

```
elif _operation == "MIRROR_Y":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
elif _operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

#selection at the end -add back the deselected mirror modifier object
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the active ob
#mirror_ob.select = 0
#one = bpy.context.selected_objects[0]
```

LUCENE

PRÁCTICA 1
Gestión de Información en la Web



Álvaro de la Flor Bonilla



```
username;  
password;  
$database;  
$charset;  
  
public function connect()  
{  
    $link = null;  
    if (!$link = mysql_connect($database, $username, $password))  
    {  
        throw new MySQLException("Error al conectar a la base de datos");  
    }  
    mysql_query("SET CHARACTER SET UTF8");  
    mysql_query("SET NAMES UTF8");  
    mysql_query("USE $database");  
    return $link;  
}
```



CONFIGURACIÓN

LENGUAJES



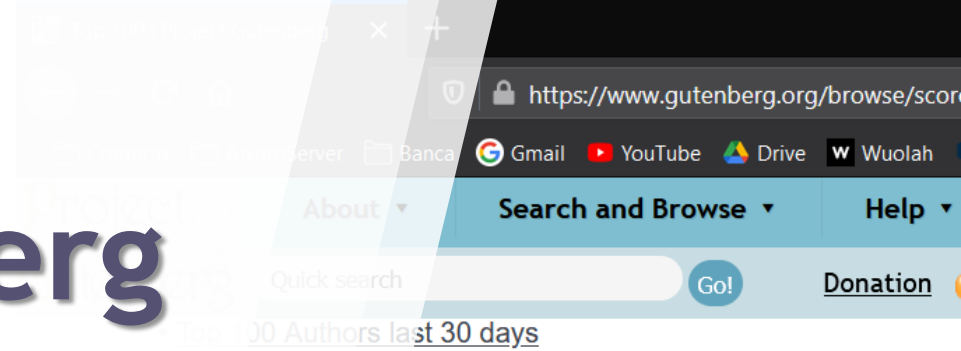
BIBLIOTECA





Project Gutenberg

SCRAPING



Top 100 EBooks yesterday

1. [Frankenstein; Or, The Modern Prometheus](#) by Mary Wollstonecraft (1818)
2. [Pride and Prejudice](#) by Jane Austen (2504)
3. [The Great Gatsby](#) by F. Scott Fitzgerald (1392)
4. [Et dukkehjem. English](#) by Henrik Ibsen (1164)
5. [The Importance of Being Earnest: A Trivial Comedy for Serious People](#) by Oscar Wilde (1895)
6. [A Modest Proposal](#) by Jonathan Swift (1094)
7. [A Tale of Two Cities](#) by Charles Dickens (1086)
8. [Metamorphosis](#) by Franz Kafka (1062)
9. [Alice's Adventures in Wonderland](#) by Lewis Carroll (1008)
10. [The Yellow Wallpaper](#) by Charlotte Perkins Gilman (932)
11. [Moby Dick; Or, The Whale](#) by Herman Melville (906)
12. [The Picture of Dorian Gray](#) by Oscar Wilde (902)
13. [Adventures of Huckleberry Finn](#) by Mark Twain (873)
14. [Dracula](#) by Bram Stoker (856)
15. [Jane Eyre: An Autobiography](#) by Charlotte Brontë (796)
16. [The Adventures of Sherlock Holmes](#) by Arthur Conan Doyle (781)
17. [The Scarlet Letter](#) by Nathaniel Hawthorne (759)
18. [Great Expectations](#) by Charles Dickens (758)
19. [The Strange Case of Dr. Jekyll and Mr. Hyde](#) by Robert Louis Stevenson (757)
20. [Anthem](#) by Ayn Rand (699)
21. [Heart of Darkness](#) by Joseph Conrad (687)
22. [A Christmas Carol in Prose; Being a Ghost Story of Christmas](#) by Charles Dickens (641)
23. [The Awakening, and Selected Short Stories](#) by Kate Chopin (614)
24. [The Hound of the Baskinville](#) by Arthur Conan Doyle (614)

TOP 100 LIBROS

The image shows a Windows file explorer window displaying a directory named 'books' containing 116 HTML files. The files are named sequentially from '1-829.html' to '27-2501.html'. The file explorer is open to the 'books' folder, and the 'Documents' folder is selected in the left sidebar.

Overlaid on the file explorer is a Python IDE window showing a script named 'main.py'. The script uses the 'urllib', 'requests', and 'BeautifulSoup' libraries to fetch and parse data from a website. The output of the script is displayed in the 'Run' console, showing the download progress and the list of books found.

```
1 import urllib
2 from urllib.request import urlopen, urlretrieve
3 from alive_progress import alive_bar
4
5 from bs4 import BeautifulSoup
6 from pip._vendor import requests
7 import re
8
9 url = 'https://www.gutenberg.org/browse/scores/top'
10 u = urlopen(url)
11 try:
12     html = u.read().decode('utf-8')
13 finally:
14     u.close()
15
16 soup = BeautifulSoup(html)
17 lis = soup.find_all("a", href=lambda href: href and "ebooks" in href)
18 lis = list(filter(lambda x: (x.get('href')), lis))
19 lis = list(map(lambda x: (x.get('href').replace('/ebooks/', '')), lis))
20 lis = list(set(lis))
21 lis = list(filter(lambda x: (x is not None and x.isnumeric()), lis))
```

Run: main

Descargando
Posibles libros a descargar: 116
on 0: Book1
on 1: Book2
on 2: Book3
on 3: Book4
on 4: Book5
on 5: Book6
on 6: Book7
on 7: Book8

116 elementos

Unused import statement 'import urllib' 1:1 CRLF UTF-8 4 spaces Python 3.8 (scrapping) main

13:48 25/03/2021



Desarrollo

INDEX, SEARCH



Index

```
public Indexer(String indexDirectoryPath) throws IOException {  
    // Add stop words  
    List<String> words = Execute.readWords();  
    CharArraySet stopSet = StopFilter.makeStopSet(words);  
    // Builds an analyzer with stop words  
    StandardAnalyzer analyzer = new StandardAnalyzer(stopSet);  
    // A Directory provides an abstraction layer for storing a list of files. A directory contains only files (no sub-folder hierarchy)  
    Directory indexDirectory = FSDirectory.open(Paths.get(indexDirectoryPath));  
    // Holds all the configuration that is used to create an IndexWriter  
    IndexWriterConfig iwriter = new IndexWriterConfig(analyzer);  
    // An IndexWriter creates and maintains an index  
    writer = new IndexWriter(indexDirectory, iwriter);  
}
```

Search

```
// Using umbral
if (Constants.useUmbral) {
    scoreDocs = scoreDocs.stream().filter(x -> calculateIndividualScore(x.score, searchQuery, maxScore, minScore)).collect(Collectors.toList());
}
```

```
public static Boolean calculateIndividualScore(Float score, String searchQuery, Float maxScore, Float minScore) {
    Boolean res = false;
    Integer words = Arrays.stream(searchQuery.split(" ")).filter(x -> !x.equals(" ")).collect(Collectors.toList()).size();

    Float scoreNormalize = ((score + words * 0.47f) * 10) / maxScore;

    if ((1 - normalize(scoreNormalize, minScore)) < Constants.umbral) {
        res = true;
    }

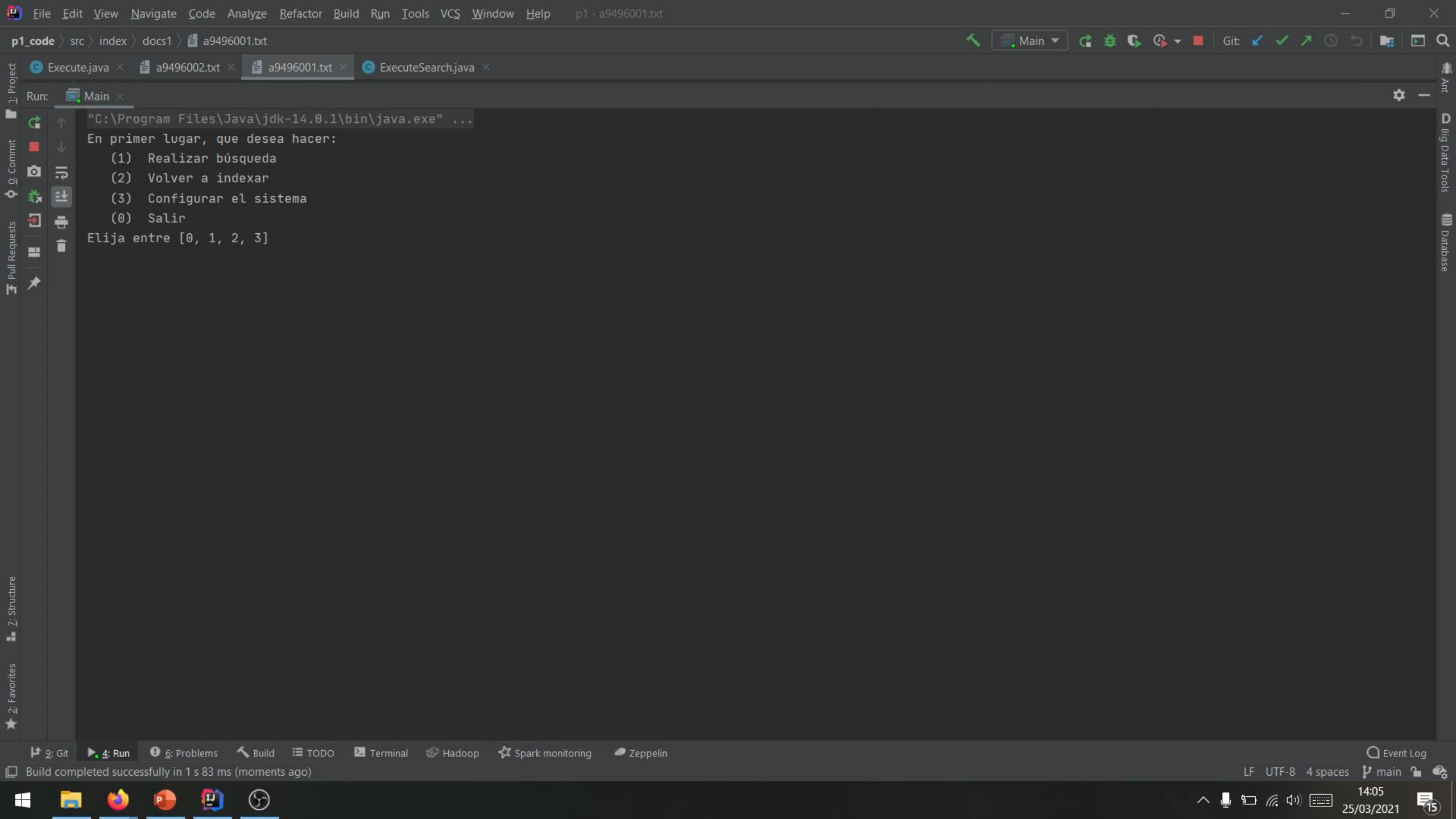
    return res;
}

public static Float normalize(Float val, Float min) { return (val - min) / (10 - min); }
```




Desarrollo

DEMO



1. w
2. w
3. w
w
w
w
o
o
o

¡GRACIAS!

¿ALGUNA DUDA?

IV
DAS

Video module

H1-Headline

Menu

|||

|||