

## SEDINTA 6 – MODULE AVANSATE ÎN PYTHON

---

### Modulul RE(expresii regulate)

---

O expresie regulata este o secventa speciala de caractere care te ajuta să gasesti într-un șir de caractere sau un set de șiruri de caractere un tipar. Se utilizeaza sintaxe speciale și este des intalnit în automatizari și testari.

Potrivirea caracterelor:

.

Se potriveste cu orice caracter.

\d

Se potriveste cu orice caracter zecimal

\D

Se potriveste cu orice caracter non-zecimal

\s

Se potriveste cu caracterul space

\S

Se potriveste cu orice caracter ce nu e space

\w

Se potriveste cu orice caracter alfanumeric.

\W

Se potriveste cu orice caracter non-alfanumeric

Exista și mecanisme repetitive:

Primul caracter repetitiv este \*. Acesta repeta de 0 sau mai multe ori un caracter sau un șir de caractere .

Spre exemplu sc\*apa va avea potrivire și cu sapa(c= 0), scapa(c=1), sccapa(c=2)...etc.

Exista o limitare interna care il impiedica să se repede de mai mult de 2 miliarde de ori.

Daca dorim să testam un șir de caractere in locul unui caracter care să se repete de cate ori este nevoie atunci folosim paranteze rotunde. `o(ma)*` va avea potrivire și cu `o` (`ma=0`), și cu `oma` (`ma=1`) și cu `omama` (`ma=2`) etc.

Un caracter repetitiv este `+`. Acesta repeta de 1 sau mai multe ori un caracter sau un șir de caractere.

Caracter repetitiv ? are rolul de a testa repetitia unui caracter sau unui șir de caractere de 0 sau o singura data.

Cel mai greu mecanism repetitiv este `{m,n}`, unde `m` este de cate ori trebuie minimum să se repete un caracter sau un șir de caractere, iar `n` reprezinta numarul maxim de repetari. Dacă am avea expresia regulata `c1{1,2}b` acesta s-ar potrivi doar cu `c1b` și `c11b`.

Exista și variatiuni ale caracterelor speciale cu caracterele repetitive:

ex: `*`. - zero sau mai multe caractere (oricare).

De asemenea ar trebui să știm ca dacă dorim să cautăm un caracter special, trebuie să folosim caracterul `/` (secventa de evadare) care anuleaza acel caracter special ca în interiorul oricarui șir de caractere unde doream să afișăm " (ghilimele duble) sau ' (ghilimele simple). Daca nu iti amintesti te rog reciteste materialele ce au fost studiate in sedinta 1.

Exista și doua functii ale modulului care pot fi utilizate pentru a cauta prin sirurile de caractere dorite:

- `re.search(pattern, string)`

pattern- șir de caractere pentru potrivire

string- în ce șir să caute.

Returneaza `None` daca nu gaseste nimic.

`re.search()` cauta tiparul dat pe fiecare linie in cazul in care sirul de caractere este pe mai multe linii

`re.search()` cauta tiparul dat prin tot randul

- `re.match(pattern, string)`

pattern- șir de caractere pentru potrivire

string- în ce șir să caute.

Returneaza `None` daca nu gaseste nimic.

`re.match()` cauta tiparul dat pe o singura linie standard (dar poate cauta si pe multiple linii) in cazul in care sirul de caractere este pe mai multe linii

`re.match()` cauta tiparul dat doar la inceputul liniei

Mai jos putem gasi un exemplu unde folosim `match`.

```
# Program RE 1
# Explica regular exp
# Ion Studentul - 1/26/13

import re

sir = """Aseara am mancat un hotdog mare."""

m = re.search("(d\\w+)", sir)
if m:
    print(m.groups())
```

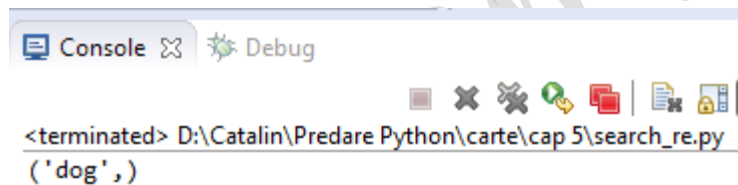
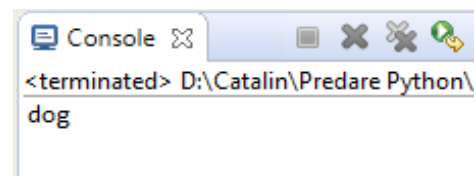


Fig. 11

Asa cum se poate vedea în Fig. 11 programul returneaza cuvantul gasit ce incepe cu d.

Modulul `re` este foarte intalnit în testare, unde trebuie să cautam în output-ul testelor cuvinte precum fail sau error. Am putea inlocui in acel if sa ne returneze doar cuvantul cu care a gasit potrivirea. Acest lucru se poate face astfel:

```
if m:
    print(m.group(0))
```



Daca folosesc groups() tot mereu ar trebui sa verific daca sirul cautat este intre paranteze rotunde.

Daca folosesc group(0) nu sunt obligat sa folosesc paranteze rotunde pentru a cauta un sir. In imaginea alaturata puteti vedea ce se intampla daca nu folosesc paranteze rotunde la un search. Prin urmare e recomandat ca de fiecare data sa folosesc group(0) pentru a returna potrivirea gasita.

```
>>> x = re.search("ac(\w)*",sir)
>>> x.group(0)
'acasa'
>>> x.group(1)
'a'
>>> x.groups()
('a',)
>>> x = re.search("(ac(\w)*)",sir)
>>> x.groups()
('acasa', 'a')
>>> x.group(0)
'acasa'
>>>
```

O alta functie foarte folosita este split.

Iata sintaxă pentru split:

```
re.split(pattern, string)
```

pattern- şir de caractere pentru potrivire

string- în ce şir să caute.

Aplicarea re.split returneaza o lista cu trei elemente unde al doilea element este potrivirea tiparului.

```
# Program RE 2
# Explica regular exp
# Ion Studentul - 1/26/13

import re

sir = """Aseara am mancat un hotdog mare."""

m = re.split("(d\w+)", sir)
if m:
    print m
```

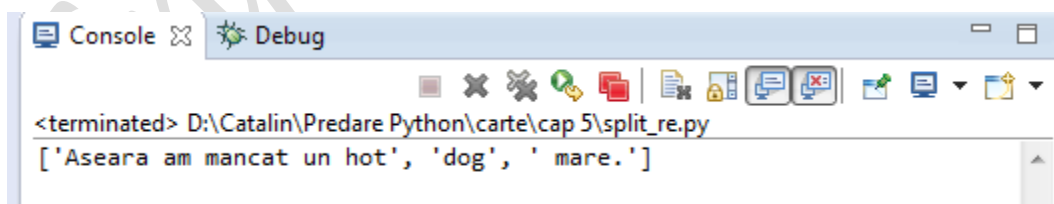


Fig .12

Asa cum se poate vedea functia split returneaza o lista cu elementele despartite de pattern.

Inlocuirea de text se poate face cu functia sub. Aceasta are sintaxa:

`re.sub(pattern,replace, string)`

pattern- șir de caractere pentru potrivire

replace – ce inlocuim

string- cu ce inlocuim.

Iată în Fig. 13 și un exemplu de utilizare a `re.sub()` ce returnează șirul unde s-a aplicat înlocuirea.

```
>>> a = """ata
asa
ara"""
>>> inloc = re.sub("a.a", "info", a)
>>> print inloc
info
info
info
>>>
```

Fig.13

Ultimul exemplu este un `re.search` unde tiparul este caracterul paranteză deschisă `<<(>>` ce este un caracter special ce are rolul de grupare. Dacă dorim să anulăm acest rol trebuie să adăugăm un backslash în fața acestui caracter.

```
>>> sir = " test (infoacademy) salut "
>>> x = re.search("\(\w+", sir)
>>> x.group(0)
'(infoacademy'
```

Fig.14

Doresc să subliniez că Python are și alte funcționalități integrate în modulul `re`. Mai multe informații găsiți [aici](#) în cazul în care doriți să extindeți cunoștințele voastre.

## Modulul socket

---

Python ofera doua nivele de acces al serviciilor de retea. La nivelul de jos poti accesa suportul de baza pentru lucrul cu porturi de access (socket). Accesata modalitate permite flexibilitatea de a implementa servere sau/si clienti pentru portocoalele orientate pe conexiune(TCP) sau cele neorientate pe conexiune(UDP).

Majoritatea traficului de retea (indrasnesc să spun 99% din tot traficul unei retele) are în spate unul din protocoalele TCP sau UDP. Acestea fac parte dintr-o suita de protocoale numita TCP/IP stack. Aceste doua protocoale determina cum curg datele intr-o sesiune de comunicare. Prin urmare, în cele ce urmeaza vom discuta un pic diferente și asemanari dintre TCP și UDP, dar și despre aplicatiile de retea.

Aplicatiile de retea (network-aware) sunt impartite dupa mai multe categorii:

- Dupa interactiunea cu utilizatorul:
  - GUI (Graphical User interface) – interfata vizuala cu butoane
  - CLI (Command Line Interface) – Aplicatie de tip consola sau terminal prin care interactiunea cu userul se face prin argumente(numite și parametrii).
- Dupa interactiunea cu sistemul de operare:
  - Ruleaza folosind sistemului grafic și interactiunea se face direct
  - Ruleaza ca un serviciu al sistemului de operare și interactiunea cu aceasta aplicatie se face indirect (fie printr-o alta aplicatie, fie fara interactiune fiind o aplicatie ce doar trebuie oprita sau pornita). De cele mai multe ori aceasta aplicatie este subordonata aplicatiei parinte ce are o interfata grafica.
- Dupa interactiunea cu alte dispozitive:
  - Client-Server. Un dispozitiv are o aplicatie ce este mereu pornita și asteapta cereri de la Client. Acest dispozitiv se numeste Server.
  - Peer- to-Peer (P2P). Nu exista un server ce asteapta mereu dupa cererile clientului, ci mai degraba rolurile sunt inversate dupa necesitate.

In primul rand trebuie să retinem ca aplicatiile sunt construite pt. a utiliza doar unul din aceste protocoale TCP sau UDP (rar amandoua simultan, aceste aplicatii din urma fiind aplicatii create pt. a mentine retea, astfel se incearca să mareasca posibilitatea de a fi contactat serverul intr-una din modalitati).

Aceste aplicatii folosesc un socket sau un port number(numar de port) în ambele cazuri (TCP sau UDP). Aceste socket-uri sunt utilizate pentru a putea multiplexa conexiunile logice pe baza unei singure conexiuni fizice. O astfel de multiplexare este deschiderea unor tab-uri multiple catre aceasi pagina intr-un browser. Exista porturi standard pentru aplicatii. Mai jos regasim un tabel cu cele mai utilizate tipuri de aplicatii și socket-urile lor standard.

Protocol	Tip aplicatie	Socket
TCP	HTTP	80
TCP	HTTPS	443
TCP	DNS	53
TCP	E-MAIL	25,110,143
TCP	FTP	20,21
TCP	SSH	22
TCP	TELNET	23
UDP	DNS	53
UDP	NTP	123
UDP	SNMP	161,162

Tabelul 1

Libraria socket ofera clase specifice pentru administrarea transportului informatiei și administrarea conversatiei.

Socket-urile au termeni specifici:

TERMEN	DESCRIERE
<b>FAMILIE</b>	Indica familia de protocoale ce este utilizată ca mecanism de adresare. Acestea sunt constante precum AF_INET, AF_INET6 sau AF_UNIX. AF_INET este utilizat pentru IPv4 – comunicare între dispozitive. AF_INET6 este utilizat pentru IPv6 – comunicare între dispozitive. AF_UNIX este utilizat pentru conexiuni interne (comunicarea între două procese).
<b>SOCKET TYPE</b>	Tipul comunicării între cele două dispozitive. Uzual vom utiliza SOCK_STREAM pentru TCP și SOCK_DGRAM pentru UDP.
<b>PROTOCOL</b>	Uzual zero; Aceasta valoare poate fi utilizată pentru a identifica diferite protocoale din interiorul domeniului și tipul acesteia.
<b>HOSTNAME</b>	Identificatorul unei interfete de rețea: <ul style="list-style-type: none"> <li>Un șir de caractere, ce identifică acel dispozitiv în rețea, se regăsește de formă X.X.X.X unde x este un număr între 0-255 (Atenție! Există și adrese exceptate de la utilizare). Aceasta adresă se numește adresă IPv4. Putem utiliza și o adresă IPv6 ce e de formă Y:Y:Y:Y:Y:Y:Y:Y unde Y este un grup de 4 caractere hexazecimale</li> </ul>

	<ul style="list-style-type: none"> <li>• Un șir de caractere "&lt;broadcast&gt;", ce specifica o adresa INADDR_BROADCAST. Aceasta adresa este speciala în sensul ca comunica cu toti care pot primi acest mesaj.</li> <li>• Un șir de caractere gol(de zero caractere)acesta specifica INADDR_ANY adica orice adresa.</li> <li>• Un numar binar de tip 0101, interpretat ca o adresa binara.</li> </ul>
<b>PORT</b>	Fiecare server asculta dupa cererile clientilor pe unul sau mai multe poturi.Un port number (numit si socket) poate fi un numar fix sau poate fi un nume de de aplicatie(ca serviciu exemplu http, ftp).

Tabelul 2

Pentru a crea un socket trebuie să importam modulul socket, apoi să utilizam functia socket disoponibilila în acest modul. Avem urmatoarea sintaxă generala:

```
s = socket.socket (socket_family, socket_type, protocol=0)
```

Iata o descriere a parametrilor:

- **socket\_family:** Acesta este fie AF\_UNIX, AF\_INET sau AF\_INET6 cum am explicat mai devreme.
- **socket\_type:** aceasta optiune poate fi SOCK\_STREAM sau SOCK\_DGRAM.
- **protocol:** Aceasta optiune este 0.

Dupa ce am creat un obiect *socket* poti utiliza functiile necesare pt a crea programul client sau server. Mai jos putem regasi o lista a acestor functii:

### Metode Server Socket:

Metoda	Descriere
s.bind()	Aceasta metoda leaga adresa (hostname, perechi de port number) cu socketul.
s.listen()	Aceasta metoda seteaza și porneste TCP listener – un mic serviciu de ascultare a mesajelor de pe retea.
s.accept()	Aceasta metoda accepta pasiv o conexiune, asteptand pana conexiunea TCP este primita (blocking).

### Metode Client Socket:

Metoda	Descriere
s.connect()	Aceasta metoda initiaza o conexiune TCP cu un server.



**Metode Generale Socket:**

Metoda	Descriere
s.recv()	Aceasta metoda primeste mesaje TCP
s.send()	Aceasta metoda trimite mesaje TCP
s.recvfrom()	Aceasta metoda primeste mesaje UDP
s.sendto()	Aceasta metoda trimite mesaje UDP
s.close()	Aceasta metoda inchide socket-ul
socket.gethostname()	Aceasta metoda returneaza hostname-ul.

Sa cream exemple cu aceste functii. Pentru a scrie un program pentru servere de internet trebuie să utilizam functia socket pt. a crea un obiect. Cu ajutorul acestui obiect putem controla toate functiile din modul. La acest obiect trebuie să apelam functia bind pentru a specifica un port pentru serviciul dorit. Apoi trebuie să apelam functia accept de fiecare data cand acest program primeste o conexiune de la un client. Functia accept returneaza și un obiect unic pentru a controla acea conexiune.

Server.py:

```
# Program Socket program1 Server
# Explica functiile Socket
# Ion Studentul - 1/26/13
host = "192.168.1.5"

print "Program cerere asistenta client"
import socket          # importa modulul socket

s = socket.socket()     # Creaza un obiect socket

port = 12345            # Reserva un port local pentru serviciul taufor your
                        # service.
s.bind((host, port))    # setam legatura(bind-ul)

s.listen(5)             # Acum asteptam conexiunea clientului.
                        # Argumentul specifica numarul maxim de conexiuni de
                        # asteptare in coada si
                        # ar trebuie sa fie mai amre de 0 (Uzual 5).

while True:
    conex_client, adresa = s.accept() # Stabileste o conexiune cu clientul.
    print 'Am primit solicitare de la ', adresa
    conex_client.send('Iti multumim pentru cerere!')
    conex_client.send('Am inregistrat cererea ta si vom raspunde cat mai repede!')
    conex_client.close()              # Inchidem conexiunea

raw_input("Apasa enter sa iesi")
```

Client.py:

```
# Program Socket program1 Client
```

```
# Explica functiile Socket
# Ion Studentul - 1/26/13

print "Program cerere asistenta client\n"
import socket          # importa modulul socket

s = socket.socket()     # Creaza un obiect socket

host = "192.168.1.6"    # Vom trimite cererea al un server dat
port = 12345           # Rezerva un port pentru aplicatie

s.connect((host, port)) # Solicitam conectarea la serverul dat
print s.recv(1024)     # Afisam mesajele trimise
print s.recv(1024)     # Afisam mesajele trimise
s.close                # Inchidem conexiunea

raw_input("Apasa enter sa iesi")
```

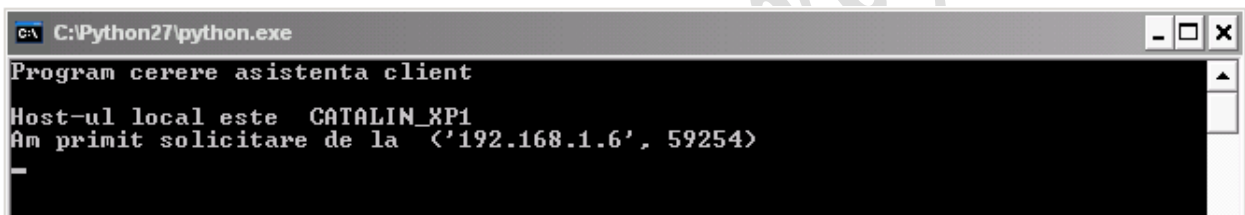


Fig.14

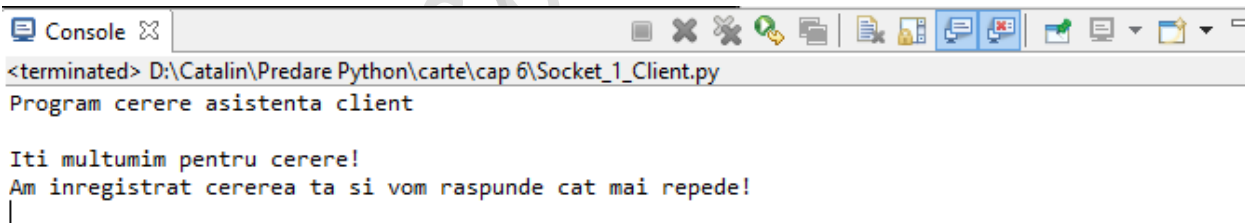


Fig.15

Chiar dacă putem crea multiple programe folosind socket trebuie să reținem că în cadrul comunității Python s-au realizat multiple module ce simplifică programarea acestora pentru multe din protocoalele știute. De aceea am să insist pe aceste module pentru a putea crea ce aplicații aveți nevoie. Totuși trebuie reținut că modulul socket stă la baza fiecărui modul despre care o să vorbim în secțiunea ce urmează și are ca rol comunicarea dintre dispozitive. De asemenea trebuie să fiți conștienți de modulul socket și de capacitățile lui. Acest modul va fi folosit cu siguranță dacă creați un tip de aplicație nouă cu socket-uri(port number) non-standard.

## NTP

NTP (Network Time Protocol) este un serviciu de retea ce are ca scop sincronizarea ceasului diferitelor dispozitive de retea. Dar de ce este atat de important timpul incat sa-l sincronizam? Oare nu este o cerinta exagerata, chiar de multe ori inutila? Din pacate este o necesitate să sincronizam dispozitivele de retea și asta datorita dinamicitatii rețelei. Problema este ca dacă facturezi, spre exemplu, traficul pe secunda, poti ajunge în situatii neplacute cand apar desincronizari ale echipamentelor. Pentru nu ajunge în astfel de situatii exista acest serviciu numit NTP.

Pentru mai multe detalii despre NTP accesati:

<http://tools.ietf.org/html/rfc958>

Python are de asemenea și un modul pentru NTP numit ntplib. Acesta este disponibil de pe pagina:

<https://pypi.python.org/packages/source/n/ntplib/ntplib-0.3.2.tar.gz>

Dupa instalare trebuie să importam modulul ntplib.

Pentru a sincroniza ceasul utilizam un server local din Romania. Mai multe detalii despre server regasisti accesand link-ul:

<https://support.ntp.org/bin/view/Servers/PublicTimeServer000389>

Iata cum solicitam timpul de la un server NTP din Romania.

```
# Program NTP Client
# Explica functiile ntplib
# Ion Studentul - 1/26/13

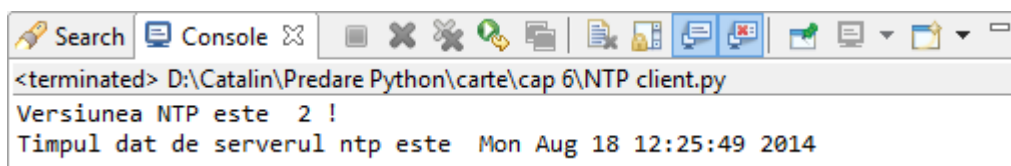
#importam modulele utilizate
import ntplib
import time

#cream un obiect ntplib de tip client
obj = ntplib.NTPClient()

rasp = obj.request("ntp3.usv.ro", 2 , 123, 5)

print "Versiunea NTP este ",rasp.version,"!"

print "Timpul dat de serverul ntp este ", time.ctime(rasp.tx_time)
```



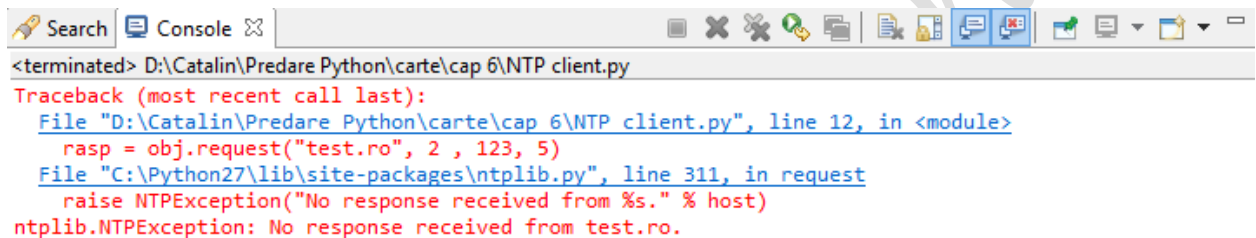
```
<terminated> D:\Catalin\Predare Python\carte\cap 6\NTP client.py
Versiunea NTP este 2 !
Timpul dat de serverul ntp este Mon Aug 18 12:25:49 2014
```

Fig.16

Cererea catre serverul NTP contine patru parametrii:

- host – Aici este ntp3.usv.ro. Reprezinta serverul ntp de la care solicitam informatia.
- Versiunea – Aici este utilizată varianta 2.
- Port – Aici este utilizat port no. NTP standard, adica 123.
- Timeout – Aici este utilizată valoarea 5. Reprezinta dupa cat timp va genera un mesaj de timeout.

In cazul în care host-ul pe care il apelam sau port no. este gresit atunci functia request va ridica o eroare ce poate intrerupe rularea programului.



```
<terminated> D:\Catalin\Predare Python\carte\cap 6\NTP client.py
Traceback (most recent call last):
  File "D:\Catalin\Predare Python\carte\cap 6\NTP client.py", line 12, in <module>
    rasp = obj.request("test.ro", 2, 123, 5)
  File "C:\Python27\lib\site-packages\ntplib.py", line 311, in request
    raise NTPException("No response received from %s." % host)
ntplib.NTPException: No response received from test.ro.
```

Fig. 17

Acelasi rezultat il vom avea si daca serverul nu raspunde. Utilizati aces modul intregat intr-o constructie de tip try-else.

## Telnet

Modulul telnet lib ofera, de asemenea, access la dispozitive remote prin intermediul aplicatiei de tip telnet. Acest modul, ca și cel precedent, are la baza modulul socket.

Creăm un obiect cu ajutorul telnetlib.Telnet(HOSTNAME) unde HOSTNAME poate o adresa IP.

Dupa crearea acestui obiect putem manipula comunicatia prin metodele read\_until() si write().

Sa vedem un exemplu în care utilizam modulul telnetlib.

```
# Program Telnet Client
# Explica functiile telnetlib
# Ion Studentul - 1/26/13
```

```
import telnetlib
```

```
HOST = "192.168.1.50"
user = "user"
password = "cisco"
```

```

telnet_obj = telnetlib.Telnet(HOST)

telnet_obj.read_until("Login: " , 5)
telnet_obj.write(user + "\r\n")

telnet_obj.read_until("Password: " ,5)
telnet_obj.write(password + "\r\n")

print telnet_obj.read_until(">",10)

telnet_obj.write("dir"+ "\r\n")

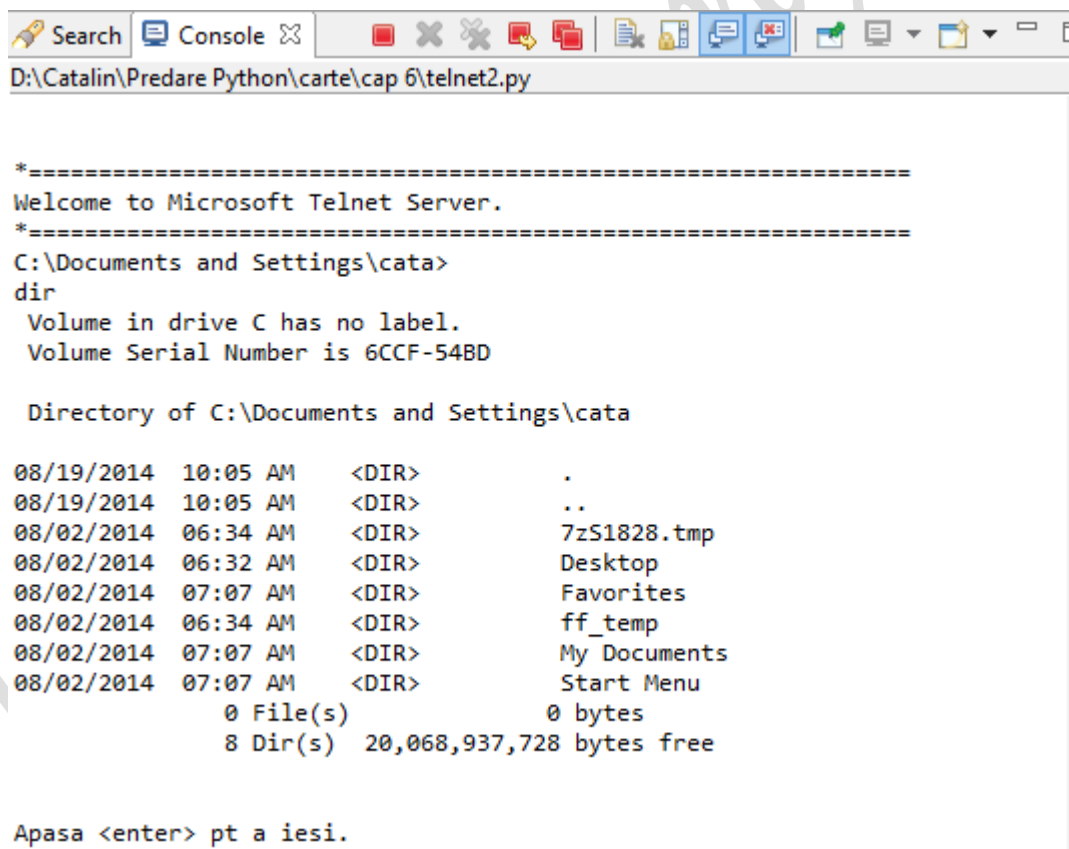
print telnet_obj.read_until("free",10)

telnet_obj.write("exit" + "\r\n")

telnet_obj.close()

raw_input("\n\nApasa <enter> pt a iesi.")

```



```

=====
Welcome to Microsoft Telnet Server.
=====
C:\Documents and Settings\cata>
dir
Volume in drive C has no label.
Volume Serial Number is 6CCF-54BD

Directory of C:\Documents and Settings\cata

08/19/2014  10:05 AM    <DIR>          .
08/19/2014  10:05 AM    <DIR>          ..
08/02/2014  06:34 AM    <DIR>          7zS1828.tmp
08/02/2014  06:32 AM    <DIR>          Desktop
08/02/2014  07:07 AM    <DIR>          Favorites
08/02/2014  06:34 AM    <DIR>          ff_temp
08/02/2014  07:07 AM    <DIR>          My Documents
08/02/2014  07:07 AM    <DIR>          Start Menu
               0 File(s)                0 bytes
               8 Dir(s)  20,068,937,728 bytes free

Apasa <enter> pt a iesi.

```

Fig. 18

Asa cum se poate vedea în Fig. 18 rulara acestui program va genera un output al comenzii dir în windows.

Incepem programul prin a importa modulul telnetlib.

In urmatoarele linii putem definirea variabilelor HOST(dispozitiv accesat), user(user folosit la conectarea prin telnet) și password (parolca folosita pentru conectarea la dispozitivul remote).

Vom crea un obiect telnet prin apelarea comenzii `telnetlib.Telnet(HOST)`.

Sintaxa `telnet_obj.read_until("sir de caractere",numar)` are rolul de a astepta un text returnat de la dispozitivul la care ne-am conectat. Sirul de caractere reprezinta punctul în care asteptarea se termina deoarece la intalnirea acelui șir de caractere aceasta comanda afiseaza totul. Dacă dorim să asptepte dupa acest șir de caractere putem seta un numar de tip integer ce preprezinta secunde de asteptare. Daca nu exista nici o potrivire cu acel șir de caractere returneaza tot ce a primit în schimb; posibil un șir de caractere gol. Va ridica eroarea `EOFError` dacă conexiunea este inchisa.

Sintaxa `telnet_obj.write(buffer)` are rolul de a scrie catre socket tot sirul de caractere din variabila `buffer`.

Programul de mai sus este o combinatie a acestor doua comenzi pentru a ajunge la rezultatul scontat. Ultima comanda trimisa este `exit` pentru a inchide conexiunea telnet.

Ulterior sintaxă `telnet_obj.close()` este apelata pentru a inchide conexiunea.

## FTP

---

Modulul `ftplib` din Python permite programatorilor să scrie programe care realizeaza o varietate de task-uri FTP automate, și asta datorita faptului ca te poti conecta cu usurinta la un server FTP pt. a extrage fisiere și a le procesa local.

Pentru a utiliza `ftplib` în Python trebuie mai intai să importam modulul. Acest modul este unul standard, deci poate fi importat fara a fi nevoie de o instalare în prealabil.

Pentru a crea un obiect `ftplib` să apelam comanda: `ftp = ftplib.FTP('HOST')`. Așa cum ati ghicit `HOST` poate fi o adresa IP sau numele dispozitivului. Prin apelarea [`ftp.login\(\)`](#) se realizeaza conectarea la serverul `ftp` declarat la crearea obiectului.

Cu ajutorul comenzii `ftp.retrlines('list')` extragem fisierele din serverul `ftp`.

Daca dorim să inchidem conexiunea cu serverul `ftp` și să stergem obiectul `ftp` atunci apelam `ftp.close()`.

Mai jos se regaseste o captura cu utilizarea de baza a modulului `ftplib`.

```
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import ftplib
>>> ftp = ftplib.FTP('192.168.1.50')
>>> ftp.login()
'230 Anonymous user logged in.'
>>> ftp.retrlines('LIST')
01-13-06 02:42AM                28521 Blue hills.jpg
08-19-14 06:48PM                <DIR>      poze
01-13-06 02:16AM                71189 Sunset.jpg
01-13-06 02:54AM                83794 Water lilies.jpg
01-13-06 02:24AM               105542 Winter.jpg
'226 Transfer complete.'
>>> ftp.close()
>>>
```

Fig. 19

În cele ce urmeaza vom vedea cum putem extrage un fisier de pe un server ftp.

```
# Program ftp Client
# Explica functiile telnetlib
# Ion Studentul - 1/26/13

import ftplib

filename = "Sunset.jpg"

ftp = ftplib.FTP('192.168.1.50')

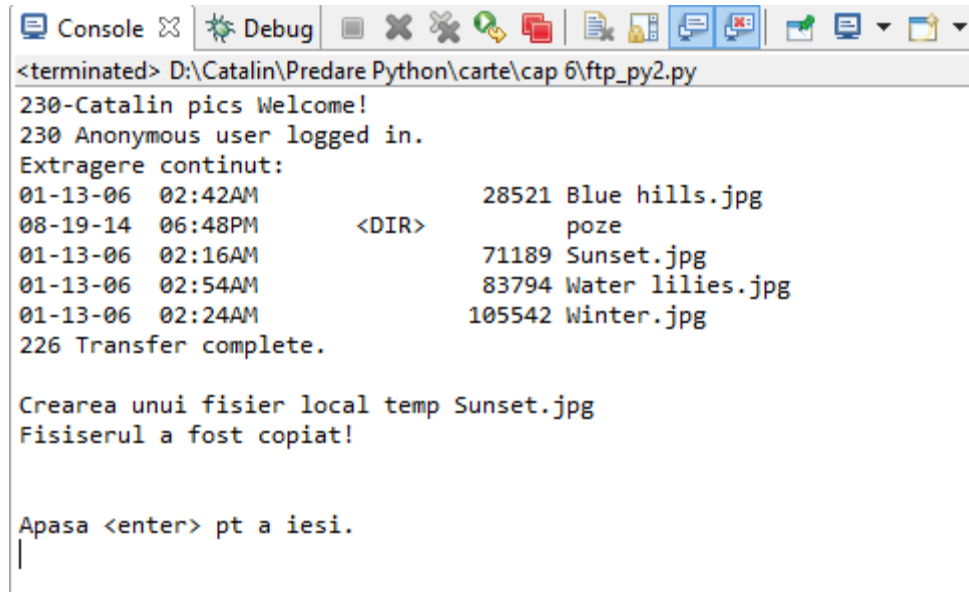
print ftp.login()

print "Extragere continut:\n",ftp.retrlines('LIST')

print '\nCrearea unui fisier local temp ' + filename

fisier = open(filename, 'wb')
ftp.retrbinary("RETR "+ filename ,fisier.write)
fisier.close()
print "Fisierul a fost copiat!"
ftp.close()

raw_input("\n\nApasa <enter> pt a iesi.")
```



```

<terminated> D:\Catalin\Predare Python\carte\cap 6\ftp_py2.py
230-Catalin pics Welcome!
230 Anonymous user logged in.
Extragere continut:
01-13-06 02:42AM                28521 Blue hills.jpg
08-19-14 06:48PM                <DIR>         poze
01-13-06 02:16AM                71189 Sunset.jpg
01-13-06 02:54AM                83794 Water lilies.jpg
01-13-06 02:24AM               105542 Winter.jpg
226 Transfer complete.

Crearea unui fisier local temp Sunset.jpg
Fisierul a fost copiat!

Apasa <enter> pt a iesi.

```

Fig. 20

Se poate observa ca trebuie să cream un obiect fisier. Acest obiect fisier deservește pentru a putea face o copie locala a fisierului de pe server (*fișier = open(filename, 'wb')*)

De asemenea putem vedea ca apeland comanda de mai jos putem extrage informatiile detinute de fisierul de pe server. Apoi aceste informatii vor fi copiate continutul din fisierul de pe server intr-un fisier local setat în obiectul fisier.

```
ftp.retrbinary("RETR "+ filename ,fișier.write)
```

## HTML

Internetul a devenit atat de integrat in vietile noastre incat este posibil să gasim aproape orice pe internet. Aceste pagini sunt scrise în asa fel incat paginile pot fi incarcate pe diferite platforme și programe de tip browser web. Trebuie să fim constienti ca un browser web pentru telefo poate avea resurse hardware mult mai mici decat un laptop.

Prin urmare limbajul http este pe cat se poate de flexibil. Exista doua tipuri de module în ceea ce priveste http. Sunt modulele ce pot oferi access paginilor web (fie client http, fie server http) și exista module care creaza paginile web cu ajutorul Python (creand un cadru ce faciliteaza dezvoltarea rapida de site-uri).

Pentru a crea un mic client ce extrage automat pagini web putem folosi modulul `httplib`. Deci trebuie și putem să importam acest modul în mod direct deoarece este un modul standard. Deoarece `httplib` (este denumit `http.client` în Python vers 3.x ) este doar un modul utilizat ca background pentru `urllib`, ar trebui să ne îndreptăm atenția către `urllib`. Totuși datorită unei documentații destul de reduse pentru multiplele tipuri de task-uri necesare unui modul de tip `http client`, ne îndreptăm atenția către modulul `resources`.



Acesta are la baza urllib, dar detine o documentatie pentru fiecare caz intalnit de un http client, prin urmare se poate utiliza cu incredere pentru a construi un web browser.

Urmeaza să facem o cerere catre un domeniu unde vom aplica diferite metode catre un obiect de tip requests. Domeniul trebuie specificat ca un șir de caractere cand cream obiectul adica sub formă unui URL. Un Uniform Resource Locator este o secvență de caractere standardizată, folosită pentru denumirea, localizarea și identificarea unor resurse de pe Internet, inclusiv documente text, imagini, clipuri video, expuneri de diapozitive, etc Schema care este folosită este:

<protocol>://<nume\_DNS>/<nume\_local>

unde

protocol - este protocolul folosit (de cele mai multe ori http),

nume\_DNS - este numele domeniului pe care se află resursa,

nume\_local - este format din calea și numele resursei de pe discul local

Pentru a descarca requests puteti accesa link-ul:

<https://pypi.python.org/pypi/requests>

Instalarea se realizeaza ca si modulul ntplib adica utilizati „setup.py install”

```
requests-2.5.1\requests-2.5.1>setup.py install
running install
running bdist_egg
running egg_info
writing requirements to requests.egg-info\requires.txt
writing requests.egg-info\PKG-INFO
writing top-level names to requests.egg-info\top_level.txt
writing dependency_links to requests.egg-info\dependency_links.txt
reading manifest file 'requests.egg-info\SOURCES.txt'
reading manifest template 'MANIFEST.in'
```

Exista și alti parametrii, dar sunt optionalii, cum ar fi port no. (standard 80 pentru http sau 443 pentru https) sau user/parola.

In acest exemplu simplu vom face o cerere GET, dar mai tarziu voi prezenta și alte tipuri de cereri.

```

Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import requests
>>> conex = requests.get("https://www.google.ro")
>>> conex = requests.get("https://www.google.ro:443")
>>> conex.content
'<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ro"><head><meta content="/images/google_favicon_128.png" itemprop="image"><title>Google</title><script>(function() {window.google={kEI:'\VhIU7HfGMjXyQP6uolcAQ', kEXPI:'\4791,17259,4000116,4003510,4007661,4008142,4009033,4010806,4010858,4010899,4011228,4011258,4011679,4012149,4012373,4013414,4013591,4013605,4013723,4013823,4013967,4013979,4014016,4014431,4014636,4014789,4014805,4014991,4015234,4015266,4015550,4015587,4016127,4016309,4016373,4016824,4016976,4017204,4017285,4017595,4017639,4017658,4017659,4017694,4017818,4017894,4017981,4017982,4018106,4018181,4018251,4018569,4018598,4018638,4018923,4018933,4018998,4019014,4019018,4019142,4019181,4019191,4019207,4019423,4019438,4019483,4019494,4019789,4019793,4019798,4019800,4019827,4019849,4019850,4019856,4019875,4019888,4020014,4020047,4020139,4020176,4020187,4020211,4020240,4020256,4020306,4020326,4020339,4020359,4020362,4020407,4020445,4020513,4020554,4020592,4020615,4020664,4020670,4020729,4020815,4020836,4020874,4020879,4020892,4020936,4021013,4021025,4021071,8300007,8300012,8300021,8300027,8300030,8300033,8300039,8300054,8300057,8300060,8300063,8300072,8300075,8500223,8500272,8500394,8500433,8500474,8500482,8500509,8500516,8500554,8500556,8500571,8500573,8500585,8500588,10200083,10200318,10200330,10200334,10200353,10200396,10200398,10200408,10200442,10200448,10200450,10200470,10200472,10200483,10200507,10200523,10200525,10200528,10200576,10200579,10200589,10200592'\, kCSI:{e:'\4791,17259,4000116,4003510,4007661,4008142,4009033,4010806,4010858,4010899,4011228,4011258,4011679,4012149,4012373,4013414,4013591,4013605,4013723,4013823,4013967,4013979,4014016,4014431,4014636,4014789,4014805,4014991,4015234,4015266,4015550,4015587,4016127,4016309,4016373,4016824,4016976,4017204,4017285,4017595,4017639,4017658,4017659,4017694,4017818,4017894,4017981,4017982,4018106,4018181,4018251,4018569,4018598,4018638,4018923,4018933,4018998,4019014,4019018,4019142,4019181,4019191,4019207,4019423,4019438,4019483,4019494,4019789,4019793,4019798,4019800,4019827,4019849,4019850,4019856,4019875,4019888,4020014,4020047,4020139,4020176,4020187,4020211,4020240,4020256,4020306,4020326,4020339,4020359,4020362,4020407,4020445,4020513,4020554,4020592,4020615,4020664,4020670,4020729,4020815,4020836,4020874,4020879,4020892,4020936,4021013,4021025,4021071,8300007,8300012,8300021,8300027,8300030,8300033,8300039,8300054,8300057,8300060,8300063,8300072,8300075,8500223,8500272,8500394,8500433,8500474,8500482,8500509,8500516,8500554,8500556,8500571,8500573,8500585,10200083,10200318,10200330,10200334,10200353,10200396,10200398,10200408,10200442,10200448,10200450,10200470,10200472,10200483,10200507,10200523,10200525,10200528,10200576,10200579,10200589,10200592'\, ei:'\VhIU7HfGMjXyQP6uolcAQ'\}, authuser:{}, googl
e.kHL='\ro\'});(function() {google.lc=[];google.li=0;google.getEI=function(a) {for (var b;a&&(!a.getAttribute('!')) {b=a.getAttribute('eid')});a=a.parentNode;return b}|go
ogle.kEI);google.https=function() {return"https://"+window.location.protocol;google.ml=function() {google.time=function() {return(new Date).getTime();google.log=functi
on(a,b,d,h,k) {var c=new Image,f=google.lc,e=google.li,g="";l=google.ls||"";c.onerror=c.onload=c.onabort=function() {delete f[e]};f[e]=c;d||-1|b.search("#ei")| (g="#ei
="+google.getEI(h));a=d||"/"+(k||"gen_204")+"?atyp=i&ct="+a+"&cad="+b+g+l+"&z="+google.time()}/http://i.test(a)&google.https()/?(google.ml(Error("a"), !1,{src:a, glmm:1
}), delete f[e]);(c.src=a, google.li+=1);google.y={};google.x=function(a,b) {google.y[a.id]=[a,b];return 1};google.load=function(a,b,d) {google.x({id:a,mm:1
}), google.load(a,b,d)};var m=0;})();var _gjlw=location;function _gjucc() {var a=_gjlw.href.indexOf("#");if (0<a&&(a=_gjlw.href.substring(a,0).indexOf("#&q="))|0<a.indexO
f("#&q="))&&(a=a.substring(1),-1==a.indexOf("#")) {for (var d=0;d<a.length;)(var b=d;"="+a.charAt(b)&&b;var c=a.indexOf("#&q="),b;-1==c&&(c=a.length);b=a.substring(b,c);
if (0==b.indexOf("#fp=")) a=a.substring(0,d)+a.substring(c,a.length),c=d;else if ("cad=h"==b) return 0;d=c;_gjlw.href="/search?"+a+"&cad=h";return 1}return 0}\nfunction _gj
h() {! _gjucc() &&window.google&&google.x({id:"GJH"},function() {google.nav&&google.nav.gjh&&google.nav.gjh()});window._gjh&&_gjh()};</script><style>#gbar,#user{
font-size:13px;padding-top:1px !important;#gbar{height:22px;#user{padding-bottom:7px !important;text-align:right}.gbb,.gbd{border-top:1px solid #c9d7f1;font-size:1px
}.gbb{height:0;position:absolute;top:2px;width:100%;#media all{.gbl{height:22px;margin-right:5em;vertical-align:top;#gbar{float:left}).a.gbl,a.gb4{text-decoration:und
erline !important}.a.gbl,a.gb4{color:#00c !important}.gbi,.gb4{color:#dd8e27 !important}.gbf,.gb4{color:#900 !important}</style><style>body,td,a,p,h{font-family:arial,
sans-serif}body{margin:0;overflow-y:scroll}#gog{padding:3px 8px 0;td{line-height:.9em}.gac_m td{line-height:17px}form{margin-bottom:20px}.h{color:#36c}.q{color:#00c}.t
s td{padding:0}.ts{border-collapse:collapse}em{font-weight:bold;font-style:normal}.lst{height:25px;width:496px}.gsfi,.lst{font:18px arial,sans-serif}.gsfs{font:17px ar
ial,sans-serif}.ds{display:inline-block;display:inline-block;margin:3px 0 4px;margin-left:4px}input{font-family:inherit}.a.gbl,a.gb2,a.gb3,a.gb4{color:#11c !important}bod
y{background:#fff;color:black}.a{color:#11c;text-decoration:none}.a:active,.a:active{text-decoration:underline}.fl a{color:#36c}.a:visited{color:#551a8b}.a.gbl,a.gb4{text-de
coration:underline}.a.gb3:hover{text-decoration:none}.ghead a.gb2:hover{color:#fff !important}.sblc{padding-top:5px}.sblc a{display:block;margin:2px 0;margin-left:13px;
font-size:11px}.lsbb{background:#eee;border:solid 1px;border-color:#ccc #999 #999 #ccc;height:30px}.lsbb{display:block}.ftl,.ftll a{display:inline-block;margin:0 12px}.

```

Fig. 21

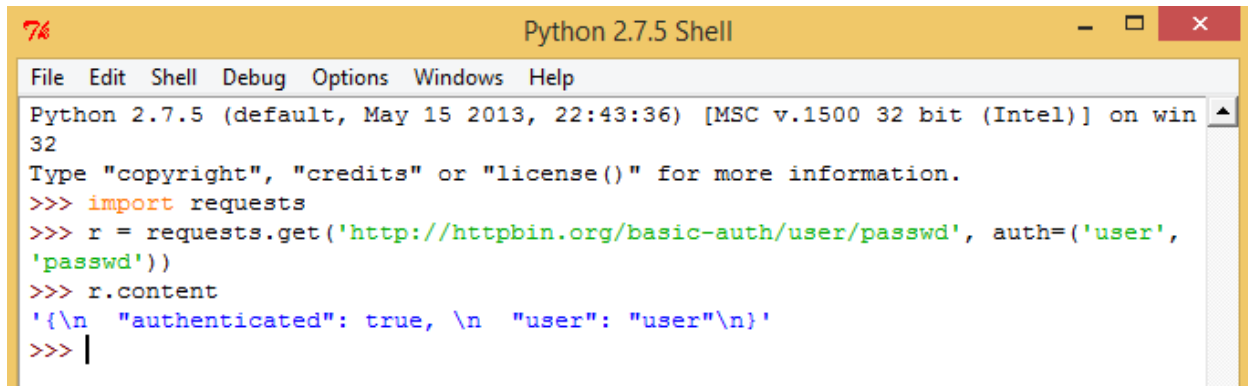
Asa cum se poate vedea variabilei conex putem sa-l aplicam alte metode, pt. a deserve orice scop. Spre exemplu, cu ajutorul metodei content putem vizualiza pagina extrasa de pe server. În cele ce urmeaza putem vedea pricipalele metode oferite de modulul requests.

Tip de cerere	Explicatie
<b>GET</b>	Metoda GET înseamnă prelua orice informație este identificata prin cerere. Dacă cererea se referă la un procedeu de producere a datelor, va fi returnat ca entitatea unui raspuns și nu textul sursă a procesului.
<b>POST</b>	Metoda POST este folosita pentru a solicita ca serverul să accepte informatii incluse în cerere. POST este conceput pentru a permite o metodă uniformă pentru a acoperi următoarele funcții:  - Adnotarea resurselor existente;

	<ul style="list-style-type: none"> <li>- Postarea unui mesaj la un avizier, newsgroup, lista de discutii, sau un grup similar de articole;</li> <li>- Extinderea unei baze de date printr-o operație de adăugare.</li> </ul> <p>Metoda POST depinde de cerere deoarece folosește URL-ul pentru a transmite informația dorită de la utilizator către server.</p>
<b>HEAD</b>	<p>Metoda HEAD este identică cu GET cu excepția faptului că serverul nu trebuie să returneze un corp mesaj (message-body) în răspuns. Informația conținută în antetele HTTP (header) trebuie să fie identică cu informațiile trimise ca răspuns la o solicitare GET. Această metodă poate fi utilizată pentru obținerea de informație despre pagina web implicată fără a transfera body-ul paginii în sine. Această metodă este deseori folosită pentru testarea link-urilor pentru a determina valabilitatea, accesibilitatea, sau modificarea recentă.</p>
<b>OPTIONS</b>	<p>Metoda OPTIUNI reprezintă o cerere pentru informații despre opțiunile de comunicare disponibile pentru cereri sau răspunsuri. Această metodă permite clientului pentru a determina opțiunile și / sau cerințele asociate cu o resursă, sau cu caracteristicile unui server, fără a presupune o acțiune de solicitare a resurselor.</p>
<b>DELETE</b>	<p>Metoda DELETE sterge o resursă alocată pe serverul web</p>
<b>PUT</b>	<p>Metoda PUT este similară cu POST în sensul că transmite informație către server în vederea prelucrării și stocării. Totuși această metodă diferă față de POST în sensul că PUT transferă efectiv fișierul selectat în cadrul unui URL către server pentru a putea fi accesat pe server chiar și de aplicații externe.</p>

Chiar dacă în exemplu de mai sus am utilizat URL-ul celebrului motor de căutare google, vom utiliza o altă pagină ce este special creată pentru testarea programelor de tip web browser, și anume <http://httpbin.org/>.

Spre exemplu, pentru a ne autentifica la un site cu ajutorul metodei get putem utiliza sintaxă ca în Fig. 22. Vizualizarea paginii extrase se poate face cu ajutorul metodei content sau text.



```

Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import requests
>>> r = requests.get('http://httpbin.org/basic-auth/user/passwd', auth=('user',
'passwd'))
>>> r.content
'{"\n  "authenticated": true, \n  "user": "user"\n}'
>>> |

```

Fig.22

Putem să determinăm dacă pagina a fost extrasă cu succes sau ce tip de eroare am întâmpinat după codul răspuns (status-code). Iată cele mai uzuale coduri întâlnite:

- 200 – Pagina a fost încărcată cu succes.
- 403 – Accesul este interzis (FORBIDDEN)
- 404 – Fișierul nu a fost găsit
- 408 – Cererea a expirat (request timeout)
- 307 – Pagina a fost temporar redirecțată (temporary\_redirect)
- 500 – INTERNAL\_SERVER\_ERROR
- 301 – MOVED\_PERMANENTLY

În Fig. 10 se poate vizualiza codul-răspuns pentru pagina solicitată în Fig.9. De asemenea, cu ajutorul `requests.codes` putem verifica starea încărcării unui fișier extras de pe serverul web fără ca programatorul să memoreze aceste coduri.

```

>>> r.status_code
200
>>> if (r.status_code == requests.codes.ok):
    print "pagina a fost încărcată cu succes!"

pagina a fost încărcată cu succes!
>>> requests.codes.INTERNAL_SERVER_ERROR
500
>>> requests.codes.REQUEST_TIMEOUT
408

```

Fig.23

Poți extrage și antetele răspuns ale serverelor (headers) ca un dicționar, deci foarte ușor de integrat. Dacă antetul căutat nu este găsit în acel dicționar (deci nu este returnat de către server) atunci va returna `None`.

```

>>> r.text
u'{\n  "authenticated": true, \n  "user": "user"\n}'
>>> r.headers
{'content-length': '46', 'server': 'unicorn/18.0', 'connection': 'keep-alive',
'access-control-allow-credentials': 'true', 'date': 'Wed, 20 Aug 2014 22:48:45 GMT', 'access-control-allow-origin': '*', 'content-type': 'application/json'}
>>> r.headers.keys()
['content-length', 'server', 'connection', 'access-control-allow-credentials', 'date', 'access-control-allow-origin', 'content-type']

```

Fig.24

Modulul requests permite să adaugi http headers la cererea ta prin adaugarea în dictionar a unei noi intrari. Pentru acest exemplu o să utilizam modulul JSON ce este utilizat pentru tehnologia cu același nume. Pentru a putea fi compatibil cu diferite sisteme de operare și web browsere tehnologiile web pot utiliza XML sau JSON cu scopul de a reprezenta obiecte și alte structuri de date. Acesta sunt folosite în special pentru a transmite date structurate prin rețea, procesul purtând numele de serializare.

JSON este un acronim în limba engleză pentru JavaScript Object Notation, și este un format de reprezentare de tip text, inteligibil pentru oameni. JSON este alternativa mai simplă, mai facilă decât limbajul XML. Eleganța formatului JSON provine din faptul că este un subset al limbajului JavaScript, fiind utilizat alături de acest limbaj. Formatul JSON a fost creat de Douglas Crockford și standardizat prin RFC 4627. Tipul de media pe care trebuie să îl transmită un document JSON este application/json. Extensia fișierelor JSON este .json.

Pentru a face o comparație a JSON cu XML să considerăm următoarele date reprezentate în ambele formate: nume: Ion Barbu, email: ionbarbu@yahoo.com, telefon: 0359454545, adresa: Oradea, strada Independentei nr 25.

În XML, datele se reprezintă în felul următor:

```

<?xml version='1.0' encoding='UTF-8'?>
  <person>
    <name>
      Ion Barbu
    </name>
    <email>
      ionbarbu@yahoo.com
    </email>
    <telephone>
      0359454545
    </telephone>
    <address>
      Oradea, strada Independentei nr 25

```

```

    </address>
  </person>

```

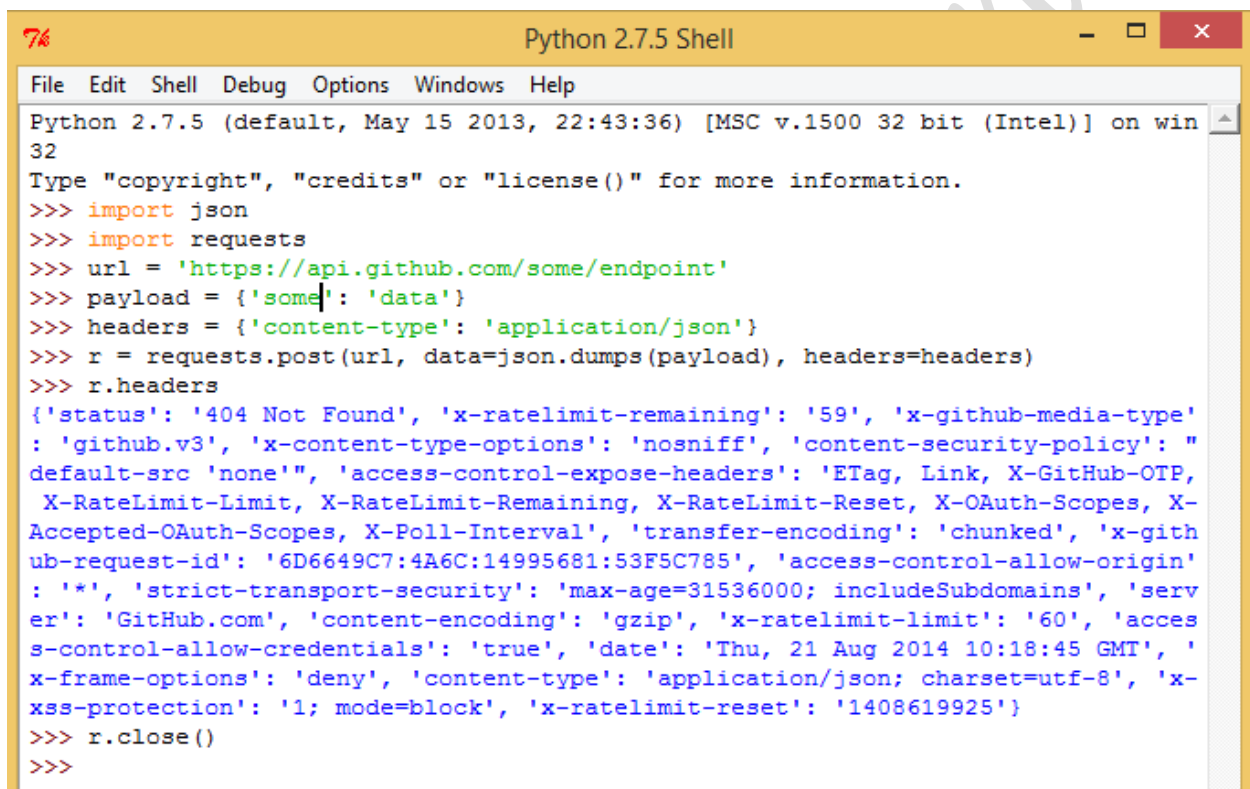
In JSON, reprezentarea este următoarea:

```

{
  "name": "Ion Barbu",
  "email": "ionbarbu@yahoo.com",
  "telephone": "0359454545",
  "address": "Oradea, strada Independentei nr 25"
}

```

Se poate vedea ca XML este mult mai voluminos, fapt ce-l face mai lent decat JSON.



```

Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import json
>>> import requests
>>> url = 'https://api.github.com/some/endpoint'
>>> payload = {'some': 'data'}
>>> headers = {'content-type': 'application/json'}
>>> r = requests.post(url, data=json.dumps(payload), headers=headers)
>>> r.headers
{'status': '404 Not Found', 'x-ratelimit-remaining': '59', 'x-github-media-type':
'github.v3', 'x-content-type-options': 'nosniff', 'content-security-policy': "
default-src 'none'", 'access-control-expose-headers': 'ETag, Link, X-GitHub-OTP,
X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset, X-OAuth-Scopes, X-
Accepted-OAuth-Scopes, X-Poll-Interval', 'transfer-encoding': 'chunked', 'x-gith
ub-request-id': '6D6649C7:4A6C:14995681:53F5C785', 'access-control-allow-origin'
: '*', 'strict-transport-security': 'max-age=31536000; includeSubdomains', 'serv
er': 'GitHub.com', 'content-encoding': 'gzip', 'x-ratelimit-limit': '60', 'acces
s-control-allow-credentials': 'true', 'date': 'Thu, 21 Aug 2014 10:18:45 GMT', '
x-frame-options': 'deny', 'content-type': 'application/json; charset=utf-8', 'x-
xss-protection': '1; mode=block', 'x-ratelimit-reset': '1408619925'}
>>> r.close()
>>>

```

Fig.25

Cu ajutorul metodei `close()` eliberam conexiunea deschisa cu serverul web.

Chiar dacă XML și JSON depasesc curriculum propus, trebuie să stim ca ele exista și pentru ce am putea să le utilizam. De asemenea, pentru procesarea de XML Python vine cu modulele standard DOM și SAX. Totusi este mult mai usoara serializarea (process numit și parsare) cu module instalabile ce au la baza modulele standard. Unul din acestea ar fi `lxml`.





```

r = requests.delete("http://httpbin.org/delete")
r = requests.head("http://httpbin.org/get")
r = requests.options("http://httpbin.org/get")

>>> r = requests.post("http://httpbin.org/post")
>>> r.status_code
200
>>> r = requests.put("http://httpbin.org/put")
>>> r.status_code
200
>>> r = requests.delete("http://httpbin.org/delete")
>>> r.status_code
200
>>> r = requests.head("http://httpbin.org/get")
>>> r.status_code
200
>>> r = requests.options("http://httpbin.org/get")
>>> r.status_code
200

```

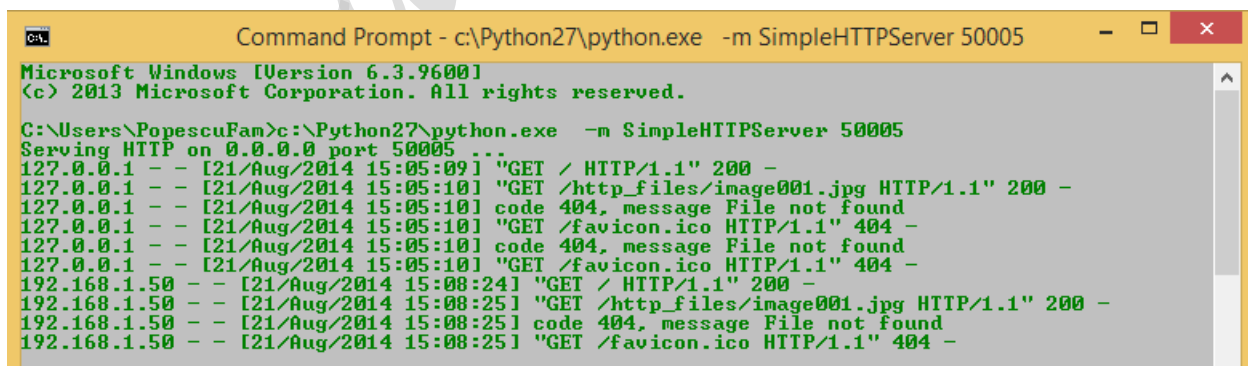
Fig.27

În ceea ce privește Python, acesta oferă multiple module pentru crearea de web servere. Cel mai simplu mod de a realiza un server web este modulul integrat SimpleHTTPServer.

Tot ce trebuie să faci este să navighezi în command prompt până la directorul ce susține pagina web index.html. Apoi trebuie să apelezi în windows:

c:\Python27\python.exe -m SimpleHTTPServer 50005

În linux trebuie să apelezi: # python exe -m SimpleHTTPServer 50005



```

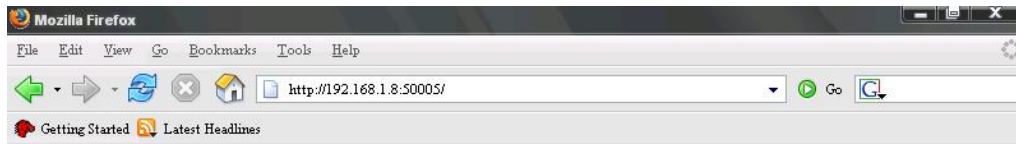
Command Prompt - c:\Python27\python.exe -m SimpleHTTPServer 50005
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\PopescuFam>c:\Python27\python.exe -m SimpleHTTPServer 50005
Serving HTTP on 0.0.0.0 port 50005 ...
127.0.0.1 - - [21/Aug/2014 15:05:09] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Aug/2014 15:05:10] "GET /http_files/image001.jpg HTTP/1.1" 200 -
127.0.0.1 - - [21/Aug/2014 15:05:10] code 404, message File not found
127.0.0.1 - - [21/Aug/2014 15:05:10] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [21/Aug/2014 15:05:10] code 404, message File not found
127.0.0.1 - - [21/Aug/2014 15:05:10] "GET /favicon.ico HTTP/1.1" 404 -
192.168.1.50 - - [21/Aug/2014 15:08:24] "GET / HTTP/1.1" 200 -
192.168.1.50 - - [21/Aug/2014 15:08:25] "GET /http_files/image001.jpg HTTP/1.1" 200 -
192.168.1.50 - - [21/Aug/2014 15:08:25] code 404, message File not found
192.168.1.50 - - [21/Aug/2014 15:08:25] "GET /favicon.ico HTTP/1.1" 404 -

```

Fig.28





Aceasta este o pagina de test pentru Serverul HTTP din Python!!!



Fig.29

## Trimiterea unui Email

Simple Mail Transfer Protocol (SMTP) este un protocol care se ocupa de transmiterea de e-mailuri și forwardarea de emailuri inter serverele de email.

Python ofera modulul `smtplib`, ce definește un client de SMTP sub formă unei sesiuni obiect ce poate fi utilizat pentru a trimite e-mail-uri catre orice masina de pe internet cu SMTP.

Iata o sintaxă simplă pt. a crea un obiect de SMTP care poate fi utilizat ulterior pentru trimiterea unui e-mail:

```
import smtplib
```

```
smtpObj = smtplib.SMTP( [host [, port [, local_hostname]]] )
```

Iata detaliile parametrilor:

**host:** Acesta este hostul care ruleaza SMTP server. Poti specifica o adresa Ip sau un domeniu precum gmail.com. Acesta este un argument optional.

port: Dacă oferim host atunci trebuie să specificăm și un port pe care serverul ascultă. Uzual portul este 25.

local\_hostname: Dacă serverul SMTP rulează local atunci poți specifica doar cuvântul cheie localhost ce indică această mașină.

Un obiect SMTP are o metodă numită sendmail, care este folosită pentru a trimite mesaje e-mail. Aceasta primește trei parametri:

Adresa de trimitere – Un șir de caractere care oferă adresa de trimitere.

Adresa destinație – Un șir de caractere care oferă adresa de destinație.

Mesajul- Un mesaj sub formă de șir de caractere ce trebuie să aibă un format special conform anumitor RFC-uri.

Iată un exemplu de cum ar putea fi un mic program ce trimite un e-mail

```
# Program Trimite e-mail
# Explica funcțiile smtplib
# Ion Studentul - 1/26/13

import smtplib

originator_email = 'hr@infoacademy.com'
destinatari = ['hr@todomain.com']

mesaj = """From: Persona care Originează e-mail <hr@infoacademy.com>
To: Ion Studentul <hr@todomain.com>
Subject: SMTP e-mail test

Salut.

Acesta este un mesaj de tip test!

O zi frumoasă,
Catalin
"""

try:
    smtpObj = smtplib.SMTP('localhost')
    smtpObj.sendmail(sender, destinatari, mesaj)
    print "Mesajul electronic a fost trimis cu succes"
except (e):
    print "Mesajul electronic nu a fost trimis cu succes"
    print e
```

Acest program nu va funcționa deoarece localhost indică mașina locală, noi neavând la dispoziție un server de email. Iar soluția nu este să realizăm noi un server de e-mail, ci

să indicăm unul utilizabil. Cum local nu avem nici un e-mail va trebui să folosim serverul de mail google.

Serverul de mail google este foarte bun deoarece nu poate fi utilizat pentru spam, poate fi accesat direct de pe internet și poate fi automatizat într-o alarmă. Vreți să primiți un e-mail când un eveniment extern se întâmplă? Realizați un script Python și îl rulați la nevoie. Astfel am modificat programul de mai sus pentru a trimite email de către un cont de google, program care îl regăsim mai jos. Înainte de a vă arăta totuși programul trebuie să avem o mică discuție în ceea ce privește caracteristicile serverului google.

Instrucțiuni Google legate de Standarde:

- (SMTP) Server - necesită TLS or SSL: smtp.gmail.com
- Folosește autentificare: DA
- Port pt. TLS/STARTTLS: 587
- Port pt. SSL: 465
- Timp de expirare al serverului : mai mare de 1 minut, recomandat 5 minute
- Account Name sau User Name: adresa de email întreaga: test@gmail.com
- Email Address: adresa de email întreaga: [test@gmail.com](mailto:test@gmail.com)
- Password: parola Gmail

Email: [mail.through.python@gmail.com](mailto:mail.through.python@gmail.com)

Parola: nu este dezvăluită

```
# Program Trimite e-mail
# Explica funcțiile smtplib
# Ion Studentul - 1/26/13

import smtplib

originator_email = 'hr@infoacademy.com'
destinatari = ['hr@todomain.com']

# Gmail Login

username = 'mail.through.python'
parola = '****'
```

```
mesaj = """From: Persona care Origineaza e-mail <hr@infoacademy.com>
To: Ion Studentul <hr@todomain.com>
Subject: SMTP e-mail test
```

Salut.

Acesta este un mesaj de tip test!

O zi frumoasa,  
Catalin  
"""

```
try:
    smtpObj = smtplib.SMTP('smtp.gmail.com:587')
    smtpObj.starttls()
    smtpObj.login(username, parola)
    smtpObj.sendmail(originator_email, destinatari, mesaj)
    print "Mesajul electronic a fost trimis cu succes"
except(), e:
    print "Mesajul electronic nu a fost trimis cu succes"
    print e
else:
    smtpObj.quit()
```

La exemplul anterior vom adauga urmatoarele:

- Indicam și un port la acel server.
- Google utilizeaza criptare prin TLS( predecesorul acestuia este TTL utilizat default pentru https)- este un algoritm de criptare pentru a se asigura ca mesajul trimis nu este citit sau modificat pe parcurs, Mai multe detalii gasiti la link-ul: [http://ro.wikipedia.org/wiki/Transport\\_Layer\\_Security](http://ro.wikipedia.org/wiki/Transport_Layer_Security)
- Apelam metoda login a obiectului creat pentru a ne autentifica la serverul google

In cazul în care va doriti să trimiteți un e-mail care să fie de forma unui html, Python ofera posibilitatea trimiterii si de mesaje complexe de tip html.

```
# Program Trimite e-mail
# Explica functiile smtplib
# Ion Studentul - 1/26/13
```

```
import smtplib
```

```
originator_email = 'hr@infoacademy.com'
destinatari = ['hr@todomain.com']
```

```
mesaj = """From: Persona care Origineaza e-mail <hr@infoacademy.com>
To: Ion Studentul <hr@todomain.com>
Subject: SMTP e-mail test
Content-type: text/html
```

```

<html>

<body>
Subject: SMTP e-mail test
<h1>Salut</h1>

<p>Acesta este un mesaj de tip test!Este un mesaj de tip HTML.</p>

<p><strong>O zi frumoasa,
Catalin</strong></p>

</body>
</html>
"""

# Gmail Login

username = 'mail.through.python'
parola = '****'

try:
    smtpObj = smtplib.SMTP('smtp.gmail.com:587')
    smtpObj.starttls()
    smtpObj.login(username, parola)
    smtpObj.sendmail(originator_email, destinatari, mesaj)
    print "Mesajul electronic a fost trimis cu succes"
except(),e:
    print "Mesajul electronic nu a fost trimis cu succes"
    print e
else:
    smtpObj.quit()

```

În programul de mai sus se poate vedea ca mesajul este de tip HTML. Surplusul acestui program în comparație cu cele de mai sus reprezintă adăugarea explicită a tipului mesajului ca fiind html:

*Content-type: text/html*

Datorită soluțiilor free ce vin cu orice sistem de operare, din păcate nu există o utilitate pentru primirea de e-mailuri. Totuși această operație se poate face cu poplib.

Python este des utilizat ca un adjuvant sau chiar bază (core, shell) în sistemele de testare, de automatizare sau de informare a unor evenimente. Prin urmare, cea mai bună cale de a trimite informație este prin email, oferind astfel o flexibilitate imensă.

De asemenea, este util să trimitem email-uri pentru orice aplicație cream pentru a oferi o modalitate de a ne informa de eventuale crash-uri (caderi ale programului).

## Desenarea de grafice(chart) în Python

Pentru desenarea de grafice putem utiliza mai multe solutii, dar una din cele mai bune și flexibile solutii este pygal. Acesta este un modul aditional ce trebuie instalat, deci mai jos se regaseste un link:

<https://pypi.python.org/packages/source/p/pygal/pygal-1.5.0.tar.gz>

PyGal utilizeaza crearea de grafice direct sub formă de fisier imagine de tip SVG.

Scalable Vector Graphics (SVG) ( "grafica vectoriala proportionabila") este un limbaj pentru descrierea de imagini 2D folosind XML. Este un standard al organizatiei W3C a cărui proiectare a început în anul 1999. Permite definirea imaginilor prin 3 metode: text, grafică vectoriala și "bitmap-uri" (fișiere în formatul BMP).

Deși există aplicații specializate pentru crearea și editarea de SVG-uri, în acest scop poate fi folosit orice editor text. Vizualizarea unei imagini SVG poate fi realizată cu orice browser, deci și integrarea graficelor într-un site este ușor de realizat.

În cele ce urmează vom putea analiza primul nostru program pygal.

```
# Program generare grafic pygal
# Explica functiile pygal
# Ion Studentul - 1/26/13

import pygal

import pygal
bar_chart = pygal.Bar()
bar_chart.add('Vineri', [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55])
bar_chart.render_to_file('grafic1.svg')

raw_input("\n\nApasa <enter> pt a iesi.")
```

În directorul unde acest program este rulat se va genera un fisier de tip svg numit grafic1. Dacă îl deschidem cu ajutorul unui browser vom vedea ceva similar ca în Fig.17

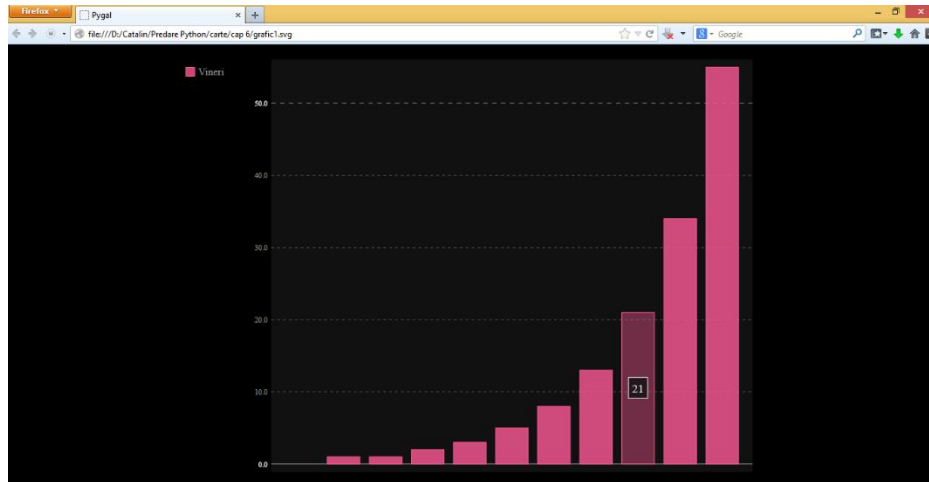


Fig.30

Daca dorim să adaugam mai multe randuri nu trebuie decat să reutilizam add()

```
# Explica functiile pygal
# Ion Studentul - 1/26/13
```

```
import pygal
bar_chart = pygal.Bar()
bar_chart.add('Bandwidth-Mbps', [0.5, 7, 1, 0, 0.8, 12, 3, 8, 13, 20, 24])
bar_chart.add('Latency-ms', [4, 10, 5, 0, 4, 20, 8, 13, 21, 34, 55])
bar_chart.render_to_file('grafic2.svg')

raw_input("\n\nApasa <enter> pt a iesi.")
```

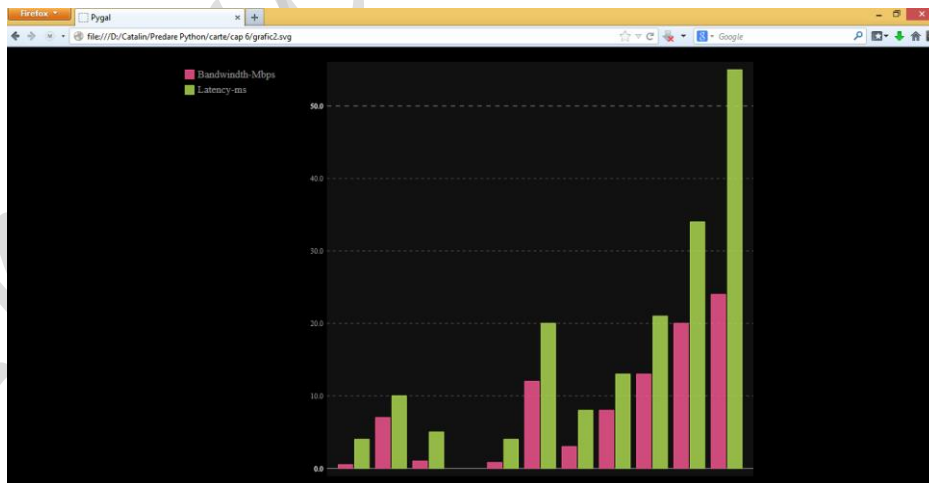


Fig.31

Daca modificati linia pygal.Bar() ca în modul de mai jos, vom adauga un titlu la grafic

```
bar_chart = pygal.Bar(title='Locatie Bucuresti')
```

Daca dorim un titlu pentru axa orizontala atunci trebuie să apelăm `x_title="titlu"`, iar pentru axa longitudinală putem folosi `y_title="titlu"`.

Pentru a modifica mărimea fontului tuturor titlurilor putem utiliza `title_font_size=numar`

```
# Program generare grafic pygal
# Explica functiile pygal
# Ion Studentul - 1/26/13

import pygal

bar_chart = pygal.Bar(title='Locatie: Bucuresti',x_title='Timp',
                      y_title='Elemente masurate',title_font_size=24)
bar_chart.add('Bandwidth-Mbps', [0.5, 7, 1, 0, 0.8, 12, 3, 8, 13, 20, 24])
bar_chart.add('Latency-ms', [4, 10, 5, 0, 4, 20, 8, 13, 21, 34, 55])
bar_chart.render_to_file('grafic2.svg')

raw_input("\n\nApasa <enter> pt a iesi.")
```

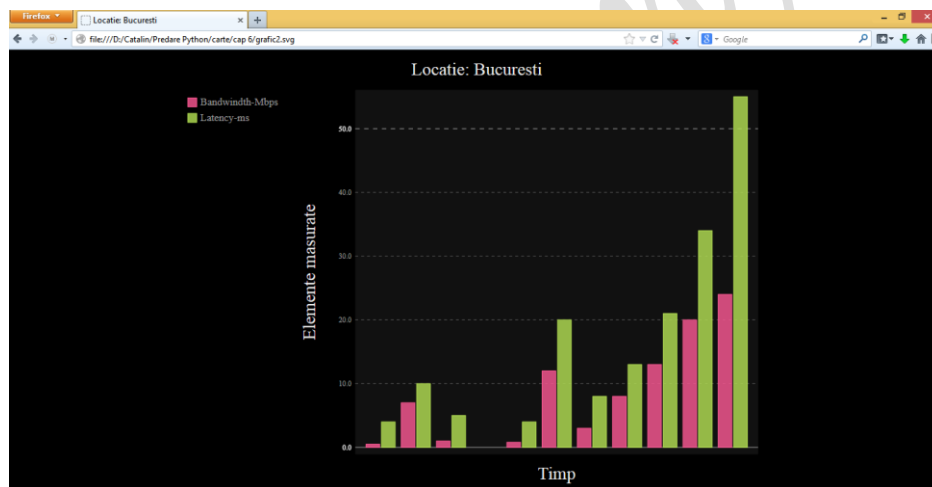


Fig.32

Așa cum e normal la fiecare punct de pe axa orizontală putem asocia un anumit șir de caractere. Aceasta practică este posibilă cu ajutorul metodei `x_labels=[]`. Fiecare punct de pe grafic va fi asociat în aceeași ordine cu șirul de caractere (label) din graficul generat. Generarea de label-uri fără puncte în grafic nu va genera o eroare, ci prelungeste graficul cu acele label-uri (completând automat cu zero în cadrul valorilor)

```
# Program generare grafic pygal
# Explica functiile pygal
# Ion Studentul - 1/26/13

import pygal

bar_chart = pygal.Bar(title='Locatie: Bucuresti',x_title='Timp',
```



```

        y_title='Elemente masurate',title_font_size=24)
bar_chart.add('Bandwidth-Mbps', [0.5, 7, 1, 0, 0.8, 12, 3, 8, 13, 20, 24])
bar_chart.add('Latency-ms', [4, 10, 5, 0, 4, 20, 8, 13, 21, 34, 55])
bar_chart.x_labels = [
    '10:15:10',
    '10:15:20',
    '10:15:30',
    '10:15:40']

bar_chart.render_to_file('grafic3.svg')

raw_input("\n\nApasa <enter> pt a iesi.")

```

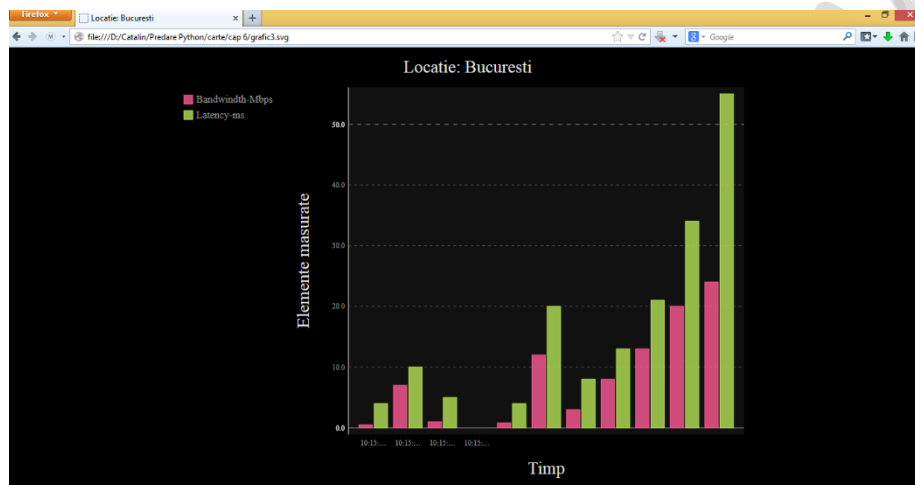


Fig.33

Pentru a scala automat cifrele și label-urile putem utiliza `human_readable`. Aceasta valoare este standard `False`. Au fost modificate anumite valori pentru a putea vedea cum functioneaza aceasta optiune.

```

# Program generare grafic pygal
# Explica functiile pygal
# Ion Studentul - 1/26/13

import pygal

bar_chart = pygal.Bar(title='Locatie: Bucuresti',x_title='Timp',
    y_title='Elemente masurate',title_font_size=24,human_readable=True)
bar_chart.add('Bandwidth-bps', [500000, 7000000, 1000000, 0, 800000, 12000000,
    3000000, 8000000, 13000000, 20000000, 24000000])
bar_chart.add('Latency-ms', [4, 10, 5, 0, 4, 20, 8, 13, 21, 34, 55])
bar_chart.render_to_file('grafic4.svg')

raw_input("\n\nApasa <enter> pt a iesi.")

```

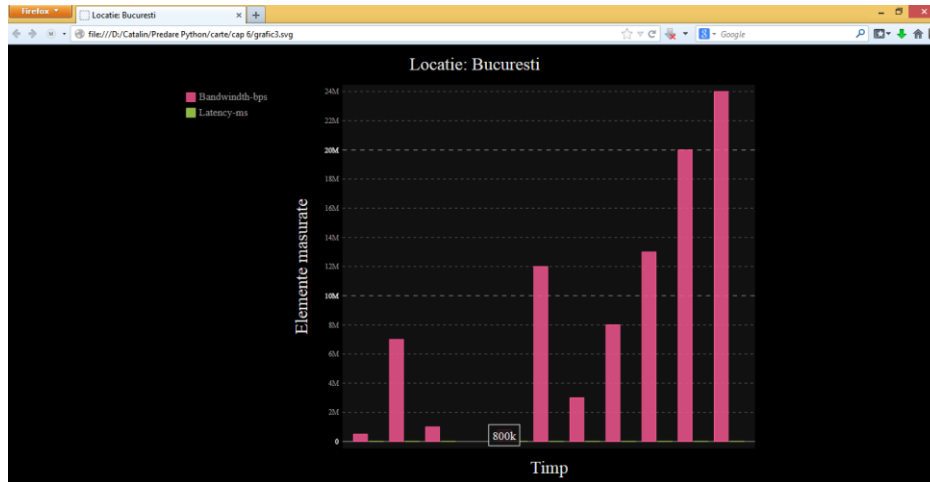


Fig.34

Vedem ca fiecare valoare este scalata separat, deci avem și valori în kilo, și valori în Mega.

PyGal este creat pentru a putea fi utilizat în automatizari. Dacă setam un text sub `no_data_text` atunci vom putea vedea acel text doar dacă nu exista puncte de desenat pe grafic. De asemenea putem vedea ca fiecare caracteristica se poate scrie ca argument de intrare la crearea obiectului `pygal` sau dupa creare.

```
# Program generare grafic pygal
# Explica functiile pygal
# Ion Studentul - 1/26/13

import pygal

bar_chart = pygal.Bar()
bar_chart.no_data_text = "Fara Date"
bar_chart.title='Locatie: Bucuresti'
bar_chart.human_readable=True
bar_chart.y_title='Elemente masurate'
bar_chart.x_title='Timp'
bar_chart.add('Bandwidth-bps',[])
bar_chart.render_to_file('grafic4.svg')

raw_input("\n\nApasa <enter> pt a iesi.")
```



Fig.35

În cele ce urmează vom vedea cum putem realiza multiple tipuri de grafice:

`chart = pygal.Bar()` # grafice de tip bară (barele sunt așezate una lângă alta)

`chart = pygal.StackedBar()` # grafice de tip bară (barele sunt așezate una peste alta)

`chart = pygal.line ()` # grafice de tip puncte unite prin linii (liniile sunt așezate una lângă alta)

`chart = pygal.StackedLine()` # grafice de tip bară (liniile sunt așezate una peste alta)

Pentru Graficele de tip linii avem și parametru `fill` ce poate fi `True` sau `False` (pentru a umple spațiul dintre linie și axa Ox). Acest parametru este standard `False`.

`Chart.fill=True`

`chart = pygal.Pie()` # grafice de tip placintă

`chart = pygal.Worldmap()` # grafice de tip hartă globului.

Mai jos regăsim un program care creează un SVG cu harta globului pământesc.

```
# Program generare grafic pygal
# Explica funcțiile pygal
# Ion Studentul - 1/26/13
```

```
import pygal
```

```
worldmap_chart = pygal.Worldmap()
worldmap_chart.title = 'Some countries'
worldmap_chart.add('Orange countries', ['fr', 'fi', 'ro'])
```

```
worldmap_chart.add('Vodafone countries', [
    'ma', 'mc', 'md', 'me', 'mq',
    'mk', 'ml', 'mm', 'mn', 'mo',
    'mr', 'mt', 'mu', 'mv', 'mw',
    'mx', 'my', 'mz'])
worldmap_chart.add('U countries', ['ca', 'uc', 'ro', 'uy', 'uz'])
worldmap_chart.render_to_file('grafic7.svg')

raw_input("\n\nApasa <enter> pt a iesi.")
```

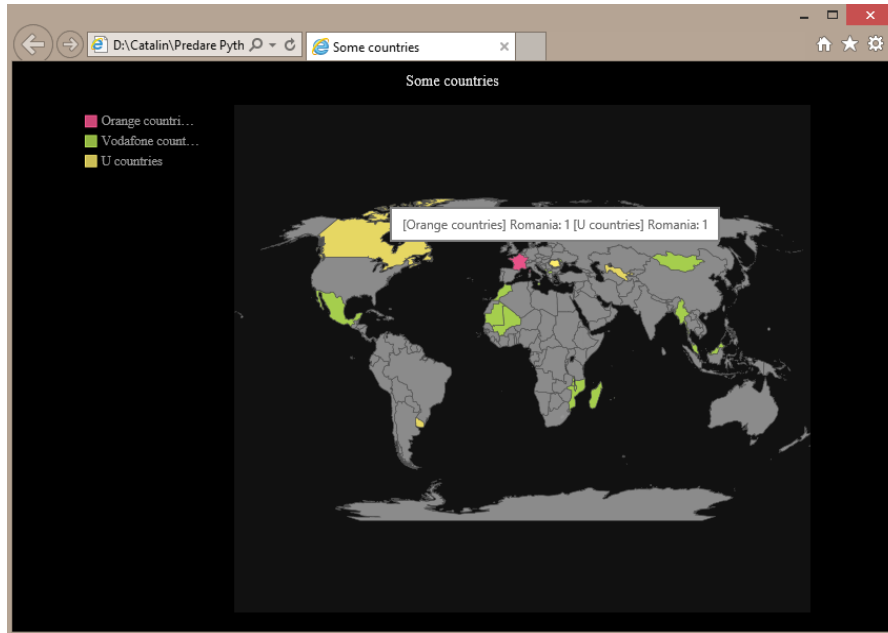


Fig.36

De asemenea puteti adauga link-uri la aceste chart-uri creand o modalitate să afisam pagini mai exacte ce contin doar informatia solicitata.

Pentru a incerca toate functiile pe care le poate pygal puteti intra pe pagina <http://cabaret.pygal.org/> și să le incercati.

In cazul în care aveti nevoie de alt format putem folosi tot pygal pentru a converti aceste grafice. Totusi pentru a converti aceste grafice pygal foloseste urmatoarele trei librarii ce trebuiesc instalate:

Cssselect:

<https://pypi.python.org/packages/source/c/cssselect/cssselect-0.9.1.tar.gz>

tinycss

<https://github.com/SimonSapin/tinycss/archive/master.zip>

lxml

<https://pypi.python.org/packages/2.7/l/lxml/lxml-2.3.win32-py2.7.exe#md5=9c02aae672870701377750121f5a6f84>

Mai jos regasim și un exemplu de astfel de program ce exportă un fișier png.

```
# Program generare grafic pygal
# Explica functiile pygal
# Ion Studentul - 1/26/13

import pygal

bar_chart = pygal.Bar(title='Locatie: Bucuresti',x_title='Timp',
                      y_title='Elemente masurate',title_font_size=24,human_readable=True)
bar_chart.add('Bandwidth-bps', [500000, 7000000, 1000000, 0, 800000, 12000000,
3000000, 8000000, 13000000, 20000000, 24000000])
bar_chart.add('Latency-ms', [4, 10, 5, 0, 4, 20, 8, 13, 21, 34, 55])
bar_chart.x_labels = [
    '10:15:10',
    '10:15:20',
    '10:15:30',
    '10:15:40']

bar_chart.render_to_png('grafic10.png')

raw_input("\n\nApasa <enter> pt a iesi.")
```

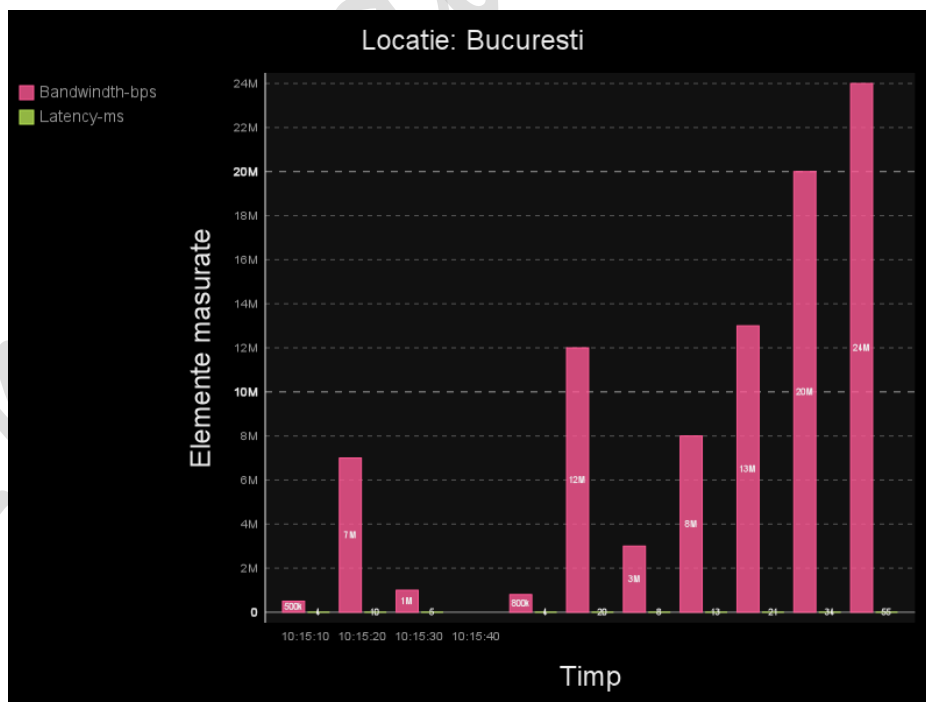


Fig.37