Python Tutor: Guía Maestra de Resolución y FAQ

Respuestas, Pistas y Soluciones para todos los ejercicios.

Índice de Contenidos

Módulo 00: Bienvenida y Guía

Ejercicio 1: Guía de Inicio y Créditos

Q: ¿De qué trata este ejercicio?

🚀 Módulo 00 — Bienvenida y Guía del Curso

(Versión Hacker Futurista)

👋 Bienvenido al Curso de Python orientado a Ciberseguridad

Autor: Carlos Domínguez **Asistente Tecnológico:** IA Generativa

@r. Has accedido al entorno de entrenamiento donde aprenderás Python desde cero hasta un nivel avanz
real en:

🛠 Automatización
🔐 Ciberseguridad y criptografía
🧠 Análisis técnico
⚙️ Creación de herramientas y scripts operativos

o aprenderás a programar... **Aprenderás a pensar como un desarrollador, como un analista, como un h**

🎯 ¿Qué encontrarás en este curso?

Cada módulo contiene:

Explicaciones claras
Ejemplos reales
Ejercicios interactivos
Código "escribiéndose en tiempo real" estilo hacking

Módulo 01: Introducción

Ejercicio 2: Hola Mundo y Print

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Hola Mundo y Print

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 01 - print()
# El módulo print() es el módulo que nos permite imprimir en consola
# Sirve para mostrar información en consola y te va a acompañar
# TODA TU VIDA. Desde hoy hasta el fin de los tiempos
###

# Este es un ejemplo básico de cómo imprimir un texto en consola
print("¡Hola, Twitch!")

# También puedes usar comillas simples para imprimir texto
print('Esto también funciona con una comilla')
```

Ejercicio 3: Tipos de Datos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Tipos de Datos

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 02 - types()
# Python tiene varios tipos de datos
# int, float, complex, str, bool, NoneType, list, tuple, dict, range, set...
###



"""
El comando `type()` devuelve el tipo de un objeto en Python
"""
print("int:")  # Enteros (números sin parte decimal)
print(type(10))  # Número entero positivo
print(type(0))  # El número cero también es un entero
print(type(-5))  # Número entero negativo
print(type(723842472378427893478923987 4))  # Python permite enteros de gran tamaño
```

Ejercicio 4: Conversión de Tipos (Casting)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Conversión de Tipos (Casting)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 03 - casting de types
# Transformar un tipo de un valor a otro
###



print("Conversión de tipos")

# Convertir una cadena que contiene un número a un entero y sumarlo con otro entero
print(2 + int("100"))  # Convierte "100" a entero y suma 2. Resultado: 102

# Convertir un entero a cadena para concatenarlo con otra cadena
nt("100" + str(2))  # Convierte el número 2 a cadena y lo concatena. Resultado: "10

# Convertir una cadena con un número decimal a tipo float
```

Ejercicio 5: Variables

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Variables

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
##
# 04 - Variables
# Las variables sirven para guardar datos en memoria.
# Python es un lenguaje de tipado dinámico y de tipado fuerte.
###



asignar una variable solo hace falta poner el nombre de la variable y asignarle ur
my_name = "midudev"
print(my_name)  # Imprime el valor de la variable my_name

age = 32
print(age)  # Imprime el valor de la variable age)

# Reasignar un nuevo valor a una variable existente
```

Ejercicio 6: Entrada de Datos (Input)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Entrada de Datos (Input)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 05 - Entrada de usuario (input()) - Versión simplificada
# La función input() permite obtener datos del usuario a través de la consola.
###



# Para obtener datos del usuario se usa la función input()
# La función input() recibe un mensaje que se muestra al usuario
# y devuelve el valor introducido por el usuario
# nombre = input("Hola, ¿cómo te llamas?\n")
nombre = "Carlos" # Valor hardcodeado para demo
print(f"Hola {nombre}, encantado de conocerte")

# Ten en cuenta que la función input() devuelve un string
# Así que si queremos obtener un número se debe convertir el string a un número
```

Ejercicio 7: Ejercicios Básicos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Ejercicios Básicos

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# exercises.py
# Ejercicios para practicar los conceptos aprendidos en las lecciones.
###



print("\nEjercicio 1: Imprimir mensajes")
print("Escribe un programa que imprima tu nombre y tu ciudad en líneas separadas.")

### Completa aquí

print("--------------")

print("\nEjercicio 2: Muestra los tipos de datos de las siguientes variables:")
print("Usa el comando 'type()' para determinar el tipo de datos de cada variable.")
```

Ejercicio 8: Condicionales (If)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Condicionales (If)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 01 - Sentencias condicionales (if, elif, else)
# Permiten ejecutar bloques de código solo si se cumplen ciertas condiciones.
###



print("\n Sentencia simple condicional")
# Podemos usar la palabra clave "if" para ejecutar un bloque de código
# solo si se cumple una condición.
edad = 18
if edad >= 18:
print("Eres mayor de edad")
print("¡Felicidades!")

# Si no se cumple la condición, no se ejecuta el bloque de código
```

Ejercicio 9: Booleanos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Booleanos

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 02 - Booleanos
# Valores lógicos: True (verdadero) y False (falso).
# Fundamentales para el control de flujo y la lógica en programación.
###



# Los booleanos representan valores de verdad: True o False.
print("\nValores booleanos básicos:")
print(True)
print(False)

# Operadores de comparación: devuelven un valor booleano.
print("\nOperadores de comparación:")
print("5 > 3:", 5 > 3)        # True
```

Ejercicio 10: Listas

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Listas

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 03 - Listas
# Secuencias mutables de elementos.
# Pueden contener elementos de diferentes tipos.
###



# Creación de listas
print("\nCrear listas")
lista1 = [1, 2, 3, 4, 5] # lista de enteros
lista2 = ["manzanas", "peras", "plátanos"] # lista de cadenas
lista3 = [1, "hola", 3.14, True] # lista de tipos mixtos

lista_vacia = []
lista_de_listas = [[1, 2], ['calcetin', 4]]
```

Ejercicio 11: Métodos de Listas

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Métodos de Listas

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 04 - Listas Métodos
# Los métodos más importantes para trabajar con listas
###



# Creamos una lista con valores
lista1 = ['a', 'b', 'c', 'd']

# Añadir o insertar elementos a la lista
lista1.append('e') # Añade un elemento al final
print(lista1)

insert(1, '@') # Inserta un elemento en la posición que le indiquemos como primer a
print(lista1)
```

Ejercicio 12: Bucle While

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Bucle While

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 01 - Bucles (while)
Permiten ejecutar un bloque de código repetidamente mientras se cumpla una condici
###



print("\n Bucle while:")

# Bucle con una simple condición
contador = 0

while contador <= 5:
    print(contador)
contador += 1 # es super importante para evitar un bucle infinito
```

Ejercicio 13: Bucle For

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Bucle For

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 02 - Bucles (for)
rmiten ejecutar un bloque de código repetidamente mientras ITERA un iterable o una
###



print("\nBucle for:")

# Iterar una lista
frutas = ["manzana", "pera", "mandarina"]
for fruta in frutas:
    print(fruta)

# Iterar sobre cualquier cosa que sea iterable
cadena = "midudev"
```

Ejercicio 14: Rango (Range)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Rango (Range)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
                                ###
                            # 03 - range()
Permite crear una secuencia de números. Puede ser útil para for, pero no solo para
                                ###



                        print("\nrange():")

                # Generado una secuencia de números del 0 al 9
                        for num in range(10):
                                print(num)

                        # range(inicio, fin)
                        for num in range(5, 10):
                                print(num)
```

Ejercicio 15: Funciones

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Funciones

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 04 - Funciones
# Bloques de código reutilizables y parametrizables para hacer tareas especificas
###



# """ Definición de una función

# def nombre_de_la_funcion(parametro1, parametro2, ...):
#   # docstring
#   # cuerpo de la función
#   return valor_de_retorno # opcional

# """
```

Ejercicio 16: Reto: Los 4 Fantásticos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Reto: Los 4 Fantásticos

Q: ¿Cuál es el objetivo principal?

Implementa la función para resolver el reto.

Q: Estoy atascado, ¿alguna pista?

Usa text.upper() y text.count().

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
"""
¿Está en Equilibrio la Alianza entre Reed Richards y Johnny Storm?

Objetivo:
ta función debe contar cuántas veces aparece la letra R (para Reed Richards) y cuán

- Si la cantidad de R y la cantidad de J son iguales, retorna True.
- Si no, retorna False.
- Si no hay ni R ni J, retorna True.
"""

def check_is_balanced(text):
    # --- TU CÓDIGO AQUÍ ---
# 1. Convierte el texto a mayúsculas para facilitar el conteo
    # 2. Cuenta las 'R' y las 'J'
    # 3. Compara y retorna el resultado
```

Ejercicio 17: Reto: Jurassic Park

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Reto: Jurassic Park

Q: ¿Cuál es el objetivo principal?

Implementa la función para resolver el reto.

Q: Estoy atascado, ¿alguna pista?

Usa el operador módulo (%) para verificar si es par.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
    """
    Objetivo:
teros y devuelva la suma total de los huevos que pertenecen a los dinosaurios carní
    """

def count_carnivore_dinosaur_eggs(egg_list) -> int:
    """
    Devuelve la suma de los números pares en la lista.
    """
    total_carnivore_eggs = 0

    # --- TU CÓDIGO AQUÍ ---
    # Itera sobre la lista y suma solo los números pares

    return total_carnivore_eggs
```

Ejercicio 18: Reto: Primera Suma

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Reto: Primera Suma

Q: ¿Cuál es el objetivo principal?

Implementa la función para resolver el reto.

Q: Estoy atascado, ¿alguna pista?

Intenta usar un diccionario para guardar los números vistos.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
                                """
tra los dos primeros números del array que sumen el número goal y devuelve sus índi

                                nums = [4, 5, 6, 2]
                                goal = 8

                            find_first_sum(nums, goal)  # [2, 3] (porque 6 + 2 = 8)
                                """


                                def find_first_sum(nums, goal):
                                    # --- TU CÓDIGO AQUÍ ---
                                # Encuentra dos números que sumen 'goal'
                                # Devuelve una lista con sus índices [i, j]
                                    pass

                                nums = [4, 5, 6, 2]
```

Ejercicio 19: Diccionarios

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Diccionarios

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
###
# 04 - Dictionaries
# Los diccionarios son colecciones de pares clave-valor.
# Sirven para almacenar datos relacionados.
###



# ejemplo tipico de diccionario
persona = {
    "nombre": "midudev",
    "edad": 25,
    "es_estudiante": True,
    "calificaciones": [7, 8, 9],
    "socials": {
        "twitter": "@midudev",
```

Ejercicio 20: Reto: Batalla Pokémon

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Reto: Batalla Pokémon

Q: ¿Cuál es el objetivo principal?

Implementa la función para resolver el reto.

Q: Estoy atascado, ¿alguna pista?

La suma total de cada lista determina el ganador en este caso simplificado.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
"""
Simula una batalla entre dos listas de números.
- Si lista_a[i] > lista_b[i], suma la diferencia al siguiente de A.
- Si lista_b[i] > lista_a[i], suma la diferencia al siguiente de B.
- Si son iguales, ambos se eliminan.

Devuelve el ganador ("Xa" o "Xb") o "x" si hay empate.
"""

def battle(lista_a, lista_b):
    # --- TU CÓDIGO AQUÍ ---
    # Implementa la lógica de la batalla
ta: Puedes simplificarlo sumando todos los elementos de cada lista y comparando los
    pass

lista_a = [4, 4, 4]
```

Módulo 02: Conceptos Fundamentales

Ejercicio 21: Hola Mundo (Repaso)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Hola Mundo (Repaso)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3
""" Primer Programa en Python
Solo para verificar si podemos ejecutar python3 correctamente
"""

name = "Sanjeev"
print("hello "+name+"\n")


# --- EJERCICIO INTERACTIVO ---
# 1. Ejecuta el código tal como está para ver el resultado.
# 2. Modifica el código para cambiar el comportamiento.
```

Ejercicio 22: Números

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Números

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python # Configurando el shebang

# Este tutorial cubrirá el concepto de tipos numéricos en Python3.x
# Hay tres tipos numéricos en Python
# 1. Int (Entero)
# 2. Float (Flotante/Decimal)
# 3. Complex (Complejo)

# Int (Entero)
positive_int = 55
negative_int = -1039
zero = 0
print(positive_int)
print(negative_int)
print(zero)
print(type(negative_int))
```

Ejercicio 23: Manejo de Errores (Try/Except)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Manejo de Errores (Try/Except)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
import requests as req

base_url="https://github.com/"
username = "deepraj1729"

url = base_url+username

# Petición GET a github para el nombre de usuario
try:
res = req.get(url)
if res.status_code == 404:
print("Error 404. Página no encontrada")
elif res.status_code == 200:
print("Estado: OK")

except Exception as e:
```

Ejercicio 24: Comentarios y Docstrings

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Comentarios y Docstrings

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Los comentarios son esenciales para escribir código mantenible.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3

# Los comentarios son líneas que el intérprete de Python ignora.
# Sirven para explicar el código a otros humanos (o a tu "yo" del futuro).

# 1. Comentarios de una sola línea (usan #)
variable = 10 # Esto es una variable

# 2. Comentarios multilínea (no existen oficialmente, pero se usan strings)
"""
Esto es un string multilínea que no se asigna a ninguna variable.
Python lo ignora, por lo que funciona como un comentario de bloque.
"""

# 3. Docstrings (Cadenas de documentación)
# Se usan para documentar funciones, clases y módulos.
```

Ejercicio 25: Conversión Numérica

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Conversión Numérica

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Este tutorial cubrirá el concepto de conversión de tipos numéricos en Python3.x

# Int
positive_int = 55

# Float
negative_float = -2.9987654

# Complex
complex_num = 1j

# convertir de int a float:
positive_float = float(positive_int)
```

Ejercicio 26: Operaciones Numéricas Extra

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Operaciones Numéricas Extra

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
from decimal import Decimal as D

# En este ejemplo, cubriremos
# 1. Cómo funcionan los números float y por qué su comparación te sorprende
# 2. Cómo usar el formato Decimal como en la escuela usando el módulo decimal
# 3. Representación binaria, octal y hexadecimal
# 4. Cualquier operación matemática de entero y float resultará en float

pueden ser de cualquier longitud, un número de punto flotante es preciso solo hast
print((1.1 + 2.2) == 3.3)
print(1.1 + 2.2)

# Salida: Decimal('3.3')
print(D('1.1') + D('2.2'))
```

Ejercicio 27: Cadenas de Texto (Strings)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Cadenas de Texto (Strings)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Python tiene la clase str para representar y manejar cadenas

first_name = "Sanjeev"
last_name  = 'Jaiswal'
nick_name  = '''Jassi'''
address    = """ Dirección de correo, ¿verdad?
si es así, es Hyderabad, Madhapur.
Pin: 500081"""

mobile_num = 9618961800

print("Nombre:", first_name)
print("Nombre: " + first_name) # Concatenación de cadenas
print("Dirección multilínea: " + address)
```

Ejercicio 28: Formato de Strings

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Formato de Strings

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# Python tiene la clase str para representar y manejar cadenas

first_name = "Sanjeev"
last_name  = 'Jaiswal'
nick_name  = '''Jassi'''
address    = """ Dirección de correo, ¿verdad?
si es así, es Hyderabad, Madhapur.
Pin: 500081"""

mobile_num = 9618961800

chr(37) + ":  %d es mi número %s es mi apodo. Tengo %.2f grandes para %s" % (mobil
print(text)

# Argumentos por posición
```

Ejercicio 29: Métodos de Strings

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Métodos de Strings

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
                        #!/usr/bin/python
    # Python tiene la clase str para representar y manejar cadenas

                         first_name = "Sanjeev"
                         last_name  = 'jaiswal'
                        nick_name  = '''Jassi'''
                address    = """ Dirección de correo, ¿verdad?
                        si es así, es Hyderabad, Madhapur.
                                Pin: 500081"""

                        mobile_num = 9618961800

                         greetings = 'Hola'
ongitud de la cadena 'Hola' es: " + str(len(greetings))) # len() para la longitud d

                # ejemplos de funciones lower(), upper() y capitalize()
```

Ejercicio 30: Estructuras Condicionales

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Estructuras Condicionales

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Las sentencias de control son uno de los bloques fundamentales de Python
# Cubriremos
# 1. if elif else
# 2. while, for, range
# 3. break, continue, pass

# 1. Ejemplo de if, elif, else
# Puede haber cero o más sentencias elif
# else es opcional

# url = input("Introduce tu sitio web: ")
url = "https://cybercloud.guru" # Valor hardcodeado para demo

if 'http' in url:
```

Ejercicio 31: Bucle While Avanzado

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Bucle While Avanzado

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
import random

# Sintaxis del bucle While
# while expresion:
#   sentencia(s)
# Puedes usar el bloque else con while también
# while expresion:
#   sentencia(s)
# else:
#   sentencia(s)

guess_num_range = 20
num_to_be_guessed = int(guess_num_range * random.random()) + 1
guess = 0
```

Ejercicio 32: Bucle For Avanzado

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Bucle For Avanzado

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# sintaxis del bucle for
# for variable_iteradora in secuencia:
#   sentencia(s)
es usar el bloque else con el bucle for igual que usamos con while en el ejemplo ar
# for  in :
#
# else:
#

port_details = {
    '22': 'ssh',
    '21': 'ftp',
   '23': 'telnet',
    '80': 'http',
```

Ejercicio 33: Uso de Range

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Uso de Range

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

genera números enteros entre el entero de inicio dado y el entero de parada, es de
el bucle for, podemos iterar sobre una secuencia de números producida por la funció
# Solo permite números de tipo entero como argumentos.
podemos proporcionar un parámetro de tipo string o float dentro de la función rang
# Los argumentos pueden ser positivos (+ve) o negativos (-ve).
No acepta '0' como valor de paso. Si el paso es '0', la función lanza un ValueErro

for step in range(10, 100, 10):
print(step)


print("\nOtro ejemplo para iterar sobre una lista usando range")
port_lists = [21, 22, 23, 25, 53, 80, 443, 3306, 8080, 9002, 27017]
```

Ejercicio 34: Control de Bucles (Break/Continue)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Control de Bucles (Break/Continue)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Ejemplo de pass, break, continue
s: no hace nada. útil cuando se prueba algo o cuando ese bloque de código no es nec
 # break: termina el bucle actual y reanuda la ejecución en la siguiente sentencia
# continue: devuelve el control al principio del bucle


port_details = {
    '21': 'ftp',
   '23': 'telnet',
    '80': 'http',
   '443': 'https',
   '3306': 'mysql'
        }
```

Ejercicio 35: Condicionales en Una Línea

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Condicionales en Una Línea

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Nota: Esto es opcional pero bueno saberlo.
iene solo una expresión, entonces podemos usar una expresión condicional. Las expre

# name = input("¿Cuál es tu primer nombre? ")
name = "Christopher" # Valor hardcodeado para demo

# 1) Llamar a `print` con una cadena diferente usando una sola expresión condiciona
print(
  "Tu nombre es tan largo o más largo que el nombre promedio en los Estados Unidos
) if len(name) >= 6 else print (
    "Tu nombre es más corto que el nombre promedio en los Estados Unidos"
)

# 2) Establecer `message` usando una sola expresión condicional
```

Ejercicio 36: Operadores Aritméticos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Operadores Aritméticos

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

x = 21
y = 5

# Salida: x + y = 26
# Suma
print('x + y =',x+y)

# Salida: x - y = 16
# Resta
print('x - y =',x-y)

# Salida: x * y = 105
# Multiplicación
print('x * y =',x*y)
```

Ejercicio 37: Operadores de Comparación

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Operadores de Comparación

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Operadores de comparación con números
x = 19
y = 91

print('{} > {} es'.format(x,y),x>y)
print('{} < {} es'.format(x, y),x= {} es'.format(x,y),x>=y)
print('{} <= {} es'.format(x,y),x<=y)

# Veamos cómo funciona con cadenas de texto (strings)
name_title = 'Jassi'
name_lowercase = 'jassi'

print('{} > {} es'.format(name_title, name_lowercase),name_title>name_lowercase)
name_title, name_lowercase),name_title= {} es'.format(name_title, name_lowercase),n
```

Ejercicio 38: Operadores de Asignación

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Operadores de Asignación

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

l operador de asignación se usa para asignar un valor a la variable del lado izquie
# Ejemplos de operadores de asignación en Python
# Operador      Ejemplo Equivalente a
   # =            x = 5    x = 5
  # +=        x += 5       x = x + 5
  # -=        x -= 5       x = x - 5
  # *=        x *= 5       x = x * 5
  # /=        x /= 5       x = x / 5
  # %=        x %= 5       x = x % 5
 # //=       x //= 5      x = x // 5
 # **=       x **= 5      x = x ** 5
  # &=        x &= 5       x = x & 5
  # |=        x |= 5       x = x | 5
  # ^=        x ^= 5       x = x ^ 5
```

Ejercicio 39: Operadores Lógicos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Operadores Lógicos

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Operadores Lógicos en Python
Operador      Significado                                                  Ejem
   # and      True (Verdadero) si ambos operandos son verdaderos        x and y
  # or       True (Verdadero) si alguno de los operandos es verdadero      x or y
  # not      True (Verdadero) si el operando es falso (invierte el valor)  not x

x = True
y = False

# Salida: x and y es False
print('x and y es',x and y)

# Salida: x or y es True
print('x or y es',x or y)
```

Ejercicio 40: Operadores de Identidad

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Operadores de Identidad

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Operadores de Identidad
# 'is' y 'is not' son los operadores de identidad en Python.
para verificar si dos valores (o variables) están ubicados en la misma parte de la
# Que dos variables sean iguales (==) no implica que sean idénticas (is).

# Ejemplo tomado de programiz.com
x1 = 5
y1 = 5
x2 = 'Hola'
y2 = 'Hola'
x3 = [1,2,3]
y3 = [1,2,3]

# Salida: False (porque x1 y y1 SON idénticos, así que 'is not' es falso)
```

Ejercicio 41: Estructuras de Datos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Estructuras de Datos

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
s algo nuevo. En cada lenguaje de programación escucharás sobre sus estructuras de
ura de Datos no es más que cómo organizas tus datos, cómo los muestras, trabajas co
non tiene muchas estructuras de datos integradas que ya has visto como números y ca
utiremos las Estructuras de Datos (DS) más importantes que usarás más a menudo en P
                                # Ellas son:
                              # 1. Listas (Lists)
                             # 2. Tuplas (Tuples)
                          # 3. Diccionarios (Dictionaries)
                            # 4. Conjuntos (Sets)

 otras con diferentes bibliotecas de python como array.array pero no las discutirem

 La Lista es una estructura de datos mutable y se implementa como un array dinámico
t = ["http", "https", "ftp", "ssh"] # Puedes tener una lista definida o crear una l
                                print(proto_list)
```

Ejercicio 42: Listas a Fondo

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Listas a Fondo

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Lista Vacía
allowed_ports = []

# lista predefinida
lowed_ports = [22, 23, 25, 53, 80, 69, 443, 3306, 8000, 8080, 5439, 8081, 9001,270

# Iterar a través de la lista
print("Iterar a través de la lista")
for port in allowed_ports:
    print(port)

# Verificar si un elemento existe en la lista
print("\nVerificando si el puerto 5432 está presente en la lista aprobada o no")
if 5432 in allowed_ports:
```

Ejercicio 43: Diccionarios a Fondo

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Diccionarios a Fondo

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3

# Ejemplos para diccionario
    # Diccionario Vacío
       emp_dict = {}

# diccionario con claves enteras
emp_dict = {100: 'Sanjeev', 101: 'Jassi'}
            print(emp_dict)

# diccionario con claves mixtas
emp_dict = {100: 'Sanjeev', 'skills': ['Python', 'AWS']}
            print(emp_dict)

# usando la función dict()
emp_dict = dict({100: 'Sanjeev', 101: 'Jassi'})
```

Ejercicio 44: Tuplas

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Tuplas

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
n objeto inmutable en Python. Significa que una vez que se establecen los datos, no
  # Se pueden usar para datos constantes o diccionario sin clave (tuplas anidadas)

                # Inicialización de Tupla Vacía
                          tup = ()
                         print(tup)

              # Inicialización de Tupla con datos
            # No me gustó esta forma sin embargo ;)
              tup1 = 'python', 'aws', 'security'
                         print(tup1)

                  # Otra para hacer lo mismo
            tup2 = ('python', 'django', 'linux')
                         print(tup2)
```

Ejercicio 45: Conjuntos (Sets)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Conjuntos (Sets)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
cción de datos que no está ordenada, ni indexada y es única. Es uno de los 4 tipos
        # Esto se basa en un concepto de estructura de datos tabla hash.
        # No podemos acceder a sus elementos por índice como en una lista
conjuntos no pueden tener elementos mutables, de lo contrario pueden contener datos

                # Inicialización de conjunto vacío
            # usa el método set(). Usar {} creará un diccionario vacío
                            test = {}
                             # Salida
                          print(type(test))
                            sets = set()
                             # Salida
                          print(type(sets))

                    # inicialización de conjuntos
                          my_set = {1,2,3}
```

Ejercicio 46: Funciones Avanzadas

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Funciones Avanzadas

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Ejemplos de función en Python 3.x

# ¿Cuándo necesitas una función?
realizar un conjunto de tareas específicas y quieres reutilizar ese código siempre
#   También, para una mejor modularidad, legibilidad y solución de problemas

# ¿Cómo escribir una función (Sintaxis)?
'''def nombre_funcion():
{
# algo de código aquí
}
'''

# Diferentes formas de pasar parámetros
```

Ejercicio 47: Funciones de Colección

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Funciones de Colección

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# Ejemplo de lambda, map, filter

# 1) Ordenar la lista de diccionarios `people` alfabéticamente basándose en la
# clave 'name' de cada diccionario usando la función `sorted` y almacenar
# la nueva lista como `sorted_by_name`

people = [
{"name": "Kevin Bacon", "age": 61},
 {"name": "Fred Ward", "age": 77},
{"name": "finn Carter", "age": 59},
{"name": "Ariana Richards", "age": 40},
{"name": "Victor Wong", "age": 74},
]

# sorted_by_name = None
```

Ejercicio 48: Funciones Integradas

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Funciones Integradas

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#Entrada en CLI
# input_text = input("Introduce algo aquí y luego presiona enter....   ")
input_text = "Hola Python" # Valor hardcodeado
print("Ingresaste: ",input_text)

#Absoluto o Mod
n = abs(-12)
print("El valor absoluto de -12 es: ",n)

#Expresión Booleana
x=12>19
print("El valor booleano de la expresión (12<19) es ",bool(x))

data = {
  "id": 1,
"name": "Ramesh",
```

Ejercicio 49: Expresiones Regulares (Regex)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Expresiones Regulares (Regex)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
import re

urls = ["https://www.facebook.com","https://www.google.com","https://www.amazon.in"

def checkValidURL(url):
    url_reg_ex = r"^(([^:/?#]+):)?(//([^/?#]*))?([^?#]*)(\?([^#]*))?(#(.*))?"
    data = re.search(url_reg_ex,url)
    if data is not None:
        return True
    return False

def parseDomain(url):
    domain = url.split("//")[1].split("www")[1].split(".")[1]
    print(domain)
```

Ejercicio 50: Introducción a Clases y Objetos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Introducción a Clases y Objetos

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Las clases permiten modelar objetos del mundo real.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
"""
Las clases son "plantillas" para crear objetos.
Los objetos agrupan datos (atributos) y comportamientos (métodos).
"""

class Laptop:
# El método __init__ es el constructor. Se ejecuta al crear un objeto.
    def __init__(self, marca, modelo, ram):
        self.marca = marca
        self.modelo = modelo
        self.ram = ram
        self.encendida = False

    def encender(self):
        self.encendida = True
        print(f"{self.marca} {self.modelo} se está encendiendo...")
```

Módulo 03: Práctica de Algoritmos

Ejercicio 51: Clima de la Ciudad

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Clima de la Ciudad

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3
ipt para obtener la temperatura y otra información de una ciudad desde una app del
import json
from datetime import datetime

# Respuesta de API simulada para evitar bloqueos por clave API inválida
city_name = "Madrid"

# Obtener la hora desde valores UTC y zona horaria proporcionados
# pasar el valor como utc + zona horaria (ambos son timestamp UTC)
def time_from_utc_with_timezone(utc_with_tz):
local_time = datetime.utcfromtimestamp(utc_with_tz)
```

Ejercicio 52: Conversor Fahrenheit a Celsius

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Conversor Fahrenheit a Celsius

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3

def fahr_to_cel(fahrenheit):
celsius = (fahrenheit - 32) / 1.8
return celsius

try:
# fahr = int(input('Introduce la temperatura en Fahrenheit por favor: '))
fahr = 100
print(f"Calculando para {fahr} Fahrenheit...")
except:
exit("Lo siento. Solo se permiten números reales")

print(fahr_to_cel(fahr))
```

Ejercicio 53: Base de Datos en Memoria

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Base de Datos en Memoria

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
# El problema requiere implementar un almacén clave-valor en memoria
# donde puedas establecer un par clave-valor y
# recuperar el valor de una clave.

# Ejemplo:
# db.set(101, ["Sanjeev", "ProdSec"])
# db.set(102, ["Deep", "DevSecOps"])
# db.get(102)
a de comandos para n entradas, establecer n entradas, luego basado en el comando im
# Ejemplo:
# Elige opciones para operaciones:
#     1. establecer/crear entradas
#     Cuántas entradas: 2
#     id_emp: entrada
#     detalles_emp: entrada([])
#     2. obtener detalles id_emp
```

Ejercicio 54: DB en Memoria (CMD)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: DB en Memoria (CMD)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
# El problema requiere implementar un almacén clave-valor en memoria
# donde puedas establecer un par clave-valor y
# recuperar el valor de una clave.

# Ejemplo:
# db.set(101, ["Sanjeev", "ProdSec"])
# db.set(102, ["Deep", "DevSecOps"])
# db.get(102)

class DB:

def __init__(self):
    self.dic = {}

def set(self, key: int, value: list) -> None:
    if not isinstance(key, int):
```

Ejercicio 55: DB en Memoria (JSON)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: DB en Memoria (JSON)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
# El problema requiere implementar un almacén clave-valor en memoria
# donde puedas establecer un par clave-valor y
# recuperar el valor de una clave.

import json
 import os

 class DB:
def __init__(self, filename='db.json'):
 self.filename = filename
 self.dic = self.load()

 def load(self):
 if os.path.exists(self.filename):
 with open(self.filename, 'r') as f:
 return json.load(f)
```

Ejercicio 56: DB Persistente

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: DB Persistente

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
# Para hacer el programa interactivo y persistente,
des usar un bucle para pedir operaciones al usuario continuamente hasta que decida
# El problema requiere implementar un almacén clave-valor en memoria
# donde puedas establecer un par clave-valor y
# recuperar el valor de una clave.

# Ejemplo:
# db.set(101, ["Sanjeev", "ProdSec"])
# db.set(102, ["Deep", "DevSecOps"])
# db.get(102)
a de comandos para n entradas, establecer n entradas, luego basado en el comando im
# Ejemplo:
# Elige opciones para operaciones:
#     1. establecer/crear entradas
#     Cuántas entradas: 2
#     id_emp: entrada
```

Ejercicio 57: Es Primo

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Es Primo

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3

# Ejercicio para if else y bucle for
# Script para decir si un número es primo o no
# Si un número es divisible solo por 1 y por sí mismo, entonces es un número primo

# Obtener entrada del usuario, como es string, necesitas convertirla a int
try:
# num = int(input('Introduce un número: '))
num = 29
print(f"Comprobando si {num} es primo...")
except:
exit("¡Solo enteros por favor!")


# Verificar si el número es negativo
```

Ejercicio 58: N Números Primos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: N Números Primos

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3

# Función para comprobar si un número es primo contra una lista de primos dada
def prime(num, primes):
    # bucle para probar si num es primo contra la lista dada
    for prime in primes:
        if  (num % prime) == 0:
            return False
    # Tenemos un nuevo número primo, añadirlo a la lista primes[]
    primes.append(num)
    return True

def n_primes(n):
    primes = []
    start_num, prime_counter = 2, 0
    while True:
```

Ejercicio 59: Primos en Rango

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Primos en Rango

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

try:
# lower = int(input('Introduce inicio del rango: '))
 # upper = int(input('Introduce fin del rango: '))
lower = 10
upper = 50
print(f"Primos entre {lower} y {upper}...")
except:
exit("Asegúrate de que los rangos sean solo enteros")

if( lower < 0 or upper < 0 ):
exit("Los rangos deben ser números positivos")

print("Los números primos entre", lower, "y", upper, "son:")
for num in range(lower, upper+1):
```

Ejercicio 60: Suma de Dos Índices

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Suma de Dos Índices

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python

# Hay un array lleno de enteros y un valor objetivo t, también entero
# Necesitas encontrar qué par de enteros suman el objetivo e imprimir sus índices
# Puedes asumir que solo hay un par que resulta en la suma objetivo

num_list = [2, 1, 3, 5, 6, 11, 2, 13, 4, 15]
target = 12

def twoSum(arr, t):
    index_dict = {}
    length = len(arr)
    index = 0

    while index < length:
        if (t - arr[index]) in index_dict:
```

Módulo 04: Automatización y Scripts

Ejercicio 61: Avatar Básico

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Avatar Básico

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -------------------------------------------------------
from py_avataaars import PyAvataaar
# Es posible que necesites instalar la librería cairo.
# Para mac: escribe `brew install cairo`
avatar = PyAvataaar()
avatar.render_png_file('basic_avatar.png')
avatar.render_svg_file('basic_avatar.svg')
```

Ejercicio 62: Avatar Personalizado

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Avatar Personalizado

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
        # --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
    # ----------------------------------------------------
                import py_avataaars as pa

                  avatar = pa.PyAvataaar(
                style=pa.AvatarStyle.TRANSPARENT,
                  skin_color=pa.SkinColor.LIGHT,
                  hair_color=pa.HairColor.BLACK,
              facial_hair_type=pa.FacialHairType.DEFAULT,
                 facial_hair_color=pa.HairColor.BLACK,
              top_type=pa.TopType.SHORT_HAIR_SHORT_FLAT,
                      hat_color=pa.Color.BLACK,
                    mouth_type=pa.MouthType.SMILE,
                    eye_type=pa.EyesType.DEFAULT,
```

Ejercicio 63: Avatar Personalizado II

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Avatar Personalizado II

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ----------------------------------------------------
import python_avatars as pa

pa.Avatar(
style=pa.AvatarStyle.TRANSPARENT,
# background_color='#FF00FF',
# Choose graphic shirt
clothing=pa.ClothingType.GRAPHIC_SHIRT,
clothing_color=pa.ClothingColor.BLUE_01,
# Important to choose this as shirt_graphic, otherwise shirt_text will be ignore
shirt_graphic=pa.ClothingGraphic.CUSTOM_TEXT,
shirt_text='Flexmind'
).render("avatar_text.svg")
```

Ejercicio 64: Avatar Aleatorio

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Avatar Aleatorio

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ----------------------------------------------------
import python_avatars as pa

# Avatar completamente aleatorio
random_avatar_1 = pa.Avatar.random()

# Avatar aleatorio excepto el sombrero
_avatar_2 = pa.Avatar.random(top=pa.HatType.HAT)  # Más atributos pueden mantenerse

# Avatar fijo pero ropa aleatoria
random_avatar_3 = pa.Avatar(
  style=pa.AvatarStyle.CIRCLE,
  hair_color=pa.HairColor.BLACK,
```

Ejercicio 65: Reloj Digital Básico

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Reloj Digital Básico

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ------------------------------------------------------
# Reloj Digital Básico (Versión Web)
# Este script genera un archivo HTML con un reloj digital funcional.

html_content = """
```

Ejercicio 66: Reloj Digital Avanzado

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Reloj Digital Avanzado

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Reloj Digital Avanzado (Temporizador Web)
# Genera un temporizador de cuenta regresiva en HTML.

seconds = 10 # Duración del temporizador

html_content = f"""
```

Ejercicio 67: Reloj Digital CLI

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Reloj Digital CLI

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Reloj Digital con Fecha (Versión Web)
# Genera un reloj que muestra fecha y hora.

html_content = """
```

Ejercicio 68: Reloj Mundial

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Reloj Mundial

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Reloj Mundial (Versión Web)
# Genera un tablero con múltiples zonas horarias.

html_content = """
```

Ejercicio 69: Captura de Cámara (OpenCV - Headless)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Captura de Cámara (OpenCV - Headless)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ----------------------------------------------------
# capturar una sola imagen de la webcam usando python
# Nota: En un entorno servidor/web, no podemos abrir una ventana (imshow).
# En su lugar, guardamos la imagen en un archivo.

import cv2 as cv
import time
import os

# inicializar la cámara
# Si no hay cámara física, esto podría fallar o devolver un frame negro.
cam_port = 0
```

Ejercicio 70: Imagen a Caricatura (Real)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Imagen a Caricatura (Real)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Procesamiento de Imagen Real con PIL
# Primero generamos una imagen base y luego aplicamos filtros.

import py_avataaars as pa
from PIL import Image, ImageFilter, ImageOps
import os

print("--- Generando imagen base ---")
# 1. Generar un avatar como imagen base
avatar = pa.PyAvataaar(
style=pa.AvatarStyle.CIRCLE,
```

Ejercicio 71: Caricatura con GUI (Web)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Caricatura con GUI (Web)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python3
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Editor de Caricaturas (Interfaz Web)
# Genera una interfaz HTML para aplicar filtros a una imagen.

html_content = """
```

Ejercicio 72: Generador QR (API Real)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Generador QR (API Real)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -------------------------------------------------
# Generador de QR usando una API pública
# No requiere instalar librerías complejas.

import requests

data = "https://www.python.org"
size = "300x300"
api_url = f"https://api.qrserver.com/v1/create-qr-code/?size={size}&data={data}"

print(f"Generando QR para: {data}")
print("Contactando API...")
```

Ejercicio 73: Info de Imagen (PIL)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Info de Imagen (PIL)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ------------------------------------------------------
# Análisis de Imagen con Pillow (PIL)

from PIL import Image
import os

# Usamos una imagen generada previamente o descargada
filename = "qr_code.png"

if not os.path.exists(filename):
print(f"El archivo {filename} no existe. Ejecuta la lección anterior primero.")
else:
```

Ejercicio 74: Análisis de Texto (Frecuencia)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Análisis de Texto (Frecuencia)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# Análisis de Frecuencia de Palabras

from collections import Counter
import re

texto = """
Python es un lenguaje de programación interpretado cuya filosofía hace hincapié
la legibilidad de su código. Se trata de un lenguaje de programación multiparadigm
ya que soporta orientación a objetos, programación imperativa y, en menor medida,
programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.
"""

print("--- Texto Original ---")
print(texto.strip())
print("-" * 20)
```

Ejercicio 75: Análisis de Sentimiento (Simple)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Análisis de Sentimiento (Simple)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# Analizador de Sentimiento Básico (Basado en palabras clave)

positivas = ["bueno", "excelente", "increíble", "feliz", "amor", "gusta", "genial"]
negativas = ["malo", "terrible", "triste", "odio", "feo", "error", "fallo"]

def analizar_sentimiento(texto):
    texto = texto.lower()
    score = 0

    for p in positivas:
        if p in texto:
            score += 1

    for n in negativas:
        if n in texto:
```

Ejercicio 76: Validador de Email (Regex)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Validador de Email (Regex)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# Validación de Email con Expresiones Regulares

import re

# Regex estándar para email
email_regex = r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"

emails = [
"usuario@example.com",
"nombre.apellido@empresa.co.uk",
"invalido@com",
"sin_arroba.com",
"user@domain"
]
```

Ejercicio 77: Generador PDF (Reporte)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Generador PDF (Reporte)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ------------------------------------------------------
# Generación de un Reporte PDF Simple
# REQUISITO: pip install fpdf2

from fpdf import FPDF

print("Generando reporte PDF...")

pdf = FPDF()
pdf.add_page()

# Título
```

Ejercicio 78: Texto a Voz Simple

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Texto a Voz Simple

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ------------------------------------------------
# Texto a Voz Simple
# REQUISITO: pip install pyttsx3

import pyttsx3

engine = pyttsx3.init() # creación del objeto

""" VELOCIDAD """
ngine.getProperty('rate')   # obteniendo detalles de la velocidad actual, por defec
#engine.setProperty('rate', 125) # Configurando nueva velocidad de voz
```

Ejercicio 79: Espiral Arcoíris (SVG Turtle)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Espiral Arcoíris (SVG Turtle)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
                    #!/usr/bin/python
        # --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
 # Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
    # ------------------------------------------------------
                # Espiral Arcoíris (SVG Turtle)
              # REQUISITO: pip install svg_turtle

                from svg_turtle import SvgTurtle

              t = SvgTurtle(width=500, height=500)
                       t.bgcolor('black')
t.speed = 0 # Ignorado en SVG pero mantenido por compatibilidad

colors = ['red', 'purple', 'blue', 'green', 'orange', 'yellow']
```

Ejercicio 80: Triángulo de Sierpinski (SVG)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Triángulo de Sierpinski (SVG)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Triángulo de Sierpinski (SVG)
# REQUISITO: pip install svg_turtle

from svg_turtle import SvgTurtle

def draw_triangle(t, points, color):
# Dibujo simple de triángulo con líneas
t.pencolor(color)
t.penup()
t.goto(points[0][0], points[0][1])
t.pendown()
```

Ejercicio 81: Arte Geométrico (SVG)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Arte Geométrico (SVG)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
                    #!/usr/bin/python
        # --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
 # Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
    # ------------------------------------------------------
                  # Arte Geométrico (SVG)
             # REQUISITO: pip install svg_turtle

               from svg_turtle import SvgTurtle

                      def draw_square(t):
                         for i in range(4):
                             t.forward(100)
                              t.right(90)

             t = SvgTurtle(width=500, height=500)
```

Ejercicio 82: Cuadrado y Círculo (SVG)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Cuadrado y Círculo (SVG)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
                    #!/usr/bin/python
        # --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
 # Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
    # ------------------------------------------------------
                # Cuadrado y Círculo (SVG)
            # REQUISITO: pip install svg_turtle

              from svg_turtle import SvgTurtle
                         import math

            def draw_circle_approx(t, radius):
             # Aproximación de círculo con líneas
             circumference = 2 * math.pi * radius
               step_length = circumference / 36
                     for _ in range(36):
```

Ejercicio 83: Cuadrado Simple (SVG)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Cuadrado Simple (SVG)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ---------------------------------------------------
# Cuadrado Simple (SVG)
# REQUISITO: pip install svg_turtle

from svg_turtle import SvgTurtle

t = SvgTurtle()
t.pencolor("green")

print("Dibujando cuadrado...")
t.forward(100)
t.right(90)
```

Ejercicio 84: Colores en Terminal (Colorama)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Colores en Terminal (Colorama)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -------------------------------------------------
# Colores en Terminal con Colorama
# REQUISITO: pip install colorama

from colorama import init, Fore, Back, Style
init(autoreset=True)


print(Fore.RED + 'texto rojo')
print(Fore.GREEN + 'texto verde')
print(Fore.BLUE + 'texto azul')
print(Fore.CYAN + 'texto cian')
```

Ejercicio 85: Descargar Archivos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Descargar Archivos

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
import sys
import requests
import re
from colorama import init, Fore, Back, Style
import validators

# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ------------------------------------------------------
# PROPÓSITO de este script
# Obtener una url con varios formatos de archivos para descargar

init(autoreset=True)

# --- CORRECCIÓN PARA EJECUCIÓN WEB ---
```

Ejercicio 86: Generar OTP

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Generar OTP

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# Generar OTP de 6 dígitos
import string
import secrets

number = string.digits
otp = ''

for i in range(6):
otp += ''.join(secrets.choice(number))

print(f'Tu OTP es: {otp}')
```

Ejercicio 87: Generador de Contraseñas

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Generador de Contraseñas

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
import string
import secrets

def get_alphabet():
letters = string.ascii_letters
digits = string.digits
special_chars = string.punctuation
alphabet = letters + digits + special_chars
return alphabet

# Restricciones de contraseña
password_len = 12
print(f"Generando contraseña de longitud: {password_len}")

if password_len <8:
print("La longitud debe ser de al menos 8 caracteres")
```

Ejercicio 88: Extraer Links de Web

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Extraer Links de Web

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
import requests
import sys
import re
from bs4 import BeautifulSoup as bs
import validators

# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ---------------------------------------------------

# --- CORRECCIÓN PARA EJECUCIÓN WEB ---
if len(sys.argv) < 2:
print("No se proporcionó URL. Usando por defecto: https://www.python.org")
sys.argv.append("https://www.python.org")
# -------------------------
```

Ejercicio 89: Info Usuario GitHub

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Info Usuario GitHub

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
import json
import requests
import sys

# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ---------------------------------------------------

# --- CORRECCIÓN PARA EJECUCIÓN WEB ---
if len(sys.argv) < 2:
    print("No se proporcionó usuario. Usando por defecto: octocat")
    sys.argv.append("octocat")
# -------------------------

username = sys.argv[1]
```

Ejercicio 90: Convertir HEIC a PNG

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Convertir HEIC a PNG

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Necesitas instalar pillow-heif para que funcione con archivos reales.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Convierte imágenes HEIC a PNG usando pillow-heif
# REQUISITO: pip install pillow-heif

from PIL import Image
import os

# Intentamos importar pillow_heif, si no está, avisamos
try:
    from pillow_heif import register_heif_opener
    register_heif_opener()
    HAS_HEIF = True
```

Ejercicio 91: Rangos IP (JSON)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Rangos IP (JSON)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
import json

# Datos truncados para brevedad
data = {
    "syncToken": "1721411590",
    "createDate": "2024-07-19-17-53-10",
    "prefixes": [
        {
            "ip_prefix": "3.5.140.0/22",
            "region": "ap-northeast-2",
            "service": "AMAZON",
            "network_border_group": "ap-northeast-2"
        },
        {
            "ip_prefix": "13.34.1.109/32",
            "region": "ap-southeast-2",
```

Ejercicio 92: Scraping con Selenium

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Scraping con Selenium

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Selenium automatiza navegadores reales. Requiere el navegador instalado.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ------------------------------------------------------
# Scraping real con Selenium
# REQUISITO: pip install selenium webdriver-manager

import time
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By

def run_scraper():
print("Iniciando Chrome...")
```

Ejercicio 93: Análisis de Datos CSV

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Análisis de Datos CSV

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

La librería csv facilita la lectura y escritura de datos tabulares.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Analizar un archivo CSV real
import csv
import os

# Creamos un archivo CSV de prueba
csv_file = "datos_ventas.csv"
with open(csv_file, "w", newline="") as f:
writer = csv.writer(f)
writer.writerow(["Producto", "Precio", "Cantidad"])
writer.writerow(["Laptop", 1200, 5])
writer.writerow(["Mouse", 25, 50])
```

Ejercicio 94: Manipulación de PDF

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Manipulación de PDF

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

pypdf permite leer, escribir y manipular archivos PDF.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -------------------------------------------------
# Manipulación real de PDF usando pypdf
# REQUISITO: pip install pypdf

from pypdf import PdfReader, PdfWriter
import os

rear un PDF simple para la demo (usando fpdf2 si está disponible, o simulando creac
try:
from fpdf import FPDF
pdf = FPDF()
pdf.add_page()
```

Ejercicio 95: PDF a Imágenes

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: PDF a Imágenes

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

pdf2image convierte páginas de PDF en objetos de imagen PIL.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Conversión real de PDF a Imagen
# REQUISITO: pip install pdf2image
# REQUISITO: Poppler instalado en el sistema (apt-get install poppler-utils)

import os

try:
from pdf2image import convert_from_path
HAS_LIB = True
except ImportError:
HAS_LIB = False
```

Ejercicio 96: Validador Regex

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Validador Regex

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# pip install regex
import re

print("Iniciando Validador Regex...")

def validate_phone_number():
    phone = "123-456-7890"
    pattern = r"\d{3}-\d{3}-\d{4}"
    if re.match(pattern, phone):
        print(f"Teléfono {phone} es válido")
    else:
        print(f"Teléfono {phone} es inválido")

def validate_username():
    user = "User_123"
```

Ejercicio 97: Renombrar Archivos Masivamente

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Renombrar Archivos Masivamente

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

os.rename(origen, destino) es la función clave.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Script real para renombrar archivos
import os
import shutil

# Configuración
dir_objetivo = "fotos_viaje_test"

# 1. Preparación: Crear directorio y archivos de prueba
if not os.path.exists(dir_objetivo):
    os.makedirs(dir_objetivo)
    # Crear archivos dummy
```

Ejercicio 98: Información del Sistema

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Información del Sistema

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
 import platform
    import sys
    import os
  import socket
   import time

Este script está probado en MacOS y funcionó bien. Debería funcionar en Linux tambi

   unumber = os.getuid()
   pnumber = os.getpid()
   where   = os.getcwd()
   now     = time.time()
 means   = time.ctime(now)

print ("Número de usuario",unumber)
```

Ejercicio 99: Recordatorio de Descanso

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Recordatorio de Descanso

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

webbrowser es una librería estándar de Python.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -------------------------------------------------
# Recordatorio real usando webbrowser
import time
import webbrowser

print("Iniciando monitor de descansos...")
print("Para la demo, esperaremos solo 3 segundos.")

# Esperar (simulando trabajo)
time.sleep(3)

print("¡Hora del descanso!")
```

Ejercicio 100: Validador de Contraseñas

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Validador de Contraseñas

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
import re

# Criterios de Contraseña
# 1. Debe contener alfanuméricos
# 2. Al menos una Letra Mayúscula
# 3. Al menos una letra minúscula
# 4. 8-20 caracteres
# 5. al menos un caracter especial !@#$%^&*_
# 6. Sin espacios en blanco por favor

hu', 'Jassi Sidhu0$','JassiSidhu0$', 'Jalantu_123*', '12Falcon#', 'Sh0rt5!','Sh0rt5

# Usando Expresiones Regulares
'^((?=.*\\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%^&*_])(?=.*[\\S])[0-9a-zA-Z!@#$%^&*_]
for password in passwords:
```

Ejercicio 101: Duración de Video

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Duración de Video

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

MoviePy es una potente librería para edición de video.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Cálculo real de duración de video usando moviepy
# REQUISITO: pip install moviepy

import os

try:
from moviepy.editor import VideoFileClip
HAS_MOVIEPY = True
except ImportError:
HAS_MOVIEPY = False
print("Instala moviepy para analizar videos reales.")
```

Módulo 05: Ciberseguridad y Criptografía

Ejercicio 102: Criptografía Básica

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Criptografía Básica

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Criptografía Básica con Fernet
# REQUISITO: pip install cryptography

from cryptography.fernet import Fernet

key = Fernet.generate_key()
cipher_suite = Fernet(key)
```

Ejercicio 103: Hashing de Contraseñas

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Hashing de Contraseñas

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# Hashing de Contraseñas con hashlib
# Aprende cómo se almacenan las contraseñas de forma segura.

import hashlib

password = "MiPasswordSeguro123"
print(f"Contraseña original: {password}")

# 1. MD5 (Inseguro, rápido)
md5_hash = hashlib.md5(password.encode()).hexdigest()
print(f"MD5:    {md5_hash}")

# 2. SHA-256 (Estándar actual)
sha256_hash = hashlib.sha256(password.encode()).hexdigest()
print(f"SHA256: {sha256_hash}")
```

Ejercicio 104: Cracking de Contraseñas (Hash)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Cracking de Contraseñas (Hash)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Los hackers comparan hashes calculados con hashes robados.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# Cracking de Contraseñas (Ataque de Diccionario/Fuerza Bruta)
scenario real, el atacante obtiene el HASH de la contraseña, no la contraseña en te

import hashlib
import time

# 1. El Hash robado (SHA-256 de un PIN de 4 dígitos)
# Supongamos que obtuvimos esto de una base de datos filtrada.
# (El hash corresponde a "4829")
TARGET_HASH = "a2c4e6f323977e58455793f20e547622995a15a303863166544165576556614 3"

print("--- Iniciando Cracking de Hash SHA-256 ---")
print(f"Objetivo: Encontrar el PIN que genera el hash: {TARGET_HASH[:10]}...")

start_time = time.time()
```

Ejercicio 105: Inyección SQL (Demo SQLite)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Inyección SQL (Demo SQLite)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# Demostración de Inyección SQL usando SQLite en memoria

import sqlite3

# Configuración de la Base de Datos
conn = sqlite3.connect(":memory:")
cursor = conn.cursor()
cursor.execute("CREATE TABLE users (id INTEGER, username TEXT, password TEXT)")
cursor.execute("INSERT INTO users VALUES (1, 'admin', 'admin123')")
cursor.execute("INSERT INTO users VALUES (2, 'user', 'pass')")

def login_vulnerable(user, pwd):
print(f"\n[Intento de Login] User: {user} | Pass: {pwd}")
# VULNERABLE: Concatenación directa de strings
query = f"SELECT * FROM users WHERE username = '{user}' AND password = '{pwd}'"
```

Ejercicio 106: Detector de Phishing (URL)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Detector de Phishing (URL)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# Analizador de URLs para detectar Phishing

import re

def analizar_url(url):
    score = 0
    razones = []

    # 1. Longitud excesiva
    if len(url) > 75:
        score += 10
        razones.append("URL muy larga")

    # 2. Uso de dirección IP
ip_regex = r"\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"
```

Ejercicio 107: Info de Cabeceras HTTP

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Info de Cabeceras HTTP

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ------------------------------------------------------
# Info de Cabeceras HTTP
# REQUISITO: pip install requests validators colorama

import requests
import sys
import validators
from colorama import init, Fore, Back, Style

init(autoreset=True)

# --- CORRECCIÓN PARA EJECUCIÓN WEB ---
```

Ejercicio 108: Cabeceras de Seguridad

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Cabeceras de Seguridad

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Cabeceras de Seguridad
# REQUISITO: pip install requests validators colorama

import requests
import sys
import validators
from colorama import init, Fore, Back, Style

init(autoreset=True)

# --- CORRECCIÓN PARA EJECUCIÓN WEB ---
```

Ejercicio 109: Escáner de Puertos (Socket)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Escáner de Puertos (Socket)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Escáner de Puertos Simple usando Sockets
# Escanea puertos comunes en localhost.

import socket
import sys

target = "127.0.0.1" # Localhost
ports = [21, 22, 80, 443, 8000, 8080]

print(f"--- Escaneando {target} ---")
```

Módulo 06: Proyectos Finales

Ejercicio 110: Proyecto 1: Web Scraper (Petición)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Proyecto 1: Web Scraper (Petición)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -------------------------------------------------------
# Proyecto Final 1: Web Scraper de Libros
# Parte 1: Realizar la petición HTTP

import requests

# Sitio de pruebas para scraping
URL = "http://books.toscrape.com/"
```

Ejercicio 111: Proyecto 1: Web Scraper (Extracción)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Proyecto 1: Web Scraper (Extracción)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# ------------------------------------------------------
# Proyecto Final 1: Web Scraper de Libros
# Parte 2: Extraer datos con BeautifulSoup

from bs4 import BeautifulSoup
import os
import csv

filename = "books.html"

if not os.path.exists(filename):
print(f"Error: {filename} no existe. Ejecuta la lección anterior.")
```

Ejercicio 112: Proyecto 2: Analizador de Logs (Lectura)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Proyecto 2: Analizador de Logs (Lectura)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -------------------------------------------------
# Proyecto Final 2: Analizador de Logs de Servidor
# Parte 1: Generar y Leer Logs

# Simulamos un archivo de logs de Apache/Nginx
log_data = """
192.168.1.10 - - [23/Nov/2025:10:00:01 +0000] "GET /index.html HTTP/1.1" 200 1024
192.168.1.11 - - [23/Nov/2025:10:00:05 +0000] "GET /about.html HTTP/1.1" 200 512
192.168.1.12 - - [23/Nov/2025:10:00:10 +0000] "GET /contact.php HTTP/1.1" 404 150
192.168.1.10 - - [23/Nov/2025:10:00:15 +0000] "POST /login HTTP/1.1" 200 0
10.0.0.5 - - [23/Nov/2025:10:01:00 +0000] "GET /admin HTTP/1.1" 403 200
192.168.1.11 - - [23/Nov/2025:10:01:05 +0000] "GET /style.css HTTP/1.1" 200 3000
```

Ejercicio 113: Proyecto 2: Analizador de Logs (Reporte)

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Proyecto 2: Analizador de Logs (Reporte)

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -----------------------------------------------------
# Proyecto Final 2: Analizador de Logs de Servidor
# Parte 2: Generar Reporte de Estadísticas

import re
from collections import Counter

filename = "server.log"

print("Analizando logs...")

ip_pattern = r"(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})"
```

Ejercicio 114: Proyecto 3: Organizador de Archivos

Q: ¿De qué trata este ejercicio?

Estudia el siguiente código: Proyecto 3: Organizador de Archivos

Q: ¿Cuál es el objetivo principal?

Ejecuta el código y analiza su funcionamiento.

Q: Estoy atascado, ¿alguna pista?

Consulta el repositorio oficial para más contexto.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
#!/usr/bin/python
# --- IMPORTANTE: EJECUCIÓN LOCAL REQUERIDA ---
# Este ejercicio requiere librerías externas o generar archivos.
# Por favor, ejecútalo en tu IDE local (VS Code, PyCharm, etc).
# -------------------------------------------------
# Proyecto Final 3: Organizador Automático de Archivos

import os
import shutil

# 1. Configuración: Crear entorno de prueba
base_dir = "downloads_test"
if not os.path.exists(base_dir):
    os.mkdir(base_dir)

# Crear archivos dummy
```

Módulo 07: Certificación

Ejercicio 115: Obtener Certificado

Q: ¿De qué trata este ejercicio?

Proyecto Final: Generador de Certificados

Has llegado al final. Como último ejercicio, utilizarás la librería `fpdf` para generar tu propio certificado.

Analiza el código proporcionado.
Cambia la variable `nombre_estudiante` con tu nombre real.
Ejecuta el código para descargar tu PDF.

Q: ¿Cuál es el objetivo principal?

Personaliza el código y ejecútalo.

Q: Estoy atascado, ¿alguna pista?

Asegúrate de cambiar el valor de nombre_estudiante.

Q: ¿Cómo es la solución o el código de ejemplo?

Aquí tienes el código de referencia para entender la lógica:

```
from fpdf import FPDF
from fpdf.enums import XPos, YPos
from datetime import datetime

# --- CONFIGURACIÓN ---
```