# Python et Unix

# CSII 2 – Examen Semestre 2, 2003

Durée : 2 heures
**Aucun documents autorisés**

## *Exercice 1 : Python (5 points)*

1.2 ) Quelle est la différence entre un tuple et une liste ? *(0,5 point)*

Le tuple n'est pas modifiable après création.

1.3 ) Soit L une liste, comment accéder : *(1 point)*

- Au i$^{éme}$ élement ?
- Aux 3 premiers élements ?
- Au dernier élement ?

(-0,5 par réponse fausse ou manquante)
L[i-1]
L[0 :3] ou L[ :3]
L[-1] ou L[len(L)] ou L[len(L) -1]

1.4 ) Qu'est-ce qu'un dictionnaire python ? Quelles sont les principales opérations qu'il permet ? *(1,5 point)*

Une liste dont les indices sont des objets arbitraires (clés).
Une table de hash faisant correspondre à une clé une valeur.
Une clé est unique dans la table de hash, les valeurs sont de types quelconques.
Principales opérations : insertion de nouveau couple clé/valeur, accès en ***temps constant*** à l'élément associé à une clé, test de présence (temps constant), (récupérer toutes les clés) …

ATTENTION : un dictionnaire n'est pas une liste de tuples de dimension 2 [(clé,valeur), (clé,valeur)…] , c'est ainsi affiché sur un print, mais la structure de données n'a rien à voir avec une liste.

1.5 ) Comment créer un module python ? Comment s'en servir dans un autre script ? *(1 point)*

Créer un fichier module.py. (0,5 point)
Utiliser une des variantes de import, from module import … (0,5 point)

1.6 ) Quelles sont les structures de contrôle python permettant de créer des boucles ? Ecrivez un petit exemple (4 lignes) illustrant l'utilisation de chacune d'elle. *(1 point)*

(1 exemple de for et de while requis)

```
for elt in ['toto','tata'] :
    print elt
```

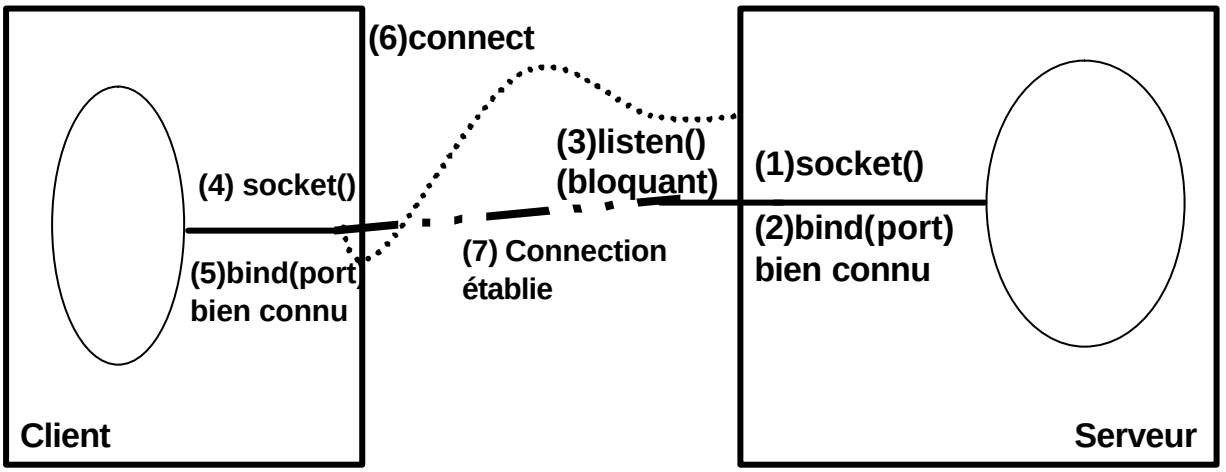
```
for i in range(5) :
        print i

while 1 :
        sleep(1)
        print "zzzzzzz"
```

## *Exercice 2 : Sockets (6 points)*

Le schéma suivant décrit l'établissement d'une connection TCP entre un client et un serveur.

2.1 ) Corrigez (en justifiant votre réponse) les erreurs qu' il contient. *(3,5 points)*




3 erreurs à corriger :

Le listen n'est pas bloquant, c'est le accept qui le suit (manquant) qui devrait l'être

Le client ne réalise pas de bind

La connection est établie sur une nouvelle socket de communication, rendue par accept

Les propositions de corrections fausses ont été pénalisées.

2.2 ) Citez deux *domaines* de socket. Expliquez pour chacun de quoi est composé une adresse de socket dans ce domaine. *(1,5 point)*
AF_INET : domaine Internet, adresse = (IP,Port)
AF_UNIX : domaine local Unix , adresse = nom de fichier

2.3 ) Quelles sont les principales caractéristiques du protocole TCP, du protocole UDP ? *(1 point)*

TCP : connecté, mode flux, pas de limite de taille, contrôle de congestion, ni pertes, ni duplication, ni déséquencement de paquets
UDP : non-connecté, limite de taille par paquet, pertes, désequencements et duplications possibles, possibilité de Multicast, efficace (surcouche IP)

## *Exercice 3 : UNIX (5 points)*

3.1 ) A quoi peut servir de recompiler son noyau ? *(1 point)*
Intégrer des modules (drivers…), mettre à jour le kernel, éliminer des choses inutiles par rapport à sa configuration…

3.2 ) Quel(s) service(s) prends en charge de l'affichage en mode fenêtré sous Unix/Linux ? En quoi cette conception diffère-t-elle d'autres OS (Windows, MacOS < MacOS X…) ? *(1 points)*
Le serveur X, les applications fenetrées étant des clients X. La plupart des autres OS intègrent le fenêtrage dans le noyau avec plusieurs conséquences essentielles :
* il est impossible d'en changer, mais en contrepartie l'aspect des applications est plus homogène car le fenêtrage est connu à priori
* Etant intégré au noyau, il a un statut privilégié ; cela permet une meilleure utilisation des ressources (carte vidéo…), mais a aussi tendance à rendre le système moins stable
* Le serveur X est conçu pour fonctionner en réseau ; c'est pourquoi on ne présuppose que très peu des capacités graphiques de la machine, en revanche il est aisé d'exporter des display sur ou depuis des machines distantes, et ce de façon transparente

3.3 ) Quel est le rôle du DNS ? Du protocole DHCP ? Quel service(s) sont rendus par Apache ? Par Samba ? Quels daemons sont responsables de ces services ? *(2 points)*
DNS : Domain Name Service, daemon named, fournit les translations de nom de domaine vers IP et réciproquement (192.168.10.12 <-> toto.tata.epsi.fr)
DHCP : daemon dhcpd, attribue des adresses IP à la demande sur un sous-réseau, permet également l'auto-configuration des paramètres réseau
Apache : daemon httpd, serveur Web port 80, un des plus utilisés au monde, permet l'hébergement de sites web
Samba : daemon smbd, nmbd : fournit la compatibilité avec les stations Windows (Netbios <-> IP) partage de fichiers et imprimantes …

3.5 ) En quoi consiste la technique de clés privées/ clé publiques ? *(2 point)*

Cryptage asymétrique ; on diffuse une clé publique aux interlocuteurs qui souhaitent nous envoyer des messages, cette clé permet de crypter des messages mais pas de les décrypter, il faut pour cela utiliser la clé privée conservée sur l'ordinateur d'origine. Ainsi la clé de décryptage ne transite pas sur le réseau, => sécurité accrue.
De nombreux outils utilisent cette technique en particulier les SSL secure socket layer, qui forment la base de ssh, scp, https etc… mais on peut aussi citer RSA ou PGP (pas DES !!)

## *Exercice 4 : Multithreading (8 points)*

Soit un sas de banque, composé de deux portes et d'une pièce intermédiaire. Le système doit assurer

1. que les deux portes du sas ne soient jamais ouvertes en même temps. On considère (hypothèse simplificatrice) qu'il est interdit de tenir la porte à un autre client.
2. que le sas ne contienne jamais plus de trois personnes

Le comportement d'un client est le suivant :

```
porteLock = thread.allocate_lock()
sasCapacite = threading.Semaphore(3)

def client () :
        exterieur.append(thread.get_ident())
        while 1 :
                sasCapacite.acquire()
                ouvrir (P1)
                entrer(exterieur,sas)
                fermer(P1)
                porteLock.release()
                time.sleep(1) # le client reste un instant dans le sas
                ouvrir (P2)
                entrer(sas,banque)
                fermer(P2)
                sasCapacite.release()
                time.sleep(random.random() * 20) # le client est au guichet
                sasCapacite.acquire()
                ouvrir (P2)
                entrer(banque,sas)
                fermer(P2)
                time.sleep(1) # le client reste un instant dans le sas
                ouvrir (P1)
                entrer(sas,exterieur)
                fermer(P1)
                sasCapacite.release()
                time.sleep(random.random() *100)
def ouvrir (porte) :
        porteLock.acquire()
def fermer (porte) :
        porteLock.release()
for i in range(10) :
        thread.start_new_thread(client,())
raw_input("Entrée pour quitter!")
```

4.1 ) Insérer des mécanismes (Semaphore, verrous, moniteurs selon ce qui vous parait approprié) permettant de respecter les contraintes 1 *(3 points)* et 2 *(3 points)*. (insérez aux endroits appropriés des appels à **acquire, release, wait, notify ou notifyall ).**

4.2 ) Ajoutez un corps principal qui instancie 10 threads clients et attends que l'utilisateur appuie sur entrée pour quitter.*(2 points)*

Attention les réponses à deux verrous (un par porte étaient fausses) ; cette solution a été proposée tellement souvent qu'elle mérite une explication : on propose :

```
1 def  ouvrir (porte):
2        if (porte == P1) :
3                semP2.acquire()
4        else :
5                semP1.acquire()
```

Or un thread T1 peut aller jusque ligne 2 et faire le test (vrai), puis être interrompu, un deuxième thread T2 prends la main et traverse en exécutant les lignes 2,4,5, quand T1 reprends la main il traverse et les deux portes sont ouvertes.

Autre erreur fréquente, acquérir le verrou porte avant le sémaphore capacitéSas, ce qui peut être fâcheux si le sas est plein, on se retrouve bloqué : plus personne d'autre ne peut ouvrir de porte (vu qu'on a le verrou « à la main »), et on ne peut plus avancer avant que le sas se vide (ce qui est devenu impossible).

Ces solutions, certes un peu fausses, ont tout de même obtenu environ la moitié des points sur cette question assez délicate.