

# GLE5g11

Class

OP

У цьому репозиторії наданий код, що розробила наша команда (main.py), код наданий ШІ (chatgpt.py), а також допоміжні файли до завдання. Цей файл є письмовим звітом.

```
def read_file(pathname: str, year: int=0) -> list:
    with open(pathname, 'r', encoding='utf-8') as my_file:
        content = my_file.readlines()
        movies = []
        for line in content[1:]:
            data = line.strip().split(';')
            movie_year = int(data[6])
            if movie_year >= year:
                movies.append(data)
        return movies
```

Перша функція зчитує файл з рейтингом фільмів, потім проходиться циклом по всіх фільмах за зазначеним роком (параметр year, оминає перший елемент списку content — маркуючий рядок), форматує стрічку відповідним чином; визначає movie\_year, перевіряє належність до year, додає до результуючого списку.

```
def top_n(data: list, genre: str='', n:int=0) -> list[tuple]:
    outcome = []
    for el in data:
        if genre in el[2]:
            actors = el[5].split(', ')
            result = {}
            for actor in actors:
                result[actor] = max(float(el[8]) for el in data if actor in el[5])
            avg_rating = sum(list(result.values()))/len(list(result.values()))
            avg_rating = float(f'{avg_rating:.1f}')
            outcome.append((el[1], float(el[8]), avg_rating))
    def some_func(outcome):
        return (outcome[2] + outcome[1])/2
    outcome = sorted(outcome)
    outcome = sorted(outcome, key=some_func, reverse=True)
    result = [(el[0], el[2]) for el in outcome]
    return result[:n]
```

Функція проходиться по елементах з data. Якщо елемент(фільм) містить відповідний жанр — формує список акторів, списковим виразом формує словник з максимальними рейтингами акторів, виконує поставлені вимоги щодо обчислення середнього значення, формує список з кортежів: (назва фільму, рейтинг, середній рейтинг акторів). Далі сортує спочатку лексикографічно, потім за низхідним порядком використовуючи допоміжну функцію some\_func для обчислення середнього значення (при сортуванні порівнюються саме ці значення). Формує result — список кортежів (Назва, Середнє значення). Повертає першу n-у кількість відповідних фільмів в рейтингу.

```
def write_file(top: list, file_name: str):
    with open(file_name, 'w+', encoding='utf-8') as file:
        for tup in top:
            tup = [str(el) for el in tup]
            file.write(', '.join(tup) + '\n')
```

Функція записує дані у файл.

Тепер до рішень ШІ.

Промпти:

Промпт для обмеження дозволеного функціоналу:

I am going to get you a list of operators you can use. You nothing much more advanced than the list I give you to solve a few problems I will give you in Python.

#### OPERATORS

\* : multiplication

/ : division (float)

// : division operator (int)

% : remainder operator (e.g. check if remainder is zero = even number)

\*\* : exponentiation

%2 == 0 : check if a number is even

a += 1, a = a + 1

a -= 1

a \*= 1

a /= 1

.2f specifies decimals

#### INPUT ARGUMENTS (feed input to function)

type(

) : *variable type*

len(

) : number of elements

min(

) : *min item in lst*

max(

) max item in lst

sum(\_) sum of items in lst

num: number

#### ON ARGUMENTS (list methods)

var.append(

) *add to end of list*

var.count(

) return number of times item occurs  
var.index(  
) *returns position of item*  
var.pop() *removes and gives back last item so that you can use it somewhere else*  
var.remove(  
) removes first occurrence of item  
var.reverse() reverses order of list  
var.sort() sorts items in increasing order

calling variables

var[0] = first element

var[-1] = last element

comparison operator : 5 <= 18 (is 5 less than or equal to 18?)

bool: TRUE or FALSE, often contains a comparison operator

< : less than

> : greater than

== equal to

!= is not equal to

<= less than or equal to

>= greater than or equal to

boolean operators

and : TRUE if both are TRUE or FALSE if one is FALSE

or : TRUE if any are true or FALSE if both are FALSE

not : TRUE if entire expression is FALSE

in : TRUE if element is in lst

not in : TRUE if element is not in lst

x or y : if x false, then y, else x

x and y : if x false, then x, else y

slicing operator : s[I:j] e.g. link[:4] slice first 4 characters

list = [' ', ' ', ' ', ' '] OR list = list(data) # turning tuples to lists

```

mutable
lst[I] = 'string'

concatenation: s + t      # name = f_name + ', ' + l_name
                s * t, t * s

```

## INPUTS

```

input(' ')
eval(input(' ')) # evaluates string to determine data type
str(int) # convert to string
input(' ' + str(int))

```

## PRINT

```

print(element, element, element) # print cannot be stored in a variable
print(f' {var}') #f-strings insert variable

sep = '; ' : define separator (default is \n), multiple arguments of one print
command
end - specify how you want to end each batch of printed arguments, at end of
ONE print command
:n reserve n spaces

```

## ESCAPE SEQUENCES

```

\n : add a space
\ : next character is a special instruction
\' or \ " : how python uses quotes inside of strings
\t : tab

```

## ONE-WAY STATEMENT (assessing two conditions)

```

if <condition 1>:
    print(' ')
    print('Goodbye.')

if <condition 2>:
    print(' ')
    print('Goodbye.')

```

## TWO-WAY STATEMENT (assessing one condition)

if <condition>:

<block of code>

else: <no condition>

<block of code>

## THREE-WAY STATEMENT

elif: <condition>:

<block of code>

## STRING METHODS

s.capitalize() returns a copy with the first character capitalized (not the same string, a copy)

s.count(target)

s.find

s.lower

s.replace

s.split

s.strip

s.upper #can combine with slice

f strings

f'{number: .2f}' # number of digits after the decimal point, first f for formatting, second for float

## ITERATIONS

for loop : executes a code block for every item of a sequence

For n in variable:

<commands>

range() : iterate over the n to n-1

range(i, n, c) : sequence with step sizes

s[:n] range from first position to n

while <condition>:

<code block until condition becomes false>

<non-indented statement>

While True:

<condition executed over and over>

if <condition>

break

if user\_wish != 'yes':

break

## FUNCTIONS

Def f(x) :                   # definition of function, name you make up, parameter

function needs to work

res = x\*\*2 + 10 # function output

return res               # returns (or prints) function

## SWAPPING VALUES

a, b = b, a

## CHARACTER TRANSLATIONS

str.maketrans(old, new) : make a translate table - use with translate method

s.translate(table) : returns a copy of string with each character mapped corrected from maketrans()

## FILE

import os           import file

os.getcwd()   gets current wd

os.chdir('path') changes current wd

chdir('path')   change path

infile.read(n)   read until n (characters wise), string

infile.read()    read all of n, store in a variable, string

infile.readline(n) read one line (linewise), list

infile.readlines(n) reads and returns a list of strings, with each being a line of the file, goes to the end of file, list

outfile.write(s)

infile.close()

open(path, 'w')

content.count('\n') to read number of lines

Dict methods

d[key] = valued  
d.items()  
d.keys()  
d.pop(key)  
d.update(d2) : new keys or updated keys in 2nd dict get added  
d.values()  
  
d.items()  
d.keys()  
d.values()  
del(d[x])

### Random

random.randrange(start, stop, step): generates a random number in range  
random.uniform(a, b) random range in this interval (a can come up but b never will)

sample(population, k) = returns a list of k randomly selected elements from a sequence or set

Shuffle(s): randomly reshuffles a collection, returns None

### Join

'separator'.join(characters) - concatenates, joining list of characters into string

### Sets

List = []

set\_constructor = set(list)

s == t

s != t

s <= t

s < t

s | t

s & t

s - t

s ^ t

s.add(item)

s.remove(item)

s.clear()



Наступний промпт був копією опису задачі

опис:

To solve this, I'll set up each function as follows:

1. `read_file(pathname: str, year: int=0)` : This function reads and filters the movie database from the specified CSV file path, returning only movies from a given year onward. I'll assume the CSV file is structured as described and that it includes headers (which we will skip).
2. `top_n(data: list, genre: str='', n: int=0)` : This function will:
  - Filter the list of movies by genre (if specified), allowing multiple genres.
  - For each actor in a selected movie, find the highest-rated movie they appear in to calculate the `actor_rating`.
  - Average the `Rating` and `actor_rating` for each movie and sort by this new rating.
  - Return a list of top-rated movies as specified.
3. `write_file(top: list, file_name: str)` : This function will write the movie titles and ratings to a specified file, each on a new line.

Однак gpt використав lambda функцію а також імпортував бібліотеку csv.  
Довелося уточнити.

промпт:

Please try again but do not import anything, do not use key=lambda. Good job though!

## Explanation of Changes

1. **No Imports:** Removed all imports. I read the file line by line, split by commas, and stripped newline characters.

2. **Sorting without Lambda:** Implemented a basic bubble sort to sort the list of tuples based on the `average_rating` in descending order and `title` in ascending order if ratings are tied.
3. **File Operations:** Explicitly opened and closed files without using `with`.

```
# Compute actor_rating as the average of highest ratings per actor
actor_ratings = []
for actor in actors:
    actor_highest_rating = 0
    for m in data:
        if actor in m[5] and float(m[8]) > actor_highest_rating:
            actor_highest_rating = float(m[8])
    actor_ratings.append(actor_highest_rating)
actor_rating = sum(actor_ratings) / len(actor_ratings) if actor_ratings else 0
```

Ключовою відмінністю була реалізація сортування за таким алгоритмом. А також відкриття/закриття файлів через `.open` `.close`

Функціональність коду `gpt` ми тестували через функцію в якій прописали `assert` за доктестами як у коді нашої групи.