Paramiko:

**import** paramiko:  
  
router = paramiko.SSHClient()  
 router.load\_system\_host\_keys()  
 router.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())  
 router.connect(host, username=user, password=passwd, allow\_agent=**False**, look\_for\_keys=**False**)  
 print(**'Router SSH connected'**)  
 **return** router

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Decorator

**import** time  
**import** datetime  
  
**def** decorator(original\_function):  
 print(**"Started "**)  
 **def** new\_function(\*args,\*\*kwargs):  
 before = datetime.datetime.now()  
 x = original\_function(\*args,\*\*kwargs)  
 print(x)  
 time.sleep(5)  
 after = datetime.datetime.now()  
 print(**"Elapsed Time = {0}"**.format(after-before))  
 **return** x  
 print(**"End"**)  
 **return** new\_function  
  
@decorator  
**def** add(x,y):  
 **return** x+y

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write a content in file and replace the string ( with out RE)

file\_name = **'example.txt'  
with** open(file\_name,**'w'**) **as** write\_file:  
 lines = [**'fast ethernet0/1\n'**,  
 **'gigabit ethernet0/1\n'**]  
 write\_file.writelines(lines)  
  
**with** open(file\_name, **'r'**) **as** read\_file:  
 **for** lines **in** read\_file.readlines():  
 print (lines.rstrip())  
 change\_lines = lines.replace(**'fast'**, **'gigabit'**)  
 print (change\_lines)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Regular Expression

value9 = **"dog cat dog"**match = re.match(**r'dog'**, value9)  
print(**"output9:"**, match.group())

Match will search the first , search will search in the entire file

value10 = **"dog cat dog"**match = re.search(**r'cat'**, value10)  
print(**"output10:"**, match.group())

Search and replace

list = [**'fa0/1'**,**'fa0/2'**,**'fa0/3'**]  
**for** i **in** list:  
 i=re.sub(**"fa"**,**'gig'**, i, re.IGNORECASE)  
 print (i)

Substitute

value2 = **"PRABHUSP121"**output2 = re.sub(**"[0-9]"**, **""**, value2)  
print(**"output2:"**, output2)

Logging into all the files in the directory and removing a text and updating the file again.

**import** os  
  
**def** remove\_word (file\_name, text):  
 lines =[]  
 new\_lines=[]  
  
 **with** open (file\_name, **'r'**)**as** read\_file:  
 lines = read\_file.readlines()  
  
 **for** line **in** lines:  
 **if** text **not in** line:  
 new\_lines.append(line)  
  
 **with** open (file\_name, **'w'**) **as** write\_file:  
 **for** line **in** new\_lines:  
 write\_file.write(line)  
  
files\_in\_dir= os.listdir(**'./'**)  
**for** files **in** files\_in\_dir:

**Converting the first letter of the word in to upper case**

s = **'fast ethernet0/1'**list = [word[0].upper() + word[1:] **for** word **in** s.split()]  
print (**" "**.join(list))

**Word Count**

str = **"My name is vivek and vivek is a Man"**word = str.split()  
word\_count = {}  
**for** i **in** word:  
 word\_count[i] = str.count(i)  
print (word\_count)

**Finding the Duplicate**

numbers = [50,34,43,24,34,10,25,10]

my\_list =[]

for i in numbers:

if i in my\_list : pass

else:

my\_list.append(i)

print(my\_list)

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Check the even number in the list**

a = [11,33,55,39,55,75,37,21,23,41,13]  
**for** num **in** a:  
 **if** num%2 == 0:  
 print(**'the list contains even number'**)  
**else**:  
 print(**'the list doesnt contain even number'**)

**def** usernameValid(uname,pwd,rpwd)

unameinp=uname

pwdinp=pwd

rpwdinp=rpwd

***self***.addUser\_element.click

***self***.name\_element.set(unameinp)

***self***.passwd\_element.set(pwdinp)

***self***.repasswd\_element.set(rpwdinp)

***self***.buttonok\_element.when\_present.click

**if** (alert\_okbutton\_element.exists?)

puts "InValid User Name"

***self***.alert\_okbutton\_element.when\_present.click

***self***.buttoncancel\_element.click

**return** ***false***

**else**

puts "Valid User Name"

**end**

***self***.filterUsername\_element.when\_present.clear

***self***.filterUsername\_element.when\_present.set(unameinp)

sleep 2

actual=***self***.usernameText\_element.text

**if** actual == unameinp

puts"Added User successfully"

**return** ***true***

**else**

puts"Add User Failed"

**return** ***false***

**end**

**end**

**Spec File**

it "should be able to Verify username validation - allowed spl characters::Tny1853692c" **do**

on UsermanagementPage **do** |action|

unamearg ="UName\_with-vldspl.char"

pwdarg ="Testpwd\_01"

rpwdarg ="Testpwd\_01"

expect(action.usernameValid(unamearg,pwdarg,rpwdarg)).to be\_true

**end**

**end**

**assert equals (**