Atelier TP

Ilyas Bambrik

Table des matières

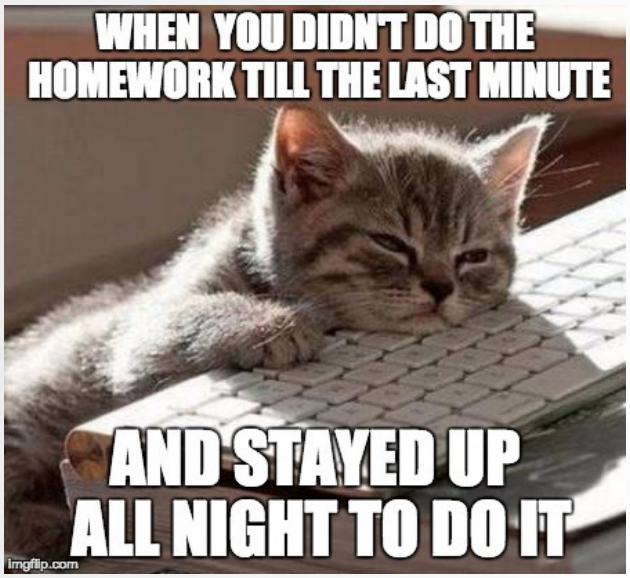
Enoncé

I

1. Remise du travail

⚠ *Attention*

- Ce devoir a pour objectif de tester les connaissances acquis lors du TP Réseaux Avancés. Ainsi ce devoi sera comptabilisé avec *6 points de la notes totale du TP*.
- Ce devoir peux se faire jusqu'à deux par groupe.
- *Date limite de remise :* une semaine après les examens.
- *Toute tentative de copiage d'une solution sera récompenser avec un 0 pour la note attribuée à la partie plagi (ci-ci s'applique pour touts les parties impliquées).*
- Chaque groupe d'étudiants a le choix entre les deux exercices (un seul exercice à faire au choix).
- Les groupes qui effectuent les deux exercices auront *+1pts* dans la note TP totale si les deux solutions sont justes.
- L'implémentation de la solution avec n'importe quel langage de programmation est permise.



2. Exercice : Client TFTP-Like

Soit le code python 2 (voir plus en bas le code équivalent en python 3) du serveur du protocole TFTP-Like suivant :

```
1 # Ecrit en python 2
2 import socket
3 import os
4 import re
```

```
SocketServeur = socket.socket()
port = 9500

# Obtention du chemin courant (getcwd() == get current working directory)
Chemin_racine=os.getcwd()
Reprtoir_racine=os.getcwd().split("\\")[-1]

# Ecout de connexion sur toutes les insterfacces (0.0.0.0)
SocketServeur.bind(("0.0.0.0", port))
SocketServeur.listen(1)
print "Lancement serveur"

while True:
   # Attente d'une connexion (accept)
   ConnexionAUnClient, addrclient = SocketServeur.accept()
   # Acceder au repertoire racine
   os.chdir(Chemin_racine)

   print "Connexion de la machine = ", addrclient
   while True:

       # Format des commandes transmis par le client : "Type_Commande:NomFichier"
       # Type_Commande == 3 changer le repertoire courant (Change Working Dirrectory CWD dans FTP et cd dans Windows et Linux)
       # Type_Commande == 2 demander la liste des fichiers dans le repetoir courant (comme la commande "dir" dans Windows et "ls" dans Linux)
       # Type_Commande == 1 telecharger un fichier( par exemple si la commande recue == "1:File.png" signifie telecharger "File.png" )
       # Type_Commande == 0 Si la commende commence par "0:" cela signifie que le client souhaite deconnecter
       # par exemple "0:", "0:CYA" et "0:leaving" signifient deconnexion
       try:
           Commande=ConnexionAUnClient.recv(1024)
           # decouper la commande par le caractere ":"
           Arguments=Commande.split(":")
           # le premier argument represente le type de la commande
           Type_Commande=int(Arguments[0])
           Nom_Fichier=Arguments[1]
           if(Type_Commande==0):# commande de deconnexion
                ConnexionAUnClient.send("200 k, bye")
                # L'envoie d'une sequence "--\r\n\r\n" delimite la fin d'une reponse
                ConnexionAUnClient.send("--\r\n\r\n")
                # Sortir de la boucle de traitement while
                break
           elif(Type_Commande==1):# commande de telechargement de fichier
               # Nom_Fichier ne doit pas contenir ".." ou / (tentative de parcourir les repertoires )
               if( ".." in Nom_Fichier or "/" in Nom_Fichier ):
                    ConnexionAUnClient.send("501 DIRECTORY TRAVERSAL DENIED")
                    ConnexionAUnClient.send("--\r\n\r\n")
                    continue # Esquiver le reste des instructions
               try:
                  # Lire le fichier designe par la commande utilisateur
                  Fichier_Upload=open(Nom_Fichier,"rb")
                  for octets in Fichier_Upload:
                      # Transmettre chaque sequence d'octets lu vers le client
                      ConnexionAUnClient.send(octets)
                  # Fermer le fichier lu
                  Fichier_Upload.close()
```

```
59                  ConnexionAUnClient.send("--\r\n\r\n")
60              except:
61                  ConnexionAUnClient.send("500 Fichier introuvable")
62                  ConnexionAUnClient.send("--\r\n\r\n")
63          elif(Type_Commande==2): # commande pour lister le contenu du
repertoir courant
64              # os.listdir(".") revoie la liste des fichier / repertoires dans
le repertoire courant (comme ls et dir)
65              Liste_Des_Fichiers=os.listdir(".")
66              ConnexionAUnClient.send("\n".join(Liste_Des_Fichiers))
67              ConnexionAUnClient.send("--\r\n\r\n")
68          elif(Type_Commande==3): # commande d'acces au repertoir en parametre
69              try:
70                  # acceder au repertoir transmis par le client ( cd
Nom_Fichier)
71                  # os.chdir renvoie une exeception si le nom du repertoir en
parametre ("Nom_Fichier") n'existe pas
72                  os.chdir(Nom_Fichier)
73                  if( Reprtoir_racine in os.listdir(".")):
74                      os.chdir(Reprtoir_racine)
75                      ConnexionAUnClient.send("501 Nope (pls, just, k?)")
76                  else:
77                      ConnexionAUnClient.send("200 OK "+Nom_Fichier)
78              except:
79                  # en cas d'exception (le repertoir demandee par le client est
inexistant)
80                  # transmettre le code 404
81                  ConnexionAUnClient.send("404 Le repertoire "+Nom_Fichier+"
est introuvable")
82              ConnexionAUnClient.send("--\r\n\r\n")
83          else:
84              # si la commande du client ne correspond a aucune des commandes
permises
85              # transmettre le code 400
86              ConnexionAUnClient.send("400 Commande inconue = "+Commande)
87              ConnexionAUnClient.send("--\r\n\r\n")
88
89      except:
90          try:
91             # envoie de code 401: syntaxe incorrecte
92             ConnexionAUnClient.send("401 Format de commande incorrect : "+
Commande)
93             ConnexionAUnClient.send("--\r\n\r\n")
94
95          except:
96              pass
97          break
98   print "Deconnexion de :",addrclient
99
100 SocketServeur.close()
101
```

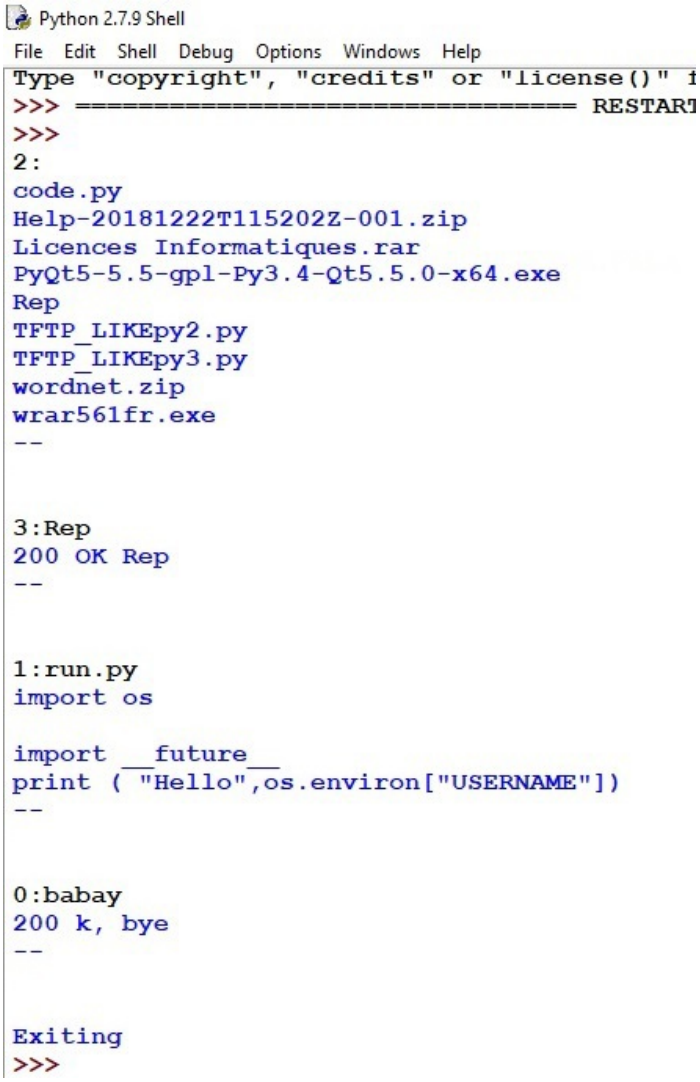
Question 1

Écrivez le code d'un client permettant la lecture / envoie des commande TFTP_Like et la récupération du résultat des commandes. (*4 pts : 2.5 points pour la commande de téléchargement de fichier*, *0.5 pour chacune des commandes 0, 2 et 3*).

Le programme client doit gérer l'occurrence des erreurs avec la clause *try - except* (comme try catch en java) ainsi que permettre l'exécution des quatre commandes.

Indice :

Voir l'exemple suivant pour le format des commande et réponse de TFTPLike :



Question 2

Changez le code du serveur afin de le rendre capable de traiter les requêtes de plusieurs clients simultanément (voir le 1er TP sur le multi-threading). *(1pts)*

Question 3

Ajouter une option du coté client pour afficher le fichier téléchargé sur console ou bien de l'enregistrer sur disque. *(1pts) [lecture d'un caractère y/n pour décider d'afficher ou d'enregistrer]*

Code équivalent python 3

```python
# Python 3!
import socket
import os
import re
SocketServeur = socket.socket()
port = 9500

# Obtention du chemin courant (getcwd()==get current working directory)
Chemin_racine=os.getcwd()
Reprtoir_racine=os.getcwd().split("\\")[-1]

# Ecout de connexion sur toutes les insterfacces (0.0.0.0)
SocketServeur.bind(("0.0.0.0", port))
SocketServeur.listen(1)
print("Lancement serveur")

while True:
   # Attente d'une connexion (accept)
   ConnexionAUnClient, addrclient = SocketServeur.accept()
   # Acceder au repertoire racine
   os.chdir(Chemin_racine)

   print("Connexion de la machine = ", addrclient)
   while True:

       # Format des commandes transmis par le client : "Type_Commande:NomFichier"
       # Type_Commande == 3 changer le repertoire courant (Change Working Dirrectory "CWD" dans FTP et "cd" dans Windows et Linux)
       # Type_Commande == 2 demander la liste des fichiers dans le repetoir courant (comme la commande "dir" dans Windows et "ls" dans Linux)
       # Type_Commande == 1 telecharger un fichier( par exemple si la commande recue == "1:File.png" signifie telecharger "File.png" )
       # Type_Commande == 0 Si la commende commence par "0:" cela signifie que le client souhaite deconnecter
       # par exemple "0:", "0:CYA" et "0:leaving" signifient deconnexion
       try:
           Commande=ConnexionAUnClient.recv(1024).decode("unicode_escape")
           # decouper la commande par le caractere ":"
           Arguments=Commande.split(":")
           # le premier argument represente le type de la commande
           Type_Commande=int(Arguments[0])
           Nom_Fichier=Arguments[1]
           if(Type_Commande==0):# commande de deconnexion
                ConnexionAUnClient.send(b"200 k, bye")
                # L'envoie d'une sequence "--\r\n\r\n" delimite la fin d'une reponse
                ConnexionAUnClient.send(b"--\r\n\r\n")
                # Sortir de la boucle de traitement while
                break
           elif(Type_Commande==1):# commande de telechargement de fichier
               # Nom_Fichier ne doit pas contenir ".." ou / (tentative de parcourir les repertoires )
               if( ".." in Nom_Fichier or "/" in Nom_Fichier ):
                    ConnexionAUnClient.send(b"501 DIRECTORY TRAVERSAL DENIED")
                    ConnexionAUnClient.send(b"--\r\n\r\n")
                    continue # Esquiver le reste des instructions
```

```
51                try:
52                   # Lire le fichier designe par la commande utilisateur
53                   Fichier_Upload=open(Nom_Fichier,"rb")
54                   for octets in Fichier_Upload:
55                       # Transmettre chaque sequence d'octets lu vers le
client
56                       ConnexionAUnClient.send(octets)
57                   # Fermer le fichier lu
58                   Fichier_Upload.close()
59                   ConnexionAUnClient.send(b"--\r\n\r\n")
60                except:
61                   ConnexionAUnClient.send(b"500 Fichier introuvable")
62                   ConnexionAUnClient.send(b"--\r\n\r\n")
63            elif(Type_Commande==2):# commande pour lister le contenu du repertoir
courant
64                # os.listdir(".") revoie la liste des fichier / repertoires dans
le repertoire courant (comme ls et dir)
65                Liste_Des_Fichiers=os.listdir(".")
66                ConnexionAUnClient.send(("\n".join(Liste_Des_Fichiers)).encode())
67                ConnexionAUnClient.send(b"--\r\n\r\n")
68            elif(Type_Commande==3):# commande d'acces au repertoir en parametre
69                try:
70                    # acceder au repertoir transmis par le client ( cd
Nom_Fichier)
71                    # os.chdir renvoie une exeception si le nom du repertoir en
parametre ("Nom_Fichier") n'existe pas
72                    os.chdir(Nom_Fichier)
73
74                    if( Reprtoir_racine in os.listdir(".")):
75                        os.chdir(Reprtoir_racine)
76                        ConnexionAUnClient.send(b"501 Nope (pls, just, k?)")
77                    else:
78                        ConnexionAUnClient.send(("200 OK "+Nom_Fichier).encode())
79                except:
80                    # en cas d'exception (le repertoir demandee par le client est
inexistant)
81                    # transmettre le code 404
82                    ConnexionAUnClient.send(("404 Le repertoire "+Nom_Fichier+"
est introuvable").encode())
83                ConnexionAUnClient.send(b"--\r\n\r\n")
84            else:
85                # si la commande du client ne correspond a aucune des commandes
permises
86                # transmettre le code 400
87                ConnexionAUnClient.send(("400 Commande inconue = "+Commande).
encode())
88                ConnexionAUnClient.send(b"--\r\n\r\n")
89
90        except:
91            try:
92               # envoie de code 401: syntaxe incorrecte
93               ConnexionAUnClient.send(("401 Format de commande incorrect : "+
Commande).encode())
94                ConnexionAUnClient.send(b"--\r\n\r\n")
95
96            except:
97                pass
98            break
99    print( "Deconnexion de :",addrclient)
100
101 SocketServeur.close()
```

3. Exercice : Télécharger une page HTTP (take 2)

Afin de télécharger le contenu d'une page, le navigateur télécharge toutes les ressources référencées par la page (fichiers *javascript, png, xls, css* etc) et les regroupes dans un sous répertoire pour pouvoir les référencer localement. Le code python 3 (*voir le code équivalant python 2 à la fin de l'énoncé*) suivant permet de télécharger une ressource *story.shtml* du serveur *www.eurosport.com*:

```
1 import http.client # bibliotheque HTTP
2 # Nom domaine du site (l'adresse IP du site fonctionnera identiquement)
3 site="www.eurosport.com"
4 # URL absolu de la ressource a obtenir
5 URL="https://www.eurosport.com/football/van-dijk-dismisses-talk-of-title-despite-
  reds-leading-the-way_sto7062094/story.shtml"
6 # Intialisation d'une connexion HTTP
7 ConnexionHTTP = None
8
9 if URL.startswith("https:"):
10     # si l'URL est un URL HTTPs, on utilise une connexion HTTPS (HTTP<u>S<
  /u>Connection)
11     ConnexionHTTP=http.client.HTTPSConnection(site)
12 else:
13     # sinon, on utilise une connexion HTTP (HTTPConnection)
14     ConnexionHTTP=http.client.HTTPConnection(site)
15 # Envoie d'une requete GET
16 ConnexionHTTP.request("GET", URL)
17 # Obtention de la ressource
18 Reponse = ConnexionHTTP.getresponse()
19
20 Content_Length=Reponse.length
21 print ("Taille reponse = ",Content_Length)
22
23 # Ouvrir un fichier en mode d'ecriture index.html en octets
24 Fichier_Pour_enregistrer_le_resultat=open("index.html","wb")
25
26 # Lecture du contenu de la ressource
27 Partie_recu=Reponse.read(1024).decode("unicode_escape")
28
29 # len(Partie_recu) == nombre de caracteres dans Partie_recu
30 # Tanque le nombre de cracateres recus est different de 0
31 while len(Partie_recu)!=0:
32     # Ecrir la partie recu sur le fichier "index.html"
33     # encode() converti un suite de caractere en une suites d'octets
34     Fichier_Pour_enregistrer_le_resultat.write(Partie_recu.encode())
35
36     # Continue de recevoir les donnees transmis par le serveur http
37     # decode("unicode_escape") converti une suite d'octets en caracteres ASCII
38     # decode("utf") pour convertir une suite d'octets en caracteres UTF-8
39     Partie_recu=Reponse.read(1024).decode("unicode_escape")
40     # Afficher la partie recu
41     print(Partie_recu,)
42
43 # Alternativement Reponse.read() peut etre utiliser directement (au lieu de
  recevoir la reponse dans des morceaux de 1024octets) pour recevoir la ressource
  complete
44 # Mais la concatenation de toute une ressource dans une seule sequence ralenti
  l'execution du programme
45
46 # Cloturer le fichier
47 Fichier_Pour_enregistrer_le_resultat.close()
```

```
48
```

Afin de rechercher les références des ressources d'une page, les expressions régulières doivent être utilisées comme suite :

```
1 # code python3
2 import re # Regular Expression (expression reguliere)
3 # Ouvrir le fichier contenant la page index.html
4 fichier=open("index.html","rb")
5 contenu_fichier=""
6 # Lecture du contenu du fichier
7 for ligne in fichier:
8     contenu_fichier+=ligne.decode("unicode_escape")
9 fichier.close()
10 # rechercher toutes les occurence dans le texte de la forme "[^"]+\.css"|"[^"]+\.
  js"|"[^"]+\.png" (une chaine de caractere se terminant par un .js entouree par "")
11 # '.' (point) == nimporte quel caractere ('\.' reference le caractere point)
12 # | == OU logique (exemple "[^"]+\.css"|"[^"]+\.js"|"[^"]+\.png" signifie "[^"]
  +\.css" OU "[^"]+\.js|" OU "[^"]+\.png"
13 # [^"] == touts cractere sauf '"'
14 # [^"]+ == suite de caracteres ne contenant pas le cracatere '"'
15 matches=re.findall('"[^"]+\.css"|"[^"]+\.js"|"[^"]+\.png"',contenu_fichier,re.M)
  # re.M == recherche Multilignes
16 for resultat in matches:
17     print (resultat)
```

L'exécution de la recherche avec expression régulière (programme précédant) sur le fichier *index.html* (qu'ont a télécharger par le 1er programme HTTP) donne le résultat suivant (liste des URLs des ressources) :

```
Python 3.4.0 (v3.4.0:04f714765c13, Mar 16 2014, 19:25:23) [MSC v.1600 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
"https://layout.eurosport.com/c/v8_5/global.css"
"https://layout.eurosport.com/c/v8_5/watchbar.css"
"https://layout.eurosport.com/c/v8_5/feed_on_air_v2.css"
"https://layout.eurosport.com/c/v8_5/storyfull_bis.css"
"https://layout.eurosport.com/i/v8/logo/logo-esp-google-rss-wen.png"
"https://layout.eurosport.com/i/v8/favicon/wen/apple-icon-57x57.png"
"https://layout.eurosport.com/i/v8/favicon/wen/apple-icon-60x60.png"
"https://layout.eurosport.com/i/v8/favicon/wen/apple-icon-72x72.png"
"https://layout.eurosport.com/i/v8/favicon/wen/apple-icon-76x76.png"
"https://layout.eurosport.com/i/v8/favicon/wen/apple-icon-114x114.png"
"https://layout.eurosport.com/i/v8/favicon/wen/apple-icon-120x120.png"
"https://layout.eurosport.com/i/v8/favicon/wen/apple-icon-144x144.png"
"https://layout.eurosport.com/i/v8/favicon/wen/apple-icon-152x152.png"
"https://layout.eurosport.com/i/v8/favicon/wen/apple-icon-180x180.png"
"https://layout.eurosport.com/i/v8/favicon/wen/apple-icon-192x192.png"
"https://layout.eurosport.com/i/v8/favicon/wen/favicon-16x16.png"
"https://layout.eurosport.com/i/v8/favicon/wen/favicon-32x32.png"
"https://layout.eurosport.com/i/v8/favicon/wen/favicon-36x36.png"
"https://layout.eurosport.com/i/v8/favicon/wen/favicon-48x48.png"
"https://layout.eurosport.com/i/v8/favicon/wen/favicon-72x72.png"
"https://layout.eurosport.com/i/v8/favicon/wen/favicon-96x96.png"
"https://layout.eurosport.com/i/v8/favicon/wen/favicon-144x144.png"
"https://layout.eurosport.com/i/v8/favicon/wen/favicon-192x192.png"
"https://layout.eurosport.com/i/v8/favicon/wen/ms-icon-144x144.png"
"https://layout.eurosport.com/j/v8_5/browser-reject.js"
"//assets.adobedtm.com/55c42c987178cce094489fba79ae9fa5af72b984/satelliteLib-e3f
d545becb77cbb376c77ed33d18dacb7157dab.js"
"//js-sec.indexww.com/ht/p/186403-120271712892857.js"
"//aka-cdn.adtech.de/dt/common/DAC.js"
"https://layout.eurosport.com/j/v8_5/merged/libs/require.js"
"https://tags.crwdcntrl.net/c/10178/cc_af.js"
>>>
```

Question 1

Écrivez un programme permettant de télécharger touts les ressources utilisées par une page HTML (fichier *.js, .css, .xss*, image *.png* et *.jpg*, etc). *(4 pts)*

Regroupez toutes les ressources téléchargées dans un sous répertoire. *(1pts)*

Indice :

Utilisez la procédure *os.mkdir("Nom_Sous_Repertoire")* pour créer un répertoire *"Nom_Sous_Repertoire"*.

Question 2

Changez les URLs des ressources dans la page HTML afin d'accéder aux copies téléchargées (sur le disque). *(1pts)*

Voir l'exemple suivant qui montre comment remplacer une chaîne de caractères correspondante à une expression régulière par un autre texte (*le même programme fonctionne dans python 2 et 3)*:

```
# cet exemple fonctionne pour python 2 et 3
import re # librerie Regular_Expression (expressions reguliere
Texte="La blablablamation est importante pour les blablameurs. Apprenez a' blamer avec une manette!"


# procedure de remplacement d'une expression reguliere par un texte: re.sub (Expression_Reguliere_Du_texte_a_remplac%e9,Texte_de_remplacement,Texte)
# Expression_Reguliere_Du_texte_a_remplac%e9="(bla)+" [bla se repete une ou plusieurs fois]
# Texte_de_remplacement = "program"
# Texte="La blablablamation est importante pour les blablameurs. Je blame avec une manette!"

# re.sub retoutre le texte apres substitution

Texte_Apres_Remplacement=re.sub("(bla)+","program",Texte)
print (Texte_Apres_Remplacement)
```

Code équivalent en python 2

```
# code python 2
import httplib # bibliotheque HTTP
# Nom domaine du site (l'adresse IP du site fonctionnera identiquement)
site="www.eurosport.com"
# URL absolu de la ressource a obtenir
URL="https://www.eurosport.com/football/van-dijk-dismisses-talk-of-title-despite-reds-leading-the-way_sto7062094/story.shtml"
# intialisation d'une connexion HTTP
ConnexionHTTP=None

if URL.startswith("https:"):
    # si l'URL est un URL HTTPs, on utilise une connexion HTTPS (HTTP<u>S</u>Connection)
    ConnexionHTTP=httplib.HTTPSConnection(site)
else:
    # sinon, on utilise une connexion HTTP (HTTPConnection)
    ConnexionHTTP=httplib.HTTPConnection(site)
# envoie d'une commande GET
ConnexionHTTP.request("GET", URL)
```

```
# obtention de la ressource
Reponse = ConnexionHTTP.getresponse()
Content_Length=Reponse.length
print "Taille reponse = ",Content_Length

# Ouvrir un fichier en mode d'ecriture index.html
Fichier_Pour_enregistrer_le_resultat=open("index.html","w")

# Lecture du contenu de la ressource
Partie_recu=Reponse.read(1024)

# len(Partie_recu) == nombre de caracteres dans Partie_recu
# Tanque le nombre de cracateres recus est different de 0
while len(Partie_recu)!=0:
    # Ecrir la partie recu sur le fichier "index.html"
    Fichier_Pour_enregistrer_le_resultat.write(Partie_recu)

    # Continue de recevoir les donnees transmis par le serveur http
    Partie_recu=Reponse.read(1024)
    # Afficher la partie recu
    print Partie_recu,
# Alternativement Reponse.read() peut etre utiliser directement (au lieu de recevoir la reponse dans des morceaux de 1024octets) pour recevoir la ressource complete
# Mais la concatenation de toute une ressource dans une seule sequence ralenti l'execution du programme

# Cloturer le fichier
Fichier_Pour_enregistrer_le_resultat.close()

```

```
# code python2
import re # Regular Expression (expression reguliere)
# Ouvrir le fichier contenant la page index.html
fichier=open("index.html")
contenu_fichier=""
# Lecture du contenu du fichier
for ligne in fichier:
    contenu_fichier+=ligne
fichier.close()
# rechercher toutes les occurence dans le texte de la forme (expression reguliere) "[^"]+\.css"|"[^"]+\.js"|"[^"]+\.png" (une chaine de caractere se terminant par un .js entouree par "")
# '.' (point) == nimporte quel caractere ('\.' reference le caractere point)
# | == OU logique (exemple "[^"]+\.css"|"[^"]+\.js"|"[^"]+\.png" signifie "[^"]+\.css" OU "[^"]+\.js|" OU "[^"]+\.png"
# [^"] == touts cractere sauf '"'
# [^"]+ == suite de caracteres ne contenant pas le cracatere '"'
matches=re.findall('"[^"]+\.css"|"[^"]+\.js"|"[^"]+\.png"',contenu_fichier,re.M)
# re.M == recherche Multilignes
# matches contiendra les chaines de caracteres correspondants a l'expression reguliere '"[^"]+\.css"|"[^"]+\.js"|"[^"]+\.png"'
for resultat in matches:
    print resultat
```