Python Exercices

Exercice 1

Afficher sur la sortie standard 'hello'

Exercice 2

Ecrire un script qui définit une fonction `display_hello` qui ne prend pas de paramètre et qui affiche le message 'hello'.

Créer une fonction via la syntaxe suivante

```
def yourfunction():
   code in your function
```

Exercice 3

Ecrire un script qui tente d'ouvrir un fichier, utiliser la fonction `open()`. Si le fichier n'existe pas, attrapper l'exception et afficher le message: 'this file does not exist'.

Pour ouvrir un fichier, rappel :

```
f = open(path_to_file, 'r')
```

Pour lire le contenu d'un fichier

```
f.read()
```

Exercice 4

Déclarer une liste d'entiers et afficher tous les éléments de la liste sur la sortie standard. Déclarer une liste initialisée :

```
intArray = [1, 10, 9, 101]
```

Parcourir une liste :

```
for item in yourArray:
  # do some thing
```

Exercice 5

Plutôt que de lire l'intégralité du fichier avec `f.read()`, vous pouvez lire le fichier ligne à ligne

```
for line in f:
  print(line)
```

Lire le fichier user.txt Lire le fichier pass.txt Afficher sur la sortie standard la combinaison de tous les éléments de user.txt avec tous les éléments de pass.txt

user.txt :

```
mysql
info
postgres
```

pass.txt :

```
12345
qwerty
webadmin
webmaster
```

Sortie attendue :

```
mysql,12345
mysql,qwerty
mysql,webadmin
mysql,webmaster
info,12345
info,qwerty
info,webadmin
info,webmaster
postgres,12345
postgres,qwerty
postgres,webadmin
postgres,webmaster
```

Exercice 6

Faire appel à une API externe

```
import requests
```

Pour utiliser requests, il faut ensuite utiliser l'un des 2 verbes (GET ou POST) avec le point d'accès fourni par l'API

```
requests.get('https://data.education.gouv.fr/api/v2/catalog/datasets?limit=10&offset=0&timez
```

URL à remplacer dans GET

La réponse d'une requête peut être directement transformée en JSON si le format de la réponse est au format JSON

```
response.json()
```

Exercice 7

Pour aller plus loin sur les structures de données disponibles dans Python, nous avons les listes et les dictionnaires.

Une liste (List), qui est un ensemble ordonné d'éléments peut se manipuler de la façon suivante

```
cves = ['cve1', 'cve2', 'cve3']
for cve in cves:
  print(cve)
```

Un dictionnaire (Dictionary) associe à chaque élément une clé

```
cve = {'name': 'cve1', 'id': 1, 'severity': 'high'}
for prop in cve:
  value = cve[prop]
  print("Prop %s[%s]" % (prop, value))
```

Un ensemble (Set) qui une collection d'éléments distincts désordonné peut se manipuler de la façon suivante

```
cveSet = {'cve1', 'cve2', 'cve3'}
```

Rappelez vous que lors d'une itération sur l'ensemble des éléments, vous ne pouvez pas partir du postulat que les éléments sont toujours dans le même ordre.

Le Tuple est une collection non modifiable d'éléments qui peut servir de clé dans un dictionnaire à condition que les éléments d'un tuple soient également non modifiables.

Exécuter le code suivant

```
d = {(x, x + 1): x for x in range(10)}
print(d)
```

Pour d, donnez les informations suivantes - Type de d parmi les structures - Type de la clé - Type de la valeur - Que renvoie `d[(1,2])` ? - Que renvoie `d[[1,2]]` ? Pourquoi ?