DS n° 04 – DS04

- Faire tous les exercices dans un même fichier **NomPrenom.py** à sauvegarder,
- Mettre en commentaire l'exercice et la question traités (ex : # Exercice 1),
- Ne pas oublier pas de commenter ce qui est fait dans votre code (ex : # Je crée une fonction pour calculer la racine d'un nombre),
- Il est possible de demander un déblocage pour certaines questions, mais celle-ci seront notées 0,
- Il faut vérifier avant de partir que le code peut s'exécuter et qu'il affiche les résultats que vous attendez. Les lignes de code qui doivent s'exécuter sont décommentées.

Présentation du thème du sujet

Les fichiers contenant des photos possèdent des informations sur celles-ci, elles sont communément appelées données `exif`. Entre autres, on y trouve :

- la date où elle a été prise,
- les coordonnées GPS de l'endroit où elle a été prise.

Le but de ce sujet est de faire un script qui va placer géographiquement sur une carte numérique les noms de fichiers photos archivés. On ajoutera des fonctionnalités à ce script au fur et à mesure de l'avancement du sujet, et la dernière partie permettra de travailler sur des informations concernant ces fichiers de photos en interrogeant une base de données. Dans tout le sujet, on supposera les modules Python importés.

I Géolocalisation de photos

Le script suivant montre que deux dates sous la forme d'une chaîne de caractères peuvent être classées.

```
x='2020:06:12 21:12:40'
y='2020:06:12 21:32:40'
print(x>y)
```

Le script pour fonctionner devra avoir en en-tête les deux lignes suivantes :

```
import os
os.chdir('photos')
```

Lors des questions Q2, Q3 et Q4, les chaînes de caractères contiennent une fois et une seule le caractère `'.'`.
Exemple : `mot1='chat.jpg'`, mot2=`'chien.jpeg'`, mot3=`'fichier.ipynb'`
Les chaînes de caractères qui commencent par `'.'` sont exclues.
Exemple :`'.ipynb'`, `'.jpg'` sont exclues.

Question 1 Écrire une fonction nommée `placeDuPoint` qui prend en entrée une chaîne de caractères et qui retourne l'indice du point dans cette chaîne caractères.

Exemple : dans `'devoir.ipynb'` le point a pour indice 6.

Solution 1.

```
def placeDuPoint(chaine):
    for i in range(len(chaine)):
        if chaine[i] == ".":
            return i

print(placeDuPoint('devoir.ipynb'))
```

Question 2 Écrire une fonction nommée `coupeExtension` qui prend en entrée une chaîne de caractères et qui retourne la chaîne de caractères précédent le caractère '.'.

Exemple : la fonction appliquée à `'devoir.ipynb'` retourne 'devoir'.

Solution 2.

```
def coupeExtension(nom_fichier):
    id_point = placeDuPoint(nom_fichier)
    return nom_fichier[:id_point]
print(coupeExtension('devoir.ipynb'))
```

Question 3 Écrire une fonction nommée `jpgTohtml` qui prend en entrée une chaîne de caractères représentant le nom d'un fichier avec son extension et qui retourne le nom avec l'extension .html.

Exemple : la fonction appliquée à `'photo.jpg'` retourne `'photo.html'`.

Solution 3.

```
def jpgTohtml(nom_fichier):
    nom_sans_extension = coupeExtension(nom_fichier)
    return nom_sans_extension + ".html"
print(jpgTohtml('photo.jpg'))
```

II Lecture de données exif

Dans cette partie, il est important de lire attentivement l'annexe 2 partie module exif.
On rappelle l'utilisation de la fonction `openImage` :

```
myimage=openImage("photo.jpg")
```

La variable `myimage` permet alors d'accéder aux données `exif` et on fera référence à cette variable dans toute la suite du sujet.
Ainsi, il faut avant de continuer ajouter au script les lignes suivantes :

```
from exif import Image

def openImage(nom): #nom est une chaîne de caractère qui représente ici une photo
    with open (nom, 'rb') as imageFile:
        return Image(imageFile)
myimage = openImage('photo.jpg')
```

La question suivante se réfère à la partie Coordonnées GPS Règles de Conversion de l'annexe 2.

Question 4 Écrire une fonction `convertTodecimale` qui prend en entrée un tuple (degré, minutes, secondes) et qui retourne le flottant correspondant. Vérifier le fonctionnement de la fonction en convertissant le tuple de l'annexe 2 : (36,10,11.78) grâce à un `print(convertTodecimale((36,10,11.78))`.

Solution 4.

```
def convertTodecimale(angle_tuple):
    degre, minutes, secondes = angle_tuple
    return degre + 1 / 60 * minutes +  1 / 3600 * secondes
print(convertTodecimale((36,10,11.78)))
```

On définit un type Point comme étant un tuple (latitude, longitude) de deux flottants signés.

Question 5 Écrire une fonction `imageToGpsPoint` qui prend en entrée une variable `myimage` et qui retourne une variable de type Point.

Solution 5.

```
def imageToGpsPoint(im):
    latitude = convertTodecimale(im.gps_latitude)
    if im.gps_latitude_ref == 'S':
        # la latitude est négative, on prend l'opposé
        latitude *= -1
    longitude = convertTodecimale(im.gps_longitude)
    if im.gps_longitude_ref == 'W':
        # la longitude est négative, on prend l'opposé
        longitude *= -1
    return (latitude, longitude)
print(imageToGpsPoint(my_image))
```

Question 6 Afficher les coordonnées GPS de toutes les photos `photos0.jpg`, ..., `photos7.jpg`.

Solution 6.

```
listePhotos = ['photo{}.jpg'.format(k) for k in range(8)]

for photo in listePhotos:
    myimage = openImage(photo)
    print(imageToGpsPoint(myimage))
```

Les coordonnées du lycée Dorian, sont les suivantes : `lycee_dorian=[48.854411, 2.392279]`.

Question 7 Écrire une fonction `calculDistance(point1,point2)` qui prend en entrée deux points définis par leur coordonnées (latitude, longitude) et qui donne la distance relative définie comme la norme du vecteur entre ces deux points.

Ainsi, la distance entre le lycée Dorian et les coordonnées GPS de la photo `photo0.jpg}` est 9.22267829505805 (attention, ce n'est pas une distance en mètre).

Solution 7.

```
lycee_dorian=[48.854411, 2.392279]

def calculDistance(point1,point2):
    return ((point1[0]-point2[0])**2+(point1[1]-point2[1])**2)**(1/2)
myimage = openImage('photo0.jpg')
print(calculDistance(imageToGpsPoint(my_image),lycee_dorian)])
```

Question 8 Générer une liste `distances` contenant l'ensemble des couples `['nomphoto',distance]` dont la distance correspond à celle entre l'endroit où la photo a été prise et le lycée Dorian. Afficher `distances[0]`.

Le résultats de l'instruction `print(distances[0])` doit être `['photo0.jpg', 9.22267829505805]`.

Solution 8.

```
distances=[]

for photo in listePhotos:
    my_image = openImage(photo)
    distances.append([photo,calculDistance(imageToGpsPoint(myimage),lycee_dorian)])

print(distances[0])
```

Question 9 A l'aide d'une fonction de tri de votre choix, classer la liste `distances` de la distance la plus petite à la plus grande. Afficher la liste `distances` ainsi triée.

Solution 9.

```
def tri_bulle(liste):
    # Parcours de 1 à la taille de la liste
    for i in range(1, len(liste)):
        # Parcours des éléments précédents
        for j in range(0, len(liste)-i):
            # On permutte les deux éléments successifs
            if liste[j][1] > liste[j+1][1] :
                liste[j], liste[j+1] = liste[j+1], liste[j]

tri_bulle(distances)
print(distances)
```

Question 10 Afficher les noms et les distances de la photo la plus proche et de la plus éloignée.

Solution 10.

```
print('La photo prise la plus près est la {}.'.format(coupeExtension(distances[0][0])))
print('La photo prise la plus loin est la {}.'.format(coupeExtension(distances[-1][0])))
```

Question 11 Écrire une fonction `listeApresGouter` qui prend en entrée une liste de photos (les photos sont représentées par leur nom qui est une chaîne de caractères) et qui retourne la liste des photos prises après 17h.

Solution 11.

```
def listeApresGouter(photos):
    avant_midi = []
    for photo in photos:
        my_image = openImage(photo)
        var_date = my_image.datetime
        heure = var_date[11:13]
        if heure > "17":
            avant_midi.append(photo)
    return avant_midi

print(listeApresGouter(listePhotos))
```

III Recherche des dossiers des photos

Dans le but de rechercher des photos dans un dossier, nous allons commencer par rechercher tous les dossiers dans un dossier principal `Arborescence`.
Pour continuer, il faudra copier le code suivant dans le script.

```
def listAbsolutePath(directory):
    for dirpath,dirs,filenames in os.walk(directory):
        return sorted([os.path.abspath(os.path.join(dirpath, d)) for d in dirs])

os.chdir('Arboresence')

s=listAbsolutePath('Arboresence')[0]
print(s)
```

Il permet de créer la fonction `listAbsolutePath(directory)` qui permet de lister les sous-dossiers contenus dans le dossier `directory`.

On rappelle donne dans la suite une méthode permettant le parcours du graphe `gr` en largeur.

```
gr={'A':['B','D','E'],'B':['A','C','E'],'C':['B','D'],'D':['A','C'],'E':['A','B']}

def parcoursLargeur(gr,s):
    files=[s]
    noeuds_decouverts=[s]
    while len(files)!=0:
        noeud_courant=files[0]
        files=files[1:]
        for voisin in gr[noeud_courant]:
            if voisin not in noeuds_decouverts:
                noeuds_decouverts.append(voisin)
                files.append(voisin)
    return(noeuds_decouverts)

print(parcoursLargeur(gr,'A'))
```

Question 12 Adapter cette méthode en utilisant la fonction `listAbsolutePath(directory)` afin de lister l'ensemble des sous-dossiers du dossier `Arborescence`. Stocker l'ensemble de ces éléments dans une liste `noeudsDecouverts`.

Solution 12.

```
files=[s]
noeudsDecouverts=[s]
while len(files)!=0:
    noeud_courant=files[0]
    files=files[1:]
    for voisin in listAbsolutePath(noeud_courant):
        if voisin not in noeuds_decouverts:
            noeudsDecouverts.append(voisin)
            files.append(voisin)
print(noeudsDecouverts)
```

Dans cette dernière partie, nous souhaitons récupérer simplement les noms des dossiers sans leur chemin absolu.

Question 13 Proposer une solution permettant de créer une chaîne de caractères contenant les noms des dossiers seuls sans leur chemin relatif, dans leur ordre d'apparition par le parcours en largeur. Vous pourrez utiliser la fonction `split`.

Solution 13.

```
chaine=''
for dir in noeuds_decouverts:
    chaine+=dir.split('/')[-1]
print(chaine)
```

ANNEXE 1
Rappels sur Python

Rappels sur les listes

```
#On suppose que maListe est une liste
len(maListe) # donne la longueur de maListe
maListe.append(objet) # ajoute l'objet à maListe
#il sera le dernier élément une fois ajouté
#soit autreListe une liste
maListe.append(autreListe)#ajoute tous les éléments de autreListe à maliste
#autreListe est non modifiée
```

Rappels sur les tuples

Un tuple est la donnée d'un n-uplet non mutable (qui ne peut pas être modifié).

```
monTuple=(donnee1,donnee2)
monTuple[0]# permet d'accéder à donnee1
monTuple[1]# permet d'accéder à donnee2
#on peut aussi accéder aux données en faisant
val1,val2=monTuple
print(val1)
>>>donnee1
print(val2)
>>>donnee2
#Pour parcourir un tuple :
for x in monTuple:
    print(x)
>>>donnee1
>>>donnee2
```

Une fonction peut retourner un tuple dont on peut en extraire le contenu directement.
Exemple :

```
def retourneTuple(x,y):
    return (x+y,x-y)
som,diff=retourneTuple(10,3)
print(som)
>>>13
print(diff)
>>>7
```

Rappels sur les chaînes de caractères

Parcours d'une chaîne :

```
mot="test"
for lettre in mot:
    print(lettre)
#Résultat console :
T
e
s
t
```

Rappels sur le Slicing :

```
print(mot[:3])
>>>Bon
print(mot[3:])
>>>jour
```

Rappel sur la fonction len :

```
print(len(mot))
>>>7
```

ANNEXE 2
Présentation des données `exif`

Module exif

Chaque fichier de photo numérique contient des informations appelées `données exif` (`exif` signifie « EXchangeable Image File » ou fichier d'échange de données). Ces informations, créées par l'appareil photo lors de la prise de vue, sont stockées dans le fichier généré par l'appareil.
Voici deux exemples d'informations accessibles :

. Date de la prise de vue.
. Coordonnées GPS du lieu de la prise de vue.

Les `données exif` sont accessibles dans un script Python en important le module `exif`, puis en accédant à `Image` dans ce module. À l'issue du script suivant, la variable `my_image` permet d'accéder aux `données exif` :

```
from exif import Image
with open("photo.jpg", 'rb') as imageFile:
    my_image= Image(imageFile)
```

Afin de faciliter le codage, on donne ci-dessous la fonction `openImage` qui sera utilisée lors de ce sujet :

```
def openImage(nom): #nom est une chaîne de caractères qui représente ici une photo
    with open(nom, 'rb') as imageFile:
        return Image(imageFile)
    imageFile.close()
#Exemple d'appel de cette fonction :
my_image=openImage("photo.jpg")
```

Suite à l'appel de la fonction, la variable `my_image` permet d'accéder aux `données exif`.

La liste des `données exif` qui nous intéressent est :

```
my_image.datetime # une date au format chaîne de caractères
my_image.gps_longitude # un tuple de trois nombres
my_image.gps_longitude_ref # une chaîne de caractères 'E' ou 'W'
my_image.gps_latitude # un tuple de trois nombres
my_image.gps_latitude_ref # une chaîne de caractères 'N' ou 'S'
```

Coordonnées GPS - Règles de conversion

Le module folium, dont on a besoin ici, utilise des coordonnées `gps` au format tuple `(latitude, longitude)` où les deux valeurs sont en décimales signées suivant l'orientation Nord-Sud ou Est-Ouest. Les données `gps` issues des données `exif` d'une image sont au format tuple `(degré, minutes, secondes)`. Il faut donc procéder à une conversion.
On donne la règle de conversion suivante :
1 degré = 1
1 minute = 1/60
1 seconde = 1/3600
L'orientation pour la latitude est `'N'` ou `'S'` et pour la longitude `'E'` ou `'W'`.
Les valeurs décimales sont signées : négative si on est `'W'` ou `'S'` et positive sinon. L'accès à cette orientation est donnée par le code Python ci-dessous :

```
my_image.gps_latitude_ref # donne 'N' ou 'S'
my_image.gps_longitude_ref # donne 'E' ou 'W'
```

Un exemple :

```
my_image.gps_latitude # donne par exemple (36,10,11.78)
my_image.gps_longitude # donne par exemple (115,8,23.38)
my_image.gps_latitude_ref # donne par exemple 'N'
my_image.gps_longitude_ref # donne par exemple 'W'
```

Ce qui donne au format décimal avec l'orientation Nord et Ouest : `36.169939,-115.13983`.

Date de prise de vue
La date de prise de vue est une chaîne de caractères de longueur 19, le format est :

```
#année:mois:jour heure:minute:seconde avec un seul espace entre jour et heure
'1967:06:17 11:15:00'
'2020:12:24 23:59:59'
```

Pour accéder à l'information de date, il suffit d'écrire le script :

```
var_date=my_image.datetime
#var_date sera une chaîne de caractères qui contient les informations de date.
```