Parcial de algoritmo y estructura

a. Desarrollar una función recursiva que permita listar # los elementos de vector/lista de manera inversa al que # están cargados. Preferentemente la función solo debe # tener un parámetro que es la lista de elementos.

def listar\_inverso(lista):

if not lista: # Si la lista está vacía, no hay elementos que imprimir

return

else:

# Imprimimos el último elemento de la lista

print(lista[-1])

# Llamamos recursivamente a la función con la lista sin el último elemento

listar\_inverso(lista[:-1])

# Ejemplo de uso

lista = [1, 2, 3, 4, 5]

print("Lista en orden inverso:")

listar\_inverso(lista)

Esta función listar\_inverso imprime los elementos de la lista en orden inverso. Funciona de la siguiente manera:

1. Verifica si la lista está vacía. Si lo está, la función termina.
2. Si la lista tiene elementos, imprime el último elemento (lista[-1]).
3. Luego, llama recursivamente a la función con la lista sin el último elemento (lista[:-1]).

Esto se repite hasta que la lista esté vacía, momento en el que la recursión termina.

me podrias pasar otra posibilidad

def listar\_inverso(lista, index):

if index < 0: # Verifica si hemos llegado al índice final de la lista

return

else:

# Imprime el elemento en el índice actual

print(lista[index])

# Llama recursivamente a la función con el índice decrementado en 1

listar\_inverso(lista, index - 1)

# Ejemplo de uso

lista = [1, 2, 3, 4, 5]

print("Lista en orden inverso:")

listar\_inverso(lista, len(lista) - 1)

Esta función también imprime los elementos de la lista en orden inverso. Funciona así:

1. Comienza desde el índice correspondiente al final de la lista (len(lista) - 1).
2. Imprime el elemento en el índice actual.
3. Llama recursivamente a la función con el índice decrementado en 1 hasta llegar al índice 0.

Puedes probar esta implementación en tu entorno de Python para verificar su funcionamiento.

dinosaurios = [ { "nombre": "Tyrannosaurus Rex", "especie": "Theropoda", "peso": "7000 kg", "descubridor": "Barnum Brown", "ano\_descubrimiento": 1902 }, { "nombre": "Triceratops", "especie": "Ceratopsidae", "peso": "6000 kg", "descubridor": "Othniel Charles Marsh", "ano\_descubrimiento": 1889 }, { "nombre": "Velociraptor", "especie": "Dromaeosauridae", "peso": "15 kg", "descubridor": "Henry Fairfield Osborn", "ano\_descubrimiento": 1924 }, { "nombre": "Brachiosaurus", "especie": "Sauropoda", "peso": "56000 kg", "descubridor": "Elmer S. Riggs", "ano\_descubrimiento": 1903 }, { "nombre": "Stegosaurus", "especie": "Stegosauridae", "peso": "5000 kg", "descubridor": "Othniel Charles Marsh", "ano\_descubrimiento": 1877 }, { "nombre": "Spinosaurus", "especie": "Spinosauridae", "peso": "10000 kg", "descubridor": "Ernst Stromer", "ano\_descubrimiento": 1912 }, { "nombre": "Allosaurus", "especie": "Theropoda", "peso": "2000 kg", "descubridor": "Othniel Charles Marsh", "ano\_descubrimiento": 1877 }, { "nombre": "Apatosaurus", "especie": "Sauropoda", "peso": "23000 kg", "descubridor": "Othniel Charles Marsh", "ano\_descubrimiento": 1877 }, { "nombre": "Diplodocus", "especie": "Sauropoda", "peso": "15000 kg", "descubridor": "Othniel Charles Marsh", "ano\_descubrimiento": 1878 }, { "nombre": "Ankylosaurus", "especie": "Ankylosauridae", "peso": "6000 kg", "descubridor": "Barnum Brown", "ano\_descubrimiento": 1908 }, { "nombre": "Parasaurolophus", "especie": "Hadrosauridae", "peso": "2500 kg", "descubridor": "William Parks", "ano\_descubrimiento": 1922 }, { "nombre": "Carnotaurus", "especie": "Theropoda", "peso": "1500 kg", "descubridor": "José Bonaparte", "ano\_descubrimiento": 1985 }, { "nombre": "Styracosaurus", "especie": "Ceratopsidae", "peso": "2700 kg", "descubridor": "Lawrence Lambe", "ano\_descubrimiento": 1913 }, { "nombre": "Therizinosaurus", "especie": "Therizinosauridae", "peso": "5000 kg", "descubridor": "Evgeny Maleev", "ano\_descubrimiento": 1954 }, { "nombre": "Pteranodon", "especie": "Pterosauria", "peso": "25 kg", "descubridor": "Othniel Charles Marsh", "ano\_descubrimiento": 1876 }, { "nombre": "Quetzalcoatlus", "especie": "Pterosauria", "peso": "200 kg", "descubridor": "Douglas A. Lawson", "ano\_descubrimiento": 1971 }, { "nombre": "Plesiosaurus", "especie": "Plesiosauria", "peso": "450 kg", "descubridor": "Mary Anning", "ano\_descubrimiento": 1824 }, { "nombre": "Mosasaurus", "especie": "Mosasauridae", "peso": "15000 kg", "descubridor": "William Conybeare", "ano\_descubrimiento": 1829 }, ]

#a) Contar cuantas especies hay;

def contar\_especies(dinosaurios):

especies = set() # Usamos un conjunto para garantizar especies únicas

for dino in dinosaurios:

especies.add(dino["especie"])

return len(especies)

# Contar cuántas especies hay

num\_especies = contar\_especies(dinosaurios)

print("Número de especies:", num\_especies)

Este código primero inicializa un conjunto vacío especies. Luego, itera sobre la lista dinosaurios, y para cada dinosaurio, agrega su especie al conjunto especies. Finalmente, devuelve la longitud de ese conjunto, que es el número de especies únicas en la lista.

#b) **Determinar cuantos descubridores distintos hay;**

def contar\_descubridores(dinosaurios):

descubridores = set() # Usamos un conjunto para garantizar descubridores únicos

for dino in dinosaurios:

descubridores.add(dino["descubridor"])

return len(descubridores)

# Contar cuántos descubridores distintos hay

num\_descubridores = contar\_descubridores(dinosaurios)

print("Número de descubridores distintos:", num\_descubridores)

Este código sigue un enfoque similar al anterior. Inicializa un conjunto vacío descubridores, luego itera sobre la lista de dinosaurios y agrega cada descubridor al conjunto descubridores. Finalmente, devuelve la longitud de ese conjunto, que es el número de descubridores distintos en la lista.

**c) Mostrar todos los dinosaurios que empiecen con T**

def dinosaurios\_con\_T(dinosaurios):

dinosaurios\_con\_t = [] # Lista para almacenar los dinosaurios cuyos nombres comienzan con "T"

for dino in dinosaurios:

if dino["nombre"].startswith("T"):

dinosaurios\_con\_t.append(dino)

return dinosaurios\_con\_t

# Mostrar los dinosaurios cuyos nombres empiezan con "T"

dinosaurios\_t = dinosaurios\_con\_T(dinosaurios)

print("Dinosaurios cuyos nombres empiezan con 'T':")

for dino in dinosaurios\_t:

print(dino["nombre"])

Este código crea una lista vacía dinosaurios\_con\_t para almacenar los dinosaurios cuyos nombres comienzan con "T". Luego, itera sobre la lista de dinosaurios y verifica si el nombre de cada dinosaurio comienza con "T" utilizando el método startswith. Si lo hace, agrega ese dinosaurio a la lista dinosaurios\_con\_t. Finalmente, imprime los nombres de los dinosaurios que cumplen con esta condición.

# Filtrar los dinosaurios cuyos nombres empiezan con "T"

dinosaurios\_con\_t = [dino["nombre"] for dino in dinosaurios if dino["nombre"].startswith("T")]

# Mostrar los dinosaurios cuyos nombres empiezan con "T"

print("Dinosaurios cuyos nombres empiezan con 'T':")

for nombre in dinosaurios\_con\_t:

print(nombre)

Este código crea una lista dinosaurios\_con\_t utilizando una comprensión de lista. Itera sobre la lista de dinosaurios y agrega el nombre de cada dinosaurio a dinosaurios\_con\_t si su nombre comienza con "T". Luego, imprime los nombres de los dinosaurios que cumplen con esta condición.

y otro código

# Mostrar los dinosaurios cuyos nombres empiezan con "T"

print("Dinosaurios cuyos nombres empiezan con 'T':")

for dino in dinosaurios:

if dino["nombre"].startswith("T"):

print(dino["nombre"])

En este código, iteramos sobre la lista de dinosaurios usando un bucle for. Para cada dinosaurio, verificamos si su nombre comienza con la letra "T" utilizando el método startswith. Si es así, imprimimos el nombre del dinosaurio.

#c) Mostrar todos los dinosaurios que empiecen con T;

**realize ese codigo para la c pero no me compila porque**

def dinosaurio\_T(dinosaurios):

resultado=[] dinosaurios\_aux=dinosaurios()

while dinosaurios\_aux.size()>0:

dino=pila.pop()

if dino["nombre"].startswith("T"):

resultado.append(dino)

dinosaurios\_aux.push(dino)

while dinosaurios\_aux.size()>0:

pila.push(dinosaurios\_aux.pop())

return resultado

El código que proporcionaste parece estar intentando usar operaciones de una pila, pero hay un pequeño error en la línea dinosaurios\_aux=dinosaurios(). Parece que estás intentando llamar a la función dinosaurios, que en realidad es una lista, como si fuera una función.

Supongo que la intención es utilizar una instancia de la clase Stack, pero parece que no la has definido en tu código. Aquí te proporciono una versión corregida y simplificada de tu función utilizando la lista dinosaurios que has proporcionado previamente:

def dinosaurios\_con\_T(dinosaurios):

resultado = []

for dino in dinosaurios:

if dino["nombre"].startswith("T"):

resultado.append(dino)

return resultado

# Mostrar los dinosaurios cuyos nombres empiezan con "T"

dinosaurios\_t = dinosaurios\_con\_T(dinosaurios)

print("Dinosaurios cuyos nombres empiezan con 'T':")

for dino in dinosaurios\_t:

print(dino["nombre"])

**d) Mostrar todos los** dinosaurio **que pesen menos de 275 Kg**

def dinosaurios\_menos\_de\_275kg(dinosaurios):

resultado = []

for dino in dinosaurios:

peso\_str = dino["peso"].split()[0] # Extraer el peso como string (sin "kg")

peso = float(peso\_str) if peso\_str.isdigit() else None # Convertir el peso a float

if peso is not None and peso < 275:

resultado.append(dino)

return resultado

# Mostrar los dinosaurios que pesen menos de 275 Kg

dinosaurios\_ligeros = dinosaurios\_menos\_de\_275kg(dinosaurios)

print("Dinosaurios que pesan menos de 275 Kg:")

for dino in dinosaurios\_ligeros:

print(dino["nombre"])

En este código, la función dinosaurios\_menos\_de\_275kg itera sobre la lista de dinosaurios. Para cada dinosaurio, extrae el peso como una cadena y lo convierte a un número flotante (si es posible). Luego, verifica si el peso es menor a 275 Kg y, si lo es, agrega el dinosaurio a la lista resultado. Finalmente, muestra los nombres de los dinosaurios que cumplen con esta condición.

Ten en cuenta que el peso se extrae como una cadena y se convierte a flotante en caso de ser un número. Esto se hace para manejar casos donde el peso está en un formato diferente o falta en los datos originales.

Si ejecutas este código, obtendrás una lista de los nombres de los dinosaurios que pesan menos de 275 Kg. Si tienes alguna pregunta o necesitas más ayuda, ¡no dudes en decírmelo!

**Dejar en una pila aparte todos los dinosaurios que # comienzan con A, Q, S**

def dinosaurios\_comienzan\_con\_A\_Q\_S(dinosaurios):

pila\_aparte = Stack() # Crear una pila aparte para almacenar los dinosaurios

dinosaurios\_restantes = Stack() # Crear una pila para almacenar los dinosaurios que no empiezan con A, Q, S

# Iterar sobre la lista de dinosaurios

for dino in dinosaurios:

if dino["nombre"][0] in ['A', 'Q', 'S']: # Verificar si el nombre del dinosaurio comienza con A, Q, o S

pila\_aparte.push(dino) # Agregar el dinosaurio a la pila aparte

else:

dinosaurios\_restantes.push(dino) # Agregar el dinosaurio a la pila de dinosaurios restantes

return pila\_aparte, dinosaurios\_restantes

# Obtener los dinosaurios que comienzan con A, Q, S y los restantes

pila\_aparte, dinosaurios\_restantes = dinosaurios\_comienzan\_con\_A\_Q\_S(dinosaurios)

# Mostrar los dinosaurios que comienzan con A, Q, S

print("Dinosaurios que comienzan con A, Q, S:")

while not pila\_aparte.is\_empty():

dino = pila\_aparte.pop()

print(dino["nombre"])

# Mostrar los dinosaurios restantes

print("\nDinosaurios restantes:")

while not dinosaurios\_restantes.is\_empty():

dino = dinosaurios\_restantes.pop()

print(dino["nombre"])

En este código, la función dinosaurios\_comienzan\_con\_A\_Q\_S itera sobre la lista de dinosaurios y verifica si el nombre de cada dinosaurio comienza con las letras "A", "Q" o "S". Si es así, el dinosaurio se agrega a una pila aparte (pila\_aparte). Si no comienza con ninguna de estas letras, se agrega a otra pila (dinosaurios\_restantes).

Luego, el código muestra los dinosaurios que comienzan con A, Q, S y los restantes en dos secciones separadas.

Asegúrate de tener definida la clase Stack para que este código funcione correctamente. Si tienes alguna pregunta o necesitas más ayuda, ¡házmelo saber!

4-Se dispone de una lista de todos los Jedi, de cada uno # de estos se conoce su nombre, maestros, colores de sable # de luz usados y especie. implementar las funciones necesarias # para resolver las actividades enumeradas a continuación:

jedis = [ { "name": "Qui-Gon Jinn", "rank": "Jedi Master", "species": "Human", "master": "Tera Sinube/Count Dooku", "lightsaber\_color": "Green", "homeworld": "Coruscant", "birth\_year": "79ABY", "height": 1.93, "to\_darkside": None, "come\_lightside": None }, { "name": "Obi-Wan Kenobi", "rank": "Jedi Master", "species": "Human", "master": "Qui-Gon Jinn/Yoda", "lightsaber\_color": "Blue", "homeworld": "Stewjon", "birth\_year": "57ABY", "height": 1.82, "to\_darkside": None, "come\_lightside": None }, { "name": "Anakin Skywalker/Darth Vader", "rank": "Jedi Knight", "species": "Human", "master": "Obi-Wan Kenobi", "lightsaber\_color": "Blue", "homeworld": "Tatooine", "birth\_year": "41ABY", "height": 1.88, "to\_darkside": True, "come\_lightside": True }, { "name": "Quinlan Vos", "rank": "Jedi Master", "species": "Human/Kiffar", "master": "Tholme", "lightsaber\_color": "Green", "homeworld": "Kiffu", "birth\_year": "59ABY", "height": 1.91, "to\_darkside": True, "come\_lightside": False }, { "name": "Yoda", "rank": "Grand Master", "species": None, "master": None, "lightsaber\_color": "Green", "homeworld": None, "birth\_year": "896ABY", "height": 0.66, "to\_darkside": None, "come\_lightside": None }, { "name": "Mace Windu", "rank": "Jedi Master/Master of the Order", "species": "Human", "master": "Cyslin Myr", "lightsaber\_color": "Purple", "homeworld": "Haruun Kal", "birth\_year": "72ABY", "height": 1.92, "to\_darkside": None, "come\_lightside": None }, { "name": "Ki-Adi-Mundi", "rank": "Jedi Master", "species": "Cerean", "master": None, "lightsaber\_color": "Purple/Blue", "homeworld": "Cerea", "birth\_year": "92ABY", "height": 1.98, "to\_darkside": None, "come\_lightside": None }, { "name": "Plo Koon", "rank": "Jedi Master", "species": "Kel Dor", "master": None, "lightsaber\_color": "Yellow/Blue/Orange", "homeworld": "Dorin", "birth\_year": "71ABY", "height": 1.88, "to\_darkside": None, "come\_lightside": None }, { "name": "Saesee Tiin", "rank": "Jedi Master", "species": "Iktotchi", "master": None, "lightsaber\_color": "Green", "homeworld": "Iktotch", "birth\_year": None, "height": 1.93, "to\_darkside": None, "come\_lightside": None }, { "name": "Yaddle", "rank": "Jedi Master", "species": None, "master": None, "lightsaber\_color": "Green", "homeworld": None, "birth\_year": "509AYB", "height": 0.61, "to\_darkside": None, "come\_lightside": None }, { "name": "Even Piell", "rank": "Jedi Master", "species": "Lannik", "master": None, "lightsaber\_color": "Green", "homeworld": "Lannik", "birth\_year": None, "height": 1.22, "to\_darkside": None, "come\_lightside": None }, { "name": "Oppo Rancisis", "rank": "Jedi Master", "species": "Thisspiasian", "master": "Yaddle", "lightsaber\_color": "Green", "homeworld": "Thisspias", "birth\_year": "206ABY", "height": 1.38, "to\_darkside": None, "come\_lightside": None }, { "name": "Adi Gallia", "rank": "Jedi Master", "species": "Tholothian", "master": None, "lightsaber\_color": "Green/Blue", "homeworld": "Coruscant", "birth\_year": None, "height": 1.84, "to\_darkside": None, "come\_lightside": None }, { "name": "Yarael Poof", "rank": "Jedi Master", "species": "Quermian", "master": None, "lightsaber\_color": "Blue", "homeworld": "Quermia", "birth\_year": None, "height": 2.64, "to\_darkside": None, "come\_lightside": None }, { "name": "Eeth Koth", "rank": "Jedi Master", "species": "Zabrak", "master": None, "lightsaber\_color": "Green/Blue", "homeworld": "Iridonia", "birth\_year": None, "height": 1.71, "to\_darkside": None, "come\_lightside": None }, { "name": "Depa Billaba", "rank": "Jedi Master", "species": "Chalactan", "master": None, "lightsaber\_color": "Green", "homeworld": "Chalacta", "birth\_year": None, "height": 1.68, "to\_darkside": None, "come\_lightside": None }, { "name": "Kit Fisto", "rank": "Jedi Master", "species": "Nautolan", "master": None, "lightsaber\_color": "Green", "homeworld": "Glee Anselm", "birth\_year": None, "height": 1.96, "to\_darkside": None, "come\_lightside": None }, { "name": "Luminara Unduli", "rank": "Jedi Master", "species": "Mirialan", "master": None, "lightsaber\_color": "Green", "homeworld": "Mirial", "birth\_year": "58ABY", "height": 1.76, "to\_darkside": None, "come\_lightside": None }, { "name": "Barriss Offee", "rank": "Padawan", "species": "Mirialan", "master": "Luminara Unduli", "lightsaber\_color": "Blue", "homeworld": "Mirial", "birth\_year": "40ABY", "height": 1.66, "to\_darkside": True, "come\_lightside": False }, { "name": "Shaak Ti", "rank": "Jedi Master", "species": "Togruta", "master": None, "lightsaber\_color": "Blue", "homeworld": "Shili", "birth\_year": None, "height": 1.87, "to\_darkside": None, "come\_lightside": None }, { "name": "Coleman Trebor", "rank": "Jedi Master", "species": "Vurk", "master": None, "lightsaber\_color": "Green", "homeworld": "Sembla", "birth\_year": None, "height": 2.13, "to\_darkside": None, "come\_lightside": None }, { "name": "Jocasta Nu", "rank": "Jedi Master", "species": "Human", "master": None, "lightsaber\_color": "Blue", "homeworld": "Coruscant", "birth\_year": None, "height": 1.69, "to\_darkside": None, "come\_lightside": None }, { "name": "Aayla Secura", "rank": "Jedi Knight", "species": "Twi'lek", "master": "Quinlan Vos", "lightsaber\_color": "Blue", "homeworld": "Ryloth", "birth\_year": "47ABY", "height": 1.72, "to\_darkside": None, "come\_lightside": None }, { "name": "Sifo-Dyas", "rank": "Jedi Master", "species": "Human", "master": None, "lightsaber\_color": "Green", "homeworld": "Mundos Cassandranos", "birth\_year": "75ABY", "height": None, "to\_darkside": None, "come\_lightside": None }, { "name": "Count Dooku", "rank": "Jedi Master", "species": "Human", "master": "Yoda", "lightsaber\_color": "Blue/Red", "homeworld": "Serenno", "birth\_year": "102ABY", "height": 1.93, "to\_darkside": True, "come\_lightside": False }, { "name": "Pablo-Jill", "rank": "Jedi Knight", "species": None, "master": None, "lightsaber\_color": "Blue", "homeworld": "Cúmulo Estelar Skustell", "birth\_year": None, "height": 1.60, "to\_darkside": None, "come\_lightside": None }, { "name": "Bultar Swan", "rank": "Jedi Knight", "species": "Human", "master": "Plo Koon", "lightsaber\_color": "Green", "homeworld": "Kuat", "birth\_year": None, "height": 1.68, "to\_darkside": None, "come\_lightside": None }, { "name": "Agen Kolar", "rank": "Jedi Master", "species": "Zabrak", "master": None, "lightsaber\_color": "Green/Blue", "homeworld": "Coruscant", "birth\_year": None, "height": 1.90, "to\_darkside": None, "come\_lightside": None }, { "name": "Stass Allie", "rank": "Jedi Master", "species": "Tholothian", "master": None, "lightsaber\_color": "Green", "homeworld": "Tholoth", "birth\_year": None, "height": 1.80, "to\_darkside": None, "come\_lightside": None }, { "name": "Ahsoka Tano", "rank": "Padawan", "species": "Togruta", "master": "Anakin Skywalker", "lightsaber\_color": "Green/Yellow/Blue/White", "homeworld": "Shili", "birth\_year": "36ABY", "height": 1.88, "to\_darkside": None, "come\_lightside": None }, { "name": "Asajj Ventress", "rank": "Padawan", "species": "Dathomirian", "master": "Ky Narec", "lightsaber\_color": "Yellow/Red", "homeworld": "Dathomir", "birth\_year": None, "height": 1.80, "to\_darkside": True, "come\_lightside": False }, { "name": "Ima-Gun Di", "rank": "Jedi Master", "species": "Nikto", "master": None, "lightsaber\_color": "Blue", "homeworld": None, "birth\_year": None, "height": 1.92, "to\_darkside": None, "come\_lightside": None }, { "name": "Nahdar Vebb", "rank": "Jedi Knight", "species": "Mon Calamari", "master": "Kit Fisto", "lightsaber\_color": "Blue", "homeworld": "Dac", "birth\_year": None, "height": None, "to\_darkside": None, "come\_lightside": None }, { "name": "Bolla Ropal", "rank": "Jedi Master", "species": "Rodian", "master": None, "lightsaber\_color": "Blue", "homeworld": "Rodia", "birth\_year": None, "height": 1.75, "to\_darkside": None, "come\_lightside": None }, { "name": "Ord Enisence", "rank": "Jedi Master", "species": "Skrilling", "master": None, "lightsaber\_color": "Green", "homeworld": "Agrimundo-2079", "birth\_year": None, "height": 1.83, "to\_darkside": None, "come\_lightside": None }, { "name": "Tera Sinube", "rank": "Jedi Master", "species": "Cosian", "master": None, "lightsaber\_color": "Blue", "homeworld": "Cosia", "birth\_year": "102ABY", "height": 1.83, "to\_darkside": None, "come\_lightside": None }, { "name": "Ky Narec", "rank": "Jedi Master", "species": "Human", "master": None, "lightsaber\_color": "Green", "homeworld": None, "birth\_year": None, "height": None, "to\_darkside": None, "come\_lightside": None }, { "name": "Pong Krell", "rank": "Jedi Master", "species": "Besalisk", "master": None, "lightsaber\_color": "Blue/Green", "homeworld": "Ojom", "birth\_year": None, "height": 2.36, "to\_darkside": True, "come\_lightside": False }, { "name": "Coleman Kcaj", "rank": "Jedi Master", "species": "Ongree", "master": None, "lightsaber\_color": "Green", "homeworld": "Skustell", "birth\_year": None, "height": None, "to\_darkside": None, "come\_lightside": None }, { "name": "Tiplar", "rank": "Jedi Master", "species": "Mikkian", "master": None, "lightsaber\_color": "Green", "homeworld": None, "birth\_year": None, "height": None, "to\_darkside": None, "come\_lightside": None }, { "name": "Tiplee", "rank": "Jedi Master", "species": "Mikkian", "master": None, "lightsaber\_color": "Blue", "homeworld": None, "birth\_year": None, "height": None, "to\_darkside": None, "come\_lightside": None }, { "name": "Tu-Anh", "rank": "Jedi Master", "species": "Human", "master": None, "lightsaber\_color": None, "homeworld": None, "birth\_year": None, "height": None, "to\_darkside": None, "come\_lightside": None }, { "name": "Kanan Jarrus", "rank": "Jedi Master", "species": "Human", "master": "Depa Billaba", "lightsaber\_color": "Blue", "homeworld": "Coruscant", "birth\_year": "33ABY", "height": 1.90, "to\_darkside": None, "come\_lightside": None }, { "name": "Ezra Bridger", "rank": "Padawan", "species": "Human", "master": "Kanan Jarrus", "lightsaber\_color": "Blue", "homeworld": "Lothal", "birth\_year": "19ABY", "height": 1.65, "to\_darkside": None, "come\_lightside": None }, { "name": "Luke Skywalker", "rank": "Jedi Master", "species": "Human", "master": "Obi-Wan Kenobi/Yoda", "lightsaber\_color": "Green/Blue", "homeworld": "Tatooine", "birth\_year": "19ABY", "height": 1.72, "to\_darkside": None, "come\_lightside": None }, { "name": "Leia Organa", "rank": "Jedi Master", "species": "Human", "master": "Luke Skywalker", "lightsaber\_color": "Blue", "homeworld": "Alderaan", "birth\_year": "19ABY", "height": 1.50, "to\_darkside": None, "come\_lightside": None }, { "name": "Ben Solo/Kylo Ren", "rank": "Padawan", "species": "Human", "master": "Luke Skywalker", "lightsaber\_color": "Green/Blue", "homeworld": "Chandrila", "birth\_year": "5DBY", "height": 1.89, "to\_darkside": True, "come\_lightside": True }, { "name": "Rey Skywalker", "rank": "Jedi Sentinel", "species": "Human", "master": "Luke Skywalker/Leia Organa", "lightsaber\_color": "Blue/Green/Yellow", "homeworld": "Jakku", "birth\_year": "15DYB", "height": 1.70, "to\_darkside": None, "come\_lightside": None } ]

1. listado ordenado por nombre y por especie;

jedis = [

# Lista de Jedi como se proporcionó en la pregunta

]

def sort\_jedis\_by\_name(jedis):

return sorted(jedis, key=lambda jedi: jedi['name'])

def sort\_jedis\_by\_species(jedis):

return sorted(jedis, key=lambda jedi: jedi['species'] if jedi['species'] else "")

# Ordenar por Nombre

sorted\_by\_name = sort\_jedis\_by\_name(jedis)

print("Jedi Ordenados por Nombre:")

for jedi in sorted\_by\_name:

print(jedi['name'])

# Ordenar por Especie

sorted\_by\_species = sort\_jedis\_by\_species(jedis)

print("Jedi Ordenados por Especie:")

for jedi in sorted\_by\_species:

print(f"{jedi['species']} - {jedi['name']}")

Este código definirá las funciones necesarias para ordenar la lista de Jedi tanto por nombre como por especie, y luego demostrará cómo usarlas para obtener y mostrar la lista ordenada.

**Puedes cambiar el código**

jedis\_ordenados = sorted(jedis, key=lambda x: (x["nombre"], x["species"])) for jedi in jedis\_ordenados: nombre = jedi['name'] especie = jedi['species'] print(f"Nombre: {nombre}, Especie: {especie}") print() me puedes cambiar un poco el código

# Ordenar la lista de Jedi por nombre y especie

jedis\_ordenados = sorted(jedis, key=lambda x: (x["name"], x["species"]))

# Imprimir la lista ordenada de Jedi

print("Listado de Jedi ordenados por nombre y especie:")

for jedi in jedis\_ordenados:

nombre = jedi['name']

especie = jedi['species'] if jedi['species'] else "Desconocida"

print(f"Nombre: {nombre}, Especie: {especie}")

**Explicación del Código**

1. **Ordenación de la Lista**:
   * sorted(jedis, key=lambda x: (x["name"], x["species"])): Ordena la lista de diccionarios jedis por el valor de las claves name y species.
2. **Impresión de la Lista**:
   * print("Listado de Jedi ordenados por nombre y especie:"): Imprime un encabezado para la lista.
   * for jedi in jedis\_ordenados: Itera sobre cada jedi en la lista ordenada.
   * especie = jedi['species'] if jedi['species'] else "Desconocida": Asigna "Desconocida" si el valor de la especie es None.
   * print(f"Nombre: {nombre}, Especie: {especie}"): Imprime el nombre y la especie de cada Jedi en un formato claro.

mostrar todos los padawan de Yoda y Luke Skywalker, # es decir sus aprendices;

def mostrar\_padawan\_de\_maestros(jedis, maestros):

# Crear una lista para almacenar los padawan encontrados

padawans = []

# Iterar sobre la lista de Jedi

for jedi in jedis:

# Verificar si el Jedi tiene un maestro en la lista de maestros proporcionada

if jedi["master"] and any(maestro in jedi["master"] for maestro in maestros):

padawans.append(jedi)

# Imprimir los padawan encontrados

if padawans:

print(f"Padawan de {', '.join(maestros)}:")

for padawan in padawans:

print(f"Nombre: {padawan['name']}, Maestro(s): {padawan['master']}")

else:

print(f"No se encontraron padawan de {', '.join(maestros)}")

# Lista de maestros para buscar sus padawan

maestros\_buscar = ["Yoda", "Luke Skywalker"]

# Llamar a la función para mostrar los padawan de Yoda y Luke Skywalker

mostrar\_padawan\_de\_maestros(jedis, maestros\_buscar)

**Explicación del Código**

1. **Definición de la Función**:
   * def mostrar\_padawan\_de\_maestros(jedis, maestros): Define una función que toma la lista de jedis y una lista de maestros como argumentos.
2. **Inicialización de la Lista de Padawan**:
   * padawans = []: Crea una lista vacía para almacenar los padawan encontrados.
3. **Iteración Sobre la Lista de Jedi**:
   * for jedi in jedis: Itera sobre cada Jedi en la lista de jedis.
4. **Verificación de Maestros**:
   * if jedi["master"] and any(maestro in jedi["master"] for maestro in maestros): Verifica si el Jedi tiene un maestro y si alguno de los maestros proporcionados está en la clave "master" del Jedi.
5. **Almacenamiento de Padawan**:
   * padawans.append(jedi): Si se cumple la condición anterior, agrega el Jedi a la lista de padawan.
6. **Impresión de los Padawan**:
   * if padawans: Verifica si hay padawan en la lista.
   * print(f"Padawan de {', '.join(maestros)}:"): Imprime los nombres de los maestros.
   * for padawan in padawans: Itera sobre la lista de padawan e imprime su nombre y maestro(s).

**Resultado Esperado**

El resultado mostrará los padawan que tuvieron a Yoda y/o Luke Skywalker como maestros, por ejemplo:

**si sirve este codigo me lo podrias cabiar un poco**

for jedi in jedis: if jedi["name"] in ["Ahsoka Tano", "Kit Fisto"]: maestro = jedi.get("master", "Desconocido") color\_sable = jedi.get("lightsaber\_color", "Desconocido") print(f"Nombre: {jedi['name']}, Maestro: {maestro}, Color de sable: {color\_sable}, Especie: {jedi['species']}") print()

# Definir una función para mostrar padawans específicos de ciertos maestros

def mostrar\_padawans(jedis, padawans\_a\_buscar):

for jedi in jedis:

if jedi["name"] in padawans\_a\_buscar:

maestro = jedi.get("master", "Desconocido")

color\_sable = jedi.get("lightsaber\_color", "Desconocido")

especie = jedi.get("species", "Desconocido")

print(f"Nombre: {jedi['name']}, Maestro: {maestro}, Color de sable: {color\_sable}, Especie: {especie}")

print() # Línea en blanco al final para mejor separación visual

# Lista de padawans a buscar

padawans\_a\_buscar = ["Ahsoka Tano", "Kit Fisto"]

# Llamar a la función con la lista de jedis y los padawans a buscar

mostrar\_padawans(jedis, padawans\_a\_buscar)

En este código, he hecho lo siguiente:

1. **Definición de la Función**:
   * def mostrar\_padawans(jedis, padawans\_a\_buscar): Define una función que toma la lista de jedis y una lista de nombres de padawans a buscar como argumentos.
2. **Iteración Sobre la Lista de Jedi**:
   * for jedi in jedis: Itera sobre cada Jedi en la lista de jedis.
3. **Verificación de Nombres de Padawans**:
   * if jedi["name"] in padawans\_a\_buscar: Verifica si el nombre del Jedi está en la lista de padawans a buscar.
4. **Obtención de Detalles del Jedi**:
   * maestro = jedi.get("master", "Desconocido"): Obtiene el nombre del maestro del Jedi, o "Desconocido" si no tiene un maestro.
   * color\_sable = jedi.get("lightsaber\_color", "Desconocido"): Obtiene el color del sable de luz del Jedi, o "Desconocido" si no tiene un color de sable especificado.
   * especie = jedi.get("species", "Desconocido"): Obtiene la especie del Jedi, o "Desconocido" si no tiene una especie especificada.
5. **Impresión de Detalles del Jedi**:
   * print(f"Nombre: {jedi['name']}, Maestro: {maestro}, Color de sable: {color\_sable}, Especie: {especie}"): Imprime los detalles del Jedi.
6. **Línea en Blanco al Final**:
   * print(): Añade una línea en blanco al final para mejor separación visual en la salida.

Puedes llamar a esta función con diferentes listas de padawans a buscar según necesites.

**c- mostrar todos los padawan de Yoda y Luke Skywalker, es decir sus aprendices**;

padawans\_yoda = [] for jedi in jedis: if jedi["master"] is not None and "Yoda" in jedi["master"]: padawans\_yoda.append(jedi["name"]) print("Padawans de Yoda:", end=" ") if padawans\_yoda: print(", ".join(padawans\_yoda)) else: print("Ninguno") padawans\_luke = [] for jedi in jedis: if jedi["master"] is not None and "Luke Skywalker" in jedi["master"]: padawans\_luke.append(jedi["name"]) print("Padawans de Luke Skywalker:", end=" ") if padawans\_luke: print(", ".join(padawans\_luke)) else: print("Ninguno") print() podrias modificarme un poco ese codigo que realize para la actividad 3

Claro, aquí tienes una versión modificada de tu código para mostrar todos los padawan de Yoda y Luke Skywalker:

def obtener\_padawans(jedis, maestro\_buscado):

padawans = []

for jedi in jedis:

if jedi.get("master") and maestro\_buscado in jedi["master"]:

padawans.append(jedi["name"])

return padawans

padawans\_yoda = obtener\_padawans(jedis, "Yoda")

padawans\_luke = obtener\_padawans(jedis, "Luke Skywalker")

def imprimir\_padawans(maestro, padawans):

print(f"Padawans de {maestro}:", end=" ")

if padawans:

print(", ".join(padawans))

else:

print("Ninguno")

imprimir\_padawans("Yoda", padawans\_yoda)

imprimir\_padawans("Luke Skywalker", padawans\_luke)

print()

**Modificaciones realizadas:**

1. **Definición de una función para obtener los padawans**:
   * def obtener\_padawans(jedis, maestro\_buscado): Define una función que toma la lista de jedis y el nombre del maestro buscado, y devuelve una lista de nombres de padawans.
2. **Uso de la función para obtener padawans**:
   * padawans\_yoda = obtener\_padawans(jedis, "Yoda"): Obtiene los padawans de Yoda.
   * padawans\_luke = obtener\_padawans(jedis, "Luke Skywalker"): Obtiene los padawans de Luke Skywalker.
3. **Definición de una función para imprimir los padawans**:
   * def imprimir\_padawans(maestro, padawans): Define una función que toma el nombre del maestro y una lista de padawans, e imprime los nombres de los padawans.
4. **Uso de la función para imprimir los padawans**:
   * imprimir\_padawans("Yoda", padawans\_yoda): Imprime los padawans de Yoda.
   * imprimir\_padawans("Luke Skywalker", padawans\_luke): Imprime los padawans de Luke Skywalker.

**Beneficios de las modificaciones:**

* **Reutilización de código**: Las funciones obtener\_padawans e imprimir\_padawans pueden reutilizarse para otros maestros y padawans si es necesario.
* **Claridad y mantenimiento**: El código es más claro y fácil de mantener al separar la lógica en funciones específicas.
* **Flexibilidad**: Puedes fácilmente modificar o extender las funciones para incluir más información o cambiar el formato de salida.

-mostrar los Jedi de especie humana y twi'lek;

for jedi in jedis:

if jedi["species"] == "Human" or jedi["species"] == "Twi'lek":

print(f"Nombre: {jedi['name']}, Especie: {jedi['species']}")

print()

me podrias cambiar un poco el codigo

def mostrar\_jedi\_por\_especie(jedis, especies):

for jedi in jedis:

if jedi["species"] in especies:

print(f"Nombre: {jedi['name']}, Especie: {jedi['species']}")

print()

especies\_deseadas = ["Human", "Twi'lek"]

mostrar\_jedi\_por\_especie(jedis, especies\_deseadas)

**Modificaciones realizadas:**

1. **Definición de una función para mostrar los Jedi por especie**:
   * def mostrar\_jedi\_por\_especie(jedis, especies): Define una función que toma la lista de jedis y una lista de especies deseadas, e imprime los Jedi que pertenecen a esas especies.
2. **Uso de la función con la lista de especies deseadas**:
   * especies\_deseadas = ["Human", "Twi'lek"]: Define una lista de especies que se desean mostrar.
   * mostrar\_jedi\_por\_especie(jedis, especies\_deseadas): Llama a la función mostrar\_jedi\_por\_especie con la lista de jedis y la lista de especies deseadas como argumentos.

**Beneficios de las modificaciones:**

* **Reutilización de código**: La función mostrar\_jedi\_por\_especie puede ser utilizada para mostrar Jedi de cualquier combinación de especies.
* **Claridad y mantenimiento**: El código es más claro y fácil de mantener al separar la lógica en una función específica.
* **Flexibilidad**: Puedes fácilmente modificar la lista de especies deseadas para mostrar Jedi de otras especies si es necesario.

mostrar los Jedi que usaron sable de luz de más de un # color;

**me podrias cambiar un poco el codigo mejorarlo**

for jedi in jedis:

if jedi["lightsaber\_color"] is not None and "/" in

jedi["lightsaber\_color"]:

print(f"Nombre: {jedi['name']}, Colores de sable:

{jedi['lightsaber\_color']}")

print()

me podrias cambiar un poco el codigo mejorarlo

def mostrar\_jedi\_multiples\_colores(jedis):

for jedi in jedis:

colores\_sable = jedi["lightsaber\_color"]

if colores\_sable is not None and "/" in colores\_sable:

print(f"Nombre: {jedi['name']}, Colores de sable: {colores\_sable}")

print()

mostrar\_jedi\_multiples\_colores(jedis)

**Modificaciones realizadas:**

1. **Definición de una función para mostrar los Jedi con múltiples colores de sable**:
   * def mostrar\_jedi\_multiples\_colores(jedis): Define una función que toma la lista de jedis y muestra aquellos que usaron sables de luz de más de un color.
2. **Uso de una variable local para los colores de sable**:
   * colores\_sable = jedi["lightsaber\_color"]: Almacena los colores de sable del Jedi en una variable local para mejorar la legibilidad del código.
3. **Mejora en el nombre de la función**:
   * Cambié el nombre de la función para que sea más descriptivo y refleje mejor su propósito.

**Beneficios de las modificaciones:**

* **Abstracción de la lógica en una función**: El código ahora está encapsulado dentro de una función específica, lo que facilita su reutilización y mantenimiento.
* **Mayor legibilidad**: Al usar una variable local para los colores del sable, el código es más claro y fácil de entender.
* **Modularidad**: La función puede ser fácilmente modificada o extendida si se necesitan cambios en el futuro.

puede ser otro codigo un poco mas parecido al que yo realice

def mostrar\_jedi\_multiples\_colores(jedis):

for jedi in jedis:

if "/" in jedi.get("lightsaber\_color", ""):

nombre = jedi.get("name", "Desconocido")

colores\_sable = jedi.get("lightsaber\_color", "Desconocido")

print(f"Nombre: {nombre}, Colores de sable: {colores\_sable}")

print()

mostrar\_jedi\_multiples\_colores(jedis)

**Modificaciones realizadas:**

1. **Uso de get para manejar claves inexistentes**:
   * jedi.get("lightsaber\_color", ""): Utiliza el método get para obtener el valor de la clave "lightsaber\_color" del diccionario jedi. Si la clave no está presente, devuelve una cadena vacía para evitar errores.
2. **Mejora en la legibilidad del código**:
   * Al utilizar el método get, el código se vuelve más legible al proporcionar un valor predeterminado en caso de que la clave no exista en el diccionario.
3. **Utilización de variables locales para el nombre y los colores del sable**:
   * nombre = jedi.get("name", "Desconocido"): Almacena el nombre del Jedi en una variable local para mejorar la legibilidad del código.
   * colores\_sable = jedi.get("lightsaber\_color", "Desconocido"): Almacena los colores del sable en una variable local para facilitar su uso en la impresión.

**Beneficios de las modificaciones:**

* **Manejo seguro de claves inexistentes**: El uso del método get permite manejar de manera segura claves que podrían no existir en el diccionario, lo que evita posibles errores.
* **Mayor legibilidad**: Al utilizar variables locales para el nombre y los colores del sable, el código se vuelve más claro y fácil de entender.
* **Mantenimiento sencillo**: La estructura del código es similar a la original, lo que facilita el mantenimiento y la comprensión para quien lo lee.

3.5

puede haber un codigo sin barra divisoria

ChatGPT

Claro, puedes usar la función split() para dividir los colores del sable de luz y luego verificar si hay más de un color. Aquí tienes una versión modificada del código sin usar la barra divisoria:

def mostrar\_jedi\_multiples\_colores(jedis):

for jedi in jedis:

colores\_sable = jedi.get("lightsaber\_color", "")

if colores\_sable and len(colores\_sable.split("/")) > 1:

nombre = jedi.get("name", "Desconocido")

print(f"Nombre: {nombre}, Colores de sable: {colores\_sable}")

print()

mostrar\_jedi\_multiples\_colores(jedis)

En este código, split("/") divide la cadena de colores del sable de luz en una lista de colores individuales utilizando la barra como delimitador. Luego, verificamos si hay más de un color en la lista utilizando len() y comprobamos si es mayor que 1. Si es así, imprimimos el nombre del Jedi y sus colores de sable.

3.5

**indicar los nombre de los padawans de Qui-Gon Jin y # Mace Windu, si los tuvieron.**

Puedes hacer esto recorriendo la lista de Jedi y buscando aquellos que tuvieron a Qui-Gon Jinn o Mace Windu como maestros. Aquí tienes un código que realiza esta tarea:

def mostrar\_padawans\_maestros(jedis):

padawans\_quigon = []

padawans\_mace = []

for jedi in jedis:

maestro = jedi.get("master", "")

nombre = jedi.get("name", "Desconocido")

if "Qui-Gon Jinn" in maestro:

padawans\_quigon.append(nombre)

if "Mace Windu" in maestro:

padawans\_mace.append(nombre)

print("Padawans de Qui-Gon Jinn:", end=" ")

if padawans\_quigon:

print(", ".join(padawans\_quigon))

else:

print("Ninguno")

print("Padawans de Mace Windu:", end=" ")

if padawans\_mace:

print(", ".join(padawans\_mace))

else:

print("Ninguno")

print()

mostrar\_padawans\_maestros(jedis)

no funciona el código

padawans\_quigon = []

for jedi in jedis:

if jedi["master"] is not None and "Qui-Gon Jinn" in

jedi["master"]:

padawans\_quigon.append(jedi["name"])

print("Padawans de Qui-Gon Jinn:", end=" ")

if padawans\_quigon:

print(", ".join(padawans\_quigon))

else: print("Ninguno")

padawans\_mace = []

for jedi in jedis: if jedi["master"] is not None and isinstance(jedi["master"], str)

and "Mace Windu" in jedi["master"]:

padawans\_mace.append(jedi["name"])

print("Padawans de Mace Windu:", end=" ")

if padawans\_mace:

print(", ".join(padawans\_mace))

else: print("Ninguno")

print()

**realize este codigo me lo puedes mejorar**

def mostrar\_padawans\_maestros(jedis, maestro\_buscar):

padawans = []

for jedi in jedis:

maestro = jedi.get("master", "")

nombre = jedi.get("name", "Desconocido")

if isinstance(maestro, str) and maestro.startswith(maestro\_buscar):

padawans.append(nombre)

print(f"Padawans de {maestro\_buscar}:", end=" ")

if padawans:

print(", ".join(padawans))

else:

print("Ninguno")

mostrar\_padawans\_maestros(jedis, "Qui-Gon Jinn")

mostrar\_padawans\_maestros(jedis, "Mace Windu")

print()