TP sur les systèmes d'exploitation (Operating systems)

I. **Prise en main de l'interface :**

-avoir un courriel fonctionnel, s'en créer un si vous désirez rester anonyme.
-Se créer un compte sur https://www.pythonanywhere.com
- Aller sur le menu "Account" ajouter "djn2015" comme teacher
- De manière graphique, créer un répertoire NSI avec le menu Files
- Dans Directories, taper "NSI", clic "New directory"

- Aller dans NSI puis taper dans Files "intervalles.py", clic "New file"
Créer une fonction python par liste compréhension qui renvoie une liste qui comprend tous les éléments de a à b compris
avec un pas de n (soit n+1 éléments).

```
def intervalle(a:float,b:float,n:int)->[float]:
        return …  #compléter
if __name__=="__main__":
   c= float(input("Donner le début de l'intervalle :"))
   d= float(input("Donner la fin de l'intervalle :"))
   n= int(input("Donner le pas :"))
   print(intervalle(c,d,n))
   print("le nombre de valeurs est : ",len(intervalle(c,d,n)))
```

Retourner dans la menu "Dashboard", puis sur "Consoles" comme spécifié par l'interface, nous ne pourrons ouvrir que 2 consoles à la fois avec ce compte gratuit. Cela est suffisant car si vous avez une impossibilité d'exécution (s'est souvent le cas), alors rendez-vous sur console et fermer les consoles pour en laisser 2 au maximum.

II. **Utilisation du Shell (Bash) :**

Nous allons maintenant utiliser "Bash" dans Start a new console. Dans mon cours, je vous ai parlé de Shell, Bash est un Shell.

Rappel du cours :
Le shell est une interface système qui nous permet d'interagir avec notre OS.
Bash est le (langage de commande) shell le plus répandu et est notamment utilisé par Mac OS.
Pour envoyer nos commande, nous allons passer par une console qui se présente
sous la forme d'un écran noir.
Sous Mac, vous utiliserez l'application Terminal qui émule le comportement d'une console.

Un shell Bash connait 4 grands types de commandes :

- Les alias qui sont des surnoms qu'on va pouvoir donner à certaines commandes pour les exécuter plus rapidement ;
- Les fonctions qui sont des blocs de code portant un nom ;
- Les builtin(s) qui sont des commandes réintégrées ou prédéfinies du shell ;
- Les commandes externes qui sont des variables d'environnement.

Ex de builtins (à faire):
Pour savoir où l'on se situe sur l'arborescence du linux :
$ pwd (vous êtes sur /home/votrePseudo)

$ cd / (vous amène à la racine)
$ cd /home/votrePseudo (vous ramène sur votre dossier de travail ou cd ~)
$ cd /NSI
$ ls (liste le contenu du répertoire NSI)
$ ls -al (liste détaillée des répertoires et des fichiers, très utilsé !)

Pour afficher le contenu d'un fichier texte,
on va simplement écrire less mon_fichier.
A faire :
$ less intervalles.py (sous linux, nous utilisons souvent la touche tabulation on écrit "less in" puis tabulation, le
fichier apparaît)
la lettre "*q*" permet de sortir de l'affichage en lecture

Voici *2 commandes qui vous permettront de tout savoir sur les commandes et leurs options*.
Pour obtenir la liste complète de toutes les commandes disponibles,
nous allons utiliser la commande compgen avec une option -c comme cela :
(à faire)
>> *compgen -c*.
La commande man (manuel) va nous fournir une brève description
de l'effet d'une commande et va nous expliquer comment l'utiliser.
(à faire)
>> *man ls*
Que signifie l'option *a* :..... et l'option *l:* .....

Un certain nombre de commandes de bases sont à savoir :

- Les commandes de navigations et de visualisations déjà vus (*pwd,cd,cd .., cd /, ls -al,..*)
- La commande cp permet de copier des fichiers et des répertoires : >> *cp fichier1 fichier2*
- La commande mv permet de déplacer ou de renommer des fichiers et des répertoires :>>*mv source1 source2 répertoire*
- La commande rm permet de supprimer des fichiers et des répertoires :>> *rm fichier1 fichier2 fichier3*
- La commande mkdir permet de créer de nouveaux répertoires: >>*mkdir répertoire*

(à faire)
Créer 2 autres répertoire NSI_copie et SNT, copier intervalle.py dans ce répertoire, déplacer le contenu du répertoire NSI_copie
dans un nouveau répertoire NSI_copieBis, supprimer le contenu et le répertoire NSI_copieBis.

Un métacaractère ou joker ou encore wildcard en anglais est un caractère qui représente autre chose que lui même
pour Bash c'est-à-dire qui possède une signification spéciale qui va au delà du caractère qu'il représente.

-Le caractère * (étoile) permet de représenter soit aucun caractère,
soit un caractère quel qu'il soit, soit un ensemble de caractères : >> *cp a*.py NSI*
-Le caractère ? permet de représenter un caractère quelconque : >>*cp cours?.txt /SNT.*
-Les ensembles : >> *cp perl[0a].txt NSI*

-Les classes de caractères utilisent la syntaxe [:nom-de-classe:] et les classes les plus couramment utilisées sont :

alnum qui représente n'importe quel caractère alphanumérique : >>cp test[[:alnum:]] [[:alnum:]].txt NSI
alpha qui représente n'importe quel caractère de l'alphabet ;
lower qui représente n'importe quel lettre minuscule de l'alphabet ;
upper qui représente n'importe quel lettre majuscule de l'alphabet ;
digit qui représente n'importe quel chiffre ex : *>>cp test[[:digit:]][[:digit:]].txt NSI*
Le caractère tilde ~ permet de représenter le répertoire de base (home) d'un utilisateur :*>>cd ~*

L'extension accolade permet de créer différentes chaînes de caractères à partir d'un schéma (= modèle = motif) contenu dans une paire d'accolades.
(à faire)
>>mkdir Nsi-{lecons,devoirs, tps, tds}

La substitution de commande est une technique qui nous permet d'utiliser le résultat d'une commande comme extension. Elle va donc nous permettre d'écrire des commandes dans d'autres commandes.
(à faire et voir sur l'interface graphique de pythonanywhere le résultat)
>> cd /SNT
>> mkdir $(seq 1 7)

La sortie standard et les redirections :
On peut utiliser un chevron fermant > pour demander à la sortie standard de diriger le résultat vers un fichier
en utilisant la syntaxe suivante : *commande > mon_fichier.txt.*
(à faire)
En utilisant man, donner la signification de la commande et des options de ps -alx
>>ps -alx > /home/djn2015/NSI/psALX.txt

Pour ajouter sur un fichier existant, on utilisera le double chevron >>:
(à faire, vérifiez le résulat sur pythonanywhere)
>> pwd >> /home/djn2015/NSI/psALX.txt

Parfois, on ne veut pas faire apparaître les erreurs car elles donnent des indications alors le mieux est de les cacher dans un nouveaux fichier sortie 2:
(à faire)
>> cd \NSI
>>ls>testS.txt 2>erreur.txt

L'entrée standard et les redirections :
nous allons pouvoir rediriger l'entrée standard afin qu'elle accepte des données enregistrées dans des fichiers. Pour cela, nous allons cette fois-ci utiliser un chevron dans l'autre sens < , à utiliser comme ceci : *commande < mon_fichier.txt.*
En utilisant l'entrée et la sortie, nous pouvons y ajouter des modifications :
(à faire)
>> ls>testS.txt 2>erreur.txt

>> sort< testlS.txt >lsTriee.txt

Connecter plusieurs commandes en elles : (grep) | : touche ALTGR+6

(à faire)
>> sort | ls -al >> test1.txt

Pour sortir d'un pipe, Crtl+C

Mon premier programme Shell, dans votre répertoire courant(/home/nom):
-Créer un nouveau fichier : test1.sh
-écrire :

```
echo "Entrez votre note :"
read  note
if [ "$note" -ge 16 ]; then
   echo "très bien"
elif [ "$note" -ge 14 ]; then
   echo "bien"
elif [ "$note" -ge 12 ]; then
   echo "assez bien"
elif [ "$note" -ge 10 ]; then
   echo "moyen"
else
   echo "insuffisant"
fi
```

-Aller sur une console pour exécuter votre programme :
$ ./test1.sh

-Changer les droits d'accès du fichier test1.sh par la console de 2 manières différentes, écrire les commandes afin de rendre possible pour tous la lecture et l'exécution de test1.sh :
méthode 1 :……..
méthode 2 :……..

III. Protocole SSH (Secure SHell):

SSH est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion. Par la suite, tous les segments TCP sont authentifiés et chiffrés. Il devient donc impossible d'utiliser un sniffer (WhireShark par exemple) pour voir ce que fait l'utilisateur.

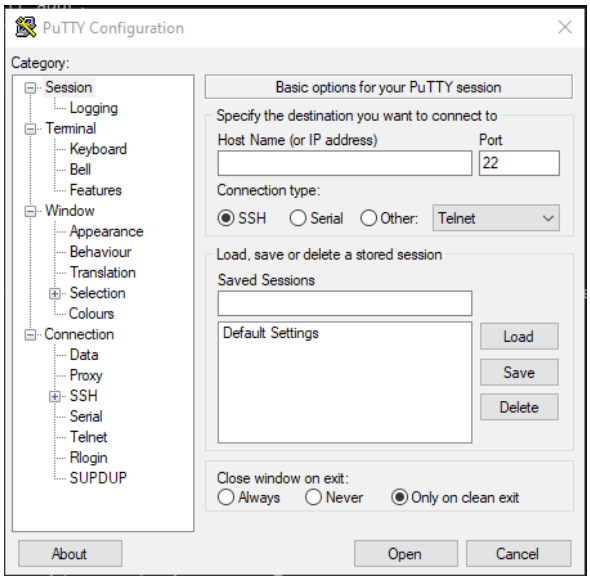
Le protocole SSH a été conçu avec l'objectif de remplacer les différents protocoles non chiffrés comme rlogin, telnet, rcp, ftp et rsh.

Par défaut, le protocole SSH utilise le port TCP 22. Il est particulièrement utilisé pour ouvrir un shell sur un ordinateur distant. Peu utilisé sur les stations Windows (on va l'utiliser avec PuTTY), SSH fait référence pour l'accès distant sur les stations Linux et Unix.

SSH peut également être utilisé pour transférer des ports TCP d'une machine vers une autre, créant ainsi un tunnel. Cette méthode est couramment utilisée afin de sécuriser une connexion qui ne l'est pas (par exemple le protocole de récupérations de courrier électronique POP3) en la faisant transférer par le biais du tunnel chiffré SSH.

Il est également possible de faire plusieurs sauts entre consoles SSH, c'est-à-dire ouvrir une console sur un serveur, puis, de là, en ouvrir une autre sur un autre serveur.

- lancer le terminal de windows (*cmd*)
- Grâce au protocole ICMP et la commande >>*ipconfig /all*
  - donner votre *@IP : …….. masque SR : ……*
  - *Nom de l'hôte :……..*
  - *@ physique (MAC) :……..*
  - *ping* sur une machine de votre réseau (vérifier la bonne connexion)
- Arp -a permet d'afficher la table ARP, c'est-à-dire la liste des @Ips ; @Macs ; type(statique ou dynamique) sur le réseau >>*arp -a*
- Si vous avez l'adresse IP du raspberryPi avec le nom DjnPI4 *ou*
  - Trouver l'@dresse Ip du RaspberryPi avec la commande *arp avec son*
  - *@mac en réseau wifi =b8-27-eb-f8-39-4f*
  - *@mac en réseau ethernet=b8-27-eb-ad-6c-1a*
- Lancer PuTTY est un client qui permet de se connecter à distance à des serveurs en utilisant les protocoles Telnet, SSH ou Rlogin. Le logiciel a été fondé par l'informaticien anglais Simon Tatham il y a plusieurs années maintenant, celui-ci se présente comme un outil libre.
- Entrer l'@Ip du RaspberryPi dans Host Name puis Open



- Dans le terminal :
  - login as: ***pi***
  - *pi@172.20.12.239's password*: ●●●●●●●● **(demandez-le au prof ;-))**
  - *pi@Djn2020:~ $* (vous êtes sur le serveur)
- Créer un script sh, en utilisant pi@Djn2020:~ $ **sudo nano exerciceTp.sh**, nano est un éditeur de texte de linux simple mais efficace (Crtl+x : quitter, Crtl+o pour enregistrer) qui permettra :
  - créer un utilisateur : votreprénom, votre mot de passe (**sudo adduser prénom**, tout le reste par défaut)
  - créer un répertoire : NSI dans le *home/votreprenom/*
  - 2 sous-répertoires : fichiers, images pour lesquels vous mettrez les droits en lecture/écriture pour tous
- Changer les droits du fichier exerciceTp.sh pour que le propriétaire et son groupe puissent exécuter ce fichier mais pas les autres.
- Lancer le script en tapant *pi@Djn2020:~ $* ***./exerciceTp.sh***
- Avec l'éditeur nano**,** écrire et enregistrer dans le dossier « fichiers » le programme intervalle.py
- Exécuter ce programme en utilisant le terminal (./python3)

- Vérifier les **users** (commande users)
  exemple : adrianu adrianu alex gabrield guillaume hugo julio lucy pi pi pi pi pi pi pi pi pi pi pi thomas et communiquer en utilisant
  $ echo "Bonjour ! " | write thomas
  $ echo "Bonjour ! " | wall (broadcast)
- Regarder les processus avec la commande : **ps -alx** (pid, ppid, NI : algo de l'ordonnanceur, lequel ?) ou la commande : **top**
- ***la commande exit permet de stopper la communication client/serveur sécurisée ssh.***