Fiche DIE Info

Généralités

Python = langage interprété (langage de programmation dont les programmes sources sont **directement exécutés par un logiciel appelé interpréteur**)
≠ **langage compilé** => doivent d'abord être traduits en instructions machines binaires avant de pouvoir être exécutés

La connexion au RPI se fait par **SSH (Secure SHell)**. On utilise une clé pour se connecter en SSH (car l'authentification par mot de passe est moins sécurisée)
L'authentification par clé fonctionne grâce à 3 composants :

- **Une clé publique** : elle sera exportée sur chaque hôte sur lequel on souhaite pouvoir se connecter ;
- **Une clé privée** : elle permet de prouver son identité aux serveurs ;
- **Une passphrase** : Permet de sécuriser la clé privée (notons la subtilité, passphrase et pas password... donc « phrase de passe » et non pas « mot de passe »).

Adresses IP

- IPv4 : notation décimale sur 32 bits / 4 octets avec 4 nombres compris entre 0 et 255, par exemple *181.174.87.53*
- IPv6 : notation hexadécimale sur 128 bits / 16 octets avec 8 groupes de 2 octets (16 bits) séparés par un signe deux-points, par exemple : *2001:0db8:0000:85a3:0000:0000:ac1f:8001*

Structure d'une page HTML

```
<!DOCTYPE html>
<html>
    <header>
        <title>Exemple de page 1</title>
    </header>
    <body>
        <h1> Ceci est le titre de la page</h1>

        Ceci est le corps de la page
    </body>
</html>
```

Commandes *shell*

- `pwd` : print working directory
- `more` : affiche le contenu d'un fichier
- `ping` : interroge une machine connectée à l'Internet afin de savoir si celle-ci est accessible
- `script` : permet d'enregistrer dans un fichier (`typescript` par défaut) tous les textes apparaissant dans le terminal. Enregistrement par `Ctrl + D`

Connexion au RPI

Nom de la machine (RPI) : `pilic[Y].irisa.fr` où `[Y]` est le numéro du RPI.
Nom d'utilisateur sur la machine : `pilicX`

Connexion en SSH `ssh pilic129@pilic19.irisa.fr -i cle_rsa`
Déconnexion `exit`

Copier des fichiers sur RPI : `scp -i cle_rsa fichier pilic129@pilic19.irisa.fr:.`
ne pas oublier le `:.` qui indique le répertoire personnel

`scp` = secure copy => chiffre les données

Copier des fichiers de RPI vers session (depuis un terminal sur la session jupyter) :
`scp -i cle_rsa_die129 pilic129@pilic19.irisa.fr:/home/pilic129/2022/dataBAM.csv .`

API (production des pages web)

Une API (Application Programming Interface) est une interface de programmation qui offre un accès à des données ou à des méthodes de traitement.

```
#Initialisation et importation des fonctions nécessaires
import sasdie
#Quelques initialisations de variables
sasdie.init()
#Création de la structure qui fournit les méthodes
c = sasdie.Sasdie()

#Connexion au serveur « prototypel1.irisa.fr »
c.setLogin(email)
c.setPasswd(numéroetudiant)
if not c.connect():
    print("La connexion a échouée")
    exit(1) # arrêt du programme

#Récupération de la clé de votre compte sur «prototypel1.irisa.fr»
c.macle()

#Publier un contenu sur le serveur # textepage est une chaine de caractères
c.publierpage_html(textepage)
```

Données de pollution

Particules Fines (PM) : Taille qui varie de 0.1 à 10 micromètres de diamètre. PM10 = diamètre < 10 µm etc.

Capteur fixe : produit des données de pollution de manière asynchrone

Résumé des sprints

Sprint 1

Récupérer un fichier et le publier sur le serveur de rendu

Sprint 2

Modifier les données d'un notebook :

```
##### Utilisation de l'API, cf plus haut #####

longitude0=-1.67883 ## ligne modifiée
latitude0=48.11563  ## ligne modifiée
if not c.publierCoordonneesGPS("longitude0","latitude0"):
    print("Erreur: position non postée !")
```

Sprint 3

Utilisation du shell. Production d'un fichier `typescript`

Sprint 4

Connexion au RPI par SSH, copier des fichiers entre le JupyterLab et le RPI

Sprint 5

Editer des fichiers avec emacs/nano

Sprint 6

Ecrire les programmes suivants :

```
print("hello world!")
print("exercice de"+"concaténation"+"de chaînes")
```

```
print(123456+876543)
for i in range(3,26):
    if i%2==0:
        print(i,"pair")
    else:
        print(i,"impair")

# initialisation liste
T=["livre","tableau","craie","ordinateur","stylo"]
chaine="extension de chaîne"
#boucle
for i in range(len(T)):
    T[i]=T[i]+chaine
#boucle d'affichage des résultats
for i in T:
    print(i)
##### Utilisation de l'API, cf plus haut #####
```

Sprint 7

Récupération des données de `dataBAM.csv` : (Time, PM10 Bam Laennec, PM2.5 Bam Laennec) => Production d'un fichier CSV contenant les 10 dernières valeurs de capteur produites par le capteur fixe

```
#Importe les méthodes nécessaires
import RPI_Sasdie_Lib

# Créer la structure de données (classe) pour l'accès aux méthodes
d = RPI_Sasdie_Lib.DataStream()

#Lit la dernière valeur produite par le capteur.
#Attention à chaque fois que vous appelez la méthode, elle ne produit pas nécessairement une nouvelle valeur. La valeur retournée (data) comporte deux champs : data.count le numéro de l'échantillon produit et data.pm25 la mesure des PM25.
contenuFichier="Numéro d'échantillon;PM25\n"
l_count=[]
while len(l_count) <10: #tant qu'on a pas 10 valeurs
    data=d.lectureDonnéesCourante() # on lit la dernière valeur
    if data.count not in l_count: # si le numéro n'était pas déjà présent dans la liste
        l_count.append(data.count) # on incrémente le compteur
        contenuFichier+=str(len(l_count))+";"+str(data.pm25)+"\n" # on ajoute au fichier

d.ecritureDuFichierCSV(contenuFichier)
```

Fichier `csv` produit :

```
Numéro d'échantillon;PM25
1;-1
2;5.44
3;4.58
...
```

Sprint 8

écrire un code Python (fichier genTabDataHTML.py) qui produit la page HTML avec un tableau contenant les données du capteur du fichier produit sur le RPI (data_pm25.csv).

```
##### Utilisation de l'API, cf plus haut #####

#lecture du fichier data_pm25.csv
#La variable donnees peut typiquement être parcourue par une boucle « for row in donnees: ». L'expression row[0] correspond à l'élément en première colonne et row[1] de celui en deuxième colonne)
donnees = c.lireDonneesPollutionRPI()

texthtml="<!DOCTYPE html>\n<html lang=\"fr\">\n  <head>\n    <meta charset=\"utf-8\">\n
<title>Données</title>\n  </head>\n  <body>\n    <h1>Données RPI</h1>\n    <table border=\"1\">\n      <tr>
<th>Numéro d'échantillon</th><th>PM25</th></tr>\n"
```

```
for row in donnees[1:]:
    texthtml+="      <tr><td>"+row[0]+"</td><td>"+row[1]+"</td></tr>\n"

texthtml+="    </table>\n  </body>\n</html>"
#Publication du contenu HTML, "textHTML" est une chaine de caractères correspondant au source de la page.

c.serveur.publierpage_html(texthtml)
```

Sprint 9

Lecture du fichier CSV `data-pm-sasdie-bus.csv` qui contient :

```
* Les coordonnées GPS, capturées par la station de mesure (mobile), du Nord-Ouest au Sud-Est, longitude puis latitude
* La période de mesure en millisecondes et au format POSIX UTC
* La valeur mesurée
* Le nom de la station mobile et du capteur (n°série de l'OPC-N3)
* La période de mesure en mode texte.

* Exemple de ligne :
-1.684122709;48.129878342;-1.683787022;48.129061521;1604994820298;1604994820298;4.289999961853027;"Bus700";"OPC_N3:20";"2020-11-10T07:53:40.298Z";"2020-11-10T07:53:40.298Z";
```

Produire une carte qui indique l'ensemble des points de mesure du fichier CSV, la moyenne des valeurs de pollution pour les PM25 qui sera utilisée comme étiquette d'un marqueur de 500 mètres de diamètre placé à la Mairie de Rennes & ajouter les positions des bars et restaurants déclarés précédemment.

```
##### Utilisation de l'API, cf plus haut #####

#lecture de data-pm-sasdie-bus.csv
csvdp = c.lireDonneesPollution("data-pm-sasdie-bus.csv")
c.creerCarte()

for row in csvdp:
    long=(float(row[0])+float(row[2]))/2 #moyenne arithmétique des longitudes
    lat=(float(row[1])+float(row[3]))/2 #moyenne arithmétique des latitudes
    etiq=row[6]
    c.macarte.ajoutCercleSurLaCarte(str(long),str(lat),500,etiq)
    #long et lat sont des chaines des caratères

#lecture des emplacements déclarés avec la fonction publierCoordonneesGPS.
csvcs = c.lireCoordonneesGPS()
for row in csvcs:
    long1=float(row[0])
    lat1=float(row[1])
    c.macarte.ajoutMarqueurSurLaCarte(str(long1),str(lat1),"bar/restaurant")

#produit le texte HTML (et javascript) de la carte
cartehtml = c.macarte.produireHTMLCarte()

#Publication de la page sur le serveur prototypel1.irisa.fr
c.serveur.publierpage_html(cartehtml)
```