Travail pratique 4
Programmation d'un client Python de type « Dropbox »

De F.-Nicola Demers

Ce travail pratique (15 %) peut être réalisé en équipe de 2 étudiant(e)s ou seul. Vous devez préciser dans le rapport l'implication de chacun(e). Les critères d'évaluation détaillés seront disponibles sur LEA. La date de remise de ce travail est le mercredi **26 avril 2016** .

Vous devez écrire deux applications client en Python 3 en objet communiquant correctement avec un serveur qui s'exécute sur la machine Ubuntu disponible sur Internet pour le cours.

Ces applications sont des clients d'un clone Dropbox. L'application serveur « de type Dropbox » est fournie par le professeur disponible sur le serveur Ubuntu en ligne.

La première application client s'exécute sur un ordinateur avec un interprète Python 3 (Windows ou Linux). La deuxième application client s'exécute sur Android. Nous verrons en cours comment développer cette application.

L'en-tête de vos fichiers doit contenir les lignes suivantes :

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

1. Applications client Python (6 points)

Sur le serveur Ubuntu 159.203.9.85, vous pouvez copier le serveur **/tp4/serveur_tp4.pyc** (version compilée) dans votre compte. Pour partir ce serveur, vous passez le paramètre du port d'écoute et le langage de communication (XML ou JSON). Voici un exemple d'exécution :

```
python3 ~/serveur_tp4.pyc 50000 xml
```

pour une communication XML ou

```
python3 ~/serveur_tp4.pyc 50000 json
```

pour une communication JSON.

Vous n'avez pas le code source de ce programme mais son comportement est expliquée dans cet énoncé.

Votre application client doit pouvoir permettre une communication en langage XML ou JSON avec le serveur. Voici un exemple d'exécution de votre premier client :

```
python3 ~/client_tp4.pyc 50000 xml
```

pour une communication XML ou

```
python3 ~/client_tp4.pyc 50000 json
```

pour une communication JSON.

Votre application client doit aussi pouvoir offrir une interface texte avec l'utilisateur (un invite) pour effectuer des tests de communication. L'exécution doit pouvoir se faire comme suit :

```
python3 ~/client_tp4.pyc 50000 json prompt
```

ou

```
python3 ~/client_tp4.pyc 50000 xml prompt
```

Voici la liste des commandes qu'on peut taper par l'invite du côté du client et les réponses qu'on obtient :

Exemples de commandes envoyés du client vers le serveur	**Exemples de réponse obtenue en ligne de commande**	**Description**
connecter?	Oui	Cela permet de vérifier si le serveur est bien en ligne.
nomServeur?	Ubuntu Dropbox 1.0	Cela permet de recevoir le nom du serveur
listeDossier?	d1/d2/d3	Cela permet de recevoir la liste des dossiers

Exemples de commandes envoyés du client vers le serveur	**Exemples de réponse obtenue en ligne de commande**	**Description**
	d4/d5/d6 ...	créés sur le serveur « Dropbox »
dossier? d1/d2/d3	Oui Non	Cela permet de vérifier l'existence d'un dossier en particulier.
creerDossier? d1/d2/d3	Oui Dossier existe déjà.	Cela permet de créer un dossier.
televerser? d1/fichier2	Ok Dossier n'existe pas. Fichier existe déjà.	Cela permet de téléverser (*upload*) d'un fichier (disponible dans le dossier d1 du client)
telecharger? d1/fichier2	Ok Dossier n'existe pas. Fichier n'existe pas.	Cela permet de télécharger (*download*) d'un fichier (disponible dans le dossier d1 du serveur). Le fichier est créé sur le client dans le dossier courant.
supprimerDossier? d1/d2 supprimerFichier? d1/fichier1	Ok Dossier n'existe pas. Fichier n'existe pas.	Cela permet de supprimer un dossier ou un fichier sur le serveur.
fichier? d2/fichier3	Oui Non	Cela permet de vérifier l'existence d'un fichier sur le serveur.
identiqueFichier? d3/fichier4	Oui	Cela permet de vérifier si un fichier local (client) est identique au fichier sur le serveur.

Exemples de commandes envoyés du client vers le serveur	Exemples de réponse obtenue en ligne de commande	Description
	Non	
fichierRecent? d4/fichier5	Oui Non	Cela permet de vérifier si un fichier local est plus récent que le même fichier sur le serveur.
miseAjour d1/d2	Ok Dossier n'existe pas.	Cela permet de mettre à jour un dossier local et le même dossier sur le serveur. Les fichiers les moins récents sont remplacés par les plus récents (côté client/côté serveur).
quitter	Bye	Le serveur coupe la communication avec le client.

Si le paramètre « prompt » n'est pas fourni à l'exécution, le client ne communique pas avec l'utilisateur. Il fonctionne alors automatiquement en mettant à jour le dossier courant au niveau du client en communiquant avec le serveur (comme Dropbox peut le faire).

Plus généralement, vos applications client doivent respecter le tableau des exemples de requêtes XML/JSON et des réponses fournies par le serveur des pages suivantes :

Exemples de message envoyé du client (XML ou JSON)	Exemples de message réponse envoyé par le serveur en XML	Exemples de message réponse envoyé par le serveur en JSON	Description
<bonjourServeur /> { "salutation": "bonjourServeur" }	<bonjourClient />	{ "salutation": "bonjourClient" }	A la connection du client au serveur, le client teste la communication en envoyant un message de salutation et le serveur répond aussi par une salutation.
<questionNomServeur />	<nomServeur>Ubuntu Dropbox 1.0</nomServeur>	{ "nomServeur": "Ubuntu Dropbox 1.0" }	Cela envoie le nom du serveur
<questionListeDossiers> d1 </questionListeDossiers> { "questionListeDossiers": "d1" }	<listeDossiers> <dossier>d1/d2/d3 </dossier> <dossier>d1/d3</dossier> <dossier>d1/d4</dossier> … </listeDossiers> <*erreurDossierInexistant* /> <*erreurDossierLecture* />	{ "listeDossiers": { "dossier": ["d1/d2/d3", "d1/d3", "d1/d4", …] } }	Cela envoie la liste des dossiers créés sur le serveur « Dropbox » à l'intérieur du dossier d1.
<questionListeFichiers> *d1* *</questionListeFichiers>* *{* *"questionListeFichiers":* *"d1"* *}*	*<listeFichiers>* *<fichier>d1/f1* *</fichier>* *<fichier>d1/f2</fichier>* *<fichier>...</fichier>* … *</listeFichiers>* <*erreurDossierInexistant* />	*{* *"listeFichiers": {* *"fichier": [* *"d1/f1",* *"d1/f2",* *"d1/f3"* *]* *}* *}*	Cela envoie la liste des fichiers créés sur le serveur « Dropbox » à l'intérieur du dossier d1. C'est un appel non récursif.

Exemples de message envoyé du client (XML ou JSON)	Exemples de message réponse envoyé par le serveur en XML	Exemples de message réponse envoyé par le serveur en JSON	Description
	<erreurDossierLecture />		
<creerDossier> d7/d8/d9 </creerDossier> *{* *"creerDossier":* *"d7/d8/d9"* *}*	<ok /> <erreurDossierExiste /> <erreurDossierInexistant />	{ "reponse": "ok" }	Cela crée un dossier sur le serveur « Dropbox ». Les dossiers d7 et d8 doivent exister.
<televerserFichier> <nom>...</nom> <dossier>...</dossier> <signature>... </signature> <contenu>...</contenu> <*date*>...</*date*> </televerserFichier> { "televerserFichier": { "nom": "...", "dossier": "...", "signature": "…" "contenu": "…" "date": "…" } }	<ok /> <erreurFichierExiste /> <erreurDossierInexistant /> <erreurSignature />	{ "reponse": "ok" } { "reponse": "erreurFichierExiste" }	Cela permet de pousser un fichier sur le serveur (*upload*). A la réception, le serveur vérifier la signature du contenu et génère une erreur si différent.
<telechargerFichier>	<fichier>	{	Cela permet de télécharger un fichier du

Exemples de message envoyé du client (XML ou JSON)	**Exemples de message réponse envoyé par le serveur en XML**	**Exemples de message réponse envoyé par le serveur en JSON**	**Description**
<nom>...</nom> <dossier>...</dossier> </telechargerFichier> { "telechargerFichier": { "nom": "...", "dossier": "…" } }	<signature>...</signature> <contenu>...</contenu> <date>...</date> </fichier> *<erreurFichierInexistant />* *<erreurFichierLecture />*	"fichier": { "signature": "...", "contenu": "...", "date": "..." } }	serveur vers le client (*download*). Le client vérifie la signature du contenu envoyé
<supprimerFichier> <nom>...</nom> <dossier>...</dossier> </supprimerFichier> { "supprimerFichier": { "nom": "...", "dossier": "…" } }	<ok /> <erreurDossierInexistant /> *<erreurFichierLecture />*	{ "reponse": "ok" }	Cela permet de supprimer un fichier sur le serveur.
<supprimerDossier> *d1/d2/d3* *</supprimerDossier>* { "supprimerDossier": "d1/d2/d3" }	*<ok />* *<erreurDossierInexistant />* *<erreurDossierLecture />*	*{ "reponse": "ok" }*	*Cela permet de supprimer un dossier sur le serveur.*
<questionFichierRecent> <nom>...</nom>	<oui />	{ "reponse": "oui" }	Cela permet de vérifier si un fichier est plus récent sur le client que sur le

Exemples de message envoyé du client (XML ou JSON)	**Exemples de message réponse envoyé par le serveur en XML**	**Exemples de message réponse envoyé par le serveur en JSON**	**Description**
<dossier>..</dossier> <date>...</date> </questionFichierRecent> { "questionFichierRecent": { "nom": "...", "dossier": "...", "date": "…" } }	<non /> <erreurFichierInexistant /> <*erreurFichierLecture* />	{ "reponse": "non" }	serveur en comparant les dates de modification du fichier.
<quitter /> { "action": "quitter" }	<ok />	{ "reponse": "ok" }	Le serveur coupe la communication avec le client.

2. Développement des tests unitaires (4 points)

Vous devez créer deux classes de test contenant l'ensemble des tests unitaires pour chaque méthode, chaque commande avec le serveur décrite ci-haut. La couverture doit être complète. Vos classes de test doivent utiliser des mocks pour éviter toute communication « réelle » avec le serveur.

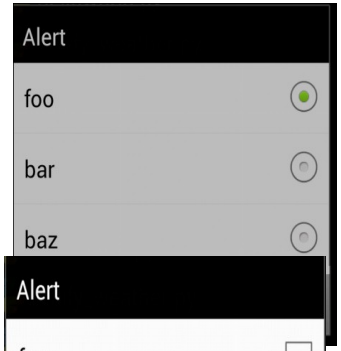
Les classes de test s'appellent `test_client_xml.py` et `test_client_json.py` La première teste toute la communication XML avec un serveur « mocké ». La deuxième classe teste toute la communication JSON avec un serveur « mocké ».

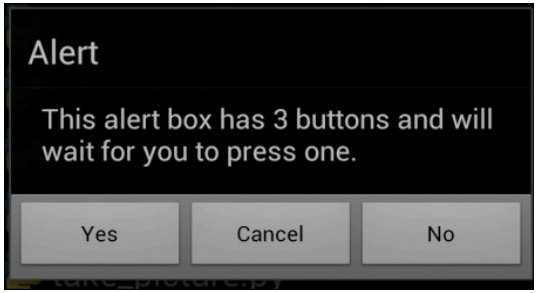
Vous devez utiliser les classes Python unittest et Mock.

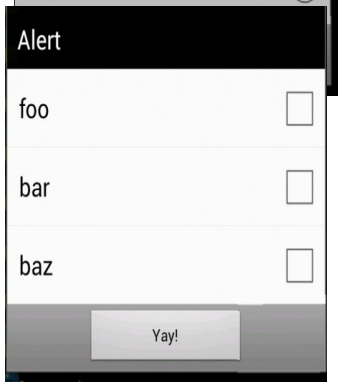
Il n'est pas nécessaire de tester la communication avec l'utilisateur. Nous nous concentrons sur le bon fonctionnement de la communication avec le serveur.

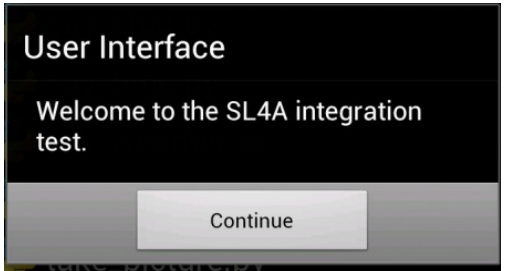
3. Construction du client Android (4 points)

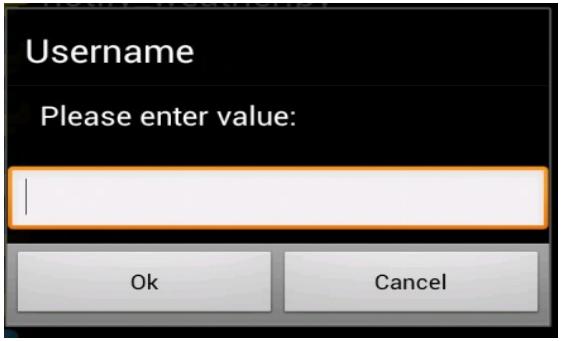
Vous devez écrire une application Android en Python nommée *client.py* qui s'exécute sur un téléphone ou une tablette Android 4/5 (vous pouvez l'exécuter sur votre poste Windows dans un émulateur durant le développement). Cette application doit ouvrir un socket sur le serveur Ubuntu sur un des ports de communication dédiés (que le professeur vous a assigné) et doit pouvoir envoyer des requêtes au serveur (ci-haut). L'application doit offrir une interface graphique simple à l'utilisateur pour l'envoi de commandes. L'application de type client doit transcrire les commandes de l'utilisateur en commande XML ou JSON (voir le tableau ci-haut) et interpréter la réponse du serveur et les afficher sur Android en mode graphique. Voici des exemples des outils d'interfaces possibles :





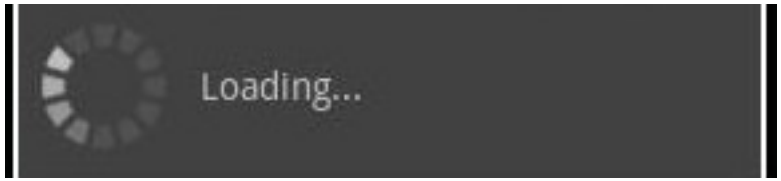


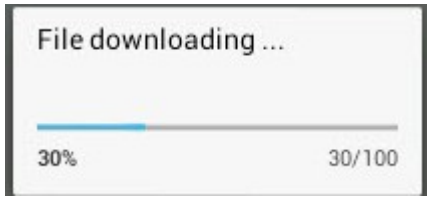




Voici les actions possibles sur l'application Android :

1. Demander à l'utilisateur s'il veut supprimer l'arborescence « Dropbox » local existante (si nécessaire).
2. Demander à l'utilisateur l'adresse IP du serveur et le port de communication. Cette information peut être enregistrée sur le téléphone dans un fichier texte ou SQLite pour éviter de le demander plus d'une fois.
3. Informer l'utilisateur avec une roue active (spin) ou barre défilante de la mise à jour avec le serveur des dossiers et fichiers locaux.





4. Informer que la mise à jour est terminée par un message.
5. Effectuer une mise à jour automatique à toutes les 5 minutes. Un nouveau message de mise à jour (roue active ou barre défilante) réapparaîtra automatiquement à chaque 5 minutes.
6. L'application ne s'arrête pas sauf si l'utilisateur ou le système d'exploitation lui-même tue l'application via Android.

4. Rapport à remettre en PDF (1 point)

Vous devez produire un rapport PDF contenant les éléments suivants :

1. Une page titre conforme aux normes du collège.
2. Une description de votre application de type client pour Android.
3. Une explication comment exécuter votre application Android.
4. Des exemples de captures d'écran pour chaque interface générée par votre application.
5. Le code source de application bien écrit et expliqué sommairement (la coloration serait de mise).

Vous remettez sur LEA un fichier ZIP contenant :

1. Une exportation ZIP de vos applications Python (assurez-vous que tous les codes sources sont présents).
2. le rapport PDF expliqué ci-haut.