Algorítmica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

Pandas

Dataframes

MovieLens Básico

https://grouplens.org/datasets/movielens/

grouplens.org/datasets/movielens/

recommended for education and development

MovieLens Latest Datasets

These datasets will change over time, and are not appropriate for reporting research results. We will keep the download links stable for automated downloads. We will not archive or make available previously released versions.

Small: 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users. Last updated 9/2018.

- README.html
- ml-latest-small.zip (size: 1 MB)

Full: 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users. Includes tag genome data with 14 million relevance scores across 1,100 tags. Last updated 9/2018.

- README.html
- ml-latest.zip (size: 265 MB)

Permalink: https://grouplens.org/datasets/movielens/latest/

MovieLens – cargando 25M

Probar abrir ratings.csv en Excel

```python
import pandas as pd

dataFrame1 = pd.read_csv("DataSets/ml-25m/ratings.csv")
dataFrame1.set_index('userId', inplace=True)

```

```
In [8]: dataFrame1.size
Out[8]: 75000285

In [9]: dataFrame1.shape
Out[9]: (25000095, 3)

In [10]: dataFrame1.head()
Out[10]:
        movieId  rating   timestamp
userId
1           296     5.0  1147880044
1           306     3.5  1147868817
1           307     5.0  1147868828
1           665     5.0  1147878820
1           899     3.5  1147868510

In [11]: dataFrame1.tail()
Out[11]:
        movieId  rating   timestamp
userId
162541    50872     4.5  1240953372
162541    55768     2.5  1240951998
162541    56176     2.0  1240950697
162541    58559     4.0  1240953434
162541    63876     5.0  1240952515
```

MovieLens - small

```python

import pandas as pd

dataRatings = pd.read_csv("DataSets/ml-latest-small/ratings.csv")
dataRatings.set_index('userId', inplace=True)

dataMovies = pd.read_csv("DataSets/ml-latest-small/movies.csv")
dataMovies.set_index('movieId', inplace=True)

dataLinks = pd.read_csv("DataSets/ml-latest-small/links.csv")
dataLinks.set_index('movieId', inplace=True)

dataTags = pd.read_csv("DataSets/ml-latest-small/tags.csv")


```

MovieLens - estructura

- 100836 ratings
- 3683 tag applications
- 9742 movies
- 610 users
- March 29, 1996 and September 24, 2018

```
In [2]: dataRatings.shape
Out[2]: (100836, 3)

In [3]: dataTags.shape
Out[3]: (3683, 4)

In [4]: dataMovies.shape
Out[4]: (9742, 2)

In [5]: dataLinks.shape
   ...:
Out[5]: (9742, 2)

In [6]: |
```

ITAM

MovieLens - información básica

```
In [4]: dataRatings.describe()
Out[4]:
             movieId         rating     timestamp
count  100836.000000  100836.000000  1.008360e+05
mean    19435.295718       3.501557  1.205946e+09
std     35530.987199       1.042529  2.162610e+08
min         1.000000       0.500000  8.281246e+08
25%      1199.000000       3.000000  1.019124e+09
50%      2991.000000       3.500000  1.186087e+09
75%      8122.000000       4.000000  1.435994e+09
max    193609.000000       5.000000  1.537799e+09
```

```
In [5]: dataMovies.describe()
Out[5]:
              title genres
count          9742   9742
unique         9737    951
top     Eros (2004)  Drama
freq              2   1053
```

```
In [6]: dataLinks.describe()
Out[6]:
             imdbId         tmdbId
count  9.742000e+03    9734.000000
mean   6.771839e+05   55162.123793
std    1.107228e+06   93653.481487
min    4.170000e+02       2.000000
25%    9.518075e+04    9665.500000
50%    1.672605e+05   16529.000000
75%    8.055685e+05   44205.750000
max    8.391976e+06  525662.000000
```

```
In [7]: dataTags.describe()
Out[7]:
            userId        movieId     timestamp
count  3683.000000    3683.000000  3.683000e+03
mean    431.149335   27252.013576  1.320032e+09
std     158.472553   43490.558803  1.721025e+08
min       2.000000       1.000000  1.137179e+09
25%     424.000000    1262.500000  1.137521e+09
50%     474.000000    4454.000000  1.269833e+09
75%     477.000000   39263.000000  1.498457e+09
max     610.000000  193565.000000  1.537099e+09
```

MovieLens - Ratings

Each line of this file after the header row represents one rating of one movie by one user, and has the following format:

```
userId,movieId,rating,timestamp
```

Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).

Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

MovieLens - Ratings

```
In [19]: dataRatings.head()
Out[19]:
         movieId  rating  timestamp
userId
1              1     4.0  964982703
1              3     4.0  964981247
1              6     4.0  964982224
1             47     5.0  964983815
1             50     5.0  964982931
```

MovieLens - Tags

Each line of this file after the header row represents one tag applied to one movie by one user, and has the following format:

```
userId,movieId,tag,timestamp
```

Tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. The meaning, value, and purpose of a particular tag is **determined by each user**.

MovieLens - Tags

```
In [20]: dataTags.head()
Out[20]:
   userId  movieId              tag   timestamp
0       2    60756            funny  1445714994
1       2    60756  Highly quotable  1445714996
2       2    60756     will ferrell  1445714992
3       2    89774     Boxing story  1445715207
4       2    89774              MMA  1445715200
```

MovieLens - Movies

Each line of this file after the header row represents one movie, and has the following format:

```
movieId,title,genres
```

Movie titles are entered manually or imported from https://www.themoviedb.org/, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.

MovieLens - Movies

```
In [18]: dataMovies.head()
Out[18]:
                                    title                                       genres
movieId
1                        Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
2                          Jumanji (1995)                   Adventure|Children|Fantasy
3                  Grumpier Old Men (1995)                                Comedy|Romance
4                 Waiting to Exhale (1995)                     Comedy|Drama|Romance
5        Father of the Bride Part II (1995)                                     Comedy
```

MovieLens - Links

Identifiers that can be used to link to other sources of movie data are contained in the file `links.csv`.

Each line of this file after the header row represents one movie, and has the following format:
`movieId,imdbId,tmdbId`

movieId is an identifier for movies used by https://movielens.org. E.g., the movie Toy Story has the link https://movielens.org/movies/1.

imdbId is an identifier for movies used by http://www.imdb.com. E.g., the movie Toy Story has the link http://www.imdb.com/title/tt0114709/.

tmdbId is an identifier for movies used by https://www.themoviedb.org. E.g., the movie Toy Story has the link https://www.themoviedb.org/movie/862.

Ejercicio

Identificar las 30 peliculas más populares y con mejores ratings de la base Movielens

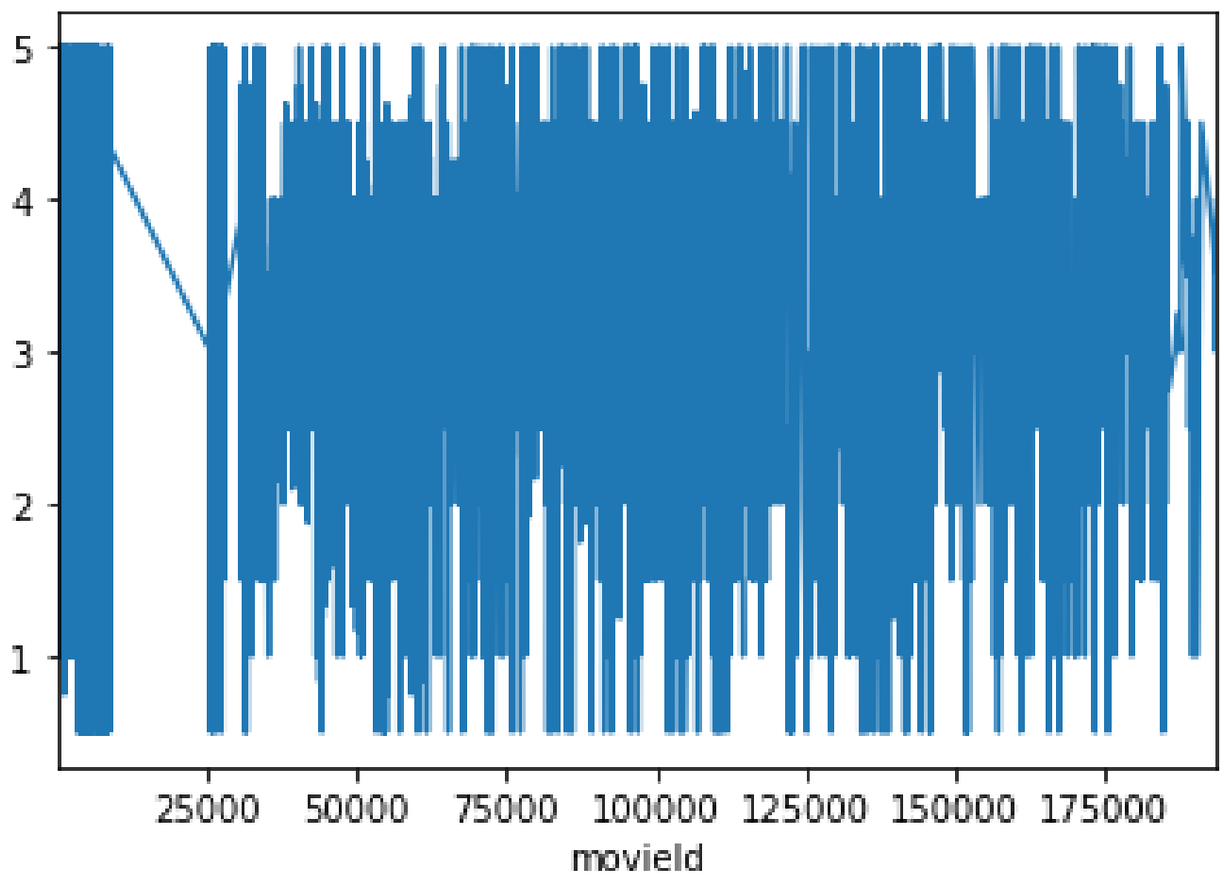
Análisis y entendimiento del problema

Análisis - groupby

```
In [85]: average_ratings = dataRatings.groupby('movieId').mean()

In [86]: average_ratings.shape
Out[86]: (9724, 2)

In [87]: average_ratings.head()
Out[87]:
           rating     timestamp
movieId
1        3.920930  1.129835e+09
2        3.431818  1.135805e+09
3        3.259615  1.005110e+09
4        2.357143  8.985789e+08
5        3.071429  9.926643e+08

In [88]: average_ratings.tail()
Out[88]:
         rating     timestamp
movieId
193581      4.0  1.537109e+09
193583      3.5  1.537110e+09
193585      3.5  1.537110e+09
193587      3.5  1.537110e+09
193609      4.0  1.537158e+09
```

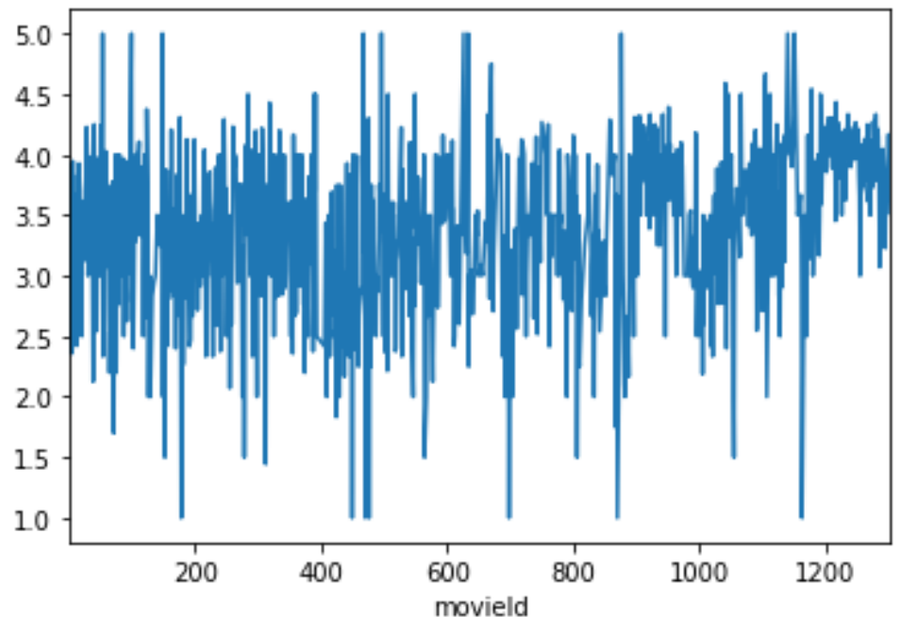
Análisis - plotting

```
In [92]: average_ratings.rating.plot()
Out[92]: <matplotlib.axes._subplots.AxesSubplot at 0x19d909d8518>
```

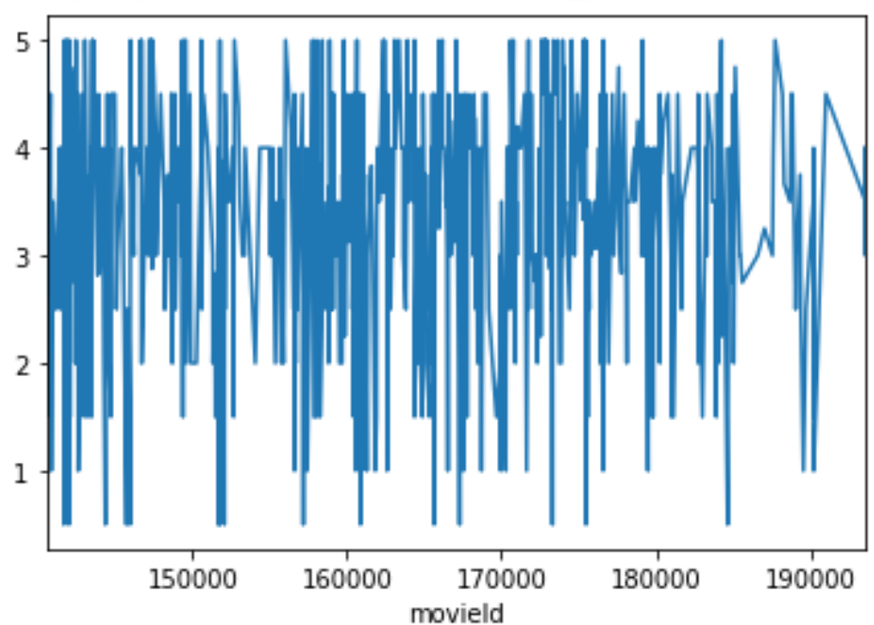


Análisis - plotting

```
In [97]: average_ratings[:1000].rating.plot()
Out[97]: <matplotlib.axes._subplots.AxesSubplot at 0x19d92d61588>
```

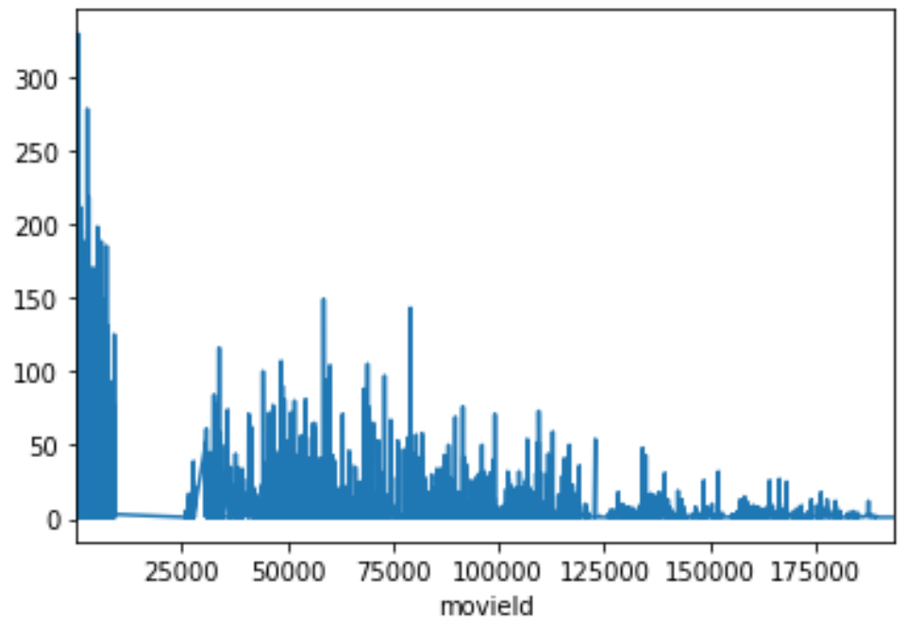



```
In [98]: average_ratings[9000:].rating.plot()
Out[98]: <matplotlib.axes._subplots.AxesSubplot at 0x19da2140ba8>
```



Análisis - count (número de ratings por película)

```
In [112]: dataRatings.groupby('movieId').rating.count().plot()
Out[112]: <matplotlib.axes._subplots.AxesSubplot at 0x19da18dd278>
```



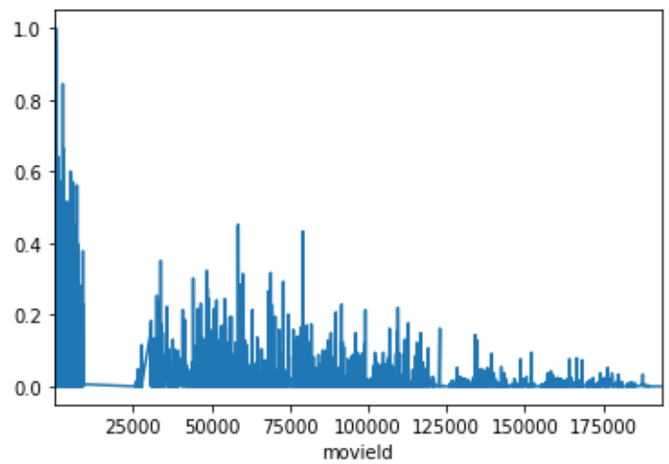

```
In [113]: dataRatings.groupby('movieId').rating.count().describe()
Out[113]:
count    9724.000000
mean       10.369807
std        22.401005
min         1.000000
25%         1.000000
50%         3.000000
75%         9.000000
max       329.000000
Name: rating, dtype: float64
```

Análisis - Agregando totales

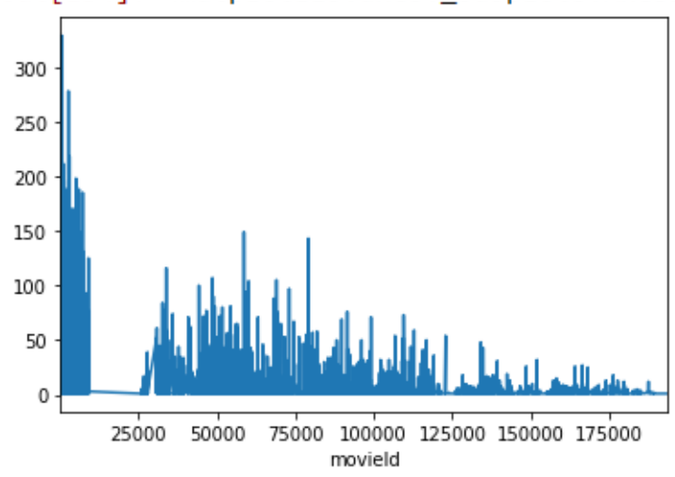
```
In [116]: average_ratings.head()
Out[116]:
           rating     timestamp
movieId
1        3.920930  1.129835e+09
2        3.431818  1.135805e+09
3        3.259615  1.005110e+09
4        2.357143  8.985789e+08
5        3.071429  9.926643e+08

In [117]: average_ratings.columns
Out[117]: Index(['rating', 'timestamp'], dtype='object')

In [118]: average_ratings['Total_ratings'] = dataRatings.groupby('movieId').rating.count()

In [119]: average_ratings.shape
Out[119]: (9724, 3)

In [120]: average_ratings.columns
Out[120]: Index(['rating', 'timestamp', 'Total_ratings'], dtype='object')

In [121]: average_ratings.head()
Out[121]:
           rating     timestamp  Total_ratings
movieId
1        3.920930  1.129835e+09            215
2        3.431818  1.135805e+09            110
3        3.259615  1.005110e+09             52
4        2.357143  8.985789e+08              7
5        3.071429  9.926643e+08             49
```

Análisis - Normalizando

```
In [132]: average_ratings['Normal'] = (average_ratings.Total_ratings - average_ratings.Total_ratings.min()) / (average_ratings.Total_ratings.max() -
average_ratings.Total_ratings.min())

In [133]: average_ratings.Normal.plot()
Out[133]: <matplotlib.axes._subplots.AxesSubplot at 0x19da1835748>
```




```
In [134]: dataRatings.groupby('movieId').rating.count().plot()
Out[134]: <matplotlib.axes._subplots.AxesSubplot at 0x19da192c358>
```



Análisis - Normalizando

```
In [135]: average_ratings.shape
Out[135]: (9724, 4)

In [136]: average_ratings.head()
Out[136]:
           rating     timestamp  Total_ratings    Normal
movieId
1        3.920930  1.129835e+09            215  0.652439
2        3.431818  1.135805e+09            110  0.332317
3        3.259615  1.005110e+09             52  0.155488
4        2.357143  8.985789e+08              7  0.018293
5        3.071429  9.926643e+08             49  0.146341

In [137]: average_ratings.tail()
Out[137]:
         rating     timestamp  Total_ratings  Normal
movieId
193581      4.0  1.537109e+09              1     0.0
193583      3.5  1.537110e+09              1     0.0
193585      3.5  1.537110e+09              1     0.0
193587      3.5  1.537110e+09              1     0.0
193609      4.0  1.537158e+09              1     0.0
```

Análisis - rating final

```
In [138]: average_ratings['ratingFinal'] = average_ratings.rating * average_ratings.Normal

In [139]: average_ratings.shape
Out[139]: (9724, 5)

In [140]: average_ratings.head()
Out[140]:
           rating     timestamp  Total_ratings    Normal  ratingFinal
movieId
1        3.920930  1.129835e+09            215  0.652439     2.558168
2        3.431818  1.135805e+09            110  0.332317     1.140452
3        3.259615  1.005110e+09             52  0.155488     0.506830
4        2.357143  8.985789e+08              7  0.018293     0.043118
5        3.071429  9.926643e+08             49  0.146341     0.449477

In [141]: average_ratings.tail()
Out[141]:
         rating     timestamp  Total_ratings  Normal  ratingFinal
movieId
193581      4.0  1.537109e+09              1     0.0          0.0
193583      3.5  1.537110e+09              1     0.0          0.0
193585      3.5  1.537110e+09              1     0.0          0.0
193587      3.5  1.537110e+09              1     0.0          0.0
193609      4.0  1.537158e+09              1     0.0          0.0
```

Análisis - Películas populares y con buen rating

```
In [145]: average_ratings.sort_values('ratingFinal', ascending=False).head(30)
Out[145]:
```

movieId	rating	timestamp	Total_ratings	Normal	ratingFinal
318	4.429022	1.189037e+09	317	0.963415	4.266985
356	4.164134	1.173755e+09	329	1.000000	4.164134
296	4.197068	1.137473e+09	307	0.932927	3.915558
2571	4.192446	1.259964e+09	278	0.844512	3.540572
593	4.161290	1.147081e+09	279	0.847561	3.526947
260	4.231076	1.189281e+09	251	0.762195	3.224905
110	4.031646	1.094719e+09	237	0.719512	2.900818
2959	4.272936	1.294796e+09	218	0.661585	2.826912
527	4.225000	1.171564e+09	220	0.667683	2.820960
480	3.750000	1.083516e+09	238	0.722561	2.709604
589	3.970982	1.105281e+09	224	0.679878	2.699784
1196	4.215640	1.197440e+09	211	0.640244	2.699038
50	4.237745	1.162501e+09	204	0.618902	2.622751
1	3.920930	1.129835e+09	215	0.652439	2.558168
1198	4.207500	1.216503e+09	200	0.606707	2.552721
2858	4.056373	1.215636e+09	204	0.618902	2.510499
858	4.289062	1.215667e+09	192	0.582317	2.497594
4993	4.106061	1.301967e+09	198	0.600610	2.466140
1210	4.137755	1.165959e+09	196	0.594512	2.459946
47	3.975369	1.140090e+09	203	0.615854	2.448246
2028	4.146277	1.213065e+09	188	0.570122	2.363883
150	3.845771	1.061333e+09	201	0.609756	2.344982
7153	4.118919	1.331073e+09	185	0.560976	2.310613
457	3.992105	1.031999e+09	190	0.576220	2.300329
5952	4.021277	1.306098e+09	188	0.570122	2.292618
608	4.116022	1.148028e+09	181	0.548780	2.258793
32	3.983051	1.082792e+09	177	0.536585	2.137247
2762	3.893855	1.217210e+09	179	0.542683	2.113128
780	3.445545	1.095549e+09	202	0.612805	2.111447
588	3.792350	1.080718e+09	183	0.554878	2.104292

Análisis - Películas populares y con buen rating
merge

```
In [148]: average_ratings = average_ratings.merge(dataMovies, on='movieId', how='left')

In [149]: average_ratings.sort_values('ratingFinal', ascending=False).head(30)
Out[149]:
```

Análisis - Películas populares y con buen rating

```
In [148]: average_ratings = average_ratings.merge(dataMovies, on='movieId', how='left')

In [149]: average_ratings.sort_values('ratingFinal', ascending=False).head(30)
Out[149]:
           rating  ...                                       genres
movieId            ...
318      4.429022  ...                                   Crime|Drama
356      4.164134  ...                      Comedy|Drama|Romance|War
296      4.197068  ...                   Comedy|Crime|Drama|Thriller
2571     4.192446  ...                       Action|Sci-Fi|Thriller
593      4.161290  ...                        Crime|Horror|Thriller
260      4.231076  ...                      Action|Adventure|Sci-Fi
110      4.031646  ...                             Action|Drama|War
2959     4.272936  ...                   Action|Crime|Drama|Thriller
527      4.225000  ...                                     Drama|War
480      3.750000  ...              Action|Adventure|Sci-Fi|Thriller
589      3.970982  ...                                 Action|Sci-Fi
1196     4.215640  ...                      Action|Adventure|Sci-Fi
50       4.237745  ...                       Crime|Mystery|Thriller
1        3.920930  ...  Adventure|Animation|Children|Comedy|Fantasy
1198     4.207500  ...                            Action|Adventure
2858     4.056373  ...                               Drama|Romance
858      4.289062  ...                                 Crime|Drama
4993     4.106061  ...                           Adventure|Fantasy
1210     4.137755  ...                      Action|Adventure|Sci-Fi
47       3.975369  ...                             Mystery|Thriller
2028     4.146277  ...                             Action|Drama|War
150      3.845771  ...                        Adventure|Drama|IMAX
7153     4.118919  ...                Action|Adventure|Drama|Fantasy
457      3.992105  ...                                     Thriller
5952     4.021277  ...                           Adventure|Fantasy
608      4.116022  ...                   Comedy|Crime|Drama|Thriller
32       3.983051  ...                      Mystery|Sci-Fi|Thriller
2762     3.893855  ...                        Drama|Horror|Mystery
780      3.445545  ...              Action|Adventure|Sci-Fi|Thriller
588      3.792350  ...  Adventure|Animation|Children|Comedy|Musical
```

Análisis - Películas populares y con buen rating

```
In [165]: average_ratings.sort_values('ratingFinal', ascending=False).head(30)[['ratingFinal','title']]
Out[165]:
         ratingFinal                                                title
movieId
318         4.266985                    Shawshank Redemption, The (1994)
356         4.164134                                 Forrest Gump (1994)
296         3.915558                                 Pulp Fiction (1994)
2571        3.540572                                  Matrix, The (1999)
593         3.526947                    Silence of the Lambs, The (1991)
260         3.224905           Star Wars: Episode IV - A New Hope (1977)
110         2.900818                                   Braveheart (1995)
2959        2.826912                                   Fight Club (1999)
527         2.820960                              Schindler's List (1993)
480         2.709604                                 Jurassic Park (1993)
589         2.699784                   Terminator 2: Judgment Day (1991)
1196        2.699038  Star Wars: Episode V - The Empire Strikes Back...
50          2.622751                         Usual Suspects, The (1995)
1           2.558168                                     Toy Story (1995)
1198        2.552721  Raiders of the Lost Ark (Indiana Jones and the...
2858        2.510499                              American Beauty (1999)
858         2.497594                               Godfather, The (1972)
4993        2.466140  Lord of the Rings: The Fellowship of the Ring,...
1210        2.459946  Star Wars: Episode VI - Return of the Jedi (1983)
47          2.448246                        Seven (a.k.a. Se7en) (1995)
2028        2.363883                         Saving Private Ryan (1998)
150         2.344982                                     Apollo 13 (1995)
7153        2.310613  Lord of the Rings: The Return of the King, The...
457         2.300329                                Fugitive, The (1993)
5952        2.292618     Lord of the Rings: The Two Towers, The (2002)
608         2.258793                                         Fargo (1996)
32          2.137247          Twelve Monkeys (a.k.a. 12 Monkeys) (1995)
2762        2.113128                              Sixth Sense, The (1999)
780         2.111447               Independence Day (a.k.a. ID4) (1996)
588         2.104292                                      Aladdin (1992)
```

Análisis - Películas populares y con buen rating

```
In [167]: average_ratings.sort_values('ratingFinal', ascending=False).head(30)[['title','genres']]
Out[167]:
```

movieId	title	genres
318	Shawshank Redemption, The (1994)	Crime|Drama
356	Forrest Gump (1994)	Comedy|Drama|Romance|War
296	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller
2571	Matrix, The (1999)	Action|Sci-Fi|Thriller
593	Silence of the Lambs, The (1991)	Crime|Horror|Thriller
260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Sci-Fi
110	Braveheart (1995)	Action|Drama|War
2959	Fight Club (1999)	Action|Crime|Drama|Thriller
527	Schindler's List (1993)	Drama|War
480	Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller
589	Terminator 2: Judgment Day (1991)	Action|Sci-Fi
1196	Star Wars: Episode V - The Empire Strikes Back...	Action|Adventure|Sci-Fi
50	Usual Suspects, The (1995)	Crime|Mystery|Thriller
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1198	Raiders of the Lost Ark (Indiana Jones and the...	Action|Adventure
2858	American Beauty (1999)	Drama|Romance
858	Godfather, The (1972)	Crime|Drama
4993	Lord of the Rings: The Fellowship of the Ring,...	Adventure|Fantasy
1210	Star Wars: Episode VI - Return of the Jedi (1983)	Action|Adventure|Sci-Fi
47	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
2028	Saving Private Ryan (1998)	Action|Drama|War
150	Apollo 13 (1995)	Adventure|Drama|IMAX
7153	Lord of the Rings: The Return of the King, The...	Action|Adventure|Drama|Fantasy
457	Fugitive, The (1993)	Thriller
5952	Lord of the Rings: The Two Towers, The (2002)	Adventure|Fantasy
608	Fargo (1996)	Comedy|Crime|Drama|Thriller
32	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller
2762	Sixth Sense, The (1999)	Drama|Horror|Mystery
780	Independence Day (a.k.a. ID4) (1996)	Action|Adventure|Sci-Fi|Thriller
588	Aladdin (1992)	Adventure|Animation|Children|Comedy|Musical

Ejercicio

- Utilizando 25M
- Escribir un programa en Python que permita encontrar las N películas populares con mejores ratings
- El programa toma como parámetro N
- El programa exporta el resultado a un archivo CSV con la siguiente estructura:
  - Posición, título de la película, rating promedio y rating ponderado

ITAM

Ejercicio

```
import pandas as pd
import sys

n = 10
if len(sys.argv) > 1:
    n = int(sys.argv[1])


dataRatings = pd.read_csv('DataSets/ml-25m/ratings.csv')
dataRatings.set_index('userId', inplace=True)

dataMovies = pd.read_csv('DataSets/ml-25m/movies.csv')
dataMovies.set_index('movieId', inplace=True)

averageRatings = pd.DataFrame(dataRatings.groupby('movieId').rating.mean())

averageRatings['Total_ratings'] = dataRatings.groupby('movieId').rating.count()

averageRatings['Normal'] = (averageRatings.Total_ratings - averageRatings.Total_ratings.min())  \
                           / (averageRatings.Total_ratings.max() - averageRatings.Total_ratings.min())

averageRatings['ratingFinal'] = averageRatings.rating * averageRatings.Normal

averageRatings = averageRatings.merge(dataMovies,on='movieId', how='left')

### Creando archivo con resultados
resultado = averageRatings.sort_values('ratingFinal', \
                                       ascending=False).head(n)[['title','rating','ratingFinal']]
resultado.index = range(1,(n+1))
resultado.index.name = 'Posicion'

resultado.to_csv('topMovies.csv')
```

Ejercicio

```
In [9]: runfile('C:/Users/irazoz/Documents/ITAM/Cursos/2020/AyP-2020-Enero-Mayo/Ejemplos/EjemploPD09.py', wdir='C:/
Users/irazoz/Documents/ITAM/Cursos/2020/AyP-2020-Enero-Mayo/Ejemplos', args='30')
```

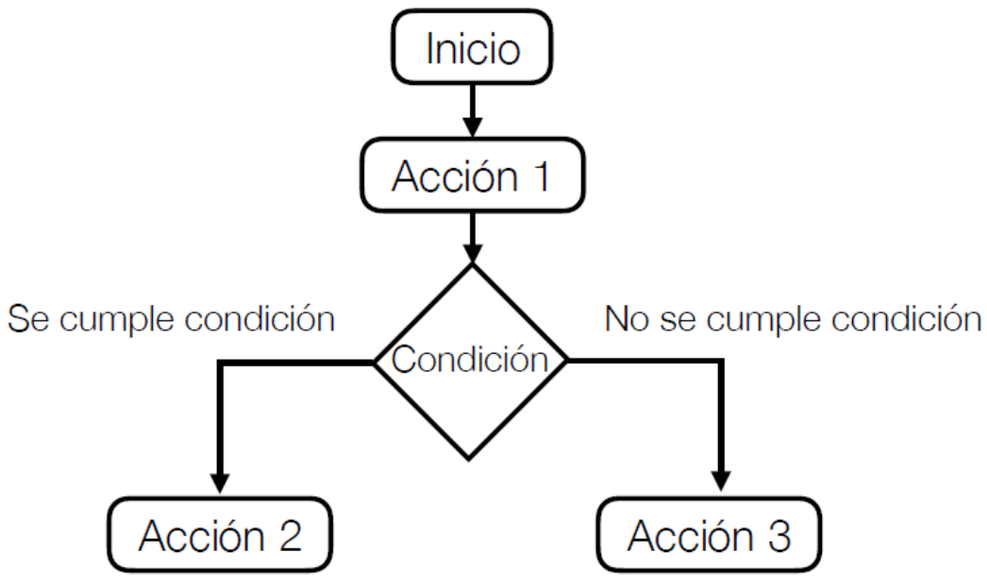
```
Posicion,title,rating,ratingFinal
1,"Shawshank Redemption, The (1994)",4.413576004516335,4.41308855594546
2,Pulp Fiction (1994),4.188912039361382,4.095408161589897
3,Forrest Gump (1994),4.048011436845787,4.048011436845787
4,"Silence of the Lambs, The (1991)",4.151341616415071,3.776197676504891
5,"Matrix, The (1999)",4.154099127610975,3.704636714945053
6,Star Wars: Episode IV - A New Hope (1977),4.120188599618726,3.4743266635341805
7,Schindler's List (1993),4.247579083279535,3.1488066317452046
8,Fight Club (1999),4.228310618821568,3.0495308834136847
9,Star Wars: Episode V - The Empire Strikes Back (1980),4.144122313069856,2.917006453
10,"Usual Suspects, The (1995)",4.284353213163313,2.9108260602133615
11,Braveheart (1995),4.002272573668559,2.9066940450046186
12,Jurassic Park (1993),3.6791749812920926,2.895978903239891
13,"Lord of the Rings: The Fellowship of the Ring, The (2001)",4.091188818716808,2.79
14,"Godfather, The (1972)",4.324336165187245,2.7857979588150057
15,Terminator 2: Judgment Day (1991),3.94637410899458,2.7786851592329245
16,Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981),4.1
17,Toy Story (1995),3.893707794587238,2.738257532116891
18,American Beauty (1999),4.107340423550448,2.706036233397674
19,Star Wars: Episode VI - Return of the Jedi (1983),3.996512919496695,2.693244612677
20,"Lord of the Rings: The Two Towers, The (2002)",4.0680511556963515,2.5528031899477
21,"Lord of the Rings: The Return of the King, The (2003)",4.0903399807075225,2.54967
22,Seven (a.k.a. Se7en) (1995),4.0791663372598626,2.532647206205212
23,"Fugitive, The (1993)",3.9721584270115637,2.4195119381712233
24,Fargo (1996),4.111421282646425,2.4127670705450655
25,Back to the Future (1985),3.9538763988305274,2.4062896812320673
26,Saving Private Ryan (1998),4.044107902443195,2.32165242228614
27,Apollo 13 (1995),3.87355561527172,2.2995106938812704
28,"Sixth Sense, The (1999)",4.009751032903046,2.298484356963641
29,Twelve Monkeys (a.k.a. 12 Monkeys) (1995),3.9057678412037236,2.255222655935192
30,Gladiator (2000),3.951473486205661,2.1653337652044886
```

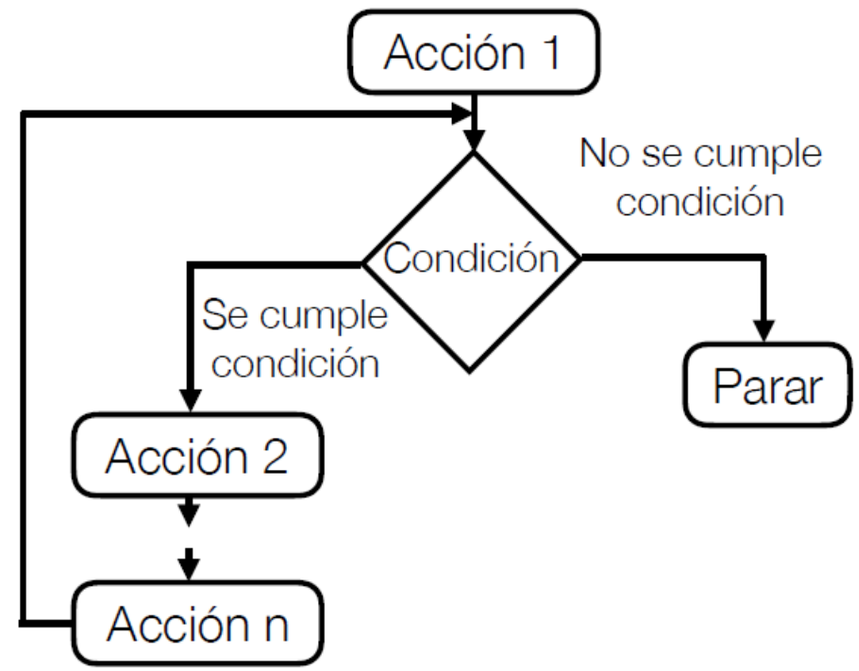
Repaso

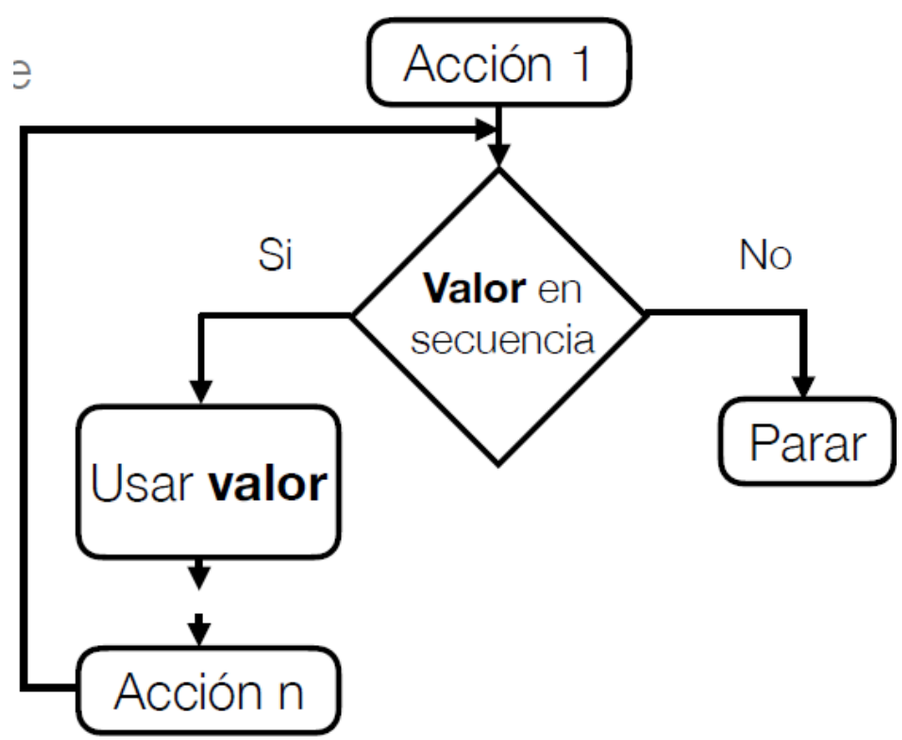
Elementos

- Diagramas de flujo
  - Estructuras de control
- Argumentos en la linea de comandos
- Estructuras de datos
  - Tuplas, Listas, Diccionarios
- Numpy
  - Arrays
- Pandas
  - Series, dataframes

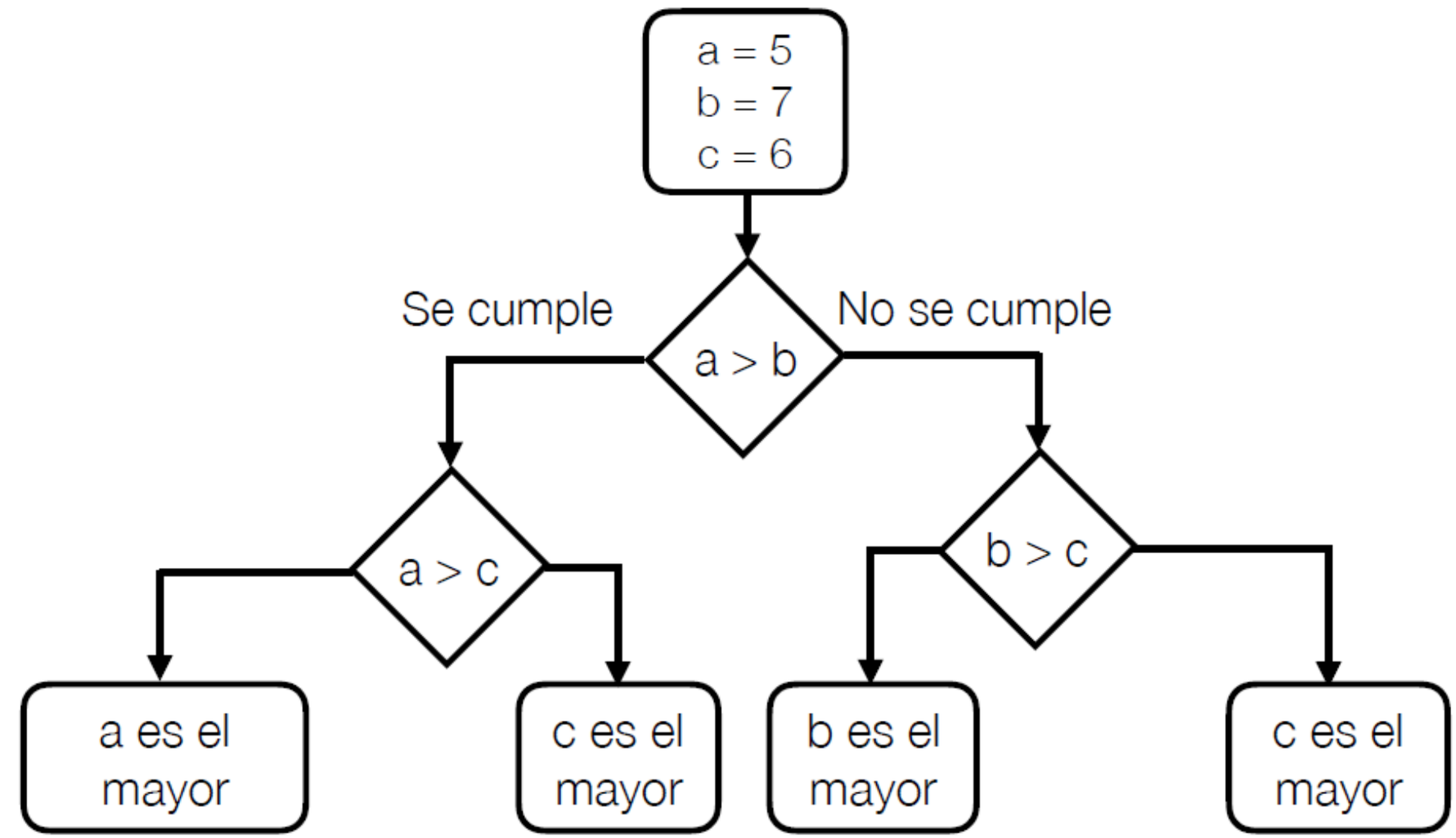
Diagramas de flujo







Diagramas de flujo



Diagramas de flujo

```
if y < 0:
    print("El valor de y es negativo")
elif y == 0:
    print("El valor de y es cero")
elif 0 < y < 10:
    print("El valor de y es positivo pero menor a diez")
else:
    print("El valor de y es igual a diez o mayor")
```

Diagramas de flujo

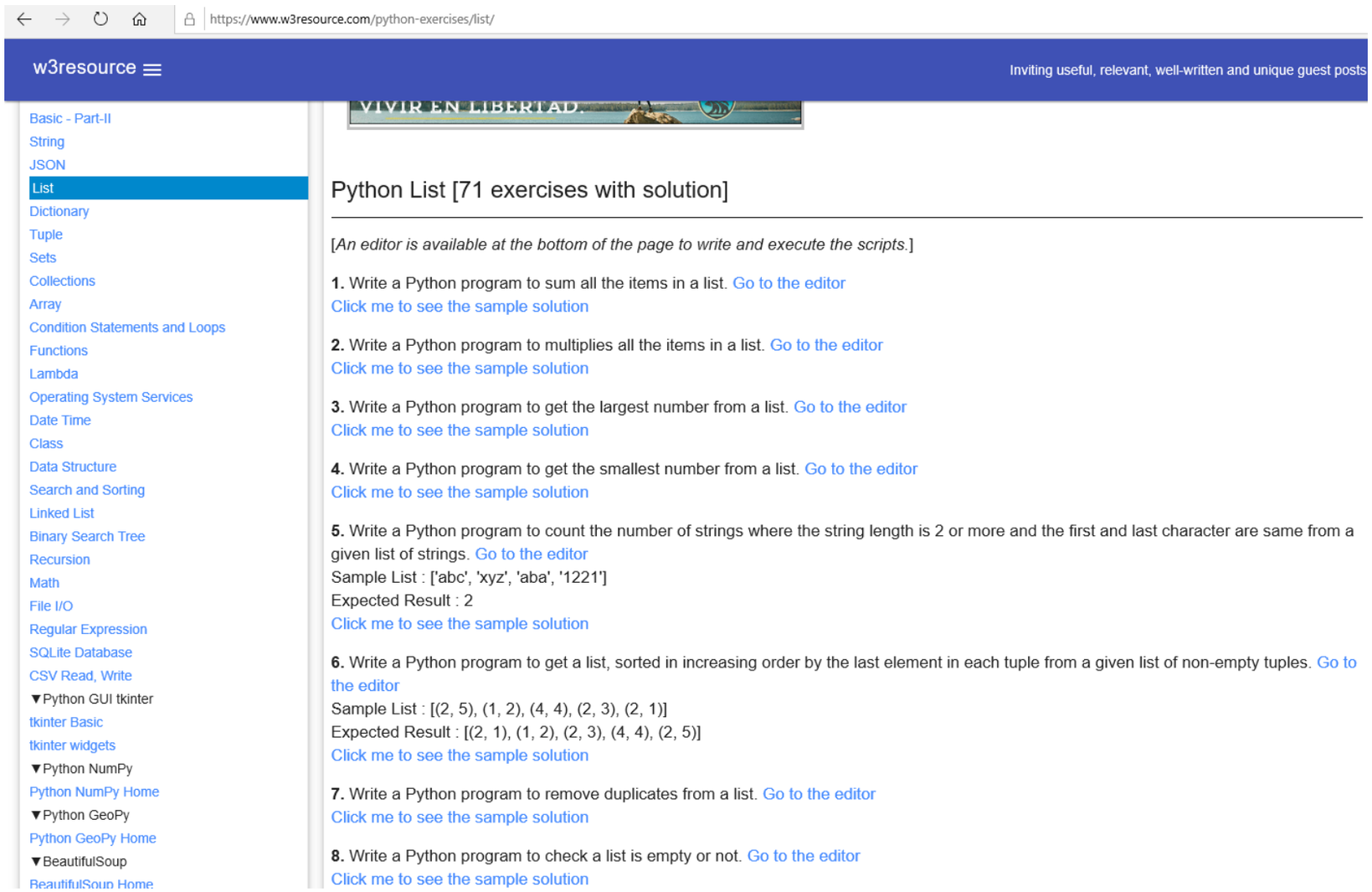
```
import sys
N = int(sys.argv[1])
print("Numeros de Fibonacci")

primero = 0
segundo = 1
print(primero)
print(segundo)
Contador = 2
while Contador <= N:
    Fibonnaci_num = primero + segundo
    print(Fibonnaci_num)
    primero = segundo
    segundo = Fibonnaci_num
    Contador += 1
```

Diagramas de flujo

```
import sys
N = int(sys.argv[1])
secuencia = [1, 2, 3, 4, 5, 6 ,7 , 8, 9 , 10]
for i in secuencia:
    print(N,'x',i,'=',N*i)
```

Diagramas de flujo

https://www.w3resource.com/python-exercises/



Diagramas de flujo

https://www.w3resource.com/python-exercises/

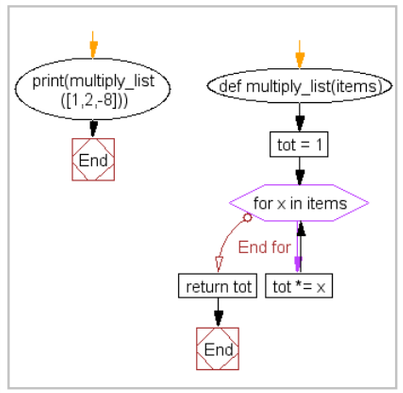
Python Code:

```python
def multiply_list(items):
    tot = 1
    for x in items:
        tot *= x
    return tot
print(multiply_list([1,2,-8]))
```

Sample Output:

```
-16
```

Flowchart:



Estructuras de datos

- Tuplas, Listas, Diccionarios
- Creación
- Propiedades

Crear diccionarios a partir de listas

```python
import numpy as np

list1 = ["A","B","C","D","E"]
list2 = [10,18,12,24,20]

dict1 = dict(zip(list1,list2))

media = np.array(list(dict1.values())).mean()

dict2 = {k:v for (k,v) in dict1.items() if v > media}

print(dict2)
```

Estructuras de datos

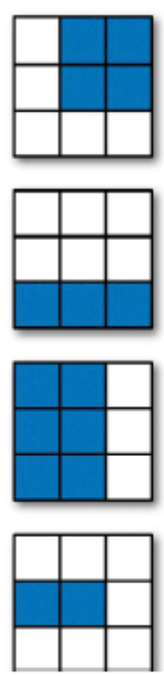
- Tuplas, Listas, Diccionarios
- Creación
- Propiedades

```
In [26]: dict1
Out[26]:
{'X23': ['Nombre1', 45, 67.9, [2, 3]],
 'Y45': ['Nombre2', 31, 237.5, [5, 3.7]],
 'W12': ['Nombre3', 10.5, 49],
 'ABC': (4, 5, 6),
 'DEF': 2}

In [27]: "W12" in dict1
Out[27]: True

In [28]: "W13" in dict1
Out[28]: False

In [28]:

In [29]: del dict1["W12"]

In [30]: dict1
Out[30]:
{'X23': ['Nombre1', 45, 67.9, [2, 3]],
 'Y45': ['Nombre2', 31, 237.5, [5, 3.7]],
 'ABC': (4, 5, 6),
 'DEF': 2}
```

Numpy

- Arreglos
- Creación

Accediendo al contenido



Expression	Shape
arr[:2, 1:]	(2, 2)
arr[2]	(3,)
arr[2, :]	(3,)
arr[2:, :]	(1, 3)
arr[:, :2]	(3, 2)
arr[1, :2]	(2,)
arr[1:2, :2]	(1, 2)

```
In [126]: arreglo2D_5 = np.random.rand(3,3)

In [127]: arreglo2D_5[0] = np.arange(1,4)

In [128]: arreglo2D_5[1] = np.arange(4,7)

In [129]: arreglo2D_5[2] = np.arange(7,10)

In [130]: arreglo2D_5
Out[130]:
array([[1., 2., 3.],
       [4., 5., 6.],
       [7., 8., 9.]])

In [131]:
```

Cambiando el contenido de un renglón

Pandas

- Series y dataframes
  - Creación, edición, procesamiento

```
import pandas as pd

dataFrame1 = pd.read_csv("DataSets/CO2.csv", encoding='latin1')
dataFrame1.set_index('Country', inplace=True)
serie1 = dataFrame1['1990']
serie2 = dataFrame1['2005']
serie3 = dataFrame1['2017']

print("Media para 1990 es: ",serie1.mean())
print("Media para 2005 es: ",serie2.mean())
print("Media para 2017 es: ",serie3.mean())

max1990 = serie1[serie1 == serie1.max()]
max2005 = serie2[serie2 == serie2.max()]
max2017 = serie3[serie3 == serie3.max()]

print("El pais que mas emitio CO2 en 1990 fue {a:s} con : {b:7.2f} toneladas".
     format(a = max1990.index[0], b = max1990[0]))

print("El pais que mas emitio CO2 en 2005 fue {a:s} con : {b:7.2f} toneladas".
     format(a = max2005.index[0], b = max2005[0]))

print("El pais que mas emitio CO2 en 2017 fue {a:s} con : {b:7.2f} toneladas".
     format(a = max2017.index[0], b = max2017[0]))
```