HTML

1 Accès à un site internet

1.1 Adresse IP

Pour accéder à une page web, on utilise généralement son nom, appelé également URL. Notre exemple sera ici le site du lycée, disponible à l'adresse pascal-lyc.spip.ac-rouen.fr.

Mais en pratique, les serveurs ne sont pas accessibles par un nom, mais par un numéro les identifiants, que l'on appelle l'adresse IP.

Par exemple, la page qui nous intéresse se trouve à l'adresse : http://pascal-lyc.spip.ac-rouen.fr

```
# ne pas faire attention à la commande, seul le résultat nous intéresse.
!getent hosts pascal-lyc.spip.ac-rouen.fr
```

```
194.167.110.9   hongkong.ac-rouen.fr pascal-lyc.spip.ac-rouen.fr
```

Évidemment, ce n'est pas la seule page disponible à cette adresse, mais nous y reviendrons plus tard.

1.2 La conversion : DNS

La communication entre les machines formant le web se fait entre machines conaissant leurs adresses IP respectives. Pour le serveur répondant à une *requête*, c'est facile : l'adresse de réponse est indiquée.

Par contre, pour la première communication vers un serveur, le client (nous) doit pouvoir trouver le bon serveur pour communiquer. On ne peut pas envisager de stocker à un seul endroit la liste de tous les sites associés aux adresses IP, le volume de données à stocker et le nombre de demandes seraient bien trop grands.

Le procédé est un peu compliqué, mais on peut le résumer par :

```
<img src="https://www.nameshield.com/wp-content/uploads/2020/03/DNS-768x419.png">
```

Soyons plus précis. Une URL (dans le shéma, "www.wikipedia.org"), se décompose en plusieurs morceaux, séparés par des points. Seuls les deux derniers nous intéressent.

```
url = truc.nom-de-domaine.tld/encore-des-truc
```

où truc peut être, lui aussi séparé par des points (voir le site du lycée, truc = pascal-lyc.spip) ou non (pour wikipedia, truc = www) Nous ignorons pour l'instant le protocole utilisé (http, https...)

Les deux parties qui nous concernent aujourd'hui sont le nom de domaine (wikipedia ou ac-rouen) ainsi que le tld (pour top level domain, ici org ou fr).

La résolution de nom (le fait de pouvoir trouver le serveur qui va nous renvoyer la bonne page) se fait en plusieurs étapes.

1.2.1 Trouver le serveur TLD

On va d'abord trouver le serveur listant tous les sites dont le tld est .org. Pour cela, il existe des serveurs (nommés racines) dont la seule fonction est de connaître tous les tld possibles associés aux serveurs connaissant les domaines associés (premier niveau : on trouve où habitent les sites se terminant en .org). Les adresses des serveurs racines sont publiques. Nous utiliserons le serveur à l'adresse 192.58.128.30

```
host -v www.wikipedia.org  192.58.128.30

Trying "www.wikipedia.org"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61680
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 12

;; QUESTION SECTION:
;www.wikipedia.org.     IN  MX

;; AUTHORITY SECTION:
org.            172800  IN  NS  a0.org.afilias-nst.info.
....

;; ADDITIONAL SECTION:
a0.org.afilias-nst.info. 172800 IN  A   199.19.56.1
....

Received 437 bytes from 192.58.128.30#53 in 14 ms
```

Nous n'avons pas encore trouvé wikipedia, mais nous avons trouvé l'authorité dont il dépend ("org") ainsi qu'une adresse pour trouver ce serveur : 199.19.56.1

```
host -v www.wikipedia.org  199.19.56.1

; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57772
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.wikipedia.org.     IN  A

;; AUTHORITY SECTION:
wikipedia.org.      86400   IN  NS  ns1.wikimedia.org.
wikipedia.org.      86400   IN  NS  ns0.wikimedia.org.
wikipedia.org.      86400   IN  NS  ns2.wikimedia.org.
```

```
;; ADDITIONAL SECTION:
ns0.wikimedia.org.  86400   IN  A   208.80.154.238
ns1.wikimedia.org.  86400   IN  A   208.80.153.231
ns2.wikimedia.org.  86400   IN  A   91.198.174.239

Received 147 bytes from 199.19.56.1#53 in 23 ms
```

Cette fois, en contactant le serveur responsable des sites en .org, nous obtenons une adresse correspondant à un serveur sachant trouver les pages de wikipedia.org : 208.80.154.238

```
host -v www.wikipedia.org  208.80.154.238

Trying "www.wikipedia.org"
Using domain server:
Name: 208.80.154.238
Address: 208.80.154.238#53
Aliases:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16878
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.wikipedia.org.     IN  A

;; ANSWER SECTION:
www.wikipedia.org.  86400   IN  CNAME   dyna.wikimedia.org.

Received 64 bytes from 208.80.154.238#53 in 98 ms

Trying "dyna.wikimedia.org"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6708
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;dyna.wikimedia.org.        IN  AAAA

;; ANSWER SECTION:
dyna.wikimedia.org. 600 IN  AAAA    2620:0:862:ed1a::1

Received 87 bytes from 208.80.154.238#53 in 96 ms
```

En deux étapes, nous obtenons l'adresse (sous un autre format, mais peu importe, c'est une adresse IP moderne) : 2620:0:862:ed1a::1

```
# remarquer les deux noms utilisés dans la réponse précédente.
!getent hosts www.wikipedia.org
```

```
2620:0:862:ed1a::1 dyna.wikimedia.org www.wikipedia.org
```

La commande getent effectue en fait toutes ces étapes automatiquement, jusqu'à recevoir une

réponse complète.

1.3 Analyse d'une URL

Un peu de python !

On souhaite, pour l'instant, extraire le TLD ainsi que le nom de domaine d'une URL.

```
    def domaine_tld(url):
        """
        Retourne le nom de domaine ainsi que le TLD d'une url donnée, sous␣
↪forme
        d'une chaine de caractère : "nom-de-domaine.tld[/des-trucs]"
        """
        # on se débarasse de tout ce qu'il y a après le premier slash
        indice_slash = -1
        i = 0
        while i < len(url):
            if url[i] == "/":
                indice_slash = i
                i = len(url)
            i = i + 1
        if indice_slash != -1:
            url_base = url[:indice_slash]
        else:
            url_base = url
        parties = url_base.split(".")
        # évitons les erreurs d'indices, en cas d'url non comprise
        if len(parties) >= 2:
            return parties[-2] + "." + parties[-1]
        print("format d'URL non compris : " + url)
        return url
```

```
    domaine_tld("prepa.blaise-pascal.fr/19-20/ptsi/info/")
```

```
'blaise-pascal.fr'
```

2 A la découverte du web !

Dans la suite, nous allons créer une chaîne de lien, représentant un petit morceau de la toile. Heureusement, python est livré avec plusieurs bibliothèques permettant d'effecter des requêtes automatiquement...

```
    import requests
```

```
    # pour effectuer une requête, on doit en plus préciser la manière dont on␣
↪veut accéder aux données.
```

```
    # Ici on utilise HTTP, mais on peut citer également HTTPS ou FTP.
    r = requests.get("http://pascal-lyc.spip.ac-rouen.fr")
```

```
    print(r.text[:150])
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xht
```

2.1 But du jeu

A partir de la page d'accueil du site du lycée, nous allons constituer une chaîne de sites liés les uns aux autres par les liens hypertext.

On souhaite changer de domaine le plus souvent possible.

Avant de se lancer, il est utile de savoir que dans une page HTML (les seules qui nous intéressent ici), les liens sont représentés sous la forme

```
<b>href=chaine-url</b>
```

où chaine-url est une chaine de caractère, avec la même convention qu'en python : elle est entourée de guillemets simples ou doubles.

Ainsi, après avoir chargé une page, on cherchera toutes les occurences de href (en minuscule ou majuscule) puis on trouvera les adresses cibles.

2.2 Premiers outils

Conversion en minuscule :

```
    s = "CeCi est une chaîne"
    s.lower()
```

```
'ceci est une chaîne'
```

Notre premier outil va être une fonction de recherche de motif dans une chaîne. On veut pouvoir chercher toutes les occurences de "href".

```
    def prochaine_occ(motif, texte, i):
        """
        Retourne l'indice (supérieur ou égal à i) du premier caractère de␣
↪texte
        tel que le début de text[i:] soit exactement motif, c'est à dire␣
↪l'indice du début de
        la prochaine occurence de motif dans la chaîne texte.
        On considère que motif et texte sont en minuscules.

        Retourne -1 si aucune occurence n'est trouvée après l'indice i.
        """
```

```
        j = i
        n = len(motif)
        m = len(texte)
```

```
    texte = r.text.lower()
```

```
    prochaine_occ("href", texte, 1000)
```

```
    texte[1151:1251]
```

```
'href="plugins-dist/mediabox/colorbox/black-striped/colorbox.css" type="text/
↪css" media="all" /><link'
```

Grâce à notre premier outil, on va pouvoir chercher la prochaine URL présente.

```
    def prochaine_url(texte, i):
        """
        Retourne la première url trouvée dans texte après l'indice i ainsi␣
↪que l'indice du
        premier caractère après cette url.

        Si aucune url n'est trouvée, retourne "", -1
        """
        j = prochaine_occ("href=", texte, i)
        if j == -1:
            return "", -1
        j = j + 5  # on se place après le =
        # teste pour savoir le type de guillemet
        if texte[j] == '"':
            fin = prochaine_occ('"', texte, j + 1)
        elif texte[j] == "'":
            fin = prochaine_occ("'", texte, j + 1)
        else:
            fin = j + 1
        if fin == -1:
            return "", -1
        return texte[j + 1: fin], fin + 1
```

Nous sommes maintenant en mesure de lister toutes les URL référencées dans une page.

```
    def liste_url(texte):
        """
```

```
        Retourne la liste des url trouvées dans texte.
        """
        L = []
        i = 0
        while i != -1:
            url, i = prochaine_url(texte, i)
            L.append(url)
        return L
```

```
    L = liste_url(texte)
    len(L)
```

```
248
```

```
    L[:5], L[-1]
```

```
(['http://spip.ac-rouen.fr/plugins/divers/charte_ministere/css/css_ministere.
↪css',
  'http://spip.ac-rouen.fr/plugins/divers/charte_ministere/css/
↪css_ministere_sommaire.css',
  'plugins-dist/mediabox/colorbox/black-striped/colorbox.css',
  'plugins-dist/porte_plume/css/barre_outils.css?1548080449',
  'local/cache-css/cssdyn-css_barre_outils_icones_css-3a28cb8f.css?
↪1591733985'],
 'spip.ac-rouen.fr/plugins/divers/charte_ministere/css/css_ministere.css')
```

Il nous faut modifier un peu notre fonction domaine_tld, pour accepter les URL qui contienent "http" ou "https".

```
    def domaine(url):
        """
        On applique la fonction domaine_tld à la partie ne contenant pas http:
↪// ou https://
        """
        if url[:4] == "http":
            debut = prochaine_occ("//", url, 0)
            return domaine_tld(url[debut+2: ])
        return domaine_tld(url)
```

L'outil suivant ne sera pas commenté. Il s'agit juste de normaliser les URL trouvées, ainsi que d'éliminer les liens qui ne pointent par vers des pages lisibles.

```
    import os.path
    from urllib.parse import urljoin, urlparse
```

```
    extensions = [".jpg", ".png", ".gif", ".css", ".js", ".json"]
    def normalise_urls(L, url):
        """
        Retourne la liste des URL absolues et filtre les liens inutiles.
```

```
        """
        L2 = []
        for u in L:
            _, ext = os.path.splitext(u)
            for banni in extensions:
                if banni in ext:
                    u = ""
            if u != "":
                res = urljoin(url, u)
                if urlparse(res).scheme in ("http", "https"):
                    L2.append(res)
        return L2
```

2.3 Outils finaux

Pour charger les différentes pages trouvées, on utilisera la fonction suivante qui fait le tri des réponses illisible et s'assure que le texte est bien en minuscule.

```
    def html(url):
        """
        Retourne le contenu html (chaîne de caractère) d'une page web␣
↪identifiée par son URL ou ""
        si aucun contenu texte n'est trouvé.
        """
        try:  # construction spéciale
            r = requests.get(url)
            if r.status_code != 200 or "text/html" not in r.headers["Content-
type"]:
                return ""
            return r.text.lower()
        except: # si une erreur apparaît, on exécute les instructions␣
↪suivantes
            return ""
```

Pour trouver tous les liens d'une page, on utilisera la fonction

```
    def trouve_liens(url):
        texte = html(url)
        L = liste_url(texte)
        return normalise_urls(L, url)
```

2.4 Notre réseau

Voici notre règle d'exploration, à partir de la page "http://pascal-lyc.spip.ac-rouen.fr" (ou google, ou tout autre page à votre convenance.). La collection de liens est vide au départ

— charger tous les liens de la page.

— ajouter tous les liens trouvés à notre collection, sans doublon.
— identifier, parmi les liens trouvés, les autres domaines présents et celui qui est majoritaire. En cas d'égalité, choisir au hasard.
— charger une page au hasard de ce domaine parmi les liens disponibles et recommencer.
— si on ne trouve aucun lien ou si aucun domaine n'est différent de la page courante, on charge un lien au hasard de notre collection.

But : atteindre 1000 liens.

Pour éviter les doublons facilement, nous allons utiliser une nouvelle structure de donnée python : les ensembles. Comme en mathématiques, un ensemble ne peut pas contenir plusieurs fois le même élément, mais les éléments ne sont pas ordonnés (on ne peut pas y accéder par un indice)

```
s = set([1,2,2,3,4])
s
```

```
{1, 2, 3, 4}
```

```
s[0]
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-24-c9c96910e542> in <module>
----> 1 s[0]


TypeError: 'set' object does not support indexing
```

```
s2 = set([4,5,6])
s.union(s2)
```

```
{1, 2, 3, 4, 5, 6}
```

Nous allons charger tous les liens d'une page, puis, pour trouver le ou les domaines majoritaires, trier la liste des domaines par ordre alphabetique. Ainsi, compter les liens pointant vers le même domaine revient à compter le nombre d'éléments égaux consécutifs dans une liste

```
def nombre_consecutifs(L, i):
    """
    Compte le nombre d'éléments égaux à L[i] et consécutifs dans L.
    Si la valeur de retour est n, cela signifie que L[i: i+n] est composé␣
↪de n fois
    le même élément et que L[i + n] est différent ou que  i + n == len(L)
    """
```

```
nombre_consecutifs([1,1,2,2,2,3,4], 2)
```

```
3
```

La prochaine étape, étant donnée une liste de domaines, est d'extraire le ou les domaines majoritaires (on s'occupera plus tard d'exclure le domaine courant)

```
def domaines_majoritaires(L):
    """
    L est une liste de nom de domaines.
    Retourne la liste des domaines qui apparaissent le plus souvent (les␣
↪domaines à égalité sont
    tous retournés dans la liste).
    """
    L.sort()
    courant = 0  # indice courant dans L
    maxi = 0  # le nombre maximal de domaines égaux trouvés
    # jusqu'à présent
    dom = []  # la liste des domaines maximaux
    while courant < len(L):
        n = nombre_consecutifs(L, courant) # nbre d'éléments consécutifs␣
↪égaux
        if n > maxi:  # on a trouvé un domaine apparaissant plus souvent !
            dom = [L[courant]]
            maxi = n
        elif n == maxi:  # cas d'égalité, deux (ou plus) domaines sont␣
↪majoritaires.
            dom.append(L[courant])
        courant = courant + n  # on avance dans L : on a traité n␣
↪éléments.
    return dom
```

```
domaines = [domaine(url) for url in L2 if domaine(url) != "ac-rouen.fr"]
```

```
domaines_majoritaires(domaines)
```

```
['ac-normandie.fr']
```

```
import random
```

```
url = "http://pascal-lyc.spip.ac-rouen.fr"
limite = 300
liens = set()  # notre ensemble de liens
```

```
    nb_essais = 0  # pour éviter de tourner en boucle, on s'assure
    # de ne pas rester bloqué dans une boucle infinie.
    # Par exemple, on pourrait tomber sur des pages sans liens et ne pas␣
↪pouvoir sortir de
    # notre petit coin de web.
    while len(liens) < limite and nb_essais < limite:
        nb_essais += 1
        print(str(nb_essais) + ". chargement " + url)
        L = trouve_liens(url)  # liste des liens trouvés.
        if len(L) == 0:  # on a rien trouvé !
            # on choisit au hasard un lien parmi ceux déjà trouvés.
            # dans ce cas, on pourrait rester bloqué !
            url = random.choice(list(liens))
        else:
            # on ajoute à notre collection les liens trouvés.
            liens = liens.union(set(L))
            # construction de la liste des autres domaines.
            domaines = []
            domaine_courant = domaine(url)
            for nv_url in L:
                nv_domaine = domaine(nv_url)
                if nv_domaine != domaine_courant:
                    domaines.append(nv_domaine)
            if len(domaines) == 0:
                domaines = [domaine_courant]
            # on chopit au hasard un domaine majoritaire
            dom = random.choice(domaines_majoritaires(domaines))
            # contruction de la liste des url pointant vers le domaine choisi
            L2 = [u for u in L if domaine(u) == dom]
            # choix d'une nouvelle page
            url = random.choice(L2)
```

```
1. chargement http://pascal-lyc.spip.ac-rouen.fr
2. chargement http://www.ac-normandie.fr/services-deconcentres/dsden-de-l-orne/
```

```
    list(liens)[:10]  # les 10 "premiers" liens
```

```
['http://pascal-lyc.spip.ac-rouen.fr/spip.php?breve55',
 'http://pascal-lyc.spip.ac-rouen.fr/spip.php?article1553#1553',
 'http://pascal-lyc.spip.ac-rouen.fr/spip.php?rubrique179',
 'http://www.ac-normandie.fr/academie/dsden/',
 'http://pascal-lyc.spip.ac-rouen.fr/spip.php?article1968',
 'http://pascal-lyc.spip.ac-rouen.fr/spip.php?breve61&amp;lang=fr',
 'http://www.ac-normandie.fr/services-deconcentres/dsden-du-calvados/',
 'http://www.ac-normandie.fr/politique-educative/l-ecole-et-la-societe/
↪generation-2024/',
 'http://pascal-lyc.spip.ac-rouen.fr/spip.php?article1594',
```

```
    'http://www.ac-normandie.fr/non-au-harcelement-174557.kjsp?
↪rh=1574953009484']
```

On peut également penser à retenir (dans un ensemble !) les domaines déjà visités plutôt que d'exclure seulement le domaine courant.

```
    def reseau(url_depart, limite):
        url = url_depart
        domaines_visites = set()  # ensemble vide
        liens = set()  # notre ensemble de liens
        nb_essais = 0  # pour éviter de tourner en boucle, on s'assure
        # de ne pas rester bloqué dans une boucle infinie.
        # Par exemple, on pourrait tomber sur des pages sans liens et ne pas␣
↪pouvoir sortir de
        # notre petit coin de web.
        while len(liens) < limite and nb_essais < limite:
            nb_essais += 1
            print(str(nb_essais) + ". chargement " + url)
            L = trouve_liens(url)  # liste des liens trouvés.
            if len(L) == 0:  # on a rien trouvé !
                # on choisit au hasard un lien parmi ceux déjà trouvés.
                # dans ce cas, on pourrait rester bloqué !
                url = random.choice(list(liens))
            else:
                # on ajoute le domaine à l'ensemble des domaines visités
                domaines_visites.add(domaine(url))
                # on ajoute à notre collection les liens trouvés.
                liens = liens.union(set(L))
                # construction de la liste des autres domaines.
                domaines = []
                for nv_url in L:
                    nv_domaine = domaine(nv_url)
                    if nv_domaine not in domaines_visites:
                        domaines.append(nv_domaine)
                if len(domaines) == 0:
                    domaines = [domaine_courant]
                # on choisit au hasard un domaine majoritaire
                dom = random.choice(domaines_majoritaires(domaines))
                # contruction de la liste des url pointant vers le domaine␣
↪choisi
                L2 = [u for u in L if domaine(u) == dom]
                # choix d'une nouvelle page
                url = random.choice(L2)
        return liens, domaines_visites
```

```
    liens, domaines = reseau("http://pascal-lyc.spip.ac-rouen.fr", 1000)
```

```
1. chargement http://pascal-lyc.spip.ac-rouen.fr
2. chargement http://www.ac-normandie.fr/services-deconcentres/dsden-de-l-orne/
```

```
3. chargement https://www.education.gouv.fr/bac-brevet-2020-les-reponses-vos-
questions-303348
4. chargement https://eduscol.education.fr/
5. chargement http://ses.ens-lyon.fr/
6. chargement http://www.touteconomie.org/index.php?arc=dc037g
7. chargement http://www.journeeseconomie.org/
8. chargement https://www.lyon-finance.org/app/uploads/2020/04/plemerrer.pdf
9. chargement http://www.ac-normandie.fr/eparents-l-application-pour-les-
parents-d-enfants-du-cp-a-la-terminale-174589.kjsp?rh=1574953009484
10. chargement http://pascal-lyc.spip.ac-rouen.fr/spip.php?breve54
11. chargement http://pascal-lyc.spip.ac-rouen.fr?lang=fr
12. chargement https://nero.l-educdenormandie.fr/
13. chargement https://fonts.googleapis.com/css?family=oswald
14. chargement https://eduscol.education.fr/pid23410/le-socle-commun-et-l-
evaluation-des-acquis.html
15. chargement https://www.reseau-canope.fr/education-prioritaire/accueil.html
```

```
domaines
```

```
{'ac-normandie.fr',
 'ac-rouen.fr',
 'ac-rouen.fr?lang=fr',
 'education.fr',
 'ens-lyon.fr',
 'gouv.fr',
 'journeeseconomie.org',
 'l-educdenormandie.fr',
 'reseau-canope.fr',
 'touteconomie.org'}
```

```
len(liens)
```

```
1014
```

Encore mieux, on choisit au hasard parmi les domaines non explorés !

```
def url_hasard(liens, domaines_visites):
    """
    Trouve, si possible, une url d'un domaine non encore visité.
    """
    nouveaux_liens = set([url for url in liens if domaine(url) not in␣
↪domaines_visites])
    if len(nouveaux_liens) > 0:
        ensemble = nouveaux_liens
    else:
        print("Aucun lien vers un nouveau domaine en stock")
        ensemble = liens
    url = random.choice([url for url in ensemble])
    return url
```

```
    def reseau2(url_depart, limite):
        url = url_depart
        domaines_visites = set()  # ensemble des domaines déjà visités
        liens = set()  # notre ensemble de liens
        nb_essais = 0  # pour éviter de tourner en boucle, on s'assure
        # de ne pas rester bloqué dans une boucle infinie.
        # Par exemple, on pourrait tomber sur des pages sans liens et ne pas␣
↪pouvoir sortir de
        # notre petit coin de web.
        while len(liens) < limite and nb_essais < limite:
            nb_essais += 1
            print(str(nb_essais) + ". chargement " + url)
            L = trouve_liens(url)  # liste des liens trouvés.
            if len(L) == 0:  # on a rien trouvé !
                # on choisit au hasard un lien parmi ceux déjà trouvés.
                # dans ce cas, on pourrait rester bloqué !
                if len(liens) > 0:
                    url = url_hasard(liens, domaines_visites)
                else:
                    return "Echec critique ! Aucun lien trouvé sur la page de␣
↪départ"
            else:
                # on ajoute le domaine à l'ensemble des domaines visités
                domaines_visites.add(domaine(url))
                # on ajoute les liens trouvés.
                liens = liens.union(set(L))
                # construction de la liste des autres domaines.
                domaines = []
                for nv_url in L:
                    nv_domaine = domaine(nv_url)
                    if nv_domaine not in domaines_visites:
                        domaines.append(nv_domaine)
                if len(domaines) == 0:  # pas de nouveau domaine trouvé ici,␣
↪dommage
                    print("Pas de nouveau domaine ici")
                    url = url_hasard(liens, domaines_visites)
                else:
                    # on choisit au hasard un domaine majoritaire
                    dom = random.choice(domaines_majoritaires(domaines))
                    # contruction de la liste des url pointant vers le␣
↪domaine choisi
                    L2 = [u for u in L if domaine(u) == dom]
                    # choix d'une nouvelle page
                    url = random.choice(L2)
        return liens, domaines_visites
```

```
liens, domaines = reseau2("http://pascal-lyc.spip.ac-rouen.fr", 1000)
```

```
1. chargement http://pascal-lyc.spip.ac-rouen.fr
2. chargement http://www.ac-normandie.fr/pedagogie/enseignements/
3. chargement http://maths.discip.ac-caen.fr
4. chargement
http://eduscol.education.fr/maths/actualites/actualites/article/pour-aller-
moins-loin.html
5. chargement http://www.cnrs.fr/
6. chargement https://www.instagram.com/p/cbsz-brdpx8/
7. chargement https://www.instagram.com/p/cbsxrikjheb/
8. chargement https://soundcloud.com/cnrs_officiel/sets/covid19-parole-a-la-
science
9. chargement https://style.sndcdn.com
10. chargement http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
11. chargement
https://us.netdonor.net/page/6650/donate/1?ea.tracking.id=license-footer
12. chargement https://www.flickr.com/people/pingnews/
13. chargement https://twitter.com/flickr
```