Parsing/Cleaning Data

Extracting relevant portions of a dataset is important. Below we are going to go over two functions that will help you parse and clean your data, split and replace. Let's say we have the following string:

```
input_string = "Age:  24    (1990)     Name: Harry Potter"
```

Python will store this string as:

```
'Age:\t24\t(1990)\tName: Harry Potter'
```

Split

Calling `string.split()` returns a list of strings using whitespace as the delimiter. If we wrote out:

```
input_string.split()
```

We would get:

```
['Age:', '24', '(1990)', 'Name:', 'Harry', 'Potter']
```

We can also chose a specific delimiter, like `'('` for example. If we wrote:

```
first_split = inputString.split('(')
```

We would get:

```
first_split = ['Age:\t24\t', '1990)\tName: Harry Potter']
```

`first_split` is a list with 2 elements based on our original string but split on the open parenthesis. We could call split again (using a closed parenthesis) to isolate the year of birth.

```
second_split = first_split[1].split(')')
```

We would get:

```
second_split = ['1990', '\tName: Harry Potter']
```

Replace

Calling `string.replace(old, new)` returns a copy of the string with all occurrences of substring old replaced by new. We could do something such as:

```
first_replace = inputString.replace('24', '32')
```

We would get:

```
first_replace = 'Age:\t32\t(1990)\tName: Harry Potter'
```

Practice Parsing/Cleaning Data Exercise

Question 1

Write a function, `get_year`, which takes a list that contains strings with the name, year, and language of a book as a parameter and returns a list of years in which the books were published.

```
Input: ['Wuthering Heights (1847) English', 'Don Quixote (1605) Spanish']
Output: ['1847', '1605']
```

Question 2

Write a function, `upper_case_cse`, which takes a string and returns the same string with all instances of the letter c, s, and e in uppercase.

```
Input: "this is a string containing many words"
Output: thiS iS a String Containing many wordS
```

Debugging Exercise: Parsing IMDB DataBase

We obtain a text file form IMDB that contains the name, year and genre of a movie.
We want to parse the text file and obtain the data as three different lists:

Data format example:

```
...
"!Next?" (1994)                    Documentary
"#1 Single" (2006)                 Reality-TV
"#ByMySide" (2012)                 Drama
"#Follow" (2011)                   Mystery
"#nitTWITS" (2011)                 Comedy
"$#*! My Dad Says" (2010)          Comedy
...
```

Initial Code

```
def parse_data():

      file = open("movie_genres.txt","r")
      movie_name = []
      movie_year = []
      movie_genre = []

      for line in file:
            line_data = line.split('(')
            movie_name.append( line_data[0].replace('"','') )
            sub_line_data = line_data[1].split(')')
            movie_year.append( int(sub_line_data[0]) )
            sub_line_data[1].replace('\t','').replace('\n','')
            movie_genre.append( sub_line_data[1] )

parse_data()
```

Function Decomposition Exercise: Parsing IMDB Database

Assume that the output of the function `parse_data_to_dicts()` returns a list of dictionaries with keys of Name, Year, and Genre. If we were to write:

```
list_of_movies = parse_data_to_dicts()
```

`list_of_movies` could be:

```
[{'Name': 'Bridesmaids', 'Year': 2011, 'Genre': 'Comedy'}, {'Name': 'Insidious', 'Year': 2010, 'Genre': 'Horror'}]
```

We want you to do the following tasks:

1. Find a year's major category. (For a specific year what was the most popular genre?)
2. Find the best (most productive) year for a certain category. (For a specific genre in which year was it the most popular?)

In each of the tasks, define what functions might be required, and also specify the input and output of those functions. After you have defined the functions, write down how you will use the functions.

Note: Think of how you can reuse the functions you created as much as possible.

For example:

Task: Find the year span of the movie database.

```
dicts_of_movies = parse_data_to_dicts()

get_year_span(dicts_of_movies)
      input: list of dictionaries
      output: tuple of (start_year, end_year)
```

```
Use get_year_span to get the range of years that movies within the given dictionaries were produced.
```