**Rapport Td 2 Blockchain Programming : Wallet Bitcoin :**

**Structure du code :**

Le programme s’exécute depuis la console en le lançant.   
Un menu console permet de naviguer et de choisir ce que l’utilisateur peut faire.

padd\_banary():

La fonction padd\_binary permet de compléter avec des 0 une string. Ceci est utile pour compléter des bits ayant une taille inférieur à 11.

byte\_to\_binary():

Permet de convertir des bytes en bits.

create\_entropy():

Fonction permettant de créer l’entropie de 128bits. Nous utilisons la librairie secrets pour ce faire.  
La fonction retourne l’entropie de 128bits et le checksum par SHA256 de cette entropie.

create\_binary\_seed():

Fonction qui retourne la liste des 12 mots mnémoniques.  
Elle sépare les 132 en 12 valeurs de 11 bits chacun.  
En faisant correspondre la valeur entière de chaque valeur avec la ligne des mots contenus dans le fichier « words.txt », elle retourne la liste mnémonique.

Import mnemonic seed():

Fonction retrouvant les 128 bits d’entropies et les 4 bits de checksum d’une seed.  
La fonction va demander à l’utilisateur de saisir les mots mnémoniques 1 par 1, puis va ensuite boucler pour chercher les valeurs entières associées à chaque mots.   
Si le mot n’est pas dans le dictionnaire elle retourne un message d’erreur.

Elle renvoie finalement les 128 bits d’entropies, en ayant padd les valeurs n’ayant pas une valeur de 11 bits, et les 4 bits de checksum.

get\_MprivK\_and\_CC():

Permet de retourner la master private key et le chain code.

Utilise SHA 512 pour hacher la mnémonique, et retourne la première moitié comme master private key et la seconde comme chain code.   
  
get\_MpubK():

Extrait la master public key. On utilise ecdsa curve elliptique.

get\_child\_key():

Permet de retourner une clé enfant à l’index n.   
Prend en paramètre l’index.

En hachant (SHA 512) la master public key + le chain code + l’index, retourne de la manière qu’avec la master private key, la private key et le chain code.