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

Now assume that we have all the data lists parsed from the IMDB text file.
We want you to do the following tasks:

1. Find a year's major category.
2. Find the best (most productive) year for a certain category.
3. Find the most popular category and its best year.

In each of the task, 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.
You may also change the data format obtained from parse_data().

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

For example:

Task: Find the year span of the movie database.

```
    parse_data()

    get_year_span( movie_years )
        input:  list of movie_years
        output: tuple of (start_year, end_year)

    Use get_year_span with the whole movie year list to get the year
span.
```

Possible Solution for Debugging Exercise

```
import pdb

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

    for line in file:
        try:
            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].replace('/','').\
                               replace('I','').\
                               replace('V','').\
                               replace('?','9')) )
            movie_genre.append( sub_line_data[1].
                               replace('\t','').\
                               replace('\n','') )
        except:
            print line
            pdb.set_trace()

parse_data()
```

Possible Solution for Function Decomposition Exercise

```
Assume output of parse_data() is a dictionary of dictionaries with
keys = ['name', 'year', 'genre']
```

1. Find a year's major category.

```
    parse_data()

    parse_year_genre_data_by_key( key, value )
        input:  string of key ('year' or 'genre')
                key value ( 2010 or 'Action' )
        output: list of respective data that matches key and value
        ex: parse_year_genre_data_by_key( 'year', 2010 )
            will give you a list of movie genre in year 2010.

    find_major_category( list )
        input:  list of data
        output: tuple of (key of the max bin, count of the max bin)

    Use parse_year_genre_data_by_key
        with the whole movie year list to get the category.
    Use find_major_category
        with the category to get the major category.
```

2. Find the best (most productive) year for a certain category.

```
    Reuse the functions as above, just using different inputs.
    Use parse_year_genre_data_by_key
        with the category to get the years.
    Use find_major_category
        with the years to get the best year of the category.
```

3. Find the most popular category and its best year.

```
    Reuse the functions as above, just using different inputs.
    Use find_major_category
        with the whole movie genre list to get most popular category.
    Use parse_year_genre_data_by_key
        with the most popular category to get the years.
    Use find_major_category
        with the years to get the best year of the category.
```