Python Intro

Christoffer Berglund

October 5, 2018

Overview

- Repetition
- HTTP requests
- Flask
- Questions about previous lectures

Repetition

Shell integration

Exit status

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

Can deal with exitstatus'es

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

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

Pexpect

Simple

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

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

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()
```

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()
```

Web interaction

Mechanize

- Python2 module
- 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

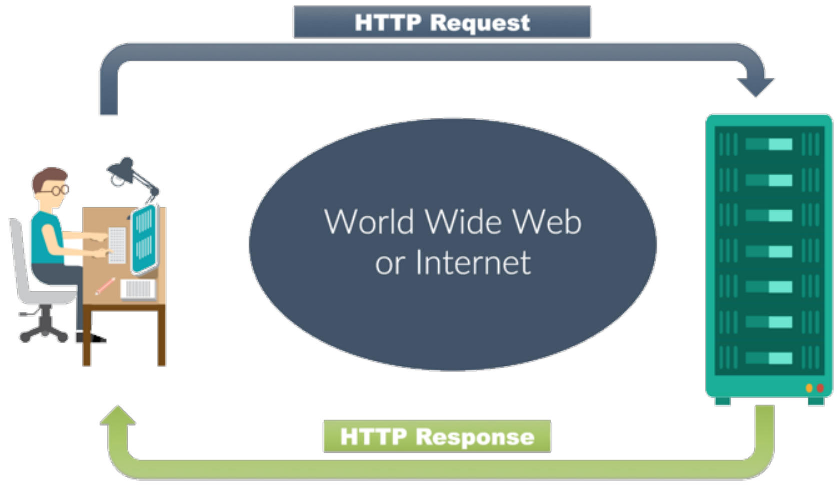
```
import mechanize
br = mechanize.Browser()
br.open('http://uia.no')
br.title()
```

HTTP Requests

HTTP requests

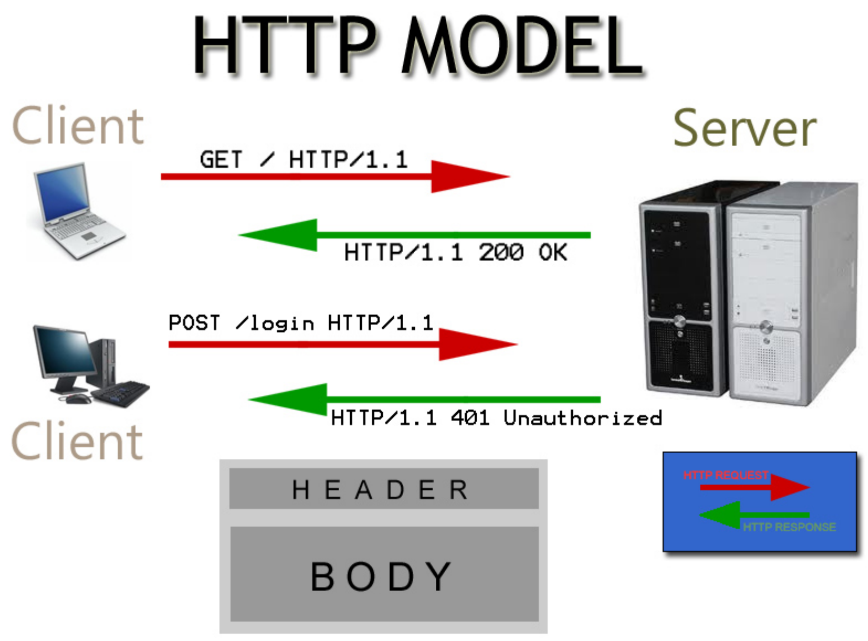
- Hypertext Transfer Protocol
- Communication between clients and servers
- Clients can be browser
- Server can be an application on a computer that hosts a website

Simple



https://www.google.no/url?sa=i&rct=j&q=&esrc=s&source=images&cd=
&cad=rja&uact=8&ved=2ahUKEwjis_zk-eTdAhWRmIsKHeaHDOIQjRx6BAgBEAU&
url=https%3A%2F%2Fwww.webnots.com%2Fwhat-is-http%2F&psig=AOvVaw2sVufnkl9aO7tl72sU71TN&
ust=1538473294330640

Advanced



https://www.google.no/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjYuZGV-uTdAhUDtYsKHZ2jA0wQjRx6BAgBEAU&url=http%3A%2F%2Fwiki.hashphp.org%2FHttpPrimer&psig=AOvVaw2sVufnkl9aO7tl72sU71TN&ust=1538473294330640

Request types

get	requests data
post	send data to server
put	send data to server to create or update
head	same as get, but without body
delete	delete resource
...	...

HTTP status codes

200	ok
201	created
400	bad request
401	unauthorized
500	internal server error

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Tools

- Browser, demo
- command line -> curl, wget
  - install: apt-get install curl wget
  - For help man curl, man wget
- graphical -> postman, wireshark

Flask

What is flask?

- Microframework
- Simple

Documentation

- Flask website: http://flask.pocoo.org/
- Flask tutorial: https://www.tutorialspoint.com/flask/

Installing and setup

- Install with pip

```
pip install Flask
```

- Setup in linux

```
export FLASK_ENV=development
export FLASK_APP=yourapp.py
flask run
```

Simple example

```
# file: simple.py
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

- To run it

```
export FLASK_ENV=development
export FLASK_APP=simple.py
flask run
```

Static html from template

- project structure

```
/application.py
/templates/
  /htmlcode.html
/static/
  /style.css
```

- to render html

```
from flask import render_template
```

Static html example

- python file

```
from flask import Flask, render_template

app = Flask(__name__)
@app.route('/index')
def index():
    return render_template('index.html', title='MyTitle')
```

- html file

```
<html>
  <head> <title>Scripting and hacking</title> </head>
  <body> <h1>Hello class</h1> </body>
</html>
```

Dynamic html, variable binding

- python file

```
from flask import Flask, render_template

app = Flask(__name__)
@app.route('/index')
def index():
    user = {'name': 'Christoffer'}
    return render_template('index.html', title='MyTitle', user=name)
```

- html file

```
<html>
  <head> <title>{{ title }}</title> </head>
  <body> <h1>Hello class, my name is {{ user.name }}!</h1> </body>
</html>
```

Control statements in html

```
<html>
  <head>
    {% if title %}
    <title> {{ title }} </title>
    {% else %}
    <title> Homepage </title>
    {% endif %}
  </head>
</html>
```

Get and Post data

```
import time

from flask import Flask, request, render_template
```

```python
app = Flask(__name__)

@app.route('/')
def my_form():
    return render_template('my-form.html')

@app.route('/', methods=['POST'])
def my_form_post():
    user = request.form['username']
    password = request.form['password']
    print(password)
    time.sleep(1)
    print("Storing {} in hacker list".format(password))
    time.sleep(1)
    print("Selling password...:", password)
    return "login " + user + " we will keep your password safe"
```

Questions about previous lectures

- Questions?

Bash

Basic

- Bourne Again Shell
- Available on Linux, Mac, Windows (Windows subsystem)
- Either from users typing at a keyboard, or as scripts.
- Anything you type into a command line, you can write as a script.

Shebang

```bash
#!/bin/bash
```

Control statements

```bash
#!/bin/bash
today=$(date +%a)
```

```bash
if [ "$today" = "Sun" ] || [ "$today" = "Fri"]; then
   echo "Today is Sunday or Friday, do complete backup"
elif [ "$today" = "Thu" ]; then
   echo "Today is Thursday, partial backup"
else
   echo "Today is not Sunday,Monday or Thursday, I will do incremental backup"
fi
```

man date man date | grep "%a"

Loops

```bash
#!/bin/bash
for i in {1..10..2}
do
   echo "Welcome for the $i th time"
done
```

Stdout, pipes and redirect

```bash
ls | grep "something" 1> file.txt 2> /dev/null
```

Powershell

Introduction

- Windows PowerShell
- Task automation framework
- Command-line shell
- Interactive editor
- Scripting language
- Based on .NET Framework
  - You can use anything from .NET
- Access to COM and WMI
- Target both local and remote

Powershell versus Linux

- **Powershell Cmdlets**
- Get-Help / man
- dir / ls
- Get-Process / ps
- Stop-Process / kill
- **Linux Equivalent**
- man
- ls
- ps
- kill

Variables

if

```
if(STATEMENT){
    echo "If-statement is true"
}
elseif(STATEMENT){
    echo "else-if statement is true"
}
else{
    echo "None of the above are true"
}
```

Pipes, Streams and Redirection - Exactly like Bash

```
ls | more
ls > file
ls 1> file
ls 2> error
```

```
$a="Hello World with variables"
echo $a
```

Python

Python vs Java

- python

```
''.join([chr(random.randint(64,128)) for i in range(10)])
```

- java

```
public class PassordGenerator{
  public String getRandomPassword() {
    StringBuffer password = new StringBuffer(20);
    int next = RandomUtils.nextInt(13) + 8;
    password.append(RandomStringUtils.randomAlphanumeric(next));
    return password.toString();
  }

  public static void main(String[] args){
    PassordGenerator pg = ne PassordGenerator();
    pg.getRandomPassword();
  }
}
```

Interpreter vs Compiler

- Interpreter interprets instructions on the fly
- Python interpreter is interactive
  - Write a line of code
  - Python process imideately
  - Python wait for new commando
- Interactive conversation
- Compiler must know the whole program
- Translate to static machine language for later execution
- Creates *.exe or *.dll in Windows

Variables

- Type inference as in Bash

Bash:

```
age=32
echo $age
```

Python:

```
age = 32
text = "Hei"
print(age)
```

Reading from users

Bash:

```
echo "Enter a number"
read a
```

Python

```
a = input("Enter a number")
```

Indenting and ifs (2)

- No curly braces

```
if(age>32):
    print("You are old")
elif (age<12):
    print("You are too young")
else:
    print("You are just right")
```

List / Arrays

Bash (array):

```
os=('linux' 'mac')
for i in "${os[@]}"
do
    echo $i
done
```

Python (list):

```
os = ['linux','mac']
for i in os:
    print(i)
```

For loops

```
tekst = "abcdef"
for bokstav in tekst:
    print(tekst)
```

or

```
for i in range(0,10):
    print(i)
```

Dictionaries (Associative Arrays)

Bash:

```
declare -A user
user=([chrisb]="Christoffer Berglund" [sigurda]="Sigurd Assev")
for i in "${!user[@]}"
do
  echo "key  : $i"
  echo "value: ${user[$i]}"
done
```

Python:

```
passwordlist = {'chrisb':'ind2978e', 'sigurda':'hemmelig123'}
for user,password in passwordlist.items():
    print(user)
    print(password)
```

Appending list

```
l = [1,2,2,3]
l.append(3)
l.remove(2)
l.pop(0)
```

l.remove = removed the number 2, not index l.pop = removes index 0

Finding in lists

```
friends = ["Luke","Leia","Han","Chewbacca"]
if "Luke" in friends:
    print("Lucky you")
```

Sorting double list

```
l = [[1,"one"], [2, "two"], [3, "three"], [4, "four"]]

def f(lst):
    return lst[1]
l.sort(key=f)
print(l)

[[4, 'four'], [1, 'one'], [3, 'three'], [2, 'two']]
```

Tuples are immutable

```
a = (1,2)
a[0] = 2
```

Union

```
a = set([0,1,2,3])
b = set([1,2,3,4,5,6])
a.union(b)
b.difference(a)
a.difference(b)
a.intersection(b)
```

union = combine
    difference = print 4,5,6
    difference = print 0
    intersection = numbers that are in both

Pythonic iterations

```
range(10)
[i for i in range(10)]
[x**2 for x in range(10)]
[i for i in range(10) if i%2==0]
```

Specialized sort

```
a = [(1, 2), (4, 1), (9, 10), (13, -3)]
a.sort(key=lambda x: x[1])
```

lambda

- One-line no-name function

Python lambda:

```
f = lambda x:x>2
```

Normal function equivalent:

```
def f(x):
    return x>2
```

Map-function

```
items = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x**2, items))
```

function - map() passes each item of the iterable to this function.

Reduce

```
from functools import reduce
product = reduce( (lambda x, y: x * y), [1, 2, 3, 4] )
```

- reduce will only work with 2 and 2 numbers, see python documentation

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()
```