Introduction to

python

for scientific computing

- Lecture 5

In [ ]:

```

```

Exercise recap!

Find the movie with the highest rating

```
# Votes | Rating | Year | Runtime | URL | Genres | Title
   126807|    8.5|1957|5280|https://images-na.ssl-images....|Drama,War|Paths of Glory
    71379|    8.2|1925|4320|https://images-na.ssl-images....|Adventure,Comedy,Drama,Family|The Gold
```

In [51]:
```
movies_file = open("../downloads/250.imdb", "r", encoding = "utf-8")

max_rating = 0
max_rating_title = ""

for line in movies_file:
    if not line[0] == "#":
        line_strip = line.strip()
        line_split = line_strip.split("|")
        rating = float(line_split[1])
        title = line_split[-1]
        if rating > max_rating:
            max_rating = rating
            max_rating_title = title

fh.close()
print(max_rating, max_rating_title)
```

```
9.3 The Shawshank Redemption
```

IMDb

Find the number of unique genres

Watch out for the upper/lower cases!

```
# Votes | Rating | Year | Runtime | URL | Genres | Title
   126807|   8.5|1957|5280|https://images-na.ssl-images....|Drama,War|Paths of Glory
    71379|   8.2|1925|4320|https://images-na.ssl-images....|Adventure,Comedy,Drama,Family|The Gold
```

In [59]:
```
movies_file = open("../downloads/250.imdb", "r", encoding = "utf-8")

genres_list = []

for line in movies_file:
    if not line[0] == "#":
        line_strip = line.strip()
        line_split = line_strip.split("|")
        genres = line_split[5]
        genres_split = genres.split(",")
        for genre in genres_split:
            if genre.lower() not in genres_list: # Drama != drama != DRAMA -> dr
ama == drama == drama
                genres_list.append(genre.lower())

n_movies = 250
print("The number of unique genres is: " + str(len(genres_list)) + " from " + st
r(n_movies) + " movies")
fh.close()
```

```
The number of unique genres is: 22 from 250 movies
```

New type of string: f-strings

f-strings are awesome, use it all the time

```
print("There are " + str(len(genres_list)) + " unique genres")
```

can be written as:

```
print(f"There are {len(genres_list)} unique genres")
```

- Remember to put the `f` before the first " sign
- Anything in the brackets will be automatically replaced inside the string

In [66]:
```
# no need for concatenating!
#n_genres = len(genres_list)
#print(f"There are {n_genres} unique genres")

# can put a variable, but also a function call (if the function returns somethin
g)
#print(f"There are {len(genres_list)} unique genres")

# can have multiple variables of different types
#n_movies = "250"
#n_movies = 250
n_movies = 250.0
print(f"There are {len(genres_list)} genres from {n_movies} movies")
```

```
There are 22 genres from 250.0 movies
```

New data type: `set`

- A set contains an unordered collection of unique and immutable objects

Syntax:

For empty set:

```
setName = set()
```

For populated sets:

```
setName = {1,2,3,4,5}
```

Common operations on `sets`

```
set.add(a)
len(set)
a in set
```

In [74]:
```
x = set()
x.add(100)
x.add(25)
x.add(3)
x.add('3.0')
x.add(3)
x
#for i in x:
#    print(type(i))
#print(x)
#x.append(4)
##mySet = {2,5,1,3}
#mySet.add(5)
#mySet.add(4)
#print(mySet)
```

Out[74]:
```
{100, 25, 3, '3.0'}
```

Find the number of unique genres

```
# Votes | Rating | Year | Runtime | URL | Genres | Title
   126807|    8.5|1957|5280|https://images-na.ssl-images....|Drama,War|Paths of Glory
    71379|    8.2|1925|4320|https://images-na.ssl-images....|Adventure,Comedy,Drama,Family|The Gold
```

Modify your code to use sets

In [71]:
```
fh   = open("../downloads/250.imdb", "r", encoding = "utf-8")

genres_list = set()

for line in fh:
    if not line[0] == "#":
        line_strip = line.strip()
        line_split = line_strip.split("|")
        genres = line_split[5]
        genres_split = genres.split(",")
        for genre in genres_split:
            genres_list.add(genre.lower())

print("There are " + str(len(genres_list)) + " unique genres")
fh.close()
```

```
There are 22 unique genres
```

IMDb

Find the number of movies per genre

```
# Votes | Rating | Year | Runtime | URL | Genres | Title
   126807|    8.5|1957|5280|https://images-na.ssl-images....|Drama,War|Paths of Glory
    71379|    8.2|1925|4320|https://images-na.ssl-images....|Adventure,Comedy,Drama,Family|The Gold
```

Hm, starting to be difficult now...

Some solutions:

- Find the genres first, put them in a list, then for each genre count how many times it appears in the file
- Mantain two lists, one containing the genres, one where at the same position a counter is increased
- Two lists, one with unique genres, one with genres repeated, the count elements of first list in second list
- A list of tuples?

```python
fh   = open("../downloads/250.imdb", "r", encoding = "utf-8")

genres_list = []

for line in fh:
    if not line[0] == "#":
        line_strip = line.strip()
        line_split = line_strip.split("|")
        genres = line_split[5]
        genres_split = genres.split(",")
        for genre in genres_split:
            this_genres_index = 0
            genre_found = False
            for (stored_genre, genre_count) in genres_list:
                print(genre, stored_genre, genre_count)
                if genre.lower() == stored_genre:
                    genre_found = True
                    genres_list[this_genres_index] = (stored_genre, genre_count
+ 1)
                    print(genres_list[this_genres_index])
                else:
                    this_genres_index += 1
            if not genre_found:
                genres_list.append((genre.lower(), 1)) # first movie for that ge
nre!


print(genres_list)
fh.close()
```

New data type: `dictionary`

- A dictionary is a mapping of unique keys to values
- Dictionaries are mutable

Syntax:

`a = {}` (create empty dictionary)

`d = {'key1':1, 'key2':2, 'key3':3}`

In [82]:
```
myDict = {'drama': 4, 'thriller': 2, 'romance': 5}
myDict
```

Out[82]:
```
{'drama': 4, 'thriller': 2, 'romance': 5}
```

Operations on Dictionaries

Dictonary	
len(d)	Number of items
d[key]	Returns the item *value* for key *key*
d[key] = value	Updating the mapping for *key* with *value*
del d[key]	Delete key from d
key in d	Membership tests
d.keys( )	Returns an iterator on the keys
d.values( )	Returns an iterator on the values
d.items( )	Returns an iterator on the pair (key, value)

In [98]:

```
myDict = {'drama': 4,
          'thriller': 2,
          'romance': 5}
```

```
{'drama': 5, 'thriller': 2, 'romance': 5}
```

```
myDict["drama"] += 1
print(myDict)
#len(myDict)
#myDict['drama']
#myDict['horror'] = 2
#del myDict['horror']
#if "adventure" in myDict:
#    print(myDict["adventure"])
#else:
#    print("Couldn't find adventure movies")
#'drama' in myDict
#myDict.keys()
#myDict.items()
#myDict.values()
```

Exercise

```
In [ ]: myDict = {'drama': 182,
                  'war': 30,
                  'adventure': 55,
                  'comedy': 46,
                  'family': 24,
                  'animation': 17,
                  'biography': 25}
```

- How many entries are there in this dictionary?
- How do you find out how many movies are in the genre 'comedy'?
- You're not interested in biographies, delete this entry
- You are however interested in fantasy, add that we have 29 movies of the genre fantasy to the list
- What genres are listed in this dictionary?
- You remembered another comedy movie, increase the number of comedies by one

Back to finding the number of movies per genre

```
# Votes | Rating | Year | Runtime | URL | Genres | Title
   126807|    8.5|1957|5280|https://images-na.ssl-images....|Drama,War|Paths of Glory
    71379|    8.2|1925|4320|https://images-na.ssl-images....|Adventure,Comedy,Drama,Family|The Gold
```

Hint! If the genre is not already in the dictionary, you have to add it first

In [100]:

```
fh          = open('../downloads/250.imdb', 'r', encoding = 'utf-8')
genreDict = {}     # create empty dictionary

for line in fh:
    if not line.startswith('#'):
        cols  = line.strip().split('|')
        genres = cols[5].strip()
        genre_split = genres.split(',')
        for genre in genre_split:
            if genre.lower() not in genreDict:
                genreDict[genre.lower()] = 1
            else:
                genreDict[genre.lower()] += 1

fh.close()
print(genreDict)
```

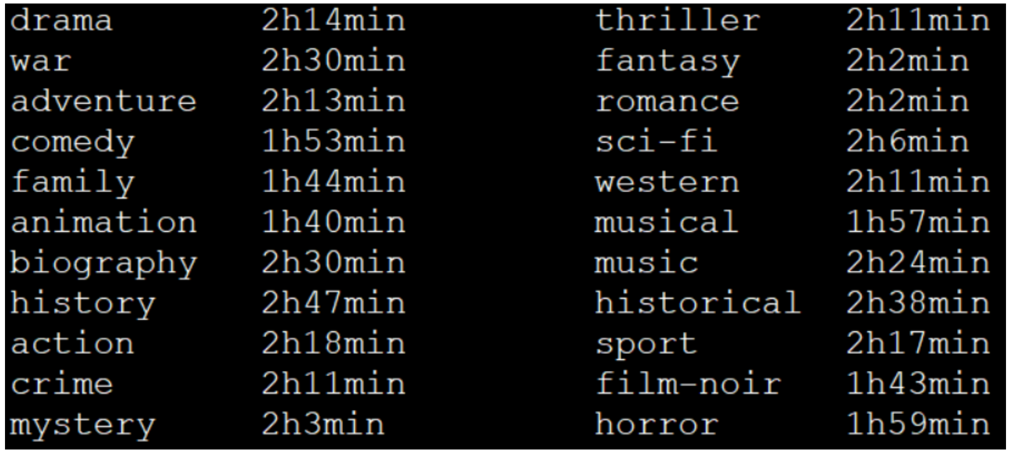
```
{'drama': 182, 'war': 30, 'adventure': 55, 'comedy': 46, 'family': 24, 'animat
ion': 17, 'biography': 25, 'history': 18, 'action': 31, 'crime': 62, 'mystery
': 41, 'thriller': 65, 'fantasy': 29, 'romance': 24, 'sci-fi': 28, 'western':
8, 'musical': 5, 'music': 3, 'historical': 1, 'sport': 7, 'film-noir': 7, 'hor
ror': 5}
```

What is the average length of the movies (hours and minutes) in each genre?

```
# Votes | Rating | Year | Runtime | URL | Genres | Title
   126807|   8.5|1957|5280|https://images-na.ssl-images....|Drama,War|Paths of Glory
    71379|   8.2|1925|4320|https://images-na.ssl-images....|Adventure,Comedy,Drama,Family|The Gold
```

Answer

What is the average length of the movies (hours and minutes) in each genre?



```
drama        2h14min        thriller    2h11min
war          2h30min        fantasy     2h2min
adventure    2h13min        romance     2h2min
comedy       1h53min        sci-fi      2h6min
family       1h44min        western     2h11min
animation    1h40min        musical     1h57min
biography    2h30min        music       2h24min
history      2h47min        historical  2h38min
action       2h18min        sport       2h17min
crime        2h11min        film-noir   1h43min
mystery      2h3min         horror      1h59min
```

Tip!
Here you have to loop twice

In [ ]:

```python
fh         = open('../downloads/250.imdb', 'r', encoding = 'utf-8')
movie_runtime = {}
movies_per_genre = {}

for line in fh:
    if not line.startswith('#'):
        cols    = line.strip().split('|')
        genres   = cols[5].strip()
        genres_split   = genres.split(',')
        runtime = int(cols[3])
        for entry in genres_split:
            if not entry.lower() in movie_runtime:
                movie_runtime[entry.lower()] = runtime
                movies_per_genre[entry.lower()] = 1
            else:
                movie_runtime[entry.lower()] += runtime
                movies_per_genre[entry.lower()] += 1

for genre in movie_runtime:
    total_genre_runtime = movie_runtime[genre]
    n_movies = movies_per_genre[genre]

    average_runtime_seconds = round(total_genre_runtime / n_movies)
    #print(f"Average runtime for genre {genre} is: {average_runtime_seconds} sec
onds")
    average_runtime_hours = average_runtime_seconds // 3600
    #leftover_minutes = (average_runtime_seconds - average_runtime_hours * 3600)
// 60
    leftover_minutes = (average_runtime_seconds % 3600) // 60
    print(f"Average runtime for genre {genre} is: {average_runtime_hours}h{lefto
ver_minutes}m")
```

```
fh        = open('../downloads/250.imdb', 'r', encoding = 'utf-8')
genreDict = {}

for line in fh:
    if not line.startswith('#'):
        cols    = line.strip().split('|')
        genre   = cols[5].strip()
        glist   = genre.split(',')
        runtime = cols[3]      # length of movie in seconds
        for entry in glist:
            if not entry.lower() in genreDict:
                genreDict[entry.lower()] = [int(runtime)]   # add a list with th
e runtime
            else:
                genreDict[entry.lower()].append(int(runtime))   # append runtime
to existing list
fh.close()
for genre in genreDict:      # loop over the genres in the dictionaries
    average = sum(genreDict[genre])/len(genreDict[genre])  # calculate average l
ength per genre
    hours   = int(average/3600)                                # format seconds
to hours
    minutes = (average - (3600*hours))/60            # format seconds to minute
s
    print('The average length for movies in genre '+genre\
          +' is '+str(hours)+'h'+str(round(minutes))+'min')
```