Parsing Challenge Documentation:

Document Statistics:

`Number of emptly entries`: 20

`Number of entries with comma`: 12264

`Number of entries without comma`: 129092

`Date formats`:

- **Days-Month_Num-Year**
- **Month_Num-Days-Year**
- **Month_Num-Year**
- **Month_Name-Year**
- **Days-Month_Name-Year**
- **Days-Month_Name**
- **Days-Month_Num**
- **Year-Months_Num-Days**
- **Year-Months_Num-Days:Timezone**
-

`Other Date Formats`:

- **Season_Name Year**
- **Quartal_Name_1(Q1, Q2, etc.) Year**
- **Quartal_Name_2(I.,II.,III.,IV. Quartal) Year**

- **Ende Month_Name Year**
- **Ende Year**
- **Anfang Month_Name Year**
- **Anfang Year**
- **Ende Month_Name / Anfang Month_Name Year**

`Date Separators` `- . / |`

Python libraries used for the parsing:

- **re**: For regular expression
- **json**: For reading and writing json files
- **locale**: For determining the locale of the strings
- **datetime.datetime**: For parsing dates into ISO format
- **dateutil.parser**: For parsing dates in case generalize the date format

Extracting date with Regular Expression:

Dictionaries for parsing the year periods / seasons:

```
season_dict = {'Fr\u00fchjahr': '20 March', \
               'Sommer': '21 April',\
```

```
                'Sp\u00e4tsommer':'August', \
                'Mitte':' 1 June', \
                'Herbst':'23 September'}

q_dict = {'Q1':'01 Januar',
          'Q2':'01 April',
          'Q3':'01 Juli',
          'Q4':'01 Oktober',\
          'I.':'01 Januar',
          'II.':'01 April',
          'III.':'01 Juli',
          'IV.':'01 Oktober'}
```

Regular expression for extracting different formats of dates

```
months_short = 'Jan|J\u00e4n|Feb|Mar|M\u00e4|Apr|May|Mai|Jun|Jul|Aug|Sept|Oct|Okt|Nov|Dec|Dez'
months_long = 'January|J\u00e4nner|Januar|February|Februar|March|M\u00e4rch|April|May|Mai|June| \
              Juni|July|Juli|August|Sepember|October|Oktober|November|December|Dezember'
months = '(' + months_short + '|' + months_long + ')'
seps = '[\s\./\|-]'

days = '(0[1-9]|[12][0-9]|3[01])'
months_num = '(0[1-9]|1[012])'
```

```
years = '[12][7890]\d{2}'
time_zone = '(T\d{2}:\d{2}:\d{2}Z)'
```

Extract the patterns of quartals (ex. Q1, Q2, I. Quartal, 4. Quartals etc) and years

- Regular expression for extracting the data in Q1, Q2 format:

```
years = '[12][7890]\d{2}'
quart_num = '(Q[1-4])'
regex_quart_num = quart_num + '\s' + years
```

- Regular expression for finding pattern of quartals in both Roman and numerics (ex., I., IV., 1., 3.):

```
years = '[12][7890]\d{2}'
quart_roman = '(I|I[IV]|I{3})\.\sQuartal'
regex_quart_roman = quart_roman + '\s' + years
```

Extract the information from the strings starting with `Ende` or `Anfang` or `Season Names`:

Strings with `Ende.` or `Anfang` or `Season Names` have the following

information:

- Only Year
- With Month and Year

`Year with seasons:`

- **Fr\u00fchjahr**
- **Sommer**
- **Sp\u00e4tsommer**
- **Mitte**
- **Herbst**

Extracting City and Country:

The task is yet to be accomplished. However, some of the strings are well structured, such as:

- **Dates separated with comma**
- **Dates separated with comma**

Extracting date from a string with a `comma (,)`:

`Number of entries with comma`: 12264
`Example:`

- **Input:** “San Juan Costa Rica, 11/27/2009”
- **Output:**

- **addr** = San Juan Costa Rica
- **date_iso** = 2009-11-27

Extracting date from a string without a `comma (,)`:

`Number of entries without comma`: 129092

`Example:`

- **Input**: “San Juan Costa Rica 11/27/2009”
- **Output**:
  - **addr** = San Juan Costa Rica
  - **date_iso** = 2009-11-27

Ranking Policy:

`Rank:1` **Good output**

`Rank:2` **Only date, No address**

`Rank:3` **No date, only address**

`Rank:4` **No date, No address**