# Project 1: Thu thập dữ liệu từ trang SoundCloud

Trang Soundcloud là một trang web cho phép người dùng upload và chia sẻ các bài hát. Ở đây, chúng ta sẽ thu thập thông tin về các nghệ sĩ, ban nhạc, podcast và người sáng tác âm nhạc trên trang SoundCloud qua 2 cách **Crawl data (Parse HTML)** và sử dụng **API**.

**Bảng phân công công việc:**

| MSSV | Họ và tên | Công việc |
|---|---|---|
| 18120429 | Phạm Trung Kiên | Thu thập các tracks bằng API |
| 19120057 | Lê Quốc Cường | Thu thập các users bằng API |
| 19120145 | Lê Đào Duy Trọng | Thu thập các Playlists bằng API |
| 19120124 | Võ Thị Cẩm Quỳnh | Cài đặt các hàm lấy dữ liệu bằng cách Parse HTML |
| 19120142 | Nguyễn Thị Phương Trang | Lên ý tưởng, kiểm thử và thực hiện thu thập dữ liệu bằng Parse HTML |

**Set up:**

- Import các thư viện cần thiết:

In [ ]:
```
import requests
import requests_cache
from bs4 import BeautifulSoup
from requests_html import HTMLSession
import time
import json
import re
import pandas as pd # Dùng để đọc và hiển thị file csv/tsv
from datetime import datetime, timedelta # Dùng để xử lý dữ liệu thời gian
```

- Setup/install cache để giúp giảm số lần phải lấy dữ liệu:

In [ ]:
```
def CheckCache(response):
    a='"incomplete_results":true'
    if a in response.text:
        return False
    else:
        return True

requests_cache.install_cache('demo-cache',expire_after=None,filter_fn=CheckCache)
```

## Part 1: API

Để thu thập thông tin trang SoundCloud bằng API, ta thực hiện đi từ playlist.

Ta sẽ lấy những playlist khả dụng bằng cách get trang `https://api-v2.soundcloud.com/playlists/[playlistID]?client_id=[cliend_id]` . Sau đó, ta sẽ cài đặt các hàm xử lý file json có từ những trang này để lấy thông tin của playlist, user và track tương ứng.

1.1. Hàm `getEmptyPlaylist(keys)` có input: `keys` (kiểu dữ liệu list) , sau khi hàm được thực hiện , các trường dữ liệu của PlayList sẽ được lưu vào keys Hàm được sử dụng để lấy các trường dữ liệu của Playlist , và sẽ trả về 1 dict với key là các trường dữ liệu vừa lấy được và value là 1 list trống:

In [ ]:
```
def getEmptyPlaylist(keys):
    playlistID=1
    url=f'https://api-v2.soundcloud.com/playlists/{playlistID}?client_id=6gsNBd4m
    r=requests.get(url)
    playList_temp = json.loads(r.text)
    _playList=playList_temp

    #chua cac key cua playlist
    #keys=[]
    junk_keys=['embeddable_by','kind','license','permalink','sharing','uri','set_
    for key in junk_keys:
        _playList.pop(key,None)
    for key in playList_temp.keys():
        if key not in junk_keys:
            if key == 'id':
                keys.insert(0,key)
            else:
                keys.append(key)
    for i in keys:
        _playList[i]=[]  #tao ra mot dict rong de save du lieu ve sau
    playListID ={'id':[]}
    _playList.pop('id',None)
    _playList={**playListID, **_playList}

    return _playList
```

1.2. Hàm `getEmptyUser(keys_user)` có input: `keys_user` (kiểu dữ liệu list) , sau khi hàm được thực hiện , các trường dữ liệu của User sẽ được lưu vào keys_user Hàm được sử dụng để lấy các trường dữ liệu của User , và sẽ trả về 1 dict với key là các trường dữ liệu vừa lấy được và value là 1 list trống

In [ ]:
```python
def getEmptyUser(keys_user):
    playlistID=1
    url=f'https://api-v2.soundcloud.com/playlists/{playlistID}?client_id=6gsNBd4m
    r=requests.get(url)
    playList_temp = json.loads(r.text)
    _userList=playList_temp['user']

    #chua cac key cua playlist
    #keys=[]
    junk_keys=['permalink','sharing','uri','urn','station_urn','station_permalink
    for key in junk_keys:
        _userList.pop(key,None)
    for key in playList_temp['user'].keys():
        if key not in junk_keys:
            if key == 'id':
                keys_user.insert(0,key)
            else:
                keys_user.append(key)
    for i in keys_user:
        _userList[i]=[]  #tao ra mot dict rong de save du lieu ve sau
    userListID ={'id':[]}
    _userList.pop('id',None)
    _userList={**userListID, **_userList}

    return _userList
```

1.3. Hàm `getEmptyTrackt(keys_track)` có input: `keys_track` (kiểu dữ liệu list) , sau khi hàm được thực hiện , các trường dữ liệu của PlayList sẽ được lưu vào keys_track Hàm được sử dụng để lấy các trường dữ liệu của Track , và sẽ trả về 1 dict với key là các trường dữ liệu vừa lấy được và value là 1 list trống

In [ ]:
```
def getEmptyTrack(keys_track):
    playlistID=1
    url=f'https://api-v2.soundcloud.com/playlists/{playlistID}?client_id=6gsNBd4m
    r=requests.get(url)
    playList_temp = json.loads(r.text)
    _trackList=playList_temp['tracks'][0]

    junk_keys=['user_id','user', 'station_urn','media','waveform_url','publisher_
    for key in junk_keys:
        _trackList.pop(key,None)
    for key in _trackList.keys():
        if key=='id':
            keys_track.insert(0,key)
        else:
            keys_track.append(key)
    keys_track.append('mime_type')
    keys_track.append('quality')
    keys_track.append('owner')
    media={'mime_type':[],'quality':[],'owner':[]}
    _trackList={**_trackList,**media}
    for key in keys_track:
        _trackList[key]=[]
    trackID ={'id':[]}
    _trackList.pop('id',None)
    _trackList={**trackID, **_trackList}
    return _trackList
```

1.4. Hàm `check_Exist(id , list_id)` có input:

- `id` : là ID của User hoặc Playlist hay Track
- `list_id` : là danh sách các id của User hoặc Playlist hay Track đã được thu thập

Để tránh trường hợp trùng lặp bộ giá trị , hàm được sử dụng để kiểm tra 1 bộ dữ liệu đã được thu thập hay chưa thông qua id , hàm trả về True hoặc False

In [ ]:
```
def checkExist(value,userList):
    if value in userList:
        return True
    else:
        return False
```

1.5. Hàm `getTrackList` có các input:

- `playListTrack` : là dict chứa các thông tin của track có được từ playlsist
- `_trackList` : là dict chứa các dữ liệu cần lấy cho track
- `keys_track` : là list chứa các key cần lấy của track

Sau khi thực hiện hàm, `_trackList` sẽ được thêm các dữ liệu của track lấy được

In [ ]:
```
def getTrackList(playListTrack,_trackList,keys_track):
    #print(playListTrack['artwork_url'])
    for key in keys_track:
        checkNone =False
        try:
            if(str(playListTrack[key])=='None'  or str(playListTrack[key])=='NaN'
                checkNone=True
        except:
            pass
        try:
            if checkNone==False:

                if key=='mime_type':
                    mime_type=playListTrack['media']['transcodings'][0]['format']
                    _trackList[key].append(mime_type)
                elif key=='quality':
                    quality=playListTrack['media']['transcodings'][0]['quality']
                    _trackList[key].append(quality)
                elif key =='owner':
                    owner=playListTrack['user']['id']
                    _trackList[key].append(owner)
                else:
                    _trackList[key].append(playListTrack[key])
            else:
                _trackList[key].append('None')
        except:
            pass
```

1.6. Hàm `getUserList` có các input:

- `playList` : là dict chứa các thông tin của playlsist
- `_userList` : là dict chứa các dữ liệu cần lấy cho user
- `keys_user` : là list chứa các key cần lấy của user

Sau khi thực hiện hàm, `_userList` sẽ được thêm các dữ liệu của user lấy được

In [ ]:
```
def getUserList(playList,_userList,keys_user):
    for key in keys_user:
        checkNone =False
        try:
            if(str(playList['user'][key])=='None'  or str(playList['user'][key])=
                checkNone=True
        except:
            pass
        if checkNone==False:

            if key=='badges':
                badges=''
                for badge in playList['user'][key]:
                    #print(type(playList['user'][key][badge]))
                    if(playList['user'][key][badge]==True):
                        badges=badges+str(badge)+','
                if badges=='':
                    badges='Nonee'
                badges=badges[:-1]
                _userList[key].append(badges)
            else:
                _userList[key].append(playList['user'][key])
        else:
            _userList[key].append('None')
```

1.7. Hàm `getPlayList` có các input:

- `playList` : là dict chứa các thông tin của playlist
- `_playList` : là dict chứa các dữ liệu cần lấy cho playlist
- `keys` : là list chứa các key cần lấy của playlist

Sau khi thực hiện hàm, `_playList` sẽ được thêm các dữ liệu của playlist lấy được

In [ ]:
```
def getPlayList(playList,_playList,keys):
    for key in keys:
        checkNone =False
        try:
            if(str(playList[key])=='None'  or str(playList[key])=='NaN'or len(pla
                checkNone=True
        except:
            pass
        if checkNone==False:
            if key=='user':
                _playList[key].append(playList[key]['id'])
            elif key=='tracks':
                tracks=''
                for id in playList[key]:
                    tracks=tracks+str(id['id'])+','
                tracks = tracks[:-1]
                _playList[key].append(tracks)
            else:
                _playList[key].append(playList[key])
        else:
            _playList[key].append('None')
```

Ở một số Playlist, tồn tại một số track không có đủ thông tin cần thu thập mà chỉ có một số thông tin cơ bản.

Hàm `checkKeysTrack(key_Track, list_Track)` dùng để kiểm tra xem một track có đủ thông tin cần lấy hay không.

In [ ]:
```
def checkKeysTrack(key_Track, list_Track):
    for i in key_Track[0:5]:
        if i not in list_Track.keys():
            return False
    if len(list_Track['media']['transcodings'])>0:
        if 'mime_type' not in list_Track['media']['transcodings'][0]['format'].ke
            return False
        if 'quality' not in list_Track['media']['transcodings'][0].keys():
            return False
    else:
        return False
    return True
```

Khai báo các biến

- `keys` : là list trường dữ liệu của Playlist
- `keys_user` : là list trường dữ liệu của User
- `keys_track` : là list trường dữ liệu của Track
- `_playList , _userList , _trackList` : là các dict để chứa dữ liệu của Playlist , User , Track`
- `i` : id của 1 Playlist
- `k` : số lượng của Playlist đã được thu thập, để kiểm soát được số lượng đã thu thập
- `target` : số bộ giá trị tối thiểu cần đạt được khi thu thập dữ liệu Khởi tạo giá trị cho i là 1 , tăng i lên 1 sau mỗi lần get API và thu thập dữ liệu nếu get API thành công. Nếu số bộ dữ liệu

chưa đạt được target ta tiếp tục tăng i

Chuyển các dữ liệu đã thu thập vào dataFrame và lưu vào file csv

In [ ]:
```
keys=[]
_playList={}
_playList=getEmptyPlaylist(keys)
keys_user=[]
_userList={}
_userList=getEmptyUser(keys_user)
keys_track=[]
_trackList =getEmptyTrack(keys_track)
i=1
k=1
target=1000
while len(_playList['id'])<=target or len(_userList['id'])<=target or len(_trackL
    url=f'https://api-v2.soundcloud.com/playlists/{i}?client_id=6gsNBd4mJwXr0LxTE
    r=requests.get(url)
    if r.ok ==True:
        #print(f'Links: {i}, Lan: {k}')  # this is for testing

        k=k+1
        playList=json.loads(r.text)
        getPlayList(playList,_playList,keys)
        if(checkExist(playList['user']['id'],_userList['id'])==False):
            getUserList(playList,_userList,keys_user)
        #just need to get 5 out of track_count
        times=0
        if(playList['track_count']>3):
            times=3
        else:
            times=playList['track_count']
        for index in range(0,times):
            playListTrack=playList['tracks'][index]
            if(checkExist(playListTrack['id'],_trackList['id'])==False and checkK
                getTrackList(playListTrack,_trackList,keys_track)
        #Your code here
        #example
        #getUser(playList,_user,keys of user)
        #getTrack(track,_track,keys of track)
       # p=len(_playList['id'])
       # u=len(_userList['id'])
       # t=len(_trackList['id'])
       # print(f'playlist: {p}, user: {u}, tracks: {t}')

    i=i+1
```

Lưu thông tin playlist, user, track vào các file csv tương ứng

In [ ]:
```
df1 = pd.DataFrame(_playList)
df2 = pd.DataFrame(_userList)
df3=pd.DataFrame(_trackList)

df1.to_csv(f'./Api_data/playlist.csv',index=False,encoding='utf-8')
df2.to_csv(f'./Api_data/userlist.csv',index=False,encoding='utf-8')
df3.to_csv(f'./Api_data/tracklist.csv',index=False,encoding='utf-8')
```

## Part 2: Parse HTML

**Khai báo các set chứa các nhãn dữ liệu trong dataframe:**

- `u_unorder` : là một set chứa các nhãn dữ liệu của dataframe users dựa theo thứ tự trong script ở kết quả trả về request trang web, chưa được sắp xếp thứ tự
- `u_order` : là một set chứa các nhãn dữ liệu của dataframe users đã được sắp xếp thứ tự
- `p_unorder` : là một set chứa các nhãn dữ liệu của dataframe playlists dựa theo thứ tự trong script ở kết quả trả về request trang web, chưa được sắp xếp thứ tự
- `p_order` : là một set chứa các nhãn dữ liệu của dataframe playlists đã được sắp xếp thứ tự
- `t_unorder` : là một set chứa các nhãn dữ liệu của dataframe tracks dựa theo thứ tự trong script ở kết quả trả về request trang web, chưa được sắp xếp thứ tự
- `t_order` : là một set chứa các nhãn dữ liệu của dataframe tracks đã được sắp xếp thứ tự

**Thông tin các nhãn dữ liệu:**

- Users:
  - `id` : id người dùng
  - `username` : tên người dùng
  - `first_name` : Tên hiển thị của người dùng
  - `last_name` : Họ hiển thị của người dùng
  - `full_name` : Tên hiển thị đầy đủ của người dùng
  - `description` : Mô tả thông tin giới thiệu của người dùng
  - `avatar_url` : đường dẫn đến avatar của người dùng
  - `country_name` : tên quốc gia của người dùng
  - `city` : thành phố hiện tại của người dùng
  - `created_at` : thời gian tạo tài khoản
  - `last_modified` : thời gian lần cuối chỉnh sửa thông tin
  - `followers_count` : số lượng người theo dõi
  - `followings_count` : số lượng người đang theo dõi
  - `groups_count` : số lượng group tham gia
  - `comments_count` : số lượng lượt bình luận nhận được
  - `likes_count` : số lượng lượt thích nhận được
  - `playlist_likes_count` : số lượng lượt thích của playlist do người dùng tạo
  - `playlist_count` : số lượng playlist do người dùng tạo ra
  - `track_count` : số lượng track do người dùng tải lên
  - `permalink_url` : đường dẫn đến trang cá nhân của người dùng
- Playlists:
  - `id` : id của playlist

    - `title` : tựa đề của playlist
    - `image_playlist` : đường dẫn đến ảnh đại diện của playlist
    - `numTracks` : số lượng track có trong playlist
    - `description` : thông tin mô tả giới thiệu playlist
    - `duration` : tổng thời gian các track của playlist
    - `genre` : thể loại
    - `label_name` : tên hãng thu playlist
    - `tag_list` : danh sách hashtag do người dùng thêm vào
    - `author` : tên người tạo playlist
    - `created_at` : thời gian tạo playlist
    - `release_date` : thời gian phát hình playlist
    - `last_modified` : thời gian lần cuối chỉnh sửa playlist
    - `likes_count` : số lượng lượt thích của playlist
    - `reposts_count` : số lượng lượt được đăng lại (chia sẻ) của playlist
    - `public` : trạng thái playlist có được truy cập công khai hay không
    - `sharing` : trạng thái playlist có được chia sẻ hay không
    - `trackids` : chuỗi các ID của các track có trong playlist
    - `first_track` : đường dẫn của track đầu tiên trong playlist
    - `permalink_url` : đường dẫn đến playlist
- Tracks:
    - `id` : id của track
    - `title` : tựa đề của track
    - `description` : thông tin mô tả, giới thiệu track
    - `track_format` : hình thức của track
    - `duration` : thời lượng track
    - `genre` : thể loại track
    - `label_name` : tên hãng thu track
    - `tag_list` : danh sách hashtag người dùng thêm vào cho track
    - `author` : tên người tạo track
    - `created_at` : thời gian tạo
    - `release_date` : thời gian phát hành
    - `last_modified` : thời gian lần cuối chỉnh sửa
    - `comment_count` : số lượng lượt bình luận
    - `download_count` : số lượng lượt tải về
    - `likes_count` : số lượng lượt thích
    - `playback_count` : số lượng lượt phát/ chạy track
    - `reposts_count` : số lượng lượt đăng lại track
    - `state` : trạng thái của track
    - `public` : trạng thái track có được truy cập công khai hay không
    - `permalink_url` : đường dẫn đến track

In [ ]:
```
#cài đặt hiển thị tất cả các cột
pd.set_option('display.max_columns', None)
#set các trường dữ liệu cài đặt cho dataframe lưu dữ liệu users, playlists, track
u_unorder=['username','country_name','avatar_url','city','comments_count','create
u_order=['id','username','first_name','last_name','full_name','description','avat
p_unorder=['author','image_playlist','numTracks','created_at','description','dura
p_order=['id','title','image_playlist','description','duration','genre','label_na
t_unorder=['comment_count','created_at','description','download_count','duration'
t_order=['id', 'title','description','track_format','duration','genre', 'label_na
```

Hàm `collect_user` có các input sau:

- `url` : là chuỗi dạng f-string cho biết đường link đến trang user cần lấy thông tin
- `users` : là dataframe chứa thông tin của user thu thập được, gồm các nhãn dữ liệu đã được khai báo
- Hàm này dùng để thu thập các thông tin của user thêm vào users, hàm trả về dataframe users và kết quả thu thập True, False

In [ ]:
```
def collect_user(url,users):
    session = HTMLSession()
    r = session.get(url)
    if(r):
        value=[]
        soup=BeautifulSoup(r.content,features='html.parser')
        for s in (soup.find_all("script")):
            #Xét script có dữ liệu cần tìm
            if("hydratable" in str(s)):
                #Lấy các dữ liệu trong meta
                st=str(s).find('"username"')
                en=str(s).find('",',st)
                value.append(str(s)[st+len('"username":"'):en])
                value.append(soup.find("meta", property="og:country-name").get('c
                #Cắt chuỗi trong script
                i=str(s).find('{"hydratable":"user","')
                j=str(s).find('"uri"')
                s=str(s)[i+len('{"hydratable":"user","data":{'):j-1]
                #Cắt chuỗi thành một mảng gồm nhiều trường dữ liệu
                r1 = re.split(r',"',s)
                for i in r1:
                    t=re.split(r'":',i)
                    #Với mỗi phần tử là trường dữ liệu tồn tại trong dataframe, t
                    if(t[0].strip('"') in list(users)):
                        #Nếu t được cắt thành 2 phần tử thì thêm t[1] là giá trị
                        #Nếu t được cắt thành 1 trường dữ liệu t[0] là tên của tr
                        #Ngược lại, nếu t rỗng thì bỏ qua
                        if(len(t)==2):
                            value.append(t[1].strip('"'))
                        elif(t[0]):
                            value.append('null')
                #Xét điều kiện kích thước của value bằng số lượng các cột của use
                if(len(value)==len(list(users))):
                    zipped = zip(list(users), value)
                    u = dict(zipped)
                    users=users.append(u,ignore_index=True)
                    return users, True
    return users, False
```

Hàm `collect_playlist` có các input sau:

- `url` : là chuỗi dạng f-string, cho biết đường link đến trang playlist cần lấy thông tin
- `playlists` : là dataframe chứa thông tin của playlist thu thập được, gồm các nhãn dữ liệu đã được khai báo
- Hàm này dùng để thu thập các thông tin của playlist thêm vào playlists, hàm trả về dataframe playlists và kết quả thu thập True, False

In [ ]:
```
def collect_playlist(url,playlists):
    session = HTMLSession()
    r = session.get(url)
    if(r):
        value=[]
        soup = BeautifulSoup(r.content,features="html.parser")
        for s in (soup.find_all("script")):
            #Xét script có dữ liệu cần tìm
            if("hydratable" in str(s)):
                #Lấy các dữ liệu trong meta
                value.append(soup.find("meta",itemprop="name").get('content'))
                value.append(soup.find("meta", property="og:image").get('content'
                value.append(soup.find("meta", itemprop="numTracks").get('content
                #Cắt chuỗi trong script
                i=str(s).find('{"hydratable":"playlist"')
                s=str(s)[i+len('{"hydratable":"playlist","data":{'):]
                j=s.find('"uri"')
                s1=s[:j-1]
                s2=s[str(s).find('"tracks":[{')+len('"tracks":[{'):]
                #Cắt chuỗi thành một mảng gồm nhiều trường dữ liệu
                r1 = re.split(r',"',s1)
                for i in r1:
                    t=re.split(r'":',i)
                    #Với mỗi phần tử là trường dữ liệu tồn tại trong dataframe, t
                    if(t[0].strip('"') in list(playlists)):
                        #Nếu t được cắt thành 2 phần tử thì thêm t[1] là giá trị
                        if(len(t)==2):
                            value.append(t[1].strip('"'))
                trackids="'"
                for k in range(min(5,int(value[2]))):
                    s2=s2[str(s2).find('"id":')+len('"id":'):]
                    end=str(s2).find(',"')
                    trackids = trackids + s2[:end]
                    if(k!=int(value[2])-1):
                        trackids=trackids+","
                for k in range(0,int(value[2])-5):
                    s2=s2[str(s2).find('{"id":')+len('{"id":'):]
                    end=str(s2).find(',"')
                    trackids = trackids + s2[:end]
                    if(k!=int(value[2])-5-1):
                        trackids=trackids+','
                trackids=trackids+"'"
                value.append(str(trackids))
                #Lấy thêm link track đầu tiên trong playlist
                value.append(f'https://soundcloud.com' + soup.find('h2',itemprop
                #Xét điều kiện kích thước của value bằng số lượng các cột của pla
                if(len(value)==len(list(playlists))):
                    zipped = zip(list(playlists), value)
                    p = dict(zipped)
                    playlists=playlists.append(p,ignore_index=True)
                    return playlists, True
    return playlists, False
```

Hàm `collect_track` có các input sau:

- `url` : là chuỗi dạng f-string, cho biết đường link đến trang track cần lấy thông tin
- `tracks` : là dataframe chứa thông tin của track thu thập được, gồm các nhãn dữ liệu đã được khai báo
- Hàm này dùng để thu thập các thông tin của track thêm vào tracks, hàm trả về dataframe tracks và kết quả thu thập True, False

In [ ]:
```
def collect_track(url, tracks):
    session = HTMLSession()
    r = session.get(url)
    if(r):
        soup = BeautifulSoup(r.content,features="html.parser")
        value=[]
        for s in (soup.find_all("script")):
            #Xét script có dữ liệu cần tìm
            if("hydratable" in str(s)):
                #Cắt chuỗi trong script
                i=str(s).find('{"hydratable":"sound"')
                s=str(s)[i+len('{"hydratable":"sound","data":{'):]
                j=s.find('"uri"')
                s=s[:j-1]
                #Cắt chuỗi thành một mảng gồm nhiều trường dữ liệu
                r1 = re.split(r',"',s)
                for i in r1:
                    #Với mỗi phần tử là trường dữ liệu tồn tại trong dataframe, t
                    t=re.split(r'":',i)
                    if(t[0].strip('"')=='id' and t[1].strip('"') in list(tracks['
                        return tracks, False
                    if(t[0].strip('"') in list(tracks)):
                        value.append(t[1].strip('"'))
                #Lấy thêm thông tin người đăng track
                author =  soup.find("link",rel="author")
                if (author):
                    author = (author.get('href')).replace('/','')
                value.append(author)
                #Xét điều kiện kích thước của value bằng số lượng các cột của tra
                if(len(value)==len(list(tracks))):
                    zipped = zip(list(tracks), value)
                    p = dict(zipped)
                    tracks=tracks.append(p,ignore_index=True)
                    return tracks, True
    return tracks,False
```

Hàm `collect_all` có các input sau:

- `users` : là dataframe chứa thông tin của user thu thập được, gồm các nhãn dữ liệu đã được khai báo
- `playlists` : là dataframe chứa thông tin của playlist thu thập được, gồm các nhãn dữ liệu đã được khai báo
- `tracks` : là dataframe chứa thông tin của track thu thập được, gồm các nhãn dữ liệu đã được khai báo
- `users_file` : là chuỗi, cho biết tên của file output dạng csv lưu thông tin users sau khi thu thập từ trang web

- `playlists_file` : là chuỗi, cho biết tên của file output dạng csv lưu thông tin playlists sau khi thu thập từ trang web
- `tracks_file` : là chuỗi, cho biết tên của file output dạng csv lưu thông tin tracks sau khi thu thập từ trang web Hàm này dùng để gọi lại các hàm `collect_user` , `collect_playlist` , `collect_track` để thu thập các dữ liệu cần thiết và lưu vào các dataframe có tên tương ứng là `users` , `playlists` , `tracks` được khai báo trong hàm.Trong quá trình thu thập dữ liệu, hàm sẽ in ra số lượng dữ liệu đã thu thập. Sau khi kết thúc quá trình thu thập, hàm trả về các dataframe sẽ được sắp xếp lại các cột dữ liệu để rõ ràng hơn đồng thời cũng lưu vào file output với các thông tin tương ứng là các nhãn dữ liệu của các dataframe trên.

In [ ]:

```
def collect_all(users_file, playlists_file, tracks_file,users,playlists,tracks):
    url = f'https://soundcloud.com/'
    index = 1002
    count_u = 0
    count_p = 0
    count_t = 0
    while (True):
        try:
            u_url=url+str(index)
            session = HTMLSession()
            u = session.get(u_url)
            if(u):
                p_url = u_url+'/sets'
                session = HTMLSession()
                p=session.get(p_url)
                #Lấy link playlists trong trang user hiện tại
                if(p.html.find('.audible', first=True)):
                    p_urls = p.html.find('.audible', first=True).absolute_links
                    for link in p_urls:
                        if('sets/' in link):
                            p_url=link
                            break
                    session = HTMLSession()
                    t = session.get(p_url)
                    t_soup=BeautifulSoup(t.content,features="html.parser")
                    try:
                        #Lấy link track đầu tiên trong playlist nếu có
                        t_url=f'https://soundcloud.com' + t_soup.find('h2',itempr
                        tracks, check_t=collect_track(t_url,tracks)
                        playlists, check_p=collect_playlist(p_url,playlists)
                        users, check_u=collect_user(u_url,users)
                        #Tăng biến đếm
                        if (check_u==True): count_u=count_u+1
                        if (check_p==True): count_p=count_p+1
                        if (check_t==True): count_t=count_t+1
                        #print('users:',count_u,'playlists:',count_p,'tracks:',co
                        #Xuất số lượng các bộ dữ liệu đã thu thập được ở hiện tại
                    except:
                        index=index+1
                        continue
                index=index+1
                #dieu kien dung
                num=1000
                if(count_u>num and count_p>num and count_t>num): break
        except:
            time.sleep(1)
    #Sắp xếp lại các cột dữ liệu trong dataframe
    users=users[u_order]
    playlists=playlists[p_order]
    tracks=tracks[t_order]
    #Ghi dữ liệu ở dataframe vào file
    users.to_csv(users_file,index=False)
    playlists.to_csv(playlists_file,index=False)
    tracks.to_csv(tracks_file,index=False)
    return users, playlists, tracks
```

**Lưu thông tin thu thập được vào các file csv tương ứng:**

In [ ]:
```
#Khởi tạo DataFrame
users=pd.DataFrame(columns=u_unorder)
playlists = pd.DataFrame(columns=p_unorder)
tracks=pd.DataFrame(columns=t_unorder)
#Thu thập dữ liệu và ghi vào file
users,playlists,tracks=collect_all('./Crawl_data/user.csv','./Crawl_data/playlist
```

In [ ]:
```
#Xuất dữ liệu được đọc từ file
u=pd.read_csv('./Crawl_data/user.csv',index_col=[0])
p=pd.read_csv('./Crawl_data/playlist.csv',index_col=[0])
t=pd.read_csv('./Crawl_data/track.csv',index_col=[0])
#display(u.head(),p.head(),t.head())
```