# Scraping Disney Movies Data from Wikipedia Website.

In [8]:

```
# Importing required Libaries.
import requests as req
from bs4 import BeautifulSoup as bs
import pandas as pd
import json
import re
from datetime import datetime
import pickle
```

Scraping the table from website into dictionary data type.

In [9]:

```
def get_content_val(row):
    if row.find('li'):
        return [li.get_text(" ",strip=True).replace("\xa0"," ") for li in row.find_all('li'
    elif row.find('br'):
        return [br for br in row.stripped_strings]
    else:
        return row.get_text(" ",strip=True).replace("\xa0"," ")

def clean_tags(soup):
    for tag in soup.find_all(['sup','span']):
        tag.decompose()
    return soup

# Getting each table from the website.
def get_info_box(urls):
    url = urls

    webpage = req.get(url)

    soup  = bs(webpage.content,'html.parser')
    table = soup.find(class_='infobox vevent')
    info_row = table.find_all('tr')
    clean_tags(soup)

    movie = {}
    for index,row in enumerate(info_row):
        if index == 0:
            movie['Title'] = row.text
        else:
            header = row.find('th')
            if header:
                content_key = row.find('th').get_text(" ",strip=True)
                content_value = get_content_val(row.find('td'))
                movie[content_key] = content_value

    return movie
```

In [10]:

```
get_info_box("https://en.wikipedia.org/wiki/Ponyo")
```

Out[10]:


```
{'Title': 'Ponyo',
 'Japanese': '',
 'Hepburn': 'Gake no Ue no Ponyo',
 'Directed by': 'Hayao Miyazaki',
 'Written by': 'Hayao Miyazaki',
 'Based on': ['The Little Mermaid', 'by', 'Hans Christian Andersen'],
 'Produced by': 'Toshio Suzuki',
 'Starring': ['Tomoko Yamaguchi',
  'Kazushige Nagashima',
  'Yūki Amami',
  'George Tokoro',
  'Yuria Nara',
  'Hiroki Doi',
  'Rumi Hiiragi',
  'Akiko Yano',
  'Kazuko Yoshiyuki',
  'Tomoko Naraoka'],
 'Cinematography': 'Atsushi Okui',
 'Edited by': 'Takeshi Seyama',
 'Music by': 'Joe Hisaishi',
 'Production company': 'Studio Ghibli',
 'Distributed by': 'Toho',
 'Release date': ['July 19, 2008'],
 'Running time': '101 minutes',
 'Country': 'Japan',
 'Language': 'Japanese',
 'Budget': ['¥', '3.4 billion', '(', 'US$', '34 million)'],
 'Box office': 'US$ 204.8 million'}
```


# Data Extraction

- Scraping from all the movies in the list of disney Movies

In [11]:

```
url = 'https://en.wikipedia.org/wiki/List_of_Walt_Disney_Pictures_films'

webpage = req.get(url)

soup  = bs(webpage.content,'html.parser')
movies = soup.select('.wikitable.sortable i a')
base_path = 'https://en.wikipedia.org/'

movie_info_box = []
for index,movie in enumerate(movies):
    if index % 10 ==0:
        print(index,"Done")
    try:
        link = movie['href']
        full_path = base_path + link
        title = movie['title']
        movie_info_box.append(get_info_box(full_path))
    except Exception as e:
        continue
```

```
0 Done
10 Done
20 Done
30 Done
40 Done
50 Done
60 Done
70 Done
80 Done
90 Done
100 Done
110 Done
120 Done
130 Done
140 Done
150 Done
160 Done
170 Done
180 Done
190 Done
200 Done
210 Done
220 Done
230 Done
240 Done
250 Done
260 Done
270 Done
280 Done
290 Done
300 Done
310 Done
320 Done
330 Done
340 Done
350 Done
360 Done
```

```
370 Done
380 Done
390 Done
400 Done
410 Done
420 Done
430 Done
440 Done
450 Done
460 Done
470 Done
480 Done
490 Done
500 Done
510 Done
520 Done
530 Done
```

Saving the data into json format

In [12]:

```
def save_data(title,data):
    with open(title , 'w', encoding='utf-8') as f:
        json.dump(data, f, ensure_ascii=False, indent=2)
```

In [13]:

```
def load_data(title):
    with open(title,encoding='utf-8') as f:
        return json.load(f)
```

In [14]:

```
save_data('movie_data_cleaned.json',movie_info_box)
```

In [15]:

```
movies = load_data('movie_data_cleaned.json')
```

# Data Cleaning

- Changing running time into integers from object data type
- Converting Budget and Box office into float using regex
- Changing date column into datetime object

In [16]:

```
def minute_to_int(running_time):
    if running_time == 'N/A':
        return None
    if isinstance(running_time,list):
        return int ((running_time[0]).split(' ')[0])
    else:
        return int(running_time.split(' ')[0])
```

In [17]:

```
for movie in movies:
    movie['Running time (int)'] = minute_to_int(movie.get('Running time','N/A'))
```

In [18]:

```
# Function to  change the money conversion
amounts = r"million|thounsand|billion|crore"
number = r"\d+(,\d{3})*\.*\d*"

word_re = rf"\$*{number}(-|\sto\s)?({number})?\s({amounts})"
value_re = rf"\${number}"

def word_to_number(word):
    value_dict  = {'million':1000000,'thousand':1000,'billion':1000000000,'crore':10000000}
    return value_dict[word]

def parse_word_syntax(string):
    value = float(re.search(number,string,flags=re.I).group().replace(',',''))
    word = word_to_number(re.search(amounts,string,flags=re.I).group())
    return value*word


def parse_value_syntax(string):
    value = float(re.search(number,string,flags=re.I).group().replace(',',''))
    return value

def money_conversion(money):
    if money == 'N/A':
        return None

    if isinstance(money,list):
        if len(money) >= 3:
            money = money[-1]
        else:
            money = money[0]

    word_syntax = re.search(word_re,money,flags=re.I)
    value_syntax = re.search(value_re,money)

    if word_syntax:
        return parse_word_syntax(word_syntax.group().lower())

    elif value_syntax:
        return parse_value_syntax(value_syntax.group())
    else:
        return None
```

In [ ]:

```
```

In [19]:

```
for movie in movies:
    movie['Budget (float)'] = money_conversion(movie.get('Budget','N/A'))
    movie['Box office (float)'] = money_conversion(movie.get('Box office','N/A'))
```

In [ ]:

```
```

In [20]:

```
def clean_date(date):
    cleaned_date =  date.split("(")[0].strip()
    if '–' in cleaned_date:
        return cleaned_date.split('–')[0]
    else:
        return cleaned_date

def date_conversion(date):
    if isinstance(date,list):
        date = date[0]
    else:
        date = date
    if date == 'N/A':
        return None

    date_str = clean_date(date)

    fmts=('%B %d, %Y',"%Y","%B %d %Y")
    for fmt in fmts:
        try:
            return datetime.strptime(date_str,fmt).date()
        except ValueError :
            pass
```

In [ ]:

```
```

In [21]:

```
for movie in movies:
    movie['Release date(datetime)'] = date_conversion(movie.get('Release date','N/A'))
```

# Saving data as a pickle format.

In [22]:

```
import pickle

def save_data(name,data):
    with open(name, 'wb') as f:
        pickle.dump(data, f)
```

In [23]:

```
def load_data(name):
    with open(name, 'rb') as f:
         return pickle.load(f)
```

In [24]:

```
save_data('disney_data-cleande.pickle',movies)
```

In [25]:

```
movies_data = load_data('disney_data-cleande.pickle')
```

# DataFrame

In [26]:

```
df = pd.DataFrame(movies)
```

In [27]:

```
df.head()
```

Out[27]:

| | Title | Production company | Distributed by | Release date | Running time | Country | Language | Box office | Runn t |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Academy Award Review of | Walt Disney Productions | United Artists | [May 19, 1937] | 41 minutes (74 minutes 1966 release) | United States | English | $45.472 | |
| **1** | Snow White and the Seven Dwarfs | Walt Disney Productions | RKO Radio Pictures | NaN | 83 minutes | United States | English | $418 million | |
| **2** | Pinocchio | Walt Disney Productions | RKO Radio Pictures | NaN | 88 minutes | United States | English | $164 million | |
| **3** | Fantasia | Walt Disney Productions | RKO Radio Pictures | [November 13, 1940] | 126 minutes | United States | English | 76.4–83.3 million (United States and Canada) | 1 |
| **4** | The Reluctant Dragon | Walt Disney Productions | RKO Radio Pictures | [June 27, 1941] | 74 minutes | United States | English | $960,000 (worldwide rentals) | |

5 rows × 47 columns

In [ ]:

```
```