the match object. If not match it returns none.

→ It matches only the first starting part.

→ If the match is not found in middle of string, it cannot matches.

Scanned with CamScanner

Ex: Program to demonstrate match() function.

```
import re
String = "She sells sea Shells on the sea shore"
pattern1 = "Sells"
if re.match(pattern1, String):
    print("Match found")
else:
    print(pattern1, "is not present in the string")
pattern2 = "She"
if re.match(pattern2, String):
    print("Match found")
else:
    print(pattern2, "is not present in the string")
```

Output:

```
Sells is not present in the string
Match found
```

Scanned with CamScanner

2. Search()

→ search() is a function in the re module. It searches the pattern anywhere in the string.

Syntax:

re.search(pattern, string, flags)

→ It is similar to match() function.

→ The function searches for first occurance of pattern within a string.

→ The flags is optional

→ If search is successful, a match object is returned and None otherwise.

program to demonstrate the use of search() function.

```
import re
string = "Im from Vignan"
pattern = "Vignan"
if re.search(pattern, string):
    print("match found")
else
    print(pattern, "is not present in the string")
```

Output

Match found

Note: This function finds a match of a pattern anywhere in the string.

Scanned with CamScanner

3. findall()

→ findall() function is used to search a string and returns a list of matches of the pattern in the string

→ If no match found, then the returned list is empty

→ Syntax:

```
list = re.findall(pattern, input_str, flags)
```

Program

```
import re
pattern = r"[a-z,A-z]+\d+"
matches = re.findall(pattern, "LXI 2013, VXI 2015, VDI 20104, Maruti Suzuki Cars in India")
for match in matches:
    print(match, end = " ")
```

Output

```
LXI 2013 VXI 2015 VDI 20104
```

Note:

The re.findall()-function returns a list of all substrings that match a pattern.

Scanned with CamScanner

4 Split ()

→ Split function is used to split the string in a fixed string.

→ They split a string based on RE module pattern

→ It adds some significent power to that string.

→ We can set the string position to split.

Ex program

```
import re
S1 = "This is AIML BRANCH"
a = re.split('is', S1)
print (a)
```

Output

```
This
is
AIML
BRANCH
```

Syntax :

```
a = re.split ("pattern, string, max)
```

Scanned with CamScanner

5. Sub()

→ The sub() function is used to search a pattern in the string and replace it with another pattern.

Syntax

re.sub(pattern, repl, string, max)

Ex: program

```
import re
s = "This is Assignment four"
a = "four"
r = "three"
s1 = re.sub(a, r, s, 1)
```

Output:

This is Assignment three.

Scanned with CamScanner

(2) (a) The functions of Thread object in threading module are

1. Lock ()
2. Semaphore ()
3. acquire ()
4. release ()
5. start ()
6. join ()

(2) (b) Regular Expression for Date of Birth

Ans:- dd/mm/yyyy

\d{1}/\d{2}/\d{4}

(2) (c) Producer- Consumer Concept

Ans:-

→ This concept is related to queue concept

→ The element which is enter first in the list it only had scope to out first.

→ The another element cannot out without coming out of the first element.

→ The concept is First In First Out [FIFO]

Scanned with CamScanner

→ Producer is the role who can produce the objects.

→ The object will enter into a space where Consumer can receive.

→ The produced Objects will recieved by Consumer after travelling into a space (or) path.

→ The first produced item will be received at first by consumer.

→ The second item cannot be recieved without recieving first item.

→ This is the concept of producer – Consumer problem.

Scanned with CamScanner

program using Producer consumer program.

```
from threading import *
from time import *

Class producer:
    def __init__(self):
        self.List = [ ]
        self.dataprodover = False
    def produce(self):
        for i in range(1,6):
            self.List.append(i)
            sleep(1)
            print('Item Produced')
        self.dataprodover = True
class Consumer:
    def __init__(self,p):
        self.prod = p;
    def Consume(self):
        while self.prod.dataprodover == False:
            sleep(5)
        print(self.prod.List)

p = Producer()
c = Consumer(p)
```

Scanned with CamScanner

```
t1 = Thread (target = p.produce)
t2 = Thread (target = c.consume)

t1.start()
t2.start()
```

Output:

```
Item Produced
Item Produced
Item Produced
Item Produced
[1, 2, 3, 4, 5]
```

---

③ (a) Regular Expression for email

Ans:

```
r"[\w.-]+@[\w.-]+"
```

② (b) Global Interpreter Lock (GIL)

Ans:

→ Execution of python code is controlled by the Python Virtual Machine (PVM)

→ only one thread is executed by the interpreter at any given time

Scanned with CamScanner

→ Access to the Python Virtual Machine is controlled by the GIL (Global Interpreter Lock)

Steps to be followed

1. Set the GIL
2. Switch in a thread to run
3. Execute either
   a. for a specified number of bytecode instructions
   b. If the thread Voluntarily yields control back to/sleeps (~~switch out thread~~) (time.sleep)
4. Put the thread back to sleep (switch out thread)
5. Unlock the GIL; and
6. Do it all over again (lathes, rinse, repeat.

Scanned with CamScanner

(8)(c) Thread.

Ans:- Thread is a light weight process. It is a unit of execution.

=# program to calculate square and cube of a given number by multiftreading.

```
import threading

def print_cube(num):
    print("Cube:", num * num * num)

def print_square(num):
    print("Square:", num * num)

def main():
    t1 = threading.Thread(target = print_square, args = (10,))
    t2 = threading.Thread(target = print_cube, args = (10,))

    t1.start()
    t2.start()
    t1.join
    t2.join
    print("Done!")

if __name__ == "__main__":
    main()
```

Scanned with CamScanner

Output:

Square : 100

Cube : 1000

Done!

Scanned with CamScanner