Corsera Python 3 Module 6

▼ JSON

json file

```
[
    {
        "id" : "01",
        "fisrt_name" : "marouane",
        "last_name" : "dbibih",
        "phone" : {
            "type" : "intl",
            "number" : "0948309"
        },
        "email" : {
            "hide" : "yes",
            "mail" : "marouane@gmail.com"
        }
    },
    {
        "id" : "02",
        "fisrt_name" : "abdessamade",
        "last_name" : "dbibih",
        "phone" : {
            "type" : "intl",
            "number" : "89579834"
        },
        "email" : {
            "hide" : "yes",
            "mail" : "abdessamade@gmail.com"
        }
    }
]
```

parsing JSON Data to python

```
import json

data = """
[
    {
        "id" : "01",
        "fisrt_name" : "marouane",
        "last_name" : "dbibih",
        "phone" : {
```

```
            "type" : "intl",
            "number" : "0948309"
        },
        "email" : {
            "hide" : "yes",
            "mail" : "marouane@gmail.com"
        }
    },
    {
        "id" : "02",
        "fisrt_name" : "abdessamade",
        "last_name" : "dbibih",
        "phone" : {
            "type" : "intl",
            "number" : "89579834"
        },
        "email" : {
            "hide" : "yes",
            "mail" : "abdessamade@gmail.com"
        }
    }
]
"""

# Parsing JSON data and retrun list and dictionnaire
infos = json.loads(data)

print("type of infos var :",type(infos))
print("Lenght",len(infos))
print(infos)

for item in infos:
    print("ID",item["id"])
    print("Fisrt name",item["fisrt_name"])
    print("Last name",item["last_name"])
    print("Phone",item["phone"]["number"])
    print("Email",item["email"]["mail"])
```

▼ **APIs**

Application Programming Interface

use web to access data on system

contract of service between two application

▼ **Google Geomtry API**

un programme Python qui interagit avec l'API de géocodage de Google (Google Geocoding API) ou un service alternatif pour obtenir des informations sur un emplacement géographique spécifié par l'utilisateur

```
# urlib utisliser pour envoyer des requete HTTP et traiter les reponse HTTP
import urllib.request,urllib.error,urllib.parse

# gerer les certificat SSL (Secure Sockets Layer) lors la communication entre des sites web securise
import ssl

# permet de travailler avec des données JSON, souvent utilisées dans les services web
import json

# If you have a Google Places API key, enter it here
# api_key = 'AIzaSy___IDByT70'
# https://developers.google.com/maps/documentation/geocoding/intro

# --api_key Marrakech--
# api_key = ChIJUZ4Xlo3urw0RuK2HT1O2UFk
```

```
# variable utilise pour stocker api key pour Google Places et comment utiliser le service de géocodage de Google
api_key = False

# si api_key is false variable service_url definie autre service de geocodage
# sinon il definie service geocodage de Google
if api_key is False:
    api_key = 42
    service_url = 'http://py4e-data.dr-chuck.net/json?'
else:
    service_url = 'https://maps.googleapis.com/maps/api/geocode/json?'


# --Ingnore SSL certificate error--

# creer un context SSL et return un objet SSLcontext qui cinfigure avec des parametre par defaut
ctx = ssl.create_default_context()

# désactiver la vérification du nom d'hôte en utilisant l'attribut check_hostname de l'objet SSLContext.
# Si cet attribut est défini sur False, le contexte SSL ne vérifiera pas que le nom d'hôte dans
# le certificat SSL correspond au nom d'hôte du site web auquel vous vous connectez.
# Cette option est souvent utilisée lors du développement ou du test d'applications qui se connectent
# à des sites web avec des certificats auto-signés ou des noms d'hôte qui ne correspondent pas
# aux noms de domaine officiels.
ctx.check_hostname = False

#le contexte SSL n'effectuera pas de vérification du certificat SSL lors de la communication avec le site web.
ctx.verify_mode = ssl.CERT_NONE

# une boucle while qui demande à l'utilisateur de saisir une adresse.
# Si l'adresse est vide, la boucle s'arrête.
while True:
    address = input("Entrer location : ")
    if len(address) < 1 :
        break

    # créent un dictionnaire vide "parms" et ajoutent l'adresse saisie par l'utilisateur au dictionnaire.
    param = dict()
    param['address'] = address

    # ajoutent la clé API à "parms" si "api_key" est défini
    if api_key is not False:
        param['key'] = api_key

    # coder les paramètres dans le dictionnaire "parms" en une chaîne de requête pour l'URL.(Codage query)
    url = service_url + urllib.parse.urlencode(param)
    # affichent l'URL de la requête qui sera envoyée à l'API de géocodage.
    print('Retrieving', url)

    # pour envoyer une requête à l'API de géocodage.
    # Le contexte SSL créé précédemment est utilisé pour gérer les certificats SSL
    query_api = urllib.request.urlopen(url,context=ctx)

    # lisent la réponse de l'API de géocodage et la décodent en une chaîne de caractères.
    data = query_api.read().decode()
    print('Retrieved',len(data),'characters')

    # Use json method load() pour charger chaine de caracthere jsno en python object.
    # si echoue en definie variable js en none
    try:
        js = json.loads(data)
    except:
        js = None

    # --  récupérer et afficher les informations de géolocalisation
    # pour une adresse donnée en utilisant l'API de géocodage de Google.--
    if not js or 'status' not in js or js['status'] != 'OK':
        print('==== Failure To Retrieve ====')
        print(data)
        continue

    # formater l'objet JSON de manière lisible par l'homme en l'affichant avec une indentation de 4 espaces
    print(json.dumps(js,indent=4))

    # Afficher les coordonees de localisation
```

```
    lat = js['results'][0]['geometry']['location']['lat']
    lng = js['results'][0]['geometry']['location']['lng']
    print('lat', lat, 'lng', lng)
    location = js['results'][0]['formatted_address']
    print(location)
```

Resulta JSON


```
Entrer location : Marrakech
Retrieving http://py4e-data.dr-chuck.net/json?address=Marrakech&key=42
Retrieved 1757 characters
{
    "results": [
        {
            "address_components": [
                {
                    "long_name": "Marrakesh",
                    "short_name": "Marrakesh",
                    "types": [
                        "locality",
                        "political"
                    ]
                },
                {
                    "long_name": "Marrakesh",
                    "short_name": "Marrakesh",
                    "types": [
                        "administrative_area_level_2",
                        "political"
                    ]
                },
                {
                    "long_name": "Marrakesh-Safi",
                    "short_name": "Marrakesh-Safi",
                    "types": [
                        "administrative_area_level_1",
                        "political"
                    ]
                },
                {
                    "long_name": "Morocco",
                    "short_name": "MA",
                    "types": [
                        "country",
                        "political"
                    ]
                }
            ],
            "formatted_address": "Marrakesh, Morocco",
            "geometry": {
                "bounds": {
                    "northeast": {
                        "lat": 31.7162668,
                        "lng": -7.887625799999999
                    },
                    "southwest": {
                        "lat": 31.5529761,
                        "lng": -8.1280804
                    }
                },
                "location": {
                    "lat": 31.6294723,
                    "lng": -7.981084500000001
                },
                "location_type": "APPROXIMATE",
                "viewport": {
                    "northeast": {
                        "lat": 31.7162668,
                        "lng": -7.887625799999999
                    },
```

```
                    "southwest": {
                        "lat": 31.5529761,
                        "lng": -8.1280804
                    }
                }
            },
            "place_id": "ChIJUZ4Xlo3urw0RuK2HT1O2UFk",
            "types": [
                "locality",
                "political"
            ]
        }
    ],
    "status": "OK"
}
lat 31.6294723 lng -7.981084500000001
Marrakesh, Morocco
```

▼ **Securing API Requests**

▼ **Exercice Extracting Data from JSON**

In this assignment you will write a Python program somewhat similar to http://www.py4e.com/code3/json2.py. The program will prompt for a URL, read the JSON data from that URL using urllib and then parse and extract the comment counts from the JSON data, compute the sum of the numbers in the file and enter the sum below:
We provide two files for this assignment. One is a sample file where we give you the sum for your testing and the other is the actual data you need to process for the assignment

```
import urllib.request, urllib.parse, urllib.error
import json

url = input('Enter location: ')
print('Retrieving', url)

# Open the URL and read the data
uh = urllib.request.urlopen(url) # open URL
data = uh.read().decode() # read the data
print('Retrieved', len(data), 'characters')

# Parse the JSON data
try:
    js = json.loads(data)
except:
    js = None

# Extract the comment counts and compute the sum
if js:
    counts = js['comments']
    print('Count:', len(counts))
    sum = 0
    for item in counts:
        sum += item['count']
    print('Sum:', sum)
else:
    print('Failed to retrieve data from', url)
```

▼ **Exercice Using the GeoJSON API**

In this assignment you will write a Python program somewhat similar to http://www.py4e.com/code3/geojson.py
. The program will prompt for a location, contact a web service and retrieve JSON for the web service and

parse that data, and retrieve the first **place_id**
 from the JSON. A place ID is a textual identifier that uniquely identifies a place as within Google Maps.

```
import urllib.request, urllib.parse, urllib.error
import json
import ssl

api_key = False
# If you have a Google Places API key, enter it here
# api_key = 'AIzaSy___IDByT70'
# https://developers.google.com/maps/documentation/geocoding/intro

if api_key is False:
    api_key = 42
    serviceurl = 'http://py4e-data.dr-chuck.net/json?'
else :
    serviceurl = 'https://maps.googleapis.com/maps/api/geocode/json?'

# Ignore SSL certificate errors
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

while True:
    address = input('Enter location: ')
    if len(address) < 1: break

    parms = dict()
    parms['address'] = address
    if api_key is not False: parms['key'] = api_key
    url = serviceurl + urllib.parse.urlencode(parms)

    print('Retrieving', url)
    uh = urllib.request.urlopen(url, context=ctx)
    data = uh.read().decode()
    print('Retrieved', len(data), 'characters')

    try:
        js = json.loads(data)
    except:
        js = None

    if not js or 'status' not in js or js['status'] != 'OK':
        print('==== Failure To Retrieve ====')
        print(data)
        continue

    print(json.dumps(js, indent=4))
    lat = js['results'][0]['geometry']['location']['lat']
    place_id = js['results'][0]['place_id']
    print("Place id",place_id)
```