|  |  |  |  |
| --- | --- | --- | --- |
| **ACADÉMIE de VERSAILLES** | *Sciences Numérique et Technologie* | |  |
|  | [*chdebernardi.fr*](http://chdebernardi.fr/) | *Seconde* | *4H30* |
| TD /TP *Autour de la gestion des données EXIF* | | Version 1.02 |
| *Lycée RICHELIEU – 64 rue George SAND – 92500 RUEIL MALMAISON* | | | |

*L’objectif de cette séquence est d’exploiter les informations « cachées » dans une image prise à l’aide d’un téléphone portable récent. Elle se compose principalement de petites manipulations et donc d’une suite de petits scripts en langage Python.*

## Activité 1 –Acquisition d’une image

1. A l’aide de votre téléphone portable, photographier un objet de la salle de TP ou choisir une image dans votre librairie d’image. Faire valider la photo par votre enseignant et insérer la dans le cadre ci-dessous. Préciser son nom.

On obtient :

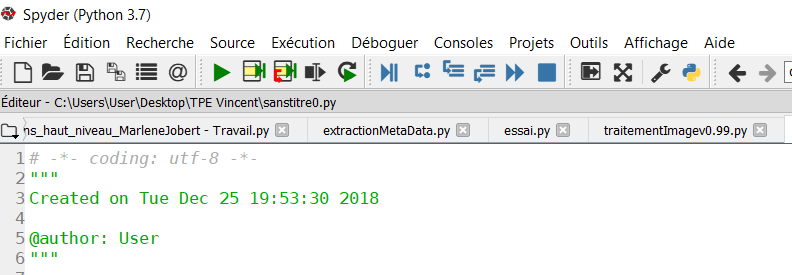
Insertion de l’image

* Sauvegarder l’image dans un répertoire portant votre nom.

## Activité 2 – Affichage de l’image et récupération des informations globales de l’image sous Python

### Affichage de l’image

* Lancer l’interface de programmation intégrée SPYDER (à partir d’ANACONDA) ou un autre ensemble de programmation. Attention, le lancement peut être très long (plusieurs minutes)



* Taper le programme ci-dessous :

from PIL import Image as im

nomFichierImage = "ouEstCe4.jpg" « Mettre le nom de votre image entre guillemets »

imageExif=im.open(nomFichierImage) # Ouverture du fichier

imageExif.show() # Affichage du fichier

print ('--------------------')

print(imageExif.size) # Attribut de taille : (largeur, hauteur)

* Sauvegarder le programme dans le **même** répertoire que l’image de travail sous le nom extractionMetaData.py.
* Lancer le programme en cliquant sur la flèche verte.

1. Que fait ce programme ? :
2. Que signifie from PIL import Image as im ?
3. Que signifie imageExif=im.open("nomFichierImage") ?
4. Que signifie imageExif.show() ?
5. Que signifie print(imageExif.size)  ?

### Activités en autonomie

1. Sur le même principe que la dernière ligne de code du programme précédent, faire afficher les attributs de format et de mode.
2. L’image choisie est-elle de type JPG ou TIFF ? Quel attribut permet-il de le savoir ?
3. L’image choisie est-elle en couleur ? Quel attribut permet-il de le savoir ?

* Enregistrer votre programme sous le même nom que précédemment.

## Activité 3 – Affichage brut des données EXIF

* A la suite de votre programme précédent, rajouter les lignes de code ci-dessous :

print ('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(imageExif.info) # Attribut d'information

print ('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

* Sauvegarder le programme dans le **même** répertoire que l’image de travail sous le nom extractionMetaData.py.
* Lancer le programme

1. Que fait cette nouvelle ligne de code ?
2. Les données affichées sont-elles directement utilisables ?
3. Que peut on en conclure si on veut utiliser ces données.

### Fin de la séance 1

## Activité 4 – Récupération et Exploitation des données EXIF

### Affichage des metadonnées EXIF de l’image

* A la suite de votre programme précédent, rajouter les lignes de code ci-dessous :

# Affichage de toutes les données contenues dans les EXIF

import piexif as piex # ATTENTION JPEG ou TIFF exclusivement

exifDict = piex.load(nomFichierImage) # Chargement du fichier image

for ifd in ("0th", "Exif", "GPS", "1st"): # Attention il s'agit d'un zéro

for tag in exifDict[ifd]:

print(piex.TAGS[ifd][tag]["name"], exifDict[ifd][tag])

print ('////////////////////')

* Sauvegarder le programme dans le **même** répertoire que l’image de travail sous le nom extractionMetaData.py.
* Lancer le programme

### Affichage des metadatas

1. Que signifie import piexif as piex ? :
2. Que signifie exifDict = piex.load(nomFichierImage) ?

Note : Les lignes de code ci-dessous sont plus difficiles à appréhender. On peut se limiter à une approche un peu superficielle.

for ifd in ("0th", "Exif", "GPS", "1st"):

for tag in exifDict[ifd]:

print(piex.TAGS[ifd][tag]["name"], exifDict[ifd][tag])

1. Combien y a-t-il de boucles imbriquées ?
2. La première boucle se fait-elle sur des valeurs numériques ?
3. Que réalise print(piex.TAGS[ifd][tag]["name"], exifDict[ifd][tag])  ?
4. Les données GPS sont-elles présentes dans ces informations ?
5. Compléter les données GPS dans le cadre ci-dessous ?

On obtient :

GPSLatitudeRef …………………

GPSLatitude …………………

GPSLongitudeRef …………………

GPSLongitude …………………

GPSAltitudeRef …………………

GPSAltitude …………………

GPSTimeStamp …………………

GPSDateStamp …………………24'

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Activité 5 – Récupération et Exploitation des données GPS

### Extraction Trame GPS complète

Les données GPS sont enregistrées dans des champs différents.

On peut noter la présence des lettres N ou S et W ou E dans les informations de géolocalisation.

* N : signifie North ; S signifie South ; E signifie East et W signifie West.
* A la suite de votre programme précédent, rajouter les lignes de code ci-dessous :

print ('Extraction des coordonnées GPS')

print (exifDict["GPS"]) # Récupération totale de l'information GPS

print (exifDict["GPS"][1]) # Récupération b'N' ou b'S'

print (exifDict["GPS"][2]) # Récupération valeur totale N ou S

print (exifDict["GPS"][3]) # Récupération b'E' ou b'W'

print (exifDict["GPS"][4]) # Récupération valeur totale E ou W

* Sauvegarder le programme dans le **même** répertoire que l’image de travail sous le nom extractionMetaData.py.
* Lancer le programme

1. Dans le cadre ci-dessous, recopier les données affichées.

On obtient :

Les données de localisation sont enregistrées sous la forme degré, minute, seconde. Elles sont associées avec un diviseur. Le couple de données se présente donc sous la forme (valeur, diviseur). La valeur du diviseur est surtout utilisée dans les secondes car elle peut être différente de 1.

1. Dans le cadre ci-dessous, préciser quelle est la valeur du diviseur des secondes.

On obtient :

### Extraction Données GPS complètes

* A la suite de votre programme précédent, rajouter les lignes de code ci-dessous :

print (exifDict["GPS"][2][0][0]) # Récupération valeur degré N ou S (d)

print (exifDict["GPS"][2][0][1]) # Récupération diviseur degré N ou S (d)

print (exifDict["GPS"][2][1][0]) # Récupération valeur minute N ou S (m)

print (exifDict["GPS"][2][1][1]) # Récupération diviseur minute N ou S (m)

print (exifDict["GPS"][2][2][0]) # Récupération valeur seconde N ou S (s)

print (exifDict["GPS"][2][2][1]) # Récupération diviseur seconde N ou S (s)

* Sauvegarder le programme dans le **même** répertoire que l’image de travail sous le nom extractionMetaData.py.
* Lancer le programme

1. Que réalise la ligne de code print (exifDict["GPS"][2][0][0]) ? :
2. Que réalise la ligne de code print (exifDict["GPS"][2][0][1]) ? :
3. Sur le même modèle que les lignes de code ci-dessus, rajouter les lignes de code pour extraire les informations relatives à la position E ou W.

### Fin de la séance 2

## Activité 6 – Conversion de format

Le format des données de géolocalisation enregistrées dans les metadata est degrés, minutes, secondes associées avec leur diviseurs respectifs. La plupart des sites WEB ou des fonctions utilisées par la suite exigent des données au format décimal. Il faut donc procéder à une étape de conversion pour passer du format dms au format dd.

### De dms vers dd

* A la suite de votre programme précédent, rajouter les lignes de code ci-dessous :

latitudeAbs=exifDict["GPS"][2][0][0]/(exifDict["GPS"][2][0][1])+\

exifDict["GPS"][2][1][0]/(60\*exifDict["GPS"][2][1][1])+\

exifDict["GPS"][2][2][0]/(3600\*exifDict["GPS"][2][2][1])

* Sauvegarder le programme dans le **même** répertoire que l’image de travail sous le nom extractionMetaData.py.
* Lancer le programme

1. Que signifie le caractère \ en fin de ligne ? :
2. Sur le même modèle que les lignes de code ci-dessus, proposer et rajouter les lignes de code permettant d’extraire les informations relatives à la longitudeAbs.
3. Proposer les lignes de code permettant de faire afficher la latitudeAbs et la longitudeAbs.

Les données liées à la latitude et à la longitude sont enregistrées uniquement à l’aide de valeurs **positives**. Les coordonnées doivent cependant être placées en positif ou en négatif suivant les informations N/S et E/W.

Les valeurs de latitude et longitude sont positives, respectivement pour N et pour E. Elles sont négatives dans les autres cas.

1. En autonomie (ou avec de la guidance), proposer les lignes de code basées sur des structures de sélection permettant de passer de latitudeAbs à latitude et de longitudeAbs à longitude suivant les valeurs contenues dans exifDict["GPS"][1] et exifDict["GPS"][3]. Faire afficher les valeurs de latitude et longitude.

On obtient (guidance partielle => rajouter uniquement la gestion de E ou W)

# Prise en compte de N ou S et E ou W

if (exifDict["GPS"][1]==b'N'):

latitude=latitudeAbs

else:

latitude=-latitudeAbs

Note : Il est possible de vérifier le calcul réalisé à partir d’un site web de conversion de dms vers dd comme par exemple <http://xjubier.free.fr/site_pages/DMS2DD_Converter.html>

## Activité 7 – Géolocalisation et affichage du nom et de carte de la prise de vue

### Détermination du nom du lieu de la place de vue

* A la suite de votre programme précédent, rajouter les lignes de code ci-dessous :

from geopy.geocoders import Nominatim

geolocator = Nominatim(user\_agent="XX") # Il faut placer une chaine de

# caractères quelconque

##### ATTENTION : Il faut configurer le proxy (passerelle) au lycée

strLat=str(latitude) # Conversion en chaîne de caractères

strLong=str(longitude) # Conversion en chaîne de caractères

strLatLong=strLat + ', ' + strLong # Concaténation des deux chaînes

location = geolocator.reverse(strLatLong) # Cette fonction attend une

# chaine de caractères

print(location.address)

* Sauvegarder le programme dans le **même** répertoire que l’image de travail sous le nom extractionMetaData.py.
* Lancer le programme

1. Que réalise la ligne de code strLat=str(latitude) ? :
2. Que réalise la ligne de code strLatLong=strLat + ', ' + strLong ?
3. Que réalise la ligne de code location = geolocator.reverse(strLatLong) ?
4. Dans le cadre ci-dessous, recopier les données affichées et en particulier le lieu de prise de la photo.

On obtient :

Note : Si vous avez de l’avance par rapport à vos camarades, vous pouvez essayer de connaitre le lieu de la prise de vues des photos de vos camarades.

### Obtention de la carte du lieu de la prise de vue

* A la suite de votre programme précédent, rajouter les lignes de code ci-dessous :

# Création de la carte avec un niveau de Zoom de 17

import smopy

sMap = smopy.Map((latitude, longitude), z=17) # (lat\_min, lon\_min, lat\_max, lon\_max).

x, y = sMap.to\_pixels(latitude, longitude)

ax = sMap.show\_mpl(figsize=(8, 6))

ax.plot(x, y, 'or', ms=10, mew=2);

* Sauvegarder le programme dans le **même** répertoire que l’image de travail sous le nom extractionMetaData.py.
* Lancer le programme

1. Que réalise l’ensemble de ces lignes de code ? :
2. Dans le cadre ci-dessous, recopier la carte obtenue à partir des coordonnées GPS.

On obtient :

1. En autonomie, modifier le niveau de zoom en le passant par exemple à la valeur 6 pour obtenir une vue plus élargie et dans le cadre ci-dessous, recopier la carte obtenue à partir des coordonnées GPS. Vous pourrez au choix modifier le niveau de zoom existant ou créer une seconde vue avec le nouveau niveau de Zoom.

On obtient :

Note : Si vous avez de l’avance par rapport à vos camarades, vous pouvez essayer de connaitre le lieu de la prise de vues des photos de vos camarades.

### Pour aller un peu plus loin – Création d’un fichier image contenant la carte

* A la suite de votre programme précédent, rajouter les lignes de code ci-dessous :

# Sauvegarde de la carte avec un niveau de Zoom de 17

import matplotlib.pyplot as plt

nomCarteSauvegarde=nomFichierImage+'CarteZ17.png' # Pour repl.it

plt.savefig(nomCarteSauvegarde) ### Pour repl.it (Attention très long environ 5mn sur repl.it)

(im.open(nomCarteSauvegarde)).show() ### Pour afficher la carte comme un image

* Sauvegarder le programme dans le **même** répertoire que l’image de travail sous le nom extractionMetaData.py.
* Lancer le programme

1. Quel est le nom du fichier obtenu, quel est son emplacement ? :
2. En autonomie, produire les lignes de code permettant de sauvegarder la carte avec un niveau de zoom de 6.

### Fin de la séance 3

## Activité 8 – Pour aller un peu plus « plus » loin : Lancement d’un navigateur et création d’URL

* A la suite de votre programme précédent, rajouter les lignes de code ci-dessous :

import webbrowser

# Construction de l'URL avec un niveau de Zoom de 17

### Attention sur certains navigateurs, il est possible qu’il faille //// à la place de //.

adresse = "https://www.openstreetmap.org/?mlat="\

+str(latitude)+"&mlon="+str(longitude)+"&zoom=17"

webbrowser.open(adresse) # Appel du navigateur par défaut

* Sauvegarder le programme dans le **même** répertoire que l’image de travail sous le nom extractionMetaData.py.
* Lancer le programme

1. Quel est le navigateur ouvert par défaut ?

La ligne de code commençant par adresse = permet de construire une URL (Uniform Resource Locator). Cette URL contient un point d’interrogation (?) et une partie après ce caractère spécial. Une recherche rapide sur Wikipedia permet de trouver l’information suivante.

*Après le point d’interrogation, les informations suivantes correspondent à des données passées lors de l’appel de l’URL.* *Elles sont séparées par le caractère &.*

1. Quelles sont les données passées dans cette adresse internet ?
2. En autonomie, produire les lignes de code permettant de créer et d’ouvrir l’URL précédente avec un niveau de zoom de 6.

## Activité 9 – Pour aller vraiment beaucoup plus loin : Mise à disposition du code

On peut placer son code sur des sites « participatifs » comme repl.it. Pour cela il suffit de créer un compte et de copier le code en ayant au préalable choisi son langage de développement.