Regular Expressions

- Regular Expression Functions
- Special Characters and Functions
- Creating character sets and repetitions
- Capture and non-Capture groups
- Backreferences

Regular Expression

- Stronger form of pattern matching
- **re** module available in python or regular expression

- *re.match(pattern, string)* : matches at start of string, None if no match
- *re.search(pattern, string)* : matches first occurrence anywhere in string
- *re.findall(pattern, string)/ finditer()* : returns all matches

Examples

- Write regex to search word **Python** in a string.
- String starts with Python
- String ends with Python

Regular Expression Special Characters – 1

- ^ : matches start of string
  - WAR to check whether string starts with word 'Python' or not.
- $: matches end of string
  - WAR to check that string should end with '!' symbol
- . (dot) : matches any single character except newline

Special Sequences

- \d : match any digit :[0-9]
- \D : match any non-digit :[^0-9]
- \b : matches a word boundary
- \s : match any white space character : [\n\t]
- \S : match any non-whitespace character : [^ \n\t]
- \w : match any alphanumeric including _ : [_a-zA-Z0-9]
- \W : match any non-alphanumeric : [^_a-zA-Z0-9]

Questions

- Find all 3 digit numbers in a string
- Match ipv4 ip Address: ex: 192.168.254.001
- WAR to check valid phone no: Valid Phone no is of the form “xxxx xxx xxx”

Regular Expression Special Characters – 1

- * : matches preceding RE 0 or more times
- + : matches preceding RE 1 or more times
- ? : matches preceding RE 0 or 1 times(non greedy)
- \ : is used as an escape sequence

Questions

- WAR to match a string that starts and ends with the word 'the', make it case insensitive.
- WAR a regex to validate PAN numbers:
  AAAAANNNNA
  First five are alphabets, next 4 are numbers and last one is alphabet

Including and excluding specific characters

- [] : specify the characters to be included inside the []
- [x-y] : specify ranges. ex: [a-z], [0-9]
- [^] : specify inverse set ex: [^a-b], [^,.]

Questions

- WAR that returns all the vowels
- Create a pattern, for strings with 4 characters. First should be alphabet, second can be alphanumeric, third should be digit, last should be ,(comma) or .(dot)
- Pattern for 5 character string with first and last character should not be digit. Second character should not be a vowel and white space. Third should not be lower case character and fourth should not be Uppercase character.

Repetitions

- {n} : matches exactly n repetitions of preceding re.
- {m, n} : match m up to n repetitions.
- {m, } : match min m repetitions.
- {, n} : : max n repetitions.

Questions

- Rewrite the IpV4 regular expression.
- Rewrite the Phone Number Regular Expression.
- Write RE to match a string of length 10, with it should have first five characters as only alphabets, next 4 as alphabets or digits, and the last one is dot '.' .

Groups and non-capture groups and alternative

- () : used to group as a single repetition unit
- (?:) : makes the group as non-capturing group
- | : works as or operator

Questions

- Rewrite the IpV4 regular expression to capture all the subnets.
- Write a RE that parses these kind of Phone numbers and returns the country code:

  +91 1111 222 123

  +cc dddd ddd ddd

  The RE should extract CC section from the Phone Number. The phone no should be a valid no.

Backreferences

- Backreference means to check for some pattern that had occurred previously in the expression
- Uses \n syntax, i.e. \1 matches the first capture group and so on.
- ([a-z]) …. \1

Questions

- Write RE to match a string of length 5, such that it should be a palindrome, also capture all the matched characters
- Write RE to check for XML syntax that the closing tag is correct or not.

Multithreading

- Steps to create a thread
- Some thread functions
- Locking critical sections and RLock vs Lock
- Event and Condition
- Queue based Synchronization

Run any Function on thread

- Import threading module:

```
from threading import Thread
```

- Create a thread:

```
<thread object> = Thread(target = <target function>, args=(<args tuple>))
```

- Start the Thread

```
<thread object>.start()
```

- Join the Thread (optional)

```
<thread object>.join()
```

The join function

- Join function allows waiting till a thread finishes

  <thread object>.join()

- Join() is a blocking call
- Execution of the remaining code blocks till the thread on which join was called finishes.

Threading functions and sleep

- currentThread()
- get_ident()
- setName()
- getName()
- import time
  time.sleep(<time in seconds>)

Lock – Locking Critical Section

- acquire() - acquire a lock; blocks if lock already held by some other thread till any thread releases it
- release() - release the lock
- locked() - tell whether already locked or not

- Ex: Two threads increment the same value in parallel might cause invalid value updates

Using Context Manager with Lock

-
```
with <lock_object>:
    # critical section of code
```

- Forgetting to release a lock from a thread causes all the other threads waiting for that lock to **block** infinitely.
- The **with** statement handles automatic acquisition and release of the lock object.

RLock vs Lock

- lock_object = threading.Lock()

```
with lock_object  :                 #lock acquired once
        with lock_object :          #same lock acquired again will block
                # critical section of code
```

- Simple Lock objects block if acquired again by the same thread.
- RLock is Re-entrant lock :

  it doesn't block if acquired by the **same thread** again and again

Event and Condition for synchronization

- Event:
  - set()
  - wait()
  - clear()
- Condition: Provides locking along with Event based synchronization

  Condition = Event + Lock

  Multiple conditions can share a common lock.
  - acquire()
  - release()
  - wait()

Queue based synchronization

- The Queue class in python is thread safe
- The get and put methods to add and remove data from the queue are blocking calls.
- Available inside the queue module.