**Trabajo Práctico N° 1**

**Introducción**



**Grupo N°01**

**Integrantes:**

| **N y A** | **DNI** | **LU** | **Correo** |
| --- | --- | --- | --- |
| Martinez, Maximiliano Joaquin | 4071080 | 8005 | 40710800@fi.unju.edu.ar |
| González, Aldo Gabriel | 43276932 | 8517 | 43276932@fi.unju.edu.ar |
| Orquera, Dario Joaquin | 33482732 | 5573 | 33482732@fi.unju.edu.ar |

**Recuperación Avanzada de la Información - 2025**

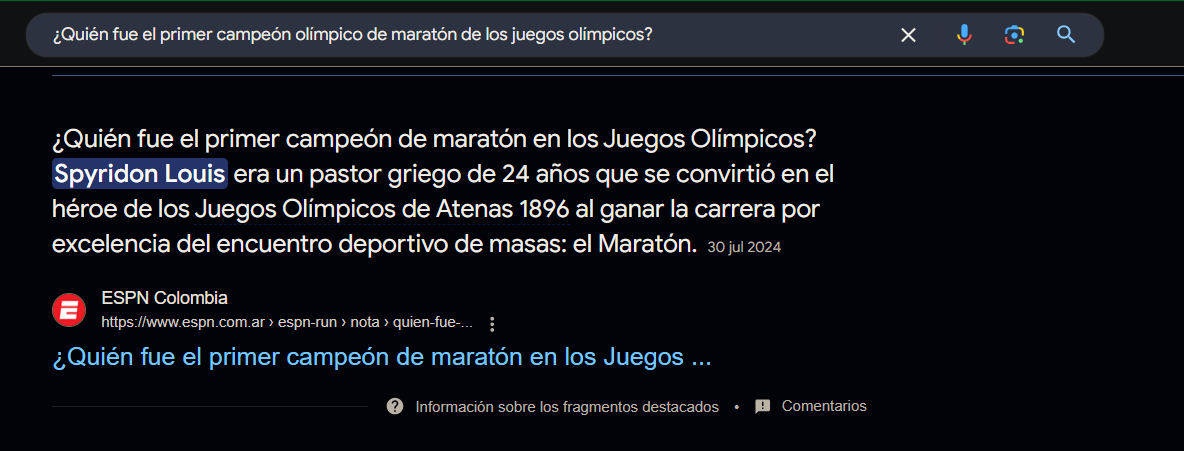
**Profesor: J. Federico Medrano**

Fecha de Presentación: 15-04-2025

**Actividades**

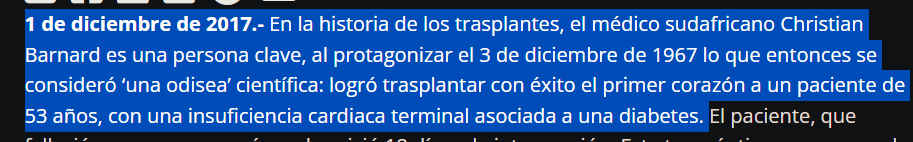
**1) Utilizando como recurso el motor de búsqueda Google trate de satisfacer las siguientes necesidades de información. Indique en cada caso la cantidad de consultas que tuvo que realizar para llegar al objetivo (si llegó), cuántas referencias de la respuesta revisó en cada caso y en cuál encontró lo buscado (si es necesario, puede escribir las consultas en inglés). Suponga que usted no sabe la respuesta a ninguna de las preguntas.**

a) ¿Quién fue el primer campeón olímpico de maratón de los juegos olímpicos?



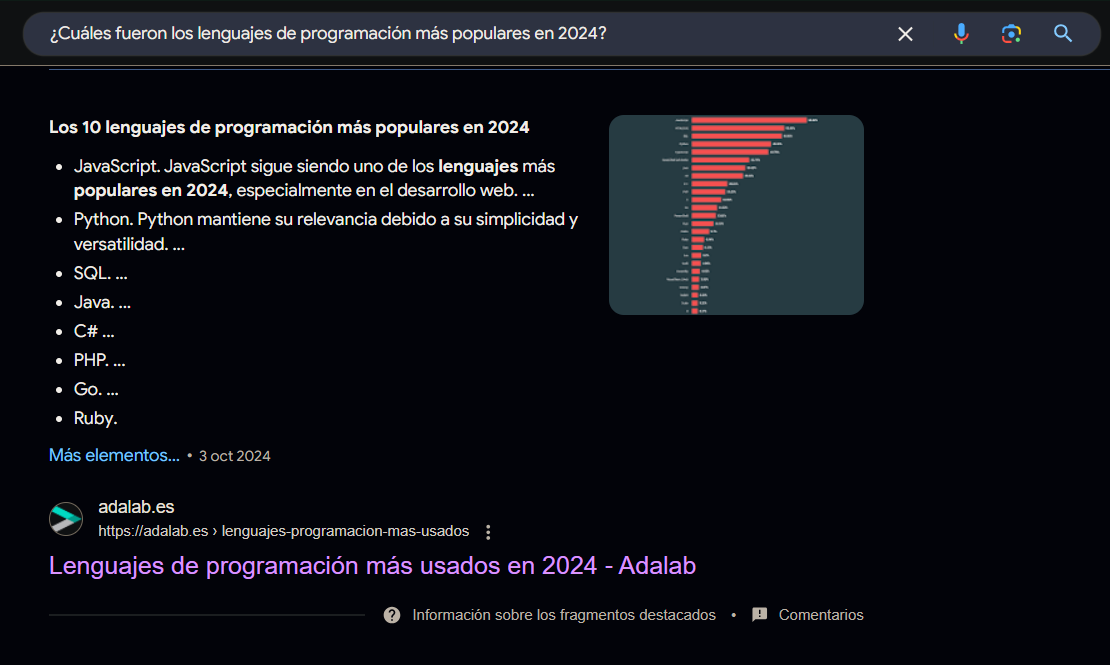
| **Pregunta** | ¿Quién fue el primer campeón olímpico de maratón de los juegos olímpicos? |
| --- | --- |
| **Cantidad de Consultas** | Se ha realizado una consulta |
| **¿Objetivo de busqueda cumplido?** | Si |
| **Cantidad de referencias de la respuesta revisadas** | Se ha revisado una referencia |
| **Referencia donde se encontró lo buscado** | [¿Quién fue el primer campeón de maratón en los Juegos Olímpicos? - ESPN](https://www.espn.com.ar/espn-run/nota/_/id/13916132/quien-fue-el-primer-campeon-de-maraton-en-los-juegos-olimpicos) |
| **Comentarios** | Google obviando lo que es la implementación de Gemini en las búsquedas, tiene como prioridad los fragmentos destacados que nos muestra información verídica sobre nuestra consulta, sin necesidad de ingresar a la pagina de donde se obtiene información |

b) ¿Cuándo se realizó el primer trasplante de corazón?



| **Pregunta** | ¿Cuándo se realizó el primer trasplante de corazón? |
| --- | --- |
| **Cantidad de Consultas** | Se ha realizado una consulta |
| **¿Objetivo de búsqueda cumplido?** | Si |
| **Cantidad de referencias de la respuesta revisadas** | Se ha revisado dos referencias (Wikipedia y Ministerio de Sanidad) |
| **Referencia donde se encontró lo buscado** | [Ministerio de Sanidad - Prensa y comunicación - Noticias](https://www.sanidad.gob.es/gabinete/notasPrensa.do?id=4259) |
| **Comentarios** | Google nos ha revelado que el primer trasplante fue realizado en 1974 cuando ese fue realizado de forma heterotópica y no es la información que se busca. |

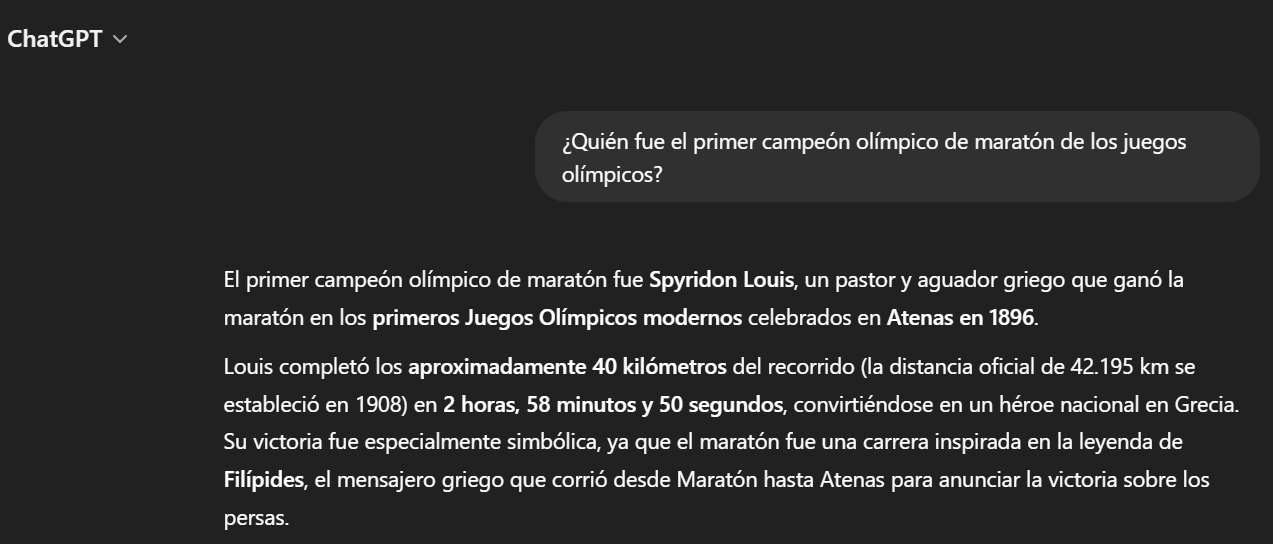
c) ¿Cuáles fueron los lenguajes de programación más populares el año pasado?

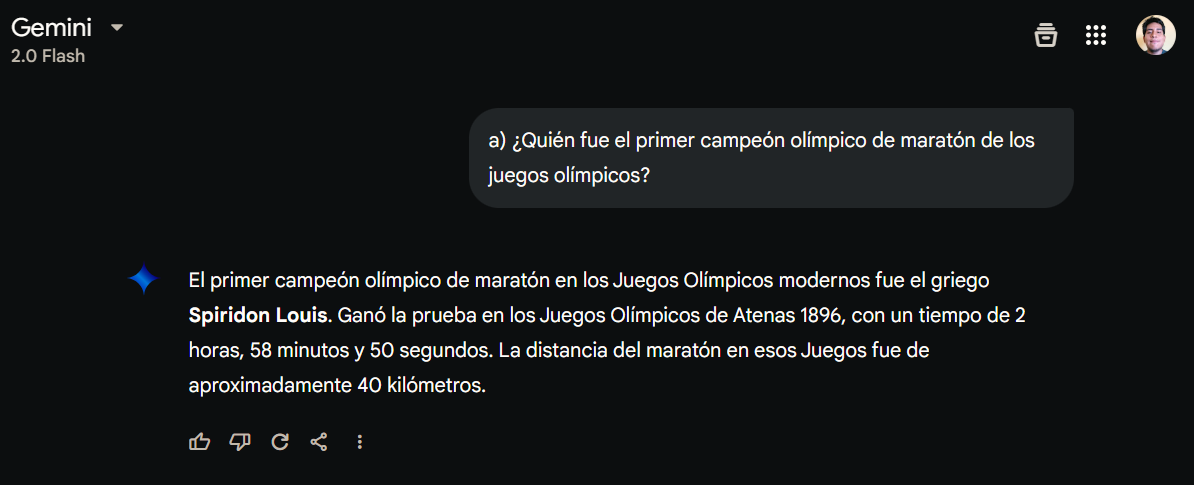


| **Pregunta** | ¿Cuáles fueron los lenguajes de programación más populares el año pasado? |
| --- | --- |
| **Cantidad de Consultas** | Se ha realizado dos consultas:   * ¿Cuáles fueron los lenguajes de programación más populares el año pasado? * ¿Cuáles fueron los lenguajes de programación más populares en 2024? |
| **¿Objetivo de búsqueda cumplido?** | Si |
| **Cantidad de referencias de la respuesta revisadas** | Se ha revisado seis referencias (cinco con la primera consulta y una con la segunda) |
| **Referencia donde se encontró lo buscado** | [Lenguajes de programación más usados en la actualidad](https://adalab.es/blog/lenguajes-programacion-mas-usados/) |
| **Comentarios** | Google nos ha dado información imprecisa con la primera consulta, ajustando esta y dándole una fecha las páginas encontradas fueron más acertadas. |

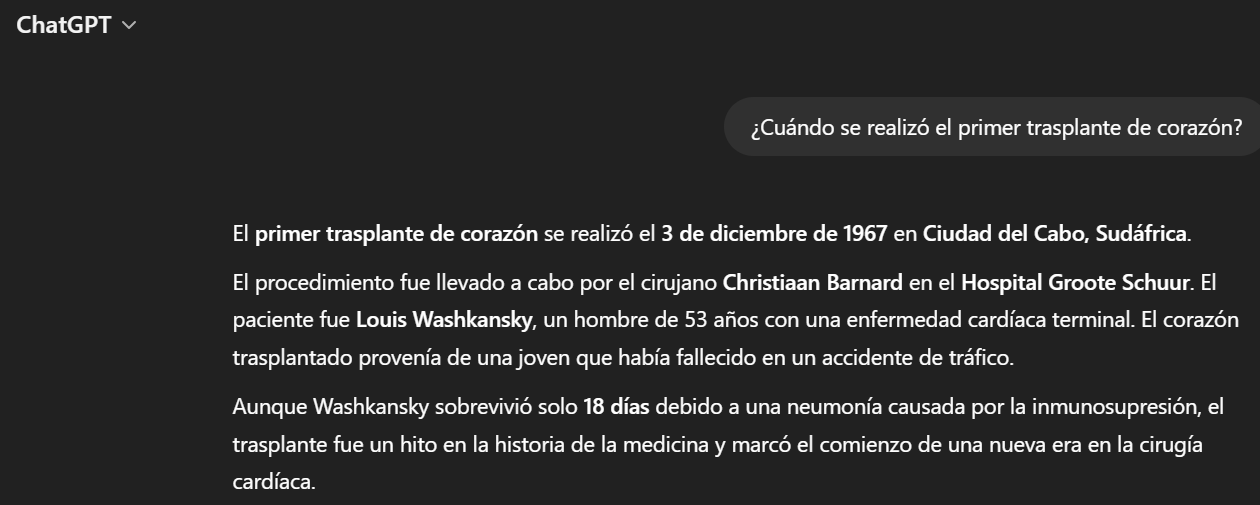
**2) Ahora realice las mismas consultas del punto 1) pero empleando ChatGPT y Gemini.**

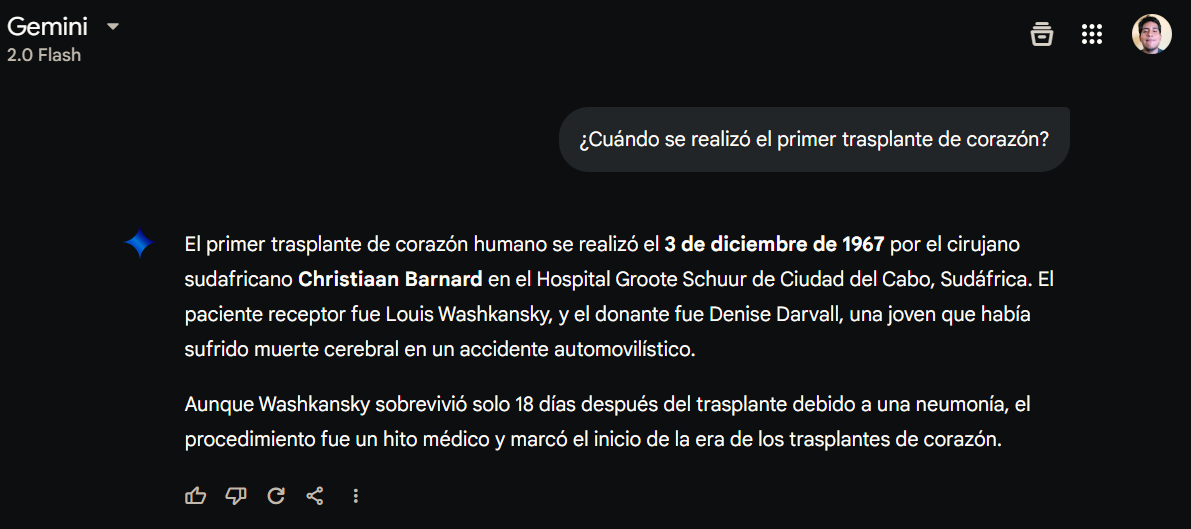
a) ¿Quién fue el primer campeón olímpico de maratón de los juegos olímpicos?



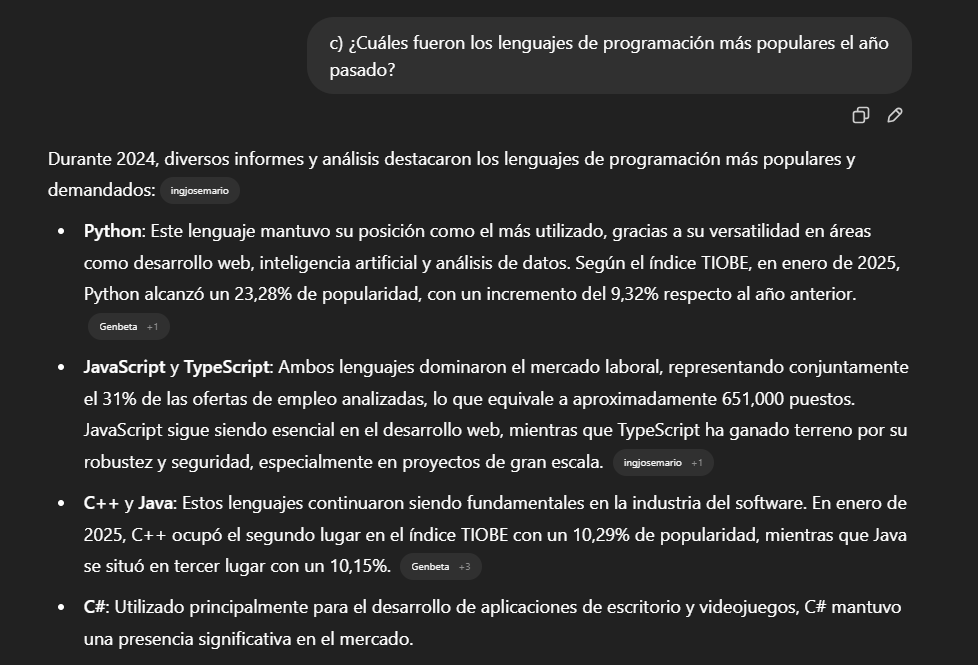


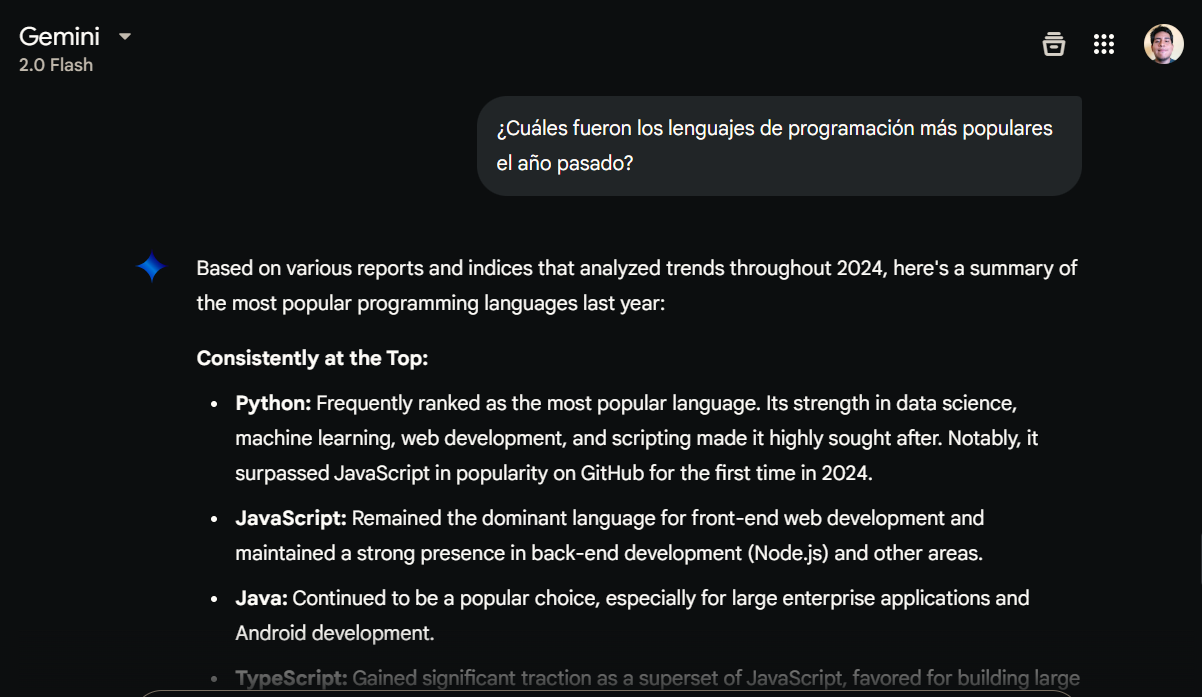
b) ¿Cuándo se realizó el primer trasplante de corazón?





c) ¿Cuáles fueron los lenguajes de programación más populares el año pasado?





**3) Cree una aplicación que “indexe” los siguientes 5 documentos. Debe formar una colección con todas las palabras, asegúrese de eliminar las stop-words. La aplicación debe leer los documentos .PDF previamente descargados a una ubicación del disco rígido y armar un fichero invertido con frecuencia y por separado un fichero invertido posicional. Con la última representación, genere 4 consultas distintas y muestre los resultados.**

<https://sedici.unlp.edu.ar/handle/10915/176466>

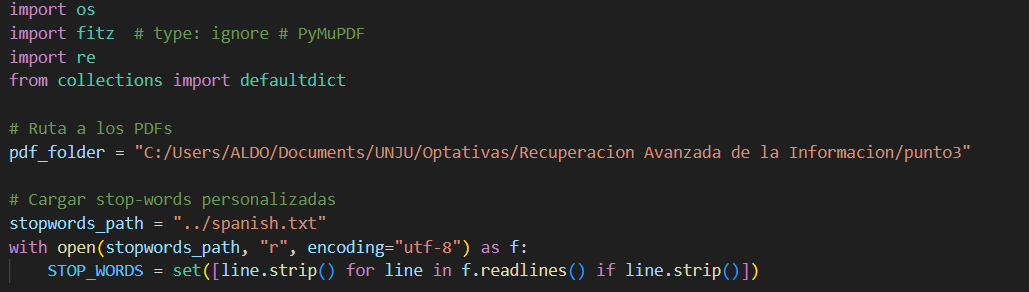
<https://sedici.unlp.edu.ar/handle/10915/176745>

<https://sedici.unlp.edu.ar/handle/10915/177333>

<https://sedici.unlp.edu.ar/handle/10915/177453>

<https://sedici.unlp.edu.ar/handle/10915/176292>

El código de Python que realizamos procesa todos los archivos PDF de una carpeta y construye dos tipos de índices que permiten analizar y buscar palabras dentro de los textos.

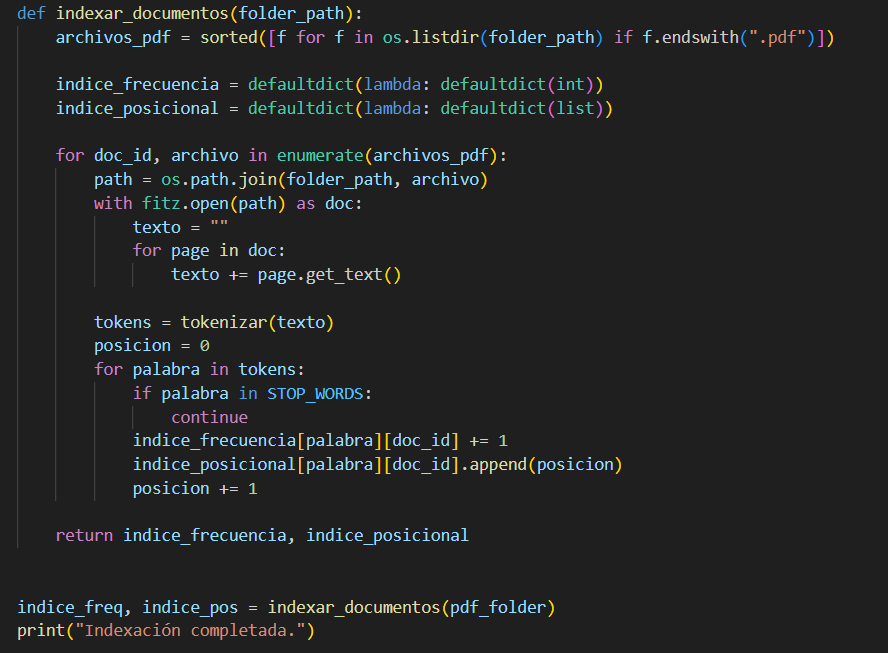


Primero, carga una lista personalizada de stop-words (palabras comunes como "el", "la", "de", etc.) que serán ignoradas en el análisis. Luego, para cada PDF:

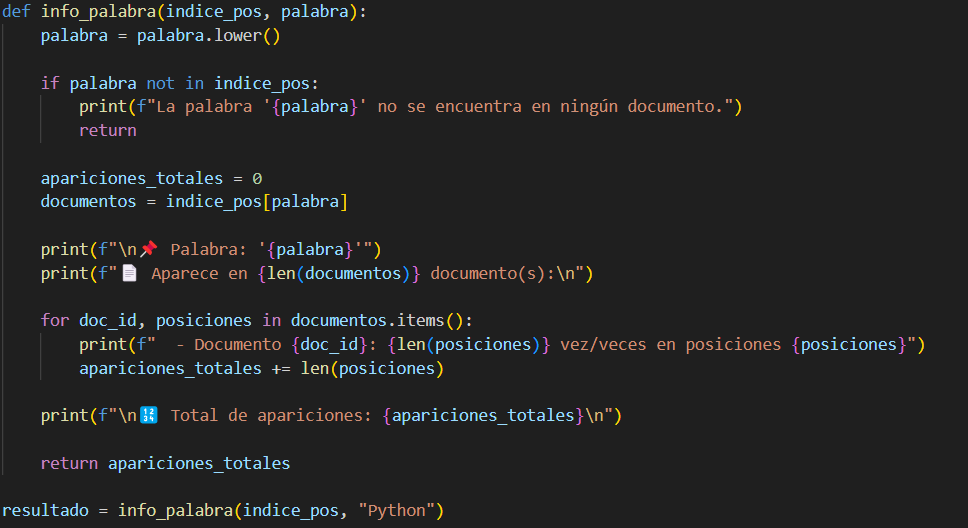
1. Extrae el texto completo usando la librería PyMuPDF.
2. Limpia el texto: lo convierte a minúsculas y elimina los signos de puntuación.
3. Tokeniza el texto: lo divide en palabras individuales.
4. Filtra las stop-words y luego:

* Guarda cuántas veces aparece cada palabra en cada documento (índice de frecuencia).
* Guarda en qué posiciones del texto aparece cada palabra (índice posicional).

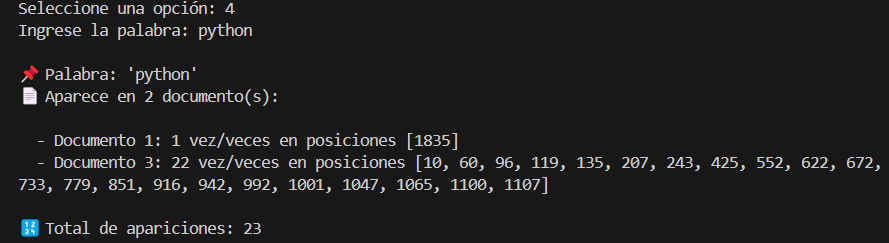
Los resultados se almacenan en estructuras de diccionario para poder consultarlos después fácilmente, por ejemplo, para hacer búsquedas por palabra o análisis de contenido. Al final, imprime un mensaje indicando que la indexación terminó correctamente.

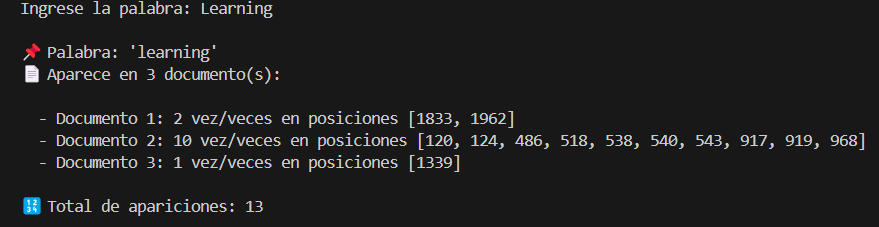


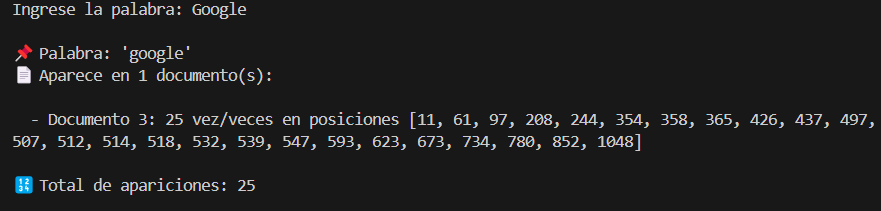
La función ***“info\_palabra”*** devuelve los documentos donde se encuentra una determinada palabra ingresada por el usuario, cuantas veces por documento aparece esa palabra, la posición donde se encuentra esa palabra en los distintos documentos y la cantidad de veces que aparece en todos los documentos.

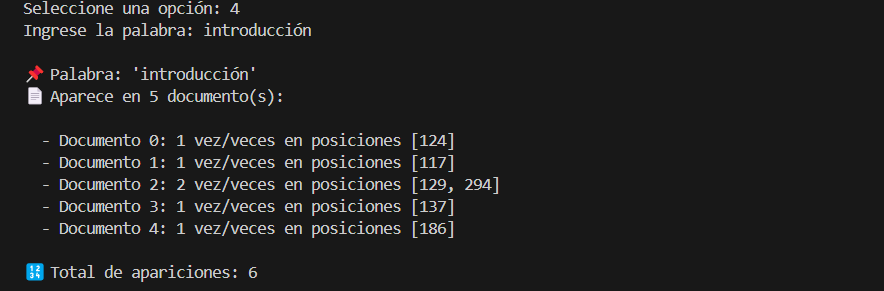


Las consultas que realizamos con la función ***“info\_palabra”*** fueron:

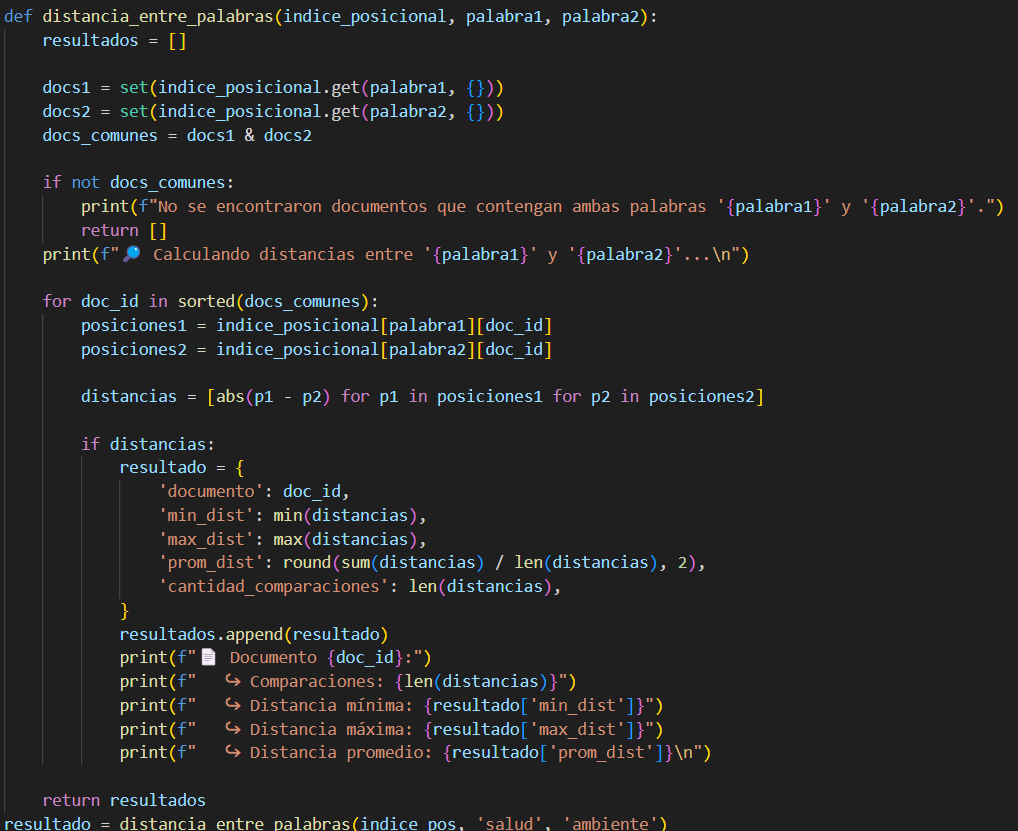






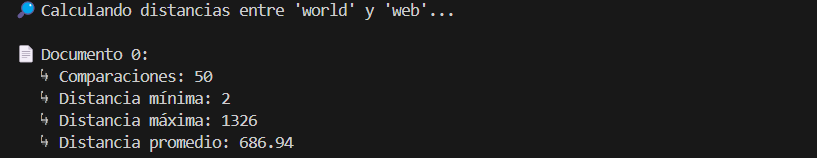


La función “***distancia\_entre\_palabras***” devuelve la distancia mínima, máxima y el promedio entre dos palabras que se encuentren en el mismo documento, se obtiene la cantidad de veces que aparecen las palabras, las posiciones en donde estas se encuentran y a partir de eso se calcula las distancias para obtener los distintos valores de distancias.

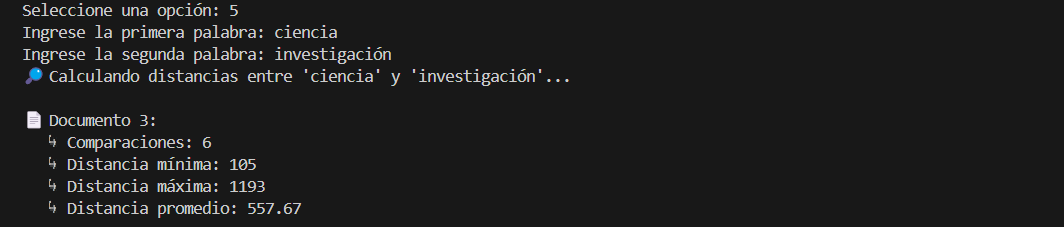


Las consultas que realizamos con estas función fueron:

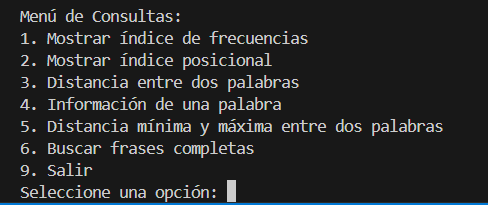
Por la distancia en la frase “World Wide Web”



En esta consulta se eligió dos palabras que pueden estar relacionadas como “ciencia” e “investigación”



Para que el usuario pueda acceder a las distintas funciones realizadas se utilizó un menú de consultas a los pdf indexados.



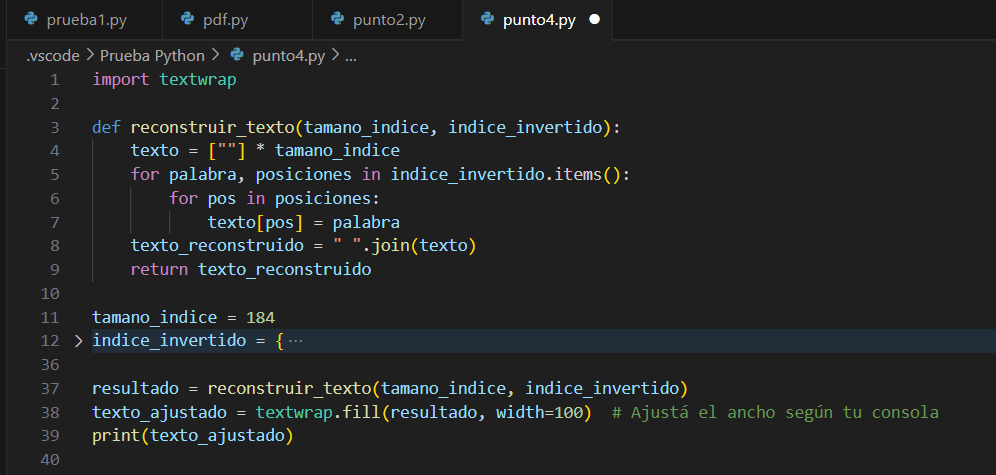
**4) El siguiente es un resumen que se encuentra construido con un índice invertido, se indica el tamaño del mismo y cada palabra posee entre corchetes la posición que ocupa dentro del texto. Cree una aplicación que permita recibir estos dos parámetros, el tamaño del índice y el índice invertido, y devuelva el resumen completo. Cree las estructuras de datos que considere necesario tanto para la reconstrucción del texto como para el pasaje de parámetros.**

**Tamaño del Índice:** 184

**Índice Invertido:**

{"COVID-19,":[0],"caused":[1],"by":[2],"the":[3,16,26,31,36,54,62,70,78,85,97,110,117,125,146,162,173,178],"coronavirus":[4],"SARSCoV2,":[5],"continues":[6],"to":[7,58,73,77,129],"pose":[8],"a":[9,170],"global":[10,37],"threat":[11],"The":[13],"consequences":[14],"from":[15,116],"pandemic":[17],"are":[18],"devastating,":[19],"not":[20,141],"only":[21],"for":[22,35,124,172],"human":[23],"health":[24,28],"and":[25,43,82,103,132],"national":[27],"systems":[29],"throughout":[30],"world,":[32],"but":[33],"also":[34,107],"economy":[38],"Bats,":[40],"pangolins,":[41],"muskrats":[42],"other":[44,136],"wild":[45],"animals":[46,128,137],"have":[47,74],"been":[48,122],"implicated":[49],"as":[50,87],"possible":[51],"hosts":[52],"of":[53,65,84,93,100,112,127,148,164,175],"virus":[55],"According":[57],"recent":[59],"research":[60],"data,":[61],"increased":[63],"concentration":[64],"particulate":[66],"air":[67,94],"matter":[68],"in":[69,145,152,177],"atmosphere":[71],"appears":[72,108],"contributed":[75],"significantly":[76],"spread":[79],"(airborne":[80],"transmission)":[81],"aggressiveness":[83],"disease,":[86],"large":[88],"cities":[89],"with":[90,135],"high":[91],"levels":[92],"pollution":[95],"showed":[96],"highest":[98],"numbers":[99],"SARSCoV2":[101],"cases":[102],"deaths":[104],"It":[106,157],"that":[109,138,161],"collapse":[111],"natural":[113],"ecosystems":[114],"resulting":[115,144],"everevolving":[118],"climate":[119],"change":[120],"has":[121,181],"crucial":[123],"migration":[126],"new":[130,149],"areas":[131],"their":[133],"interaction":[134],"they":[139],"would":[140],"normally":[142],"encounter,":[143],"emergence":[147],"pathogens,":[150],"which":[151,167],"turn":[153],"infected":[154],"humans":[155],"is":[158],"now":[159],"clear":[160],"era":[163],"“climate":[165],"medicine”":[166],"will":[168],"be":[169],"cornerstone":[171],"practice":[174],"medicine":[176],"21st":[179],"century,":[180],"already":[182],"begun" :[183]}

El código realizado para resolver el problema es el siguiente:



Se creó una función ***“reconstruir\_texto”*** con el tamaño del índice invertido como parámetros, en esta función se crea una lista denominada **“*texto*”** con el tamaño del índice, con el uso de los bucles ***for*** colocaremos cada palabra del índice en su respectivo espacio en la lista, se unen las palabras agregando un espacio con el comando join a una variable llamada ***“texto\_reconstruido”*** y se devuelve el texto. El último comando ***“textwrap”*** sirve para ajustar el texto a la pantalla.

La salida que nos devolvió el código en la terminal fue la siguiente:

