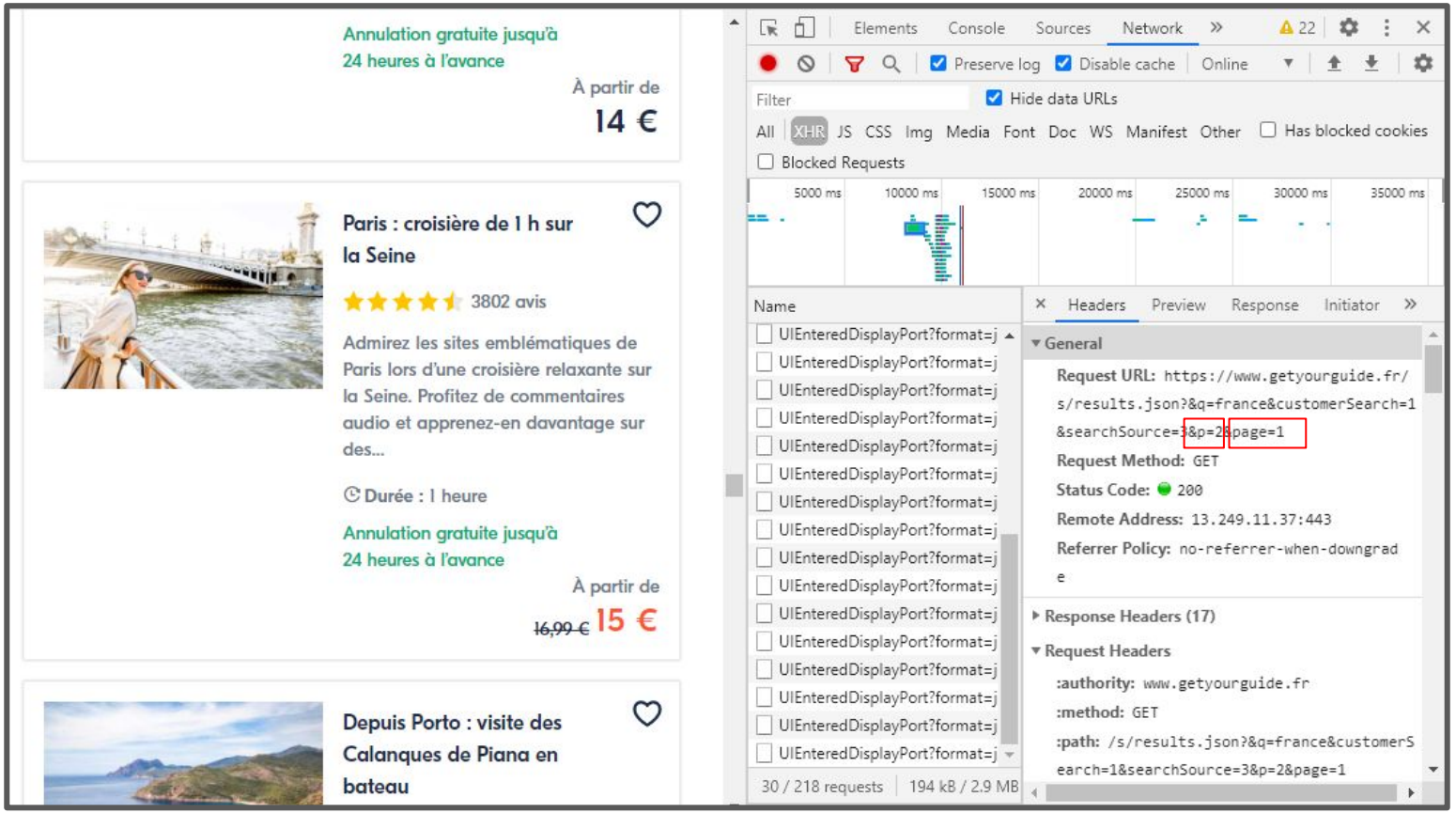
Web scraping all the french tourism “Good Plans”

As we are supposed to spend our holidays here..!




1. Web Scrapping

With 2 variables in the url



1. Web Scrapping

With 2 variables in the url

```
#Links to scrap
total_elements=1610
element_p_page=20
pagination=math.ceil(total_elements/element_p_page)+1

for i in range (pagination):
    j=math.floor(i/3.01)
    urls=[f'https://www.getyourguide.fr/s/results.json?&q=France&searchSource=2&p={i}&page={j}']
```

2. Getting data

```
headers=dict(i.split(': ') for i in headers.split('\n'))
for url in urls:
    resp=r.get(url, headers=headers)
    df=pd.DataFrame()
    step=0

    if step%5==0:
        time.sleep(2.4) # sleep every five parsed page
        print (step, 'pages loaded')

    results=resp.json()
    data=json_normalize(results['searchResults']['tours'])
    df=df.append(data)

    step+=1
    return df
```

3. Data Cleaning
1. Columns

```
def columns(df):

    df=df.drop_duplicates(subset='tourId')

    df=df.drop(columns=['tourId', 'horizontalImageUrl', 'verticalImageUrl', 'mobileImageUrl', 'horizontalSlimImageUrl', \
                        'description', 'isBestseller', 'isFeatured', 'hasDeal', 'dealMaxPercentage', 'isBoostedNewTour', 'hasBanner'
                        'hasRibbon', 'priceTag', 'detailsLink', 'isCertifiedPartner', 'isEcoCertified', 'isFirstTicket', 'isAuthoriz
                        'isGygOriginal', 'imageUrl', 'imageAlt', 'isFreeCancellation', 'smallGroup', 'privateTour', 'hasRating', \
                        'maxPossibleRating', 'totalRating', 'totalRatingTitle', 'averageRatingClass', 'ratingLink', 'languageIds',
                        'ratingStyleModifier', 'ratingStarsClasses', 'ratingTitle', 'hasDuration', 'useValidity', 'validFrom', \
                        'displayAbstract', 'displayDuration', 'displayDate', 'displayWishlist', 'isWishlistAccountWallEnabled', \
                        'displayRemoveButton', 'extraBadges', 'referrerViewPosition', 'hasDiscountedRecommendation', 'urlDetailsBtn'
                        'hideImage', 'isLikelyToSellOut', 'cardBannerMessage', 'cardBannerType', 'isPromoted', 'isElevated', \
                        'isOutdoors', 'isFamilyFriendly', 'showSkywards', 'skywardsMiles', 'assetsPath', 'isSpecialOffer', \
                        'collectionIdentifier', 'id', 'activityCardVersion', 'limit', 'isLoggedIn', 'timeSlotsExperimentEnabled', \
                        'freeCancellationFlag', 'resultSetPosition', 'highlightedOrientation', 'price.type', \
                        'experiments.hasOriginalsMoneyBackLabel','keyDetails.cancellation_policy.label', 'keyDetails.duration.label'
                        'keyDetails.cancellation_policy.description', 'keyDetails.skip_the_line.label', 'keyDetails.duration.descrip
                        'keyDetails.skip_the_line.description', 'keyDetails.instant_confirmation.label', 'keyDetails.instant_confirm

    df=df.rename(columns={'smallDescription': 'description',
                          'price.original': 'price_original', 'price.min':'price_min',
                          'averageRating':'rating'})
    return df
```

3. Data Cleaning
2. Columns

```
def cleaning(df):
    df['title']=df['title'].str.replace('&#039;', '\'')
    df['title']=df['title'].str.replace('&quot;', '\"')
    df['title']=df['title'].astype('str')

    df['price_min']=df['price_min'].str.replace(" ", "").str.strip('€\xa0\n').str.replace(',', '.').astype('float')

    df['price_original']=df['price_original'].str.replace(" ", "").str.strip('€\xa0\n').str.replace(',', '.').astype('float')

    df['rating']=df['rating'].fillna('Not rated yet')

    df['duration']=df['duration'].str.replace('heures', 'h')
    df['duration']=df['duration'].str.replace('heure', 'h')
    df['duration']=df['duration'].str.replace('jour', 'day')
    df['duration']=df['duration'].str.replace('minutes', 'min')
    df['duration']=df['duration'].str.replace(' - ', ' to ')
    df['duration']=df['duration'].str.replace(',5 h', 'h30')
    df['duration']=df['duration'].str.replace(',5', 'h30')
    df['duration']=df['duration'].str.replace('2 880,00 min', '2 days')
    df['duration']=df['duration'].str.replace('1 440,00 min', '1 day')
    df['duration']=df['duration'].str.replace('90 min', '1h30')

    return df
```

4. URLs scrapping
MAYDAY

```
#GET DATA
"""
for urls in df['url']:
    more_infos=r.get(urls, headers=headers)
    soup=BeautifulSoup(more_infos.content)

    country=list()
    region=list()
    city=list()

    place=str(soup.select('.activity__breadcrumbs ul li a'))
    names=re.findall(r'(>[A-z].*?<)', place)

    country.append(names[0].strip('><'))
    region.append(names[1].strip('><'))
    city.append(names[2].strip('><'))

df['country']=country
df['region']=region
df['city']=city
"""
```

5. Data Cleaning (end)
Columns sorting

```
def sorting(df):
    cols = df.columns.tolist()
    """
    cols.insert(1, cols.pop(cols.index('country')))
    cols.insert(2, cols.pop(cols.index('region')))
    cols.insert(3, cols.pop(cols.index('city')))
    """
    cols.insert(1, cols.pop(cols.index('title')))
    cols.insert(2, cols.pop(cols.index('description')))
    cols.insert(3, cols.pop(cols.index('price_min')))
    cols.insert(4, cols.pop(cols.index('price_original')))
    cols.insert(5, cols.pop(cols.index('rating')))
    cols.insert(6, cols.pop(cols.index('duration')))
    cols.insert(7, cols.pop(cols.index('url')))
    df = df[cols]


    df = df.reset_index(drop=True)

    return df
```

6. Apply functions

```
if __name__ == "__main__":


    import_url(urls)
    columns(df)
    cleaning(df)
    sorting(df)
```