Projet : Localiser une photographie sur une carte

Modalités :

- Par groupe de 3 élèves.
- Rendre un script Python répondant à ce qui est demandé
- Mise à disposition d'un script intermédiaire pour les plus en difficulté
- Grille d' évaluation :

1ère NSI **GRILLE D'EVALUATION :** **Noms :** **Note :** /15	Rien d'écrit	Ne fonctionne pas	Fonctionne imparfaitement	Fonctionne avec beaucoup d'aide	Fonctionne avec aide	Fonctionne sans aide	Fonctionne de manière optimale et avec les commentaires
Architecture générale du script							
Fonction dico_exif							
Fonction deg_vers_decimale							
Fonction html_geolocalisation							
Fonction images_geolocalisation()							

1. Préambule

1.1 Vous êtes agents dans une agence de cyber sécurité

Un équipe de la police scientifique dispose d'un disque dur contenant une grande quantité d'images dans un dossier.

Les investigateurs nous demandent d'analyser ces images.

Ont-elles été prises par les mêmes appareils numériques ?

Peut-on retrouver la date ou le lieu où ces photos ont été prises ?

Afin de faire des recoupements, est-il possible d'afficher sur une carte , par des puces de couleurs différentes en fonction des appareils, la géolocalisation de chaque photographie .

1.2 Objectifs

Les résultats de vos investigations devront être remis en main propre au format numérique :

- Un script python comprenant les 4 fonctions :
  - dico_exif(une_image) : Extraction du dictionnaire des informations EXIF
  - deg_vers_decimale(lat, long) : Conversion des latitude/longitude en degré, minute, seconde vers des latitude/longitude en décimal
  - html_geolocalisation(une_image) : Qui prend une image et qui génère un fichier html avec la géolocalisation.
  - images_geolocalisation(un_repertoire) : Qui prend un répertoire et qui génère un fichier html avec la géolocalisation

2. Une image et ses métadonnées :

2.1 Découverte

Un fichier «image» issu d'un téléphone contient plus que l'image. On trouve en effet des informations sur l'image elle-même (définition, résolution...) mais aussi des informations sur la prise de vue (date et heure, lieu...). Ces informations sont enregistrées en binaire dans les premiers bits de l'image.

Ces informations associées à chaque prise sur un appareil photographique numérique s'appelle les données EXIF (EXchangeable Image file Format).

A faire vous même 1. Repérer les données EXIF en hexadécimal

- Téléchargez l'image suivante :

http://ninoo.fr/LC21_22/NSI_1ere/seq5_donnees_en_tableau/dossier_avec_images/image_2.jpg

- Ouvrez-là avec le site : https://hexed.it/

La colonne centrale correspond à l'image au format hexadécimal.

En observant les colonne de droite, repérer et noter la marque et le modèle du smartphone.

…….

A faire vous même 2. Repérer les données EXIF

- Sur un site dédié aux données EXIF, reprenez l'image et ouvrez-là avec le site

http://exif.regex.info/exif.cgi

- Repérez et notez les coordonnées où la photo a été prise ainsi que la date de la prise de vue .

…….

2.2 La librairie piexif :

La librairie piexif permet de récupérer les données EXIF d'une image dans un dictionnaire.

A faire vous même 3.

```
# importation de la librairie
import piexif
# Chargement du dictionnaire EXIF
exif_dict = piexif.load("image_2.jpg")
# Affichage du dictionnaire
print(exif_dict)
```

La librairie piexif est aussi utile pour supprimer ou modifier des informations EXIF d'une image.
Pour en savoir plus : https://piexif.readthedocs.io/en/latest/functions.html

A faire vous même 4. Aide python

Pour lister les éléments d' un dictionnaire, il existe des méthodes. Testez dans la console :

```
>>> mon_dico = {'un' :'one', 'deux' :'two', 'trois' :'three'}
>>> mon_dico.keys() #Création liste avec toutes les clés
['un', 'deux', 'trois']
>>> mon_dico.values() #Création liste avec toutes les valeurs
['one', 'two', 'three']
>>> mon_dico.items() #Création liste avec des tuples
[('un' :'one'), ('deux' :'two'), ('trois' :'three')]
```

A faire vous même 5. Extraction du dict. EXIF à l'aide de Python

- Concevez un programme qui extrait les information du fichier EXIF en liste de dictionnaire et affiche celui-ci.
- Affichez les clés du dictionnaire.
- Affichez les valeurs de la première clé. Que constatez-vous ?
- Exécutez une ligne de commande qui affiche le fabricant et le modèle du téléphone.
- Retrouvez le nom de l'appareil photographique et la marque de celui-ci dans le dictionnaire en lisant celui-ci directement à l'écran . (Vous pourrez-vous aidez du document suivant : https://www.exiv2.org/tags.html)
- Avec les informations recueillies sur le site web http://exif.regex.info, observez et retrouvez les coordonnées GPS de la photographie.

A faire vous même 6. PROJET

- Écrivez une fonction dico_exif qui prend une image en paramètre et qui retourne un dictionnaire avec les coordonnées GPS, le nom de l' appareil photo, ...

3. Conversion coordonnées GPS

3.1 Rappel : Coordonnées GPS

Latitude: 0 (Equateur), c'est jusqu'à 90° au Nord et au Sud jusqu'à -90° (+ pour le Nord, – pour le Sud)

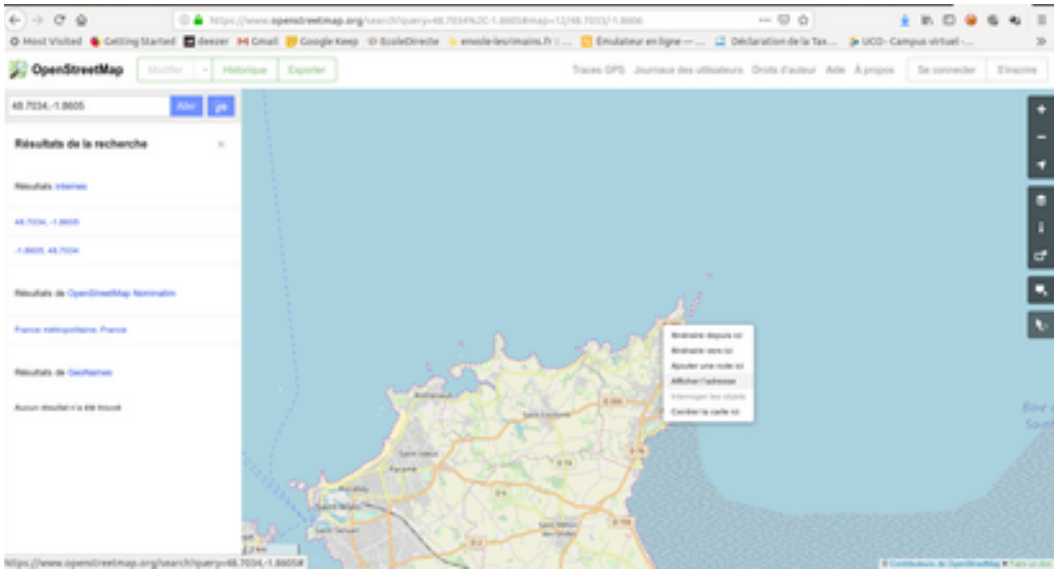
Longitude : 0 (Greenwich), c'est jusqu'à 180°à l'est et à l'ouest jusqu'à -180° (+ pour l'Est, – pour l'Ouest)

Coordonnées GPS en degrés décimaux de quelques villes :

Nom de la capitale	Latitude	Longitude
Moscou	55.75	37.62
New-York	40.78	-73.97
Paris	48.85	2.35
Rio de Janeiro	-22.91	- 43.21
Shanghai	31.23	121.49
Sydney	-33.87	151.21

Repérer des coordonnées GPS sur une carte openstreatmap :

https://www.openstreetmap.org/



3.2 Conversion coordonnées GPS

Les coordonnées GPS données par le format EXIF sont en heures, minutes, secondes, centièmes de seconde et un caractère 'N'/'S'.

Nous allons avoir besoin des coordonnées au format décimal.

A faire vous même 7. Aide python, type bytes

Une chaîne de caractère de type byte se caractérise par une écriture de la forme

```
chaine_binaire= b'ma chaine binaire'
type(chaine_binaire)  # <class 'bytes'>
```

Pour convertir cette chaîne en caractère utf-8 :

```
chaine=chaine_binaire.decode('utf-8')
type(chaine)          #<class 'str'>
```

plus d'information : http://sametmax.com/en-python-3-le-type-bytes-est-un-array-dentiers/

A faire vous même 8. Aide python

Tronquer une chaîne de caractères (fonctionne comme une liste) :

```
>>>a='ma chaine'
>>>a[0]
'm'
>>>a[3:5]
'ch'
>>>a[3:]
'chaine'
>>>a[:-2]
'ma chai'
```

A faire vous même 9. Aide python

Retirer les premiers ou les derniers caractères :

```
>>>a='mmmma chaineeeee'
>>>a.rstrip('e')
'mmmma chain'
>>>a.lstrip('m')
'a chaineeeee'
```

A faire vous même 10. PROJET

- Écrivez une fonction deg_vers_decimale qui transforme les coordonnées GPS format EXIF en coordonnées GPS décimales (pouvant être utilisées par OpenStreetMap)

4. Positionnement sur une carte

4.1 Un peu d'histoire ...

Dans les années 60, le premier SIG apparaît. Un SIG (système d'information géographique) permet de mémoriser des informations géolocalisées sur un ordinateur.

Entre les années 80 et 2000, le système GPS se développe passant de l'usage uniquement militaire à celui du grand public.

En juillet 2004, Steve Coast lance le projet OSM (OpenStreetMap), permettant par exemple de créer des cartes sous licence libre, à partir de données GPS.

En 1979, le premier tableur pour ordinateur voit le jour. Visicalc permet de gérer automatiquement des données sous forme de tableau.

4.2 Rappel seconde SNT : Localisation

En langage Python, il est possible d'utiliser une bibliothèque nommée folium. Elle sert à fabriquer des cartes au format HTML, en utilisant des fonds de carte OpenStreetMap.

A faire vous même 11.

```
1  import folium
2  carte = folium.Map(location=[-33.87,151.21],zoom_start=11)
3  folium.Marker([-33.851,151.212],popup="Harbor Bridge").add_to(carte)
4  carte.save('ville.html')
```

A l'aide du programme python, répondre aux questions suivantes :

1. Dans quelle ville du monde, la carte est-elle calibrée ? (Voir Rappel ci-dessus)
2. Convertir les coordonnées angulaires de cette ville de degrés décimaux à degrés minutes secondes.
3. Lors de l'exécution du programme, une puce apparaît. Qu' observe-t-on si l'on clique dessus ?

4.3 La librairie Folium :

A faire vous même 12.

Testez et mettez au point le script suivant :

```
# Création d'une carte open streat map
import folium

# Initialiser la carte en un lieu donné
carte= folium.Map(location=[47.078025, -2.409053],zoom_start=1)
# Marquer une punaise
folium.Marker([lat,long],popup="L'étiquette").add_to(c)
# Liste de positions GPS
Liste_positions=[(lat1,long1),(lat1,long2),(lat3,long3)]

# Création d'une ligne de positions
folium.PolyLine(Liste_positions).add_to(c)
# Enregistrement de la carte dans le répertoire courant au format html
c.save('./ma_page.html')

#Complément : choisir la couleur
colors= {'red', 'blue', 'green', 'purple', 'orange', 'darkred',
'lightred', 'beige', 'darkblue', 'darkgreen', 'cadetblue', 'darkpurple',
'white', 'pink', 'lightblue', 'lightgreen', 'gray', 'black', 'lightgray'}
# Marquer une punaise
folium.Marker([lat,long],popup="L'étiquette", color ='red').add_to(c)
```

Pour en savoir plus : https://python-visualization.github.io/folium/

A faire vous même 13.

- Créez une fonction epuration_EXIF qui prend en entrée un dictionnaire EXIF brut et en retourne un dictionnaire épuré. Par exemple :

```
epuration_EXIF(Dico)
{'marque': 'Sony', 'modele': 'G3312', 'date': '2019:04:28
21:09:07', 'lat': 48.6736, 'long': -1.859079724}
```

A faire vous même 14. PROJET ☐

- Créez une fonction html_geolocalisation qui prend une image et qui génère un fichier html contenant la carte Openstreetmap avec une punaise à l' endroit où la photo à été prise.

A faire vous même 15. PROJET ☐

- Faites l' apprentissage du module python os.path
- Créez une fonction images_geolocalisation qui prend un répertoire et qui génère un fichier html contenant la carte Openstreetmap avec des punaises pour les lieux où les photos ont été prises.
- Vous l' appliquerez à l' ensemble des photos disponibles là :
  http://ninoo.fr/LC21_22/NSI_1ere/seq5_donnees_en_tableau/dossier_avec_images/
- Générez la page html avec toutes les photos géolocaliées.