400 EJERCICIOS DE PYTHON DESDE NIVEL BÁSICO A AVANZADO CON APLICACIONES EN I.A.

Aquí veremos ejercicios resueltos con su explicación .

Terminando esta guía estarás en condiciones de aprender Inteligencia Artificial y sus aplicaciones

Mucha suerte en su camino hacia la excelencia profesional !!!

Gustavo Zaracho

Variables , operadores , expresiones , condicionales y bucles

1) Hacer un programa donde se pida un nombre por teclado y se imprima “Hola ..el_nombre_ingresado”
2) Hacer un programa que solicite por teclado dos número y muestre la suma , la resta ,la multiplicación y la división de esos números
3) Hacer un programa que solicite una edad y determine si es mayor de edad.
4) Hacer un programa que solicite una edad e imprima “Es mayor” si es mayor de edad , sino que imprima “Es menor”.
5) Hacer un programa que solicite un color por teclado e imprima “Puede pasar “ si el color ingresado es verde , “Precaución “ si es amarillo , “No pasar “ si es rojo o “Color inválido” si es cualquier otro.
6) Hacer un programa que cuente desde el 1 al 100 y los imprima uno a uno.
7) Hacer un programa que cuente del 1 al 100 inclusive e imprima sólo los números pares

8) Hacer un programa que cuente del 1 al 100 inclusive e imprima los números que son divisibles por 2 y por 5.

9) Hacer un programa que imprima una tabla de multiplicar del 2 al 9 . Cada uno debe mostrar sus valores multiplicados del 1 al 9 inclusive

10) Hacer un programa que muestre el siguiente dibujo.

```
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
```

11) Hacer un programa donde se muestre el siguiente dibujo

```
* * * * * * * * * *
*                 *
*                 *
*                 *
* * * * * * * * * *
```

12)Hacer un programa que muestre el siguiente dibujo

```
*
*  *
*  *  *
*  *  *  *
*  *  *  *  *
```

13) Idem anterior con este dibujo

```
*  *  *  *  *
*  *  *  *
*  *  *
*  *
*
```

Funciones

14)Crea un función EscribirCentrado, que reciba como parámetro un texto y lo escriba centrado en pantalla (suponiendo una anchura de 80 columnas;

pista: deberás escribir 40 - longitud/2 espacios antes del texto). Además subraya el mensaje utilizando el carácter =.

15)Crea un programa que pida dos número enteros al usuario y diga si alguno de ellos es múltiplo del otro. Crea una función EsMultiplo que reciba

los dos números, y devuelve si el primero es múltiplo del segundo.

16)Crear una función que calcule la temperatura media de un día a partir de la temperatura máxima y mínima.

Crear un programa principal, que utilizando la función anterior, vaya pidiendo la temperatura máxima y mínima de cada día y vaya mostrando la media.

El programa pedirá el número de días que se van a introducir.

17)Crea un función “ConvertirEspaciado”, que reciba como parámetro un texto y devuelve una cadena con un espacio adicional tras cada letra.

Por ejemplo, “Hola, tú” devolverá “H o l a , t ú “. Crea un programa principal donde se use dicha función.

18)Crea una función “calcularMaxMin” que recibe una lista con valores numéricos y devuelve el valor máximo y el mínimo.

Crea un programa que pida números por teclado y muestre el máximo y el mínimo, utilizando la función anterior.

19)Diseñar una función que calcule el área y el perímetro de una circunferencia. Utiliza dicha función en un programa principal que lea el radio de

una circunferencia y muestre su área y perímetro.

20)Crear una subrutina llamada “Login”, que recibe un nombre de usuario y una contraseña y te devuelve Verdadero si el nombre de usuario es “usuario1”

y la contraseña es “asdasd”. Además recibe el número de intentos que se ha intentado hacer login y si no se ha podido hacer login incremente este valor.

Crear un programa principal donde se pida un nombre de usuario y una contraseña y se intente hacer login, solamente tenemos tres oportunidades

para intentarlo.

21)Crear una función recursiva que permita calcular el factorial de un número. Realiza un programa principal donde se lea un entero y se muestre el resultado

del factorial.

22)Crear una función que calcule el MCD de dos número por el método de Euclides. El método de Euclides es el siguiente:

Se divide el número mayor entre el menor.

Si la división es exacta, el divisor es el MCD.

Si la división no es exacta, dividimos el divisor entre el resto obtenido y se continúa de esta forma hasta obtener una división exacta, siendo el último divisor el MCD.

Crea un programa principal que lea dos números enteros y muestre el MCD.

23 Escribir dos funciones que permitan calcular:

La cantidad de segundos en un tiempo dado en horas, minutos y segundos.

La cantidad de horas, minutos y segundos de un tiempo dado en segundos.

Escribe un programa principal con un menú donde se pueda elegir la opción de convertir a segundos, convertir a horas,minutos y segundos o salir del programa.

24)El día juliano correspondiente a una fecha es un número entero que indica los días que han transcurrido desde el 1 de enero del año indicado.

Queremos crear un programa principal que al introducir una fecha nos diga el día juliano que corresponde. Para ello podemos hacer las siguientes subrutinas:

LeerFecha: Nos permite leer por teclado una fecha (día, mes y año).

DiasDelMes: Recibe un mes y un año y nos dice los días de ese mes en ese año.

EsBisiesto: Recibe un año y nos dice si es bisiesto.

Calcular_Dia_Juliano: recibe una fecha y nos devuelve el día juliano.

25) Vamos a mejorar el ejercicio anterior haciendo una función para validar la fecha. De tal forma que al leer una fecha se asegura que es válida.

26)Queremos crear un programa que trabaje con fracciones a/b. Para representar una fracción vamos a utilizar dos enteros: numerador y denominador.

27) Definir una función max() que tome como argumento dos números y devuelva el mayor de ellos.(Es cierto que python tiene una función max() incorporada, pero hacerla nosotros mismos es un muy buen ejercicio.

28) Definir una función max_de_tres(), que tome tres números como argumentos y devuelva el mayor de ellos.

29) Definir una función que calcule la longitud de una lista o una cadena dada. (Es cierto que Python tiene la función len() incorporada, pero escribirla por nosotros mismos resulta un muy buen ejercicio.

30)- Escribir una función que tome un carácter y devuelva True si es una vocal, de lo contrario devuelve False.

31) Escribir una funcion sum() y una función multip() que sumen y multipliquen respectivamente todos los números de una lista. Por ejemplo: sum([1,2,3,4]) debería devolver 10 y multip([1,2,3,4])debería devolver 24.

32) Definir una función inversa() que calcule la inversión de una cadena. Por ejemplo la cadena "estoy probando" debería devolver la cadena "odnaborp yotse"

33) Definir una función es_palindromo() que reconoce palíndromos (es decir, palabras que tienen el mismo aspecto escritas invertidas), ejemplo: es_palindromo ("radar") tendría que devolver True.

34)Definir una función superposicion() que tome dos listas y devuelva True si tienen al menos 1 miembro en común o devuelva False de lo contrario. Escribir la función usando el bucle for anidado.

35) Definir una función generar_n_caracteres() que tome un entero n y devuelva el carácter multiplicado por n. Por ejemplo: generar_n_caracteres(5, "x") debería devolver "xxxxx".

36) Definir un histograma procedimiento() que tome una lista de números enteros e imprima un histograma en la pantalla. Ejemplo: procedimiento([4, 9, 7]) debería imprimir lo siguiente:

****

****

****

Listas , tuplas , conjuntos , diccionarios

~~37)~~

~~A) Solicitar al usuario que ingrese números, los cuales se guardarán en una lista. Finalizar al ingresar el número 0, el cual no debe guardarse.~~

~~B) A continuación, solicitar al usuario que ingrese un número y, si el número está en la lista, eliminar su primera ocurrencia. Mostrar un mensaje si no es posible eliminar.~~

~~C) Recorrer la lista para imprimir la sumatoria de todos los elementos.~~

~~D) Solicitar al usuario otro número y crear una lista con los elementos de la lista original que sean menores que el número dado. Imprimir esta nueva lista, iterando por ella.~~

~~E) Generar e imprimir una nueva lista que contenga como elementos a tuplas de dos elementos, cada una compuesta por un número de la lista original y la cantidad de veces que aparece en ella. Por ejemplo, si la lista original es [5,16,2,5,57,5,2] la nueva lista contendrá: [(5,3), (16,1), (2,2), (57,1)]~~

~~38)~~

~~Escribir un programa que permita procesar datos de pasajeros de viaje en una lista de tuplas con la siguiente forma: (nombre, dni, destino). Ejemplo: [("Manuel Juarez", 19823451, "Liverpool"), ("Silvana Paredes", 22709128, "Buenos Aires"), ("Rosa Ortiz", 15123978, "Glasgow"), ("Luciana Hernandez", 38981374, "Lisboa")] Además, en otra lista de tuplas se almacenan los datos de cada ciudad y el país al que pertenecen. Ejemplo: [("Buenos Aires","Argentina"), ("Glasgow","Escocia"), ("Lisboa", "Portugal"), ("Liverpool","Inglaterra"),~~

("Madrid","España")] Hacer un menú iterativo que permita al usuario realizar las siguientes operaciones:

-Agregar pasajeros a la lista de viajeros.

-Agregar ciudades a la lista de ciudades.

-Dado el DNI de un pasajero, ver a qué ciudad viaja.

-Dada una ciudad, mostrar la cantidad de pasajeros que viajan a esa ciudad.

-Dado el DNI de un pasajero, ver a qué país viaja.

-Dado un país, mostrar cuántos pasajeros viajan a ese país.

-Salir del programa.

39)

Solicitar al usuario que ingrese los nombres de pila de los alumnos de nivel primario de una escuela, finalizando al ingresar “x”. A continuación, solicitar que ingrese los nombres de los alumnos de nivel secundario, finalizando al ingresar “x”.

- Informar los nombres de todos los alumnos de nivel primario y los de nivel secundario, sin repeticiones.<>

- Informar qué nombres se repiten entre los alumnos de nivel primario y secundario.

-Informar qué nombres de nivel primario no se repiten en los de nivel secundario.

40)

Suponer una lista con datos de las compras hechas por clientes de una empresa a lo largo de un mes, la cual contiene tuplas con información de cada venta: (cliente, día del mes, monto, domicilio del cliente). Ejemplo:

[("Nuria Costa", 5, 12780.78, "Calle Las Flores 355"), ("Jorge Russo", 7, 699, "Mirasol 218"), ("Nuria Costa", 7, 532.90, "Calle Las Flores 355"), ("Julián Rodriguez", 12, 5715.99, "La Mancha 761"), ("Jorge Russo", 15, 958, "Mirasol 218")]

Escribir una función que reciba como parámetro una lista con el formato mencionado anteriormente y retorne los domicilios de cada cliente al cual se le debe enviar una factura de compra. Notar que cada cliente puede haber hecho más de una compra en el mes, por lo que la función debe retornar una estructura que contenga cada domicilio una sola vez.

41

Escribir un programa que procese strings ingresados por el usuario. La lectura finaliza cuando se hayan procesado 50 strings. Al finalizar, informar la cantidad total de ocurrencias de cada carácter, en todos los strings ingresados. Ejemplo: "r":5, "%":3, "a":8, "9":1.

¿Cómo se podrían informar las ocurrencias de las letras del alfabeto únicamente, incluyendo el valor 0 para las letras que no aparecieron?

42)

Crear un programa para gestionar datos de los socios de un club, permitiendo:

-Cargar información de los socios en un diccionario para acceder por número de socio. Los datos a almacenar son: número, nombre y apellido, fecha de ingreso (ddmmaaaa), cuota al día (s/n). El programa debe iniciar con los datos de los socios fundadores ya cargados:

Socio nº1, Amanda Núñez, ingresó: 17/03/2009, cuota al día.

Socio nº2, Bárbara Molina, ingresó: 17/03/2009, cuota al día.

Socio nº3, Lautaro Campos, ingresó: 17/03/2009, cuota al día.

-Informar cantidad de socios del club.

-Solicitar al usuario el número de un socio y registrar que ha pagado todas las cuotas adeudadas.

-Modificar la fecha de ingreso de todos los socios ingresados el 13/03/2018, para indicar que en realidad ingresaron el 14/03/2018.

-Solicitar el nombre y apellido de un socio y darlo de baja (eliminarlo del listado).

-Imprimir el listado de socios completo.

43)Contar Elementos: Escribe una función que cuente cuántas veces aparece cada elemento en una lista y lo almacene en un diccionario.

44)Calcular Promedio: Dado un diccionario de estudiantes y sus calificaciones, calcula el promedio de calificaciones.

45)Filtrar por Valor: Escribe una función que filtre un diccionario de estudiantes para obtener solo aquellos que hayan aproba

46)Unir Diccionarios: Combina dos diccionarios en uno solo.

~~**47)Contar Palabras:** Dada una cadena de texto, crea un diccionario que cuente cuántas veces aparece cada palabra.~~

~~**48)Eliminar Duplicados:** Elimina los elementos duplicados de una lista y almacénalos en un diccionario.~~

~~**49)Valor Mínimo y Máximo:** Encuentra el valor mínimo y máximo en un diccionario de números.~~

~~**50)Eliminar Claves:** Escribe una función que elimine las claves de un diccionario si su valor es mayor que un cierto umbral.~~

~~**51)Invertir Diccionario:** Crea una función que invierta las claves y los valores en un diccionario.~~

~~**52)Calcular Histograma:** Crea un histograma de letras a partir de una cadena de texto.~~

53)Diccionario Anidado: Trabaja con un diccionario anidado que represente datos de una tienda, como productos y precios.

54)Contar Vocales: Cuenta cuántas vocales hay en una cadena de texto y almacena el resultado en un diccionario.

55)Ordenar Diccionario: Ordena un diccionario por sus claves o valores.

56)Diccionario Vacío: Verifica si un diccionario está vacío o no.

57)Buscar Clave: Escribe una función que busque una clave en un diccionario y devuelva su valor o un mensaje de error si no se encuentra.

58)Eliminar Clave: Elimina una clave específica de un diccionario si existe.

59)Diccionario de Contactos: Crea un diccionario de contactos con nombres y números de teléfono.

60)Unir Listas en Diccionario: Convierte dos listas en un diccionario, utilizando una como claves y la otra como valores.

61Diccionario de Frequencias: Dada una lista de números, crea un diccionario que muestre cuántas veces aparece cada número.

62)Diccionario de Traducción: Crea un diccionario que traduzca palabras de un idioma a otro

63)Registro de Ventas: Crea un programa que registre las ventas diarias de una tienda utilizando un diccionario. El diccionario debe contener las fechas como claves y las ventas totales como valores. Utiliza una lista para almacenar los detalles de cada venta, que incluyen el producto vendido, la cantidad y el precio unitario.

64)Puntuaciones de Jugadores: Diseña un juego que registre las puntuaciones de múltiples jugadores. Crea un diccionario que almacene el nombre de cada jugador como clave y una tupla de sus puntuaciones como valor. Luego, permite que los jugadores agreguen nuevas puntuaciones a sus registros.

65)Diccionario de Estudiantes: Crea un diccionario que almacene información sobre estudiantes, donde cada estudiante tiene un nombre, una lista de cursos en los que está inscrito y una tupla con sus calificaciones para cada curso.

66)Inventario de Tienda: Desarrolla un programa que administre el inventario de una tienda. Utiliza un diccionario para almacenar los productos como claves y sus detalles (precio, cantidad en stock, etc.) como valores. Permite que los usuarios agreguen nuevos productos y actualicen la información existente.

67)Registro de Películas: Crea un diccionario que contenga información sobre películas, donde cada película tiene un título, un año de lanzamiento y una lista de actores principales representados como tuplas (nombre del actor, papel en la película).

68)Almacenamiento de Recetas: Diseña un programa que gestione recetas de cocina. Utiliza un diccionario para almacenar recetas donde cada receta tiene un nombre, una lista de ingredientes (almacenados como tuplas de nombre y cantidad), y un conjunto de instrucciones.

69)Directorio Telefónico: Crea un directorio telefónico que almacene contactos como un diccionario donde las claves son nombres y los valores son tuplas con números de teléfono y direcciones de correo electrónico.

70)Registro de Compras: Desarrolla un programa para registrar compras en una tienda en línea. Utiliza un diccionario para rastrear las compras de los clientes donde cada cliente tiene

un identificador único y una lista de productos comprados (almacenados como tuplas de nombre del producto y cantidad).

71Registro de Viajes: Diseña una aplicación para registrar viajes. Utiliza un diccionario que almacene información sobre viajes, donde cada viaje tiene un destino, una lista de lugares visitados (almacenados como tuplas de nombre y fecha) y una descripción.

72Registro de Pedidos: Crea un sistema de registro de pedidos para un restaurante. Utiliza un diccionario que almacene detalles de pedidos donde cada pedido tiene un número de pedido único, una lista de elementos del pedido (almacenados como tuplas de nombre del plato y cantidad), y un estado (pendiente, entregado, etc.).

Mas diccionarios , listas , tuplas , conjuntos

Diccionarios:

73) Crea un programa que encuentre la clave con el valor máximo en un diccionario de números enteros.

74) Implementa una función que fusione dos diccionarios sin utilizar el operador de actualización (**update**).

75) Diseña un programa que elimine todas las claves con valores pares de un diccionario.
76) Crea una función que encuentre todas las claves que tienen valores iguales en dos diccionarios diferentes.

77) Desarrolla un programa que calcule la suma de los valores en un diccionario anidado.

Tuplas:

78) Escribe una función que tome una lista de tuplas y las ordene en función del segundo elemento de cada tupla.

79) Implementa una función que encuentre la tupla con la mayor suma de elementos en una lista de tuplas.

80) Diseña un programa que combine dos listas de tuplas en una sola lista, eliminando duplicados basados en el primer elemento de cada tupla.

81) Crea una función que divida una lista de números en dos listas de tuplas, una con números pares y otra con números impares.

82) Desarrolla un programa que encuentre la tupla más común en una lista de tuplas.

Conjuntos:

83) Escribe una función que calcule la intersección de dos conjuntos sin utilizar el operador **&**.

84) Implementa una función que verifique si un conjunto es un subconjunto de otro conjunto.

85) Diseña un programa que encuentre todos los elementos únicos en una lista y los almacene en un conjunto.

86) Crea una función que calcule la unión de múltiples conjuntos.

87) Desarrolla un programa que encuentre la diferencia simétrica entre dos conjuntos.

Listas:

88) Escribe una función que elimine elementos duplicados de una lista sin cambiar su orden.

89) Implementa un programa que encuentre el elemento que más se repite en una lista.

90) Diseña una función que divida una lista en sub-listas de tamaño fijo.

91) Crea un programa que realice una rotación a la izquierda en una lista (los elementos se desplazan hacia la izquierda).

92) Desarrolla una función que encuentre la subsecuencia más larga creciente en una lista de números.

Nivel avanzado en listas , diccionarios , tuplas y conjuntos

Diccionarios y Listas:

93) Crea un programa que gestione una lista de usuarios, donde cada usuario es un diccionario con nombre, correo electrónico y una lista de pedidos realizados (almacenados como diccionarios con detalles de productos).

94) Implementa una función que reciba una lista de empleados (diccionarios) y calcule el salario promedio.

95) Diseña un programa que almacene información sobre películas en un diccionario, donde cada película tiene un título, una lista de actores principales y una tupla con la fecha de lanzamiento y la duración.

96) Desarrolla una función que encuentre el empleado con el salario más alto en una lista de empleados (diccionarios) y muestre su información.

97) Escribe un programa que gestione un diccionario de productos, donde cada producto tiene un nombre y una lista de revisiones de usuarios (almacenadas como diccionarios con comentarios y calificaciones).

Listas y Tuplas:

98) Implementa una función que tome una lista de números y devuelva una tupla con el valor mínimo y máximo.

99) Diseña un programa que organice una lista de estudiantes (diccionarios) en función de sus calificaciones (almacenadas como una tupla).

100) Crea una función que tome una lista de direcciones de correo electrónico y divida las válidas de las inválidas en dos listas separadas.

101) Desarrolla un programa que administre un diccionario de ciudades y sus temperaturas máximas diarias (almacenadas como una lista de tuplas con fechas y temperaturas).

102) Escribe una función que encuentre el elemento más común en una lista de tuplas y muestre cuántas veces aparece.

Diccionarios y Conjuntos:

103) Diseña un programa que gestione una lista de tareas pendientes de múltiples usuarios, utilizando un diccionario donde las claves son los usuarios y los valores son conjuntos de tareas.

104) Implementa una función que elimine duplicados de una lista manteniendo el orden original y almacene los elementos únicos en un conjunto.

105) Crea un programa que administre una lista de invitados a un evento, utilizando un diccionario para rastrear la asistencia (utilizando un conjunto).

106) Desarrolla una función que encuentre la intersección de múltiples conjuntos almacenados en un diccionario.

107) Escribe un programa que gestione un diccionario de productos en stock, donde las claves son los productos y los valores son conjuntos de ubicaciones donde se almacenan.

Validación de datos de entrada

108) **Suma de dos números:** Escribe una función que tome dos números como entrada y devuelva su suma después de validar que ambos sean números.

109) **Promedio de una lista:** Implementa una función que reciba una lista de números y calcule su promedio, asegurándose de que todos los elementos sean numéricos.

110) **Calificación de un examen:** Crea una función que tome una calificación como entrada y la valide para asegurarse de que esté en el rango de 0 a 100.

111) **Edad válida:** Desarrolla una función que solicite la edad del usuario y la valide para asegurarse de que sea un número positivo.

112) **División segura:** Escribe una función que realice una división entre dos números, pero primero valide que el divisor no sea cero.

113) **Búsqueda en lista:** Implementa una función que busque un elemento en una lista y valide que el elemento exista en la lista antes de buscarlo.

114) **Número positivo o negativo:** Crea una función que tome un número como entrada y determine si es positivo, negativo o cero.

115) **Número primo:** Desarrolla una función que verifique si un número es primo después de validar que sea un número entero positivo.

116) **Día de la semana:** Escribe una función que solicite al usuario un número del 1 al 7 y devuelva el nombre del día de la semana correspondiente después de validar la entrada.

117) **Convertir a mayúsculas:** Implementa una función que tome una cadena como entrada y la convierta a mayúsculas, validando que sea una cadena de texto.

118) **Longitud de una cadena:** Crea una función que calcule la longitud de una cadena después de validar que sea una cadena de texto válida.

119) **Factorial de un número:** Desarrolla una función que calcule el factorial de un número después de validar que sea un número entero no negativo.

120) **Potencia de un número:** Escribe una función que calcule la potencia de un número después de validar que tanto la base como el exponente sean números reales.

121) **Raíz cuadrada:** Implementa una función que calcule la raíz cuadrada de un número positivo después de validar la entrada.

122) **Divisores de un número:** Crea una función que encuentre todos los divisores de un número después de validar que el número sea un entero positivo.

123) **Número de Fibonacci:** Desarrolla una función que calcule el número de Fibonacci en una posición dada después de validar que la posición sea un número entero no negativo.

124) **Cálculo de área:** Escribe una función que calcule el área de una figura geométrica después de validar que los valores de entrada sean números reales positivos.

125) **Suma de elementos pares:** Implementa una función que tome una lista de números y calcule la suma de los elementos pares, validando que los elementos sean números.

126) **Ordenar una lista:** Crea una función que ordene una lista de números de manera ascendente después de validar que los elementos sean números.

127) **Eliminar elementos duplicados:** Desarrolla una función que elimine elementos duplicados de una lista después de validar que la entrada sea una lista de valores.

Funciones de orden superior

128) **Mapeo de Números:** Escribe una función de orden superior que tome una lista de números y una función como argumentos y aplique la función a cada elemento de la lista.

129) **Filtrado de Números Pares:** Implementa una función de orden superior que tome una lista de números y una función de filtro como argumentos, y devuelva una lista con solo los números pares que pasen el filtro.

130) **Suma de Números en Lista:** Diseña una función de orden superior que tome una lista de números y una función acumuladora como argumentos, y devuelva la suma acumulada de los números en la lista.

131) **Transformación de Texto:** Crea una función de orden superior que tome una cadena de texto y una función de transformación como argumentos, y aplique la función de transformación a cada palabra en la cadena.

132) **Ordenar por Clave:** Desarrolla una función de orden superior que tome una lista de objetos y una función de clave como argumentos, y ordene la lista en función de la clave proporcionada.

133) **Composición de Funciones:** Implementa una función de orden superior que tome dos funciones como argumentos y devuelva una nueva función que sea la composición de las dos funciones.

134) **Filtrado de Palabras Largas:** Diseña una función de orden superior que tome una lista de palabras y una función de filtro basada en la longitud como argumentos, y devuelva una lista con palabras que pasen el filtro.

135) **Operaciones en Lista:** Crea una función de orden superior que tome una lista de números y una función de operación binaria (como suma o multiplicación) como argumentos, y aplique la operación a todos los elementos de la lista.

136) **Reducción de Lista:** Desarrolla una función de orden superior que tome una lista de números y una función de reducción (como máximo o mínimo) como argumentos, y devuelva el resultado de aplicar la función de reducción a la lista.

137) **Ordenar por Valor de Atributo:** Implementa una función de orden superior que tome una lista de objetos y un atributo como argumentos, y ordene la lista en función del valor de ese atributo en los objetos.

138) **Filtrado de Elementos Duplicados:** Diseña una función de orden superior que tome una lista y una función de filtro que identifique elementos duplicados, y devuelva una lista con elementos únicos.

139) **Agrupar Elementos por Clave:** Crea una función de orden superior que tome una lista de objetos y una función de agrupación basada en una clave como argumentos, y devuelva un diccionario que agrupe los objetos según la clave.

140) **Operaciones de Matriz:** Desarrolla una función de orden superior que tome dos matrices y una función de operación binaria, y aplique la operación a las matrices elemento por elemento.

141) **Filtrado de Fechas:** Implementa una función de orden superior que tome una lista de fechas y una función de filtro basada en una condición de fecha, y devuelva una lista de fechas que cumplan con la condición.

142) **Conteo de Elementos:** Diseña una función de orden superior que tome una lista de objetos y una función de conteo basada en una propiedad, y devuelva un diccionario que cuente la cantidad de elementos con cada valor de la propiedad.

143) **Operaciones con Fracciones:** Crea una función de orden superior que tome dos fracciones y una función de operación binaria, y aplique la operación a las fracciones.

144) **Validación de Datos:** Desarrolla una función de orden superior que tome una lista de datos y una función de validación, y devuelva una lista de datos válidos según la función de validación.

145) **Ordenar por Frecuencia:** Implementa una función de orden superior que tome una lista de elementos y una función de frecuencia, y ordene la lista en función de la frecuencia de cada elemento.

146) **Transformación de Imágenes:** Diseña una función de orden superior que tome una imagen (representada como una matriz de píxeles) y una función de transformación, y aplique la transformación a la imagen.

147) **Generación de Números Primos:** Crea una función de orden superior que genere números primos utilizando el algoritmo de la criba de Eratóstenes.

Recursividad

Nivel Básico:

148) **Suma de Números:** Escribe una función recursiva que calcule la suma de los primeros **n** números naturales.

149) **Factorial:** Implementa una función recursiva para calcular el factorial de un número.

150) **Potencia:** Crea una función recursiva que calcule la potencia de un número dado.

151) **Fibonacci:** Desarrolla una función recursiva para calcular el término **n** de la secuencia de Fibonacci.

152) **Cuenta Regresiva:** Escribe una función recursiva que muestre una cuenta regresiva desde un número **n** hasta 1.

Nivel Intermedio:

153) **Suma de Elementos en Lista:** Implementa una función recursiva que calcule la suma de todos los elementos en una lista.

154) **Cambio de Moneda:** Diseña una función recursiva que determine todas las formas posibles de dar cambio a una cantidad dada utilizando monedas de diferentes denominaciones.

155) **Número Palindrómico:** Crea una función recursiva que verifique si una cadena de texto es un palíndromo.

156) **Máximo Común Divisor:** Desarrolla una función recursiva para calcular el máximo común divisor (MCD) de dos números.

157) **Torres de Hanoi:** Escribe una función recursiva para resolver el problema de las Torres de Hanoi con **n** discos.

Pilas y colas

Nivel Básico:

158) **Pila:** Implementa una pila utilizando una lista y crea funciones para apilar y desapilar elementos.

159) **Cola:** Crea una cola utilizando una lista y funciones para encolar y desencolar elementos.

160) **Cola Doble:** Diseña una cola doble utilizando una lista y funciones para insertar y eliminar elementos tanto al principio como al final de la cola.

161) **Pila de Palabras:** Escribe un programa que verifique si una palabra es un palíndromo utilizando una pila para comparar caracteres.

162) **Cola de Tareas:** Crea una cola de tareas pendientes y desarrolla funciones para agregar nuevas tareas y completarlas.

Nivel Intermedio:

163) **Pila con Límite:** Implementa una pila con límite de tamaño utilizando una lista y controla el desbordamiento.

164) **Cola Circular:** Diseña una cola circular utilizando una lista y funciones para encolar y desencolar elementos.

165) **Cola Doble con Prioridad:** Crea una cola doble con prioridad utilizando una lista y funciones para insertar elementos con diferentes prioridades.

166) **Pila de Expresiones:** Desarrolla un programa que evalúe una expresión matemática en notación polaca inversa (postfija) utilizando una pila.

167) **Simulación de Colas:** Escribe una simulación simple de una cola de autobuses que llegan y parten en intervalos de tiempo.

Programación Orientada a objetos

Nivel Básico:

168) **Clase Perro:** Crea una clase **Perro** con atributos como nombre, raza y edad, y métodos para ladrar y mostrar información básica.

169) **Clase Rectángulo:** Implementa una clase **Rectangulo** con atributos para el largo y ancho, y métodos para calcular el área y el perímetro.

170) **Clase Libro:** Diseña una clase **Libro** con atributos como título, autor y género, y métodos para mostrar la información del libro.

171) **Clase Vehículo:** Desarrolla una clase **Vehiculo** con atributos como marca, modelo y año, y métodos para encender y apagar el motor.

172) **Clase Estudiante:** Crea una clase **Estudiante** con atributos como nombre, edad y calificaciones, y métodos para calcular el promedio de calificaciones.

Nivel Medio:

173) **Clase Banco:** Implementa una clase **Banco** con una lista de cuentas bancarias, y métodos para agregar cuentas y calcular el saldo total.

174) **Clase Tienda:** Diseña una clase **Tienda** con una lista de productos y métodos para agregar productos y calcular el precio total de la compra.

175) **Clase Película:** Desarrolla una clase **Pelicula** con atributos como título, director y año, y métodos para agregar y listar actores.

176) **Clase Contacto:** Crea una clase **Contacto** con atributos como nombre, correo electrónico y número de teléfono, y métodos para mostrar y modificar la información de contacto.

177) **Clase Agenda:** Implementa una clase **Agenda** que almacene contactos y permita buscarlos por nombre.

178) **Clase Empleado:** Diseña una clase **Empleado** con atributos como nombre, salario y departamento, y métodos para calcular el aumento de salario.

179) **Clase Coche:** Desarrolla una clase **Coche** con atributos como marca, modelo y año, y métodos para acelerar y frenar.

180) **Clase Banco con Transferencias:** Extiende la clase **Banco** para permitir transferencias entre cuentas bancarias.

181) **Clase Alumno:** Crea una clase **Alumno** con atributos como nombre, edad y calificaciones, y métodos para calcular la calificación final.

182) **Clase Música:** Implementa una clase **Musica** con atributos como título, artista y género, y métodos para reproducir y pausar la música.

183) **Clase Restaurante:** Diseña una clase **Restaurante** con atributos como nombre y menú, y métodos para agregar platos y mostrar el menú.

184) **Clase Computadora:** Desarrolla una clase **Computadora** con atributos como marca, modelo y capacidad de almacenamiento, y métodos para instalar software.

185) **Clase Inventario:** Crea una clase **Inventario** para gestionar un inventario de productos con métodos para agregar, vender y mostrar productos.

186) **Clase Hotel:** Implementa una clase **Hotel** con atributos como nombre y habitaciones, y métodos para reservar y desocupar habitaciones.

187) **Clase Juego de Cartas:** Diseña una clase **JuegoDeCartas** para un juego de cartas simple con métodos para barajar, repartir y jugar cartas.

Listas enlazadas , colas , pilas con POO

Listas Enlazadas:

188) **Lista Enlazada Simple:** Implementa una clase **ListaEnlazada** para gestionar una lista enlazada simple que permita agregar y eliminar elementos.

189) **Inversión de Lista:** Desarrolla un método en la clase **ListaEnlazada** que invierta la lista enlazada.

190) **Eliminación de Duplicados:** Diseña un método en la clase **ListaEnlazada** que elimine los elementos duplicados de la lista.

191) **Listas Enlazadas Dobles:** Crea una clase **ListaEnlazadaDoble** para gestionar una lista enlazada doble que permita agregar y eliminar elementos en ambos sentidos.

Pilas:

192) **Pila con Listas Enlazadas:** Implementa una clase **Pila** utilizando una lista enlazada que permita realizar operaciones de apilamiento y desapilamiento.

193) **Validación de Paréntesis:** Desarrolla un método en la clase **Pila** para validar si una cadena de texto tiene paréntesis balanceados.

Colas:

194) **Cola con Listas Enlazadas:** Crea una clase **Cola** utilizando una lista enlazada que permita realizar operaciones de encolamiento y desencolamiento.

Colas Dobles:

195) **Cola Doble con Listas Enlazadas:** Implementa una clase **ColaDoble** utilizando una lista enlazada que permita insertar y eliminar elementos tanto al principio como al final de la cola.

Colas de Prioridad:

196) **Cola de Prioridad con Lista Enlazada:** Diseña una clase **ColaPrioridad** utilizando una lista enlazada que permita insertar elementos con diferentes prioridades y realizar extracciones basadas en prioridad.

Combinación de Estructuras:

197) **Administrador de Tareas:** Crea una clase **AdministradorTareas** que utilice una cola para gestionar tareas pendientes y una lista enlazada para llevar un registro de tareas completadas.

198) **Simulación de Procesos:** Implementa una clase **SimulacionProcesos** que utilice una cola de prioridad para simular la ejecución de procesos con diferentes prioridades.

199) **Búsqueda en Anchura (BFS):** Desarrolla una clase **Grafo** que utilice una cola para realizar un recorrido en anchura (BFS) en un grafo.

200) **Evaluación de Expresiones:** Crea una clase **EvaluadorExpresiones** que utilice una pila para evaluar expresiones matemáticas en notación polaca inversa (postfija).

201) **Ordenamiento con Cola de Prioridad:** Diseña una clase **OrdenadorColaPrioridad** que utilice una cola de prioridad para ordenar una lista de elementos.

202) **Planificador de Tareas:** Implementa una clase **PlanificadorTareas** que utilice una cola de prioridad para planificar tareas con diferentes niveles de urgencia.

203) **Manejo de Mensajes:** Desarrolla una clase **ManejoMensajes** que utilice una cola para gestionar mensajes entrantes y una pila para mostrar los mensajes más recientes.

204) **Gestor de Descargas:** Crea una clase **GestorDescargas** que utilice una cola para gestionar descargas de archivos y una cola de prioridad para priorizarlas.

205) **Simulación de Tráfico:** Diseña una clase **SimulacionTrafico** que utilice una cola de prioridad para simular el tráfico de vehículos con diferentes prioridades en una intersección.

206) **Almacenamiento de Datos:** Implementa una clase **AlmacenamientoDatos** que utilice una cola de prioridad para almacenar y recuperar datos con marcas de tiempo.

207) **Planificador de Vuelos:** Desarrolla una clase **PlanificadorVuelos** que utilice una cola de prioridad para programar vuelos en un aeropuerto con diferentes horarios de despegue.

Árboles

Nivel Básico:

208) **Árbol Binario Simple:** Crea una clase **Nodo** para representar nodos en un árbol binario y una clase **ArbolBinario** que permita agregar nodos y realizar un recorrido en orden.

209) **Altura de un Árbol:** Escribe una función que calcule la altura de un árbol binario.

210) **Recorrido en Profundidad:** Implementa una función para realizar un recorrido en profundidad (DFS) en un árbol.

211) **Recorrido en Anchura:** Desarrolla una función para realizar un recorrido en anchura (BFS) en un árbol.

212) **Árbol Binario de Búsqueda (BST):** Crea una clase **ArbolBST** que implemente un árbol binario de búsqueda y permita realizar inserciones y búsquedas.

Nivel Intermedio:

213) **Árbol AVL:** Diseña una clase **ArbolAVL** que implemente un árbol AVL y permita realizar inserciones y eliminaciones manteniendo el equilibrio.

214) **Recorrido Postorden:** Escribe una función para realizar un recorrido postorden en un árbol.

215) **Suma de Nodos:** Implementa una función que calcule la suma de todos los nodos en un árbol.

216) **Árbol Binario Espejo:** Desarrolla una función que cree un árbol binario espejo a partir de un árbol dado.

217) **Búsqueda en Árbol Binario:** Crea una función que busque un valor en un árbol binario y devuelva el nodo correspondiente.

Nivel Avanzado:

218) **Árbol de Segmento:** Diseña una clase **ArbolSegmento** que implemente un árbol de segmento para consultas de rangos en un arreglo.

219) **Árbol Trie:** Implementa una clase **ArbolTrie** que represente un árbol trie para almacenar palabras y permita buscar prefijos y palabras completas.

220) **Árbol de Huffman:** Desarrolla una función para construir un árbol de Huffman a partir de frecuencias de caracteres y codificar texto.

221) **Recorrido en Zigzag:** Escribe una función para realizar un recorrido en zigzag en un árbol (alternando entre izquierda y derecha en cada nivel).

222) **Árbol 2-3:** Crea una clase **Arbol23** que implemente un árbol 2-3 y permita realizar inserciones y eliminaciones manteniendo la propiedad.

Nivel Experto:

223) **Árbol Sufijo:** Diseña una clase **ArbolSufijo** que construya un árbol de sufijo a partir de una cadena de texto y permita buscar subcadenas.

224) **Árbol KD:** Implementa una clase **ArbolKD** para representar un árbol KD (árbol binario de búsqueda en k dimensiones) y realizar consultas en espacios multidimensionales.

225) **Árbol Splay:** Desarrolla una clase **ArbolSplay** que implemente un árbol Splay y permita realizar operaciones de acceso y reorganización.

226) **Árbol de Intervalo:** Crea una clase **ArbolIntervalo** que implemente un árbol de intervalo para buscar y actualizar rangos en un conjunto de valores.

227) **Árbol B+:** Diseña una clase **ArbolBPlus** que implemente un árbol B+ y permita realizar inserciones, eliminaciones y búsquedas eficientes en un conjunto ordenado de datos.

+ Recursividad

Nivel Avanzado:

228) **Recorrido de Árbol Binario:** Implementa una función recursiva que realice un recorrido en orden de un árbol binario.

229) **Generación de Subconjuntos:** Diseña una función recursiva que genere todos los subconjuntos de un conjunto dado.

230) **Cálculo de Combinaciones:** Desarrolla una función recursiva para calcular el número de combinaciones posibles de **n** elementos tomados de **k** en **k**.

231) **Recorrido de Laberinto:** Escribe una función recursiva que encuentre una ruta a través de un laberinto utilizando "backtracking".

232) **Árbol de Recursión:** Crea una función recursiva que genere un árbol de recursión para calcular el n-ésimo término de la secuencia de Fibonacci.

233) **División Entera:** Implementa una función recursiva para realizar una división entera de dos números enteros.

234) **Recorrido de Grafo:** Diseña una función recursiva que realice un recorrido en profundidad (DFS) de un grafo.

235) **Fractal:** Desarrolla una función recursiva para dibujar un fractal, como el conjunto de Mandelbrot.

236) **Ordenamiento por Fusión:** Escribe una función recursiva para realizar el ordenamiento por fusión en una lista.

237) **Recorrido en Laberinto 3D:** Crea una función recursiva que encuentre una ruta a través de un laberinto tridimensional.

+ Pilas , + Colas

Nivel Avanzado:

238) **Pila de Historiales:** Implementa una pila de historiales de navegación web, permitiendo navegar hacia atrás y hacia adelante.

239) **Cola de Mensajes:** Crea una cola de mensajes para un sistema de chat, gestionando la entrega y lectura de mensajes.

240) **Cola de Prioridad Dinámica:** Diseña una cola de prioridad que permita cambiar la prioridad de los elementos en tiempo real.

241) **Editor de Texto Undo/Redo:** Desarrolla un editor de texto con las funciones "deshacer" (undo) y "rehacer" (redo) utilizando pilas.

242) **Cola de Impresión:** Implementa una cola de impresión para administrar documentos que deben imprimirse, considerando prioridades y tiempos de espera.

243) **Pila de Tareas Recursivas:** Crea una pila para gestionar llamadas recursivas en un programa y evitar desbordamiento de la pila de llamadas.

244) **Cola de Espera en Servidor:** Diseña una cola de espera para un servidor que administra conexiones de clientes concurrentes.

245) **Pila de Navegación de Laberinto:** Escribe un programa que encuentre la ruta de navegación a través de un laberinto utilizando una pila para realizar un seguimiento de las ubicaciones visitadas.

246) **Cola de Notificaciones:** Desarrolla una cola de notificaciones para una aplicación que muestra alertas a los usuarios.

247) **Editor de Código con Funcionalidad "Deshacer" y "Rehacer":** Implementa un editor de código con funciones de "deshacer" (undo) y "rehacer" (redo) utilizando pilas para mantener el historial de cambios.

Grafos

Nivel Básico:

248) **Representación de Grafo:** Implementa una clase **Grafo** que permita representar un grafo no dirigido y agregar vértices y aristas.

249) **Búsqueda de Camino:** Escribe una función que verifique si existe un camino entre dos vértices en un grafo.

250) **Grado de un Vértice:** Desarrolla una función para calcular el grado de un vértice en un grafo no dirigido.

251) **Recorrido en Profundidad:** Implementa una función que realice un recorrido en profundidad (DFS) en un grafo.

252) **Recorrido en Anchura:** Desarrolla una función que realice un recorrido en anchura (BFS) en un grafo.

Nivel Intermedio:

249) **Ciclo en Grafo:** Escribe una función que determine si un grafo contiene algún ciclo.

250) **Árbol de Expansión Mínima:** Implementa un algoritmo para encontrar el árbol de expansión mínima en un grafo ponderado.

251) **Ruta más Corta:** Desarrolla una función que encuentre la ruta más corta entre dos vértices en un grafo ponderado.

252) **Orden Topológico:** Crea una función que encuentre un orden topológico válido en un grafo dirigido acíclico (DAG).

253) **Grafo Bipartito:** Diseña una función que determine si un grafo es bipartito o no.

Nivel Avanzado:

254) **Dijkstra y Bellman-Ford:** Implementa los algoritmos de Dijkstra y Bellman-Ford para encontrar rutas más cortas en un grafo ponderado con pesos positivos y negativos, respectivamente.

255) **Floyd-Warshall:** Desarrolla el algoritmo de Floyd-Warshall para encontrar las rutas más cortas entre todos los pares de vértices en un grafo ponderado.

256) **Flujo Máximo:** Diseña un algoritmo para encontrar el flujo máximo en un grafo de flujo con capacidad en aristas.

257) **Coloración de Grafos:** Implementa un algoritmo de coloración de grafos para asignar colores a los vértices de manera que ningún par de vértices adyacentes tenga el mismo color.

258) **Algoritmo de Kruskal:** Desarrolla el algoritmo de Kruskal para encontrar el árbol de expansión mínima en un grafo ponderado con conjuntos disjuntos.

259) **Algoritmo de Prim:** Crea el algoritmo de Prim para encontrar el árbol de expansión mínima en un grafo ponderado no dirigido.

260) **Caminos más Cortos en DAG:** Implementa un algoritmo eficiente para encontrar caminos más cortos en un grafo dirigido acíclico (DAG).

261) **Flujo de Redes:** Desarrolla un algoritmo para calcular el flujo máximo en una red de flujo con múltiples fuentes y sumideros.

262) **Ciclo Euleriano:** Diseña una función que determine si un grafo contiene un ciclo euleriano (un ciclo que visita cada arista exactamente una vez).

263) **Conectividad de Componentes:** Implementa un algoritmo para encontrar y contar las componentes conexas en un grafo no dirigido.

Matrices

Nivel Medio:

264) **Suma de Matrices:** Escribe una función que sume dos matrices y devuelva el resultado.

265) **Producto de Matrices:** Implementa una función para multiplicar dos matrices y obtener el producto.

266) **Matriz Traspuesta:** Desarrolla una función que calcule la matriz traspuesta de una matriz dada.

267) **Identificar Matriz Diagonal:** Diseña una función que determine si una matriz cuadrada es diagonal o no.

268) **Determinante de una Matriz:** Crea una función que calcule el determinante de una matriz cuadrada.

Nivel Avanzado:

269) **Inversa de una Matriz:** Implementa un algoritmo para calcular la matriz inversa de una matriz cuadrada.

270) **Matriz de Cofactores:** Desarrolla una función que calcule la matriz de cofactores de una matriz dada.

271) **Ecuaciones Lineales:** Diseña una función que resuelva un sistema de ecuaciones lineales utilizando matrices y la regla de Cramer.

272) **Factorización LU:** Implementa un algoritmo para la factorización LU de una matriz.

273) **Descomposición QR:** Crea una función que realice la descomposición QR de una matriz.

Nivel Experto:

274) **Eigenvalores y Eigenvectores:** Diseña un programa que calcule los eigenvalores y eigenvectores de una matriz.

275) **Matrices Simétricas:** Implementa un algoritmo para verificar si una matriz es simétrica o no.

276) **Matriz de Rotación 3D:** Desarrolla una función que genere una matriz de rotación 3D para una transformación espacial.

277) **Matriz de Covarianza:** Diseña un programa que calcule la matriz de covarianza a partir de un conjunto de datos.

278) **Factorización de Cholesky:** Implementa un algoritmo para la factorización de Cholesky de una matriz simétrica definida positiva.

Nivel Experto (Continuación):

279) **Matrices Sparse:** Diseña una estructura de datos para representar matrices sparse (con muchos elementos iguales a cero) y realiza operaciones eficientes en ellas.

280) **Ranking de Páginas Web:** Implementa el algoritmo PageRank utilizando matrices para calcular el ranking de páginas web.

281) **Reducción de Dimensión:** Desarrolla un algoritmo de reducción de dimensión como Análisis de Componentes Principales (PCA) utilizando matrices.

282) **Interpolación de Matrices:** Diseña una función que realice interpolación de matrices para estimar valores faltantes en una matriz.

283) **Matrices Tensoriales:** Implementa operaciones y cálculos con matrices tensoriales de orden superior.

Funciones de Hash y tablas

Nivel Básico:

284) **Función de Hash Simple:** Implementa una función de hash que convierta una cadena de texto en un número entero utilizando la suma de los valores ASCII de los caracteres.

285) **Tabla de Hash:** Crea una tabla de hash que permita agregar pares clave-valor y recuperar valores por clave.

286) **Colisión de Hash:** Diseña una función de hash básica que cause colisiones y un método para manejarlas.

287) **Buscar Elemento en Tabla:** Escribe una función que busque un elemento por clave en una tabla de hash y lo elimine si existe.

288) **Contar Colisiones:** Desarrolla una función que cuente el número de colisiones en una tabla de hash.

Nivel Intermedio:

289) **Función de Hash Mejorada:** Mejora la función de hash utilizando técnicas como el método de división o multiplicación y evalúa su rendimiento.

290) **Tabla de Hash con Resolución de Colisiones:** Implementa una tabla de hash que maneje colisiones utilizando listas enlazadas.

291) **Carga de la Tabla:** Diseña una función que calcule el factor de carga de una tabla de hash y la redimensione si es necesario.

292) **Eliminar Colisiones:** Desarrolla un algoritmo para reducir o eliminar colisiones en una tabla de hash.

293) **Colisiones de Cadena:** Crea una tabla de hash que maneje colisiones utilizando el método de "cadenas" (buckets).

Nivel Avanzado:

294) **Función de Hash Criptográfica:** Implementa una función de hash criptográfica, como SHA-256, y úsala para almacenar contraseñas de manera segura.

295) **Tabla de Hash Abierta:** Diseña una tabla de hash abierta que maneje colisiones mediante el uso de sondas y resuelva colisiones de manera eficiente.

296) **Tabla de Hash Distribuida:** Crea una tabla de hash distribuida que almacene datos en varios servidores y permita la recuperación eficiente de datos.

297) **Hash Map de Frecuencia:** Desarrolla una tabla de hash que cuente la frecuencia de elementos en una lista o conjunto de datos.

298) **Tabla de Hash Persistente:** Implementa una tabla de hash que permita versionar datos y consultar versiones anteriores.

Nivel Experto:

299) **Estructura de Hash Compacta:** Diseña una estructura de hash compacta que optimice el uso de memoria para grandes conjuntos de datos.

300) **Tabla de Hash con Búsqueda Aproximada:** Crea una tabla de hash que permita buscar elementos aproximados utilizando técnicas como el hashing localidad-sensitivo (LSH).

301) **Tabla de Hash Inmutable:** Implementa una tabla de hash inmutable que no pueda ser modificada una vez creada.

302) **Hashing de Datos Grandes:** Desarrolla un algoritmo de hashing que funcione eficientemente para datos grandes que no caben en memoria principal.

303) **Hash Map con Comprimido de Huffman:** Diseña una tabla de hash que utilice el algoritmo de compresión de Huffman para representar las claves de manera eficiente.

+ Archivos

Lectura de Archivos:

304) Lee un archivo de texto y muestra su contenido en la consola.

305) Cuenta la cantidad de líneas en un archivo de texto.

306) Encuentra y muestra las palabras únicas en un archivo de texto.

307) Lee un archivo CSV y almacena sus datos en una lista de diccionarios.

308) Lee un archivo JSON y conviértelo en un diccionario Python.

Escritura de Archivos:

309) Crea un archivo de texto y escribe una lista de nombres en él.

310) Guarda datos en un archivo CSV a partir de una lista de diccionarios.

311) Escribe un diccionario en un archivo JSON.

Procesamiento de Archivos:

312) Calcula la suma de todos los números en un archivo de números separados por comas.

313) Encuentra la palabra más larga en un archivo de texto.

314) Dada una lista de nombres en un archivo, organízalos alfabéticamente y guárdalos en otro archivo.

315) Lee un archivo XML y extrae información específica de él.

Estructuras de Datos:

316) Lee un archivo de registros de estudiantes y almacena la información en una lista de objetos Estudiante.

317) Crea una tabla hash para almacenar información de empleados a partir de un archivo CSV.

318) Lee un archivo de registros de ventas y almacena los datos en una matriz.

319) Construye un árbol binario de búsqueda a partir de un archivo de datos numéricos.

Búsquedas y Filtrado:

320) Lee un archivo de registros de empleados y busca a todos los empleados con un salario superior a cierta cantidad.

321) Filtra los registros de un archivo de registros de ventas por fecha y almacena los datos filtrados en otro archivo.

322) Busca y muestra todos los correos electrónicos válidos en un archivo de texto.

Resúmenes y Estadísticas:

323) Calcula la suma y el promedio de una columna de números en un archivo CSV.

324) Encuentra el valor mínimo y máximo en un archivo de datos numéricos y muestra sus ubicaciones.

325) Calcula el total de ventas por producto a partir de un archivo de registros de ventas y almacena los resultados en un diccionario.

Transformación de Datos:

326) Convierte un archivo de registros de empleados en un archivo JSON.

327) Lee un archivo CSV con coordenadas geográficas y conviértelo en un archivo KML para su visualización en Google Earth.

328) Lee un archivo de datos en formato JSON y crea un archivo CSV con los mismos datos.

Procesamiento Avanzado:

329) Extrae datos de imágenes a partir de un archivo ZIP que contiene múltiples imágenes.

330) Analiza un archivo de registro de acceso web y muestra las estadísticas de uso por usuario.

331) Crea un programa que pueda realizar operaciones de consulta SQL en un archivo de base de datos SQLite.

Procesamiento de Grandes Volúmenes de Datos:

332) Lee un archivo de registro de transacciones financiero y calcula el saldo final de cada cuenta de cliente.

333) Procesa un archivo de registro de sensores IoT en tiempo real y almacena los datos en una base de datos NoSQL.

Ciencia de datos , Machine Learning , IA

Análisis de Datos:

334) Dado un conjunto de datos en formato CSV que contiene información de ventas, calcula el total de ventas por producto.

335) Lee un archivo JSON con datos demográficos de clientes y encuentra el grupo de edad promedio.

336) A partir de una tabla de datos en Excel, encuentra el valor promedio de una columna específica.

337) Lee un archivo de registro de acceso web y determina las páginas más visitadas.

338) Dado un conjunto de datos en formato CSV con información sobre el clima, encuentra la temperatura promedio por mes.

Preprocesamiento de Datos:

339) Limpia un conjunto de datos eliminando valores nulos o duplicados.

340) Normaliza una columna de datos numéricos para que tengan una media de 0 y una desviación estándar de 1.

341) Realiza la codificación one-hot de una columna de datos categóricos.

342) Divide un conjunto de datos en entrenamiento y prueba en una proporción específica.

343) Aplica una función de escala logarítmica a una columna de datos.

Visualización de Datos:

344) Crea un gráfico de barras que muestre la distribución de edades en una población.

345) Genera un gráfico de dispersión que muestre la relación entre dos variables en un conjunto de datos.

346) Crea un mapa de calor que visualice la correlación entre las características de un conjunto de datos.

347) Realiza un gráfico de líneas que muestre la tendencia de ventas mensuales a lo largo del tiempo.

348) Crea un histograma que muestre la distribución de ingresos en un conjunto de datos.

Modelado de Datos:

349) Entrena un modelo de regresión lineal para predecir el precio de una casa en función de sus características.

350) Implementa un clasificador K-Nearest Neighbors (K-NN) para etiquetar puntos en un conjunto de datos.

351) Realiza una reducción de dimensionalidad utilizando el análisis de componentes principales (PCA) en un conjunto de datos.

352) Entrena un modelo de machine learning para clasificar reseñas de productos como positivas o negativas.

353) Implementa un algoritmo de clustering, como K-Means, para agrupar datos similares en un conjunto de datos.

Machine Learning

Conceptos Básicos de Machine Learning:

354) Define machine learning y explica la diferencia entre aprendizaje supervisado y no supervisado.

355) Describe los pasos fundamentales en el proceso de machine learning.

356) ¿Qué es un conjunto de entrenamiento (training set) y para qué se utiliza en machine learning?

357) Explica la diferencia entre un modelo y un algoritmo en machine learning.

358) ¿Qué significa la "generalización" en el contexto del machine learning?

Introducción a Redes Neuronales:

359) Define una neurona artificial y describe su función en una red neuronal.

360) ¿Qué es una red neuronal artificial (ANN) y cuál es su estructura básica?

361) Describe la función de activación ReLU (Rectified Linear Unit) en una neurona.

362) ¿Qué es el aprendizaje supervisado y cómo se aplica a las redes neuronales?

363) Explica cómo se ajustan los pesos en una red neuronal mediante el descenso del gradiente.

Implementación de Redes Neuronales Simples:

364) Implementa una neurona artificial en Python que realice una suma ponderada de entradas.

365) Crea una función de activación ReLU y aplícala a la salida de una neurona.

366) Diseña una red neuronal con una capa de entrada, una capa oculta y una capa de salida.

367) Implementa el cálculo de la función de pérdida (loss function) para una tarea de regresión.

368) Desarrolla una red neuronal para clasificar imágenes de dígitos escritos a mano utilizando el conjunto de datos MNIST.

Entrenamiento y Evaluación de Redes Neuronales:

369) Divide un conjunto de datos en entrenamiento y prueba y explica la importancia de esta división.

370) Entrena una red neuronal utilizando el algoritmo de descenso del gradiente y ajusta los hiperparámetros.

371) Evalúa el rendimiento de una red neuronal utilizando métricas como precisión, recall y F1-score.

372) Implementa la técnica de validación cruzada (cross-validation) para evaluar el modelo de manera más robusta.

373) Aprende a evitar el sobreajuste (overfitting) en redes neuronales aplicando técnicas como la regularización y el abandono (dropout).

Redes Neuronales Convolucionales (CNN):

374) Describe el propósito y la estructura básica de una capa convolucional en una CNN.

375) Implementa una capa de convolución en una red neuronal para el procesamiento de imágenes.

376) Explica cómo se utiliza el agrupamiento (pooling) para reducir la dimensionalidad en una CNN.

377) Diseña una CNN para la clasificación de imágenes utilizando el conjunto de datos CIFAR-10.

378) Comprende el concepto de transfer learning y aplica un modelo preentrenado en nuevos datos.

Redes Neuronales Recurrentes (RNN):

379) Describe el funcionamiento de una RNN y su aplicación en secuencias de datos.

380) Implementa una capa LSTM (Long Short-Term Memory) en una RNN.

381) Diseña una RNN para la generación de texto, como la predicción de palabras en un texto.

382) Explica el problema del desvanecimiento del gradiente en RNN y cómo se resuelve con LSTM y GRU.

383) Aprende sobre las aplicaciones de RNN en tareas como el procesamiento de lenguaje natural (NLP) y la traducción automática.

Inteligencia Artificial

Conceptos Básicos de IA:

384) Explica qué es la inteligencia artificial y proporciona ejemplos de aplicaciones cotidianas.

385) Define el concepto de "aprendizaje automático" y menciona sus principales categorías.

386) ¿Qué significa el término "conjunto de datos" en el contexto de la inteligencia artificial?

Clasificación:

387) Dado un conjunto de datos con etiquetas (clases), entrena un modelo simple para clasificar nuevos ejemplos.

388) Implementa un clasificador binario para predecir si un correo electrónico es spam o no spam.

Regresión:

389) Utiliza un modelo de regresión lineal para predecir el precio de una casa en función de sus características.

390) Desarrolla un modelo de regresión polinómica para ajustar una curva a un conjunto de puntos de datos.

Árboles de Decisión:

391) Crea un árbol de decisión para ayudar a decidir si debes tomar un paraguas en función de las condiciones climáticas.

392) Explica el concepto de "entropía" en la construcción de árboles de decisión.

Aprendizaje No Supervisado:

393) Realiza una agrupación (clustering) de un conjunto de datos en función de similitudes entre ejemplos.

394) Describe cómo se utiliza el algoritmo K-Means para agrupar datos en clusters.

Procesamiento de Lenguaje Natural (NLP):

395) Desarrolla un programa que cuente la frecuencia de palabras en un texto.

396) Realiza un análisis de sentimientos en reseñas de productos para determinar si son positivas o negativas.

Visión por Computadora:

397) Implementa un programa que detecte rostros en una imagen utilizando OpenCV.

398) Desarrolla un sistema de reconocimiento de dígitos escritos a mano utilizando bibliotecas de Python.

Juegos y Agentes Inteligentes:

399) Crea un juego simple en el que un agente tome decisiones para maximizar una puntuación.

400) Implementa un agente inteligente que resuelva el problema del "rompecabezas de las ocho reinas".

FIN DE LOS EJERCICIOS