EDUARDO RUIZ RAMIREZ – BASES DE DATOS – 22/02/2024

EJERCICIO 1

Original

```python

# La función recibe una lista de enteros
# Calcula la sumatoria, pero ignora el valor 1
# osea que no cuenta los 1
def contar_sin_unos(lista):
    total = len(lista)
    i = 0
    suma = 0
    while i < total:
        elemento = lista[i]
        if elemento == 1:
            continue
        suma += elemento
        i += 1
    return suma

# Pruebas (las que pasan dan True)
print(contar_sin_unos([2, 3, 4]) == 9)
print(contar_sin_unos([2, 3, 4, 1]) == 9)
print(contar_sin_unos([1, 1, 1, 1]) == 0)

```

Corregido

```python
def contar_sin_unos(lista):
    total = len(lista)
    i = 0
    suma = 0
    while i < total:
        elemento = lista[i]
        i += 1 #se pasa arriba del if
        if elemento == 1:
            continue
        suma += elemento
    return suma

numeros = int(input('escribe cuantos numeros quieres ingresar'))
indexnumeros = []

if numeros == 0:
    print('no valido')
    exit()

for _ in range(numeros):
    indexnumeros.append(int(input()))

resultado = contar_sin_unos(indexnumeros)
print(resultado)

print(contar_sin_unos([2, 3, 4]) == 9)
print(contar_sin_unos([2, 3, 4, 1]) == 9)
print(contar_sin_unos([1, 1, 1, 1]) == 0)
```

El código proporcionado originalmente se trataba solo de una función, por lo que por si solo el código no iba a hacer absolutamente nada al momento de ejecutarse, pues necesita de otro código que lo llame.
Esta función suma una cantidad de números (que el usuario especifica) y muestra el resultado, en caso de haber un 1 en los números introducidos, se ignora.
Este código cuenta con solo un error: El “i += 1” esta posicionado de manera errónea. Al momento de ejecutarse el código, caía dentro de un bucle infinito. Inmediatamente, se sospecha acerca de los while, pues como se trata de una función que repite código solo si respeta la regla que se le asigne, se supuso que el problema radicaba ahí. Los sospechosos principales fueron “suma += elemento” y “i += 1”. Se pensó que o estaban mal definidos o estaban fuera de lugar. Se intento borrarlos y ejecutar el código para ver que sucedía (el código no corría). Se intento cambiar de lugar a diferentes partes del código, fue hasta que se cambió de lugar el “i += 1” que se descubrió que ese era el causante del problema debido a que, al estar fuera de lugar, i no se incrementaba de valor, lo que causaba un bucle infinito.

EDUARDO RUIZ RAMIREZ – BASES DE DATOS – 22/02/2024

EJERCICIO 2

Original

```python
# Regresa el valor que está a la mitad de la lista de números
# Si el número de elementos es par, regresar la suma
# del dos números a la mitad
# Si algún elemento no es un número, se regresa None
def regregar_valor_mitad(lista):
    mitad = len(lista)/2
    if len(lista) % 2 == 0:
        return lista[mitad]
    else:
        suma = lista[mitad] + lista[mitad + 1]
        return suma

# Pruebas, deben pasar todas
print(regregar_valor_mitad([1, 2, 3]) == 2)
print(regregar_valor_mitad([5]) == 5)
print(regregar_valor_mitad([]) == None)
print(regresar_valor_mitad([1, 2]) == 3)
print(regregar_valor_mitad([1, 2, 3, 4, 5, 6]) == 7)
print(regregar_valor_mitad([1, 'hola', 3]) == None)
print(regregar_valor_mitad([1, 2, 3, 4, 5, 'hola']) == None)
```

Corregido

```python
def regregar_valor_mitad(lista):
    mitad = len(lista) // 2              #debe ser //, porque si es / da float y no pide float
    if len(lista) % 2 == 0:
        return lista[mitad - 1] + lista[mitad]   #se modifica la operacion para que sea resta y se cambia de lugar
    else:
        return lista[mitad]       #se cambia de lugar el return lista y el return lista con las operaciones

numeros = int(input('escribe la cantidad de numeros a introducir'))
numeroslista = []

try:
    numeros = int(numeros)
    if numeros == 0:
        print('None')
        exit()

    for i in range(numeros):
        numero = input()
        numeroslista.append(int(numero))

    if not numeroslista:
        print("None")
    else:
        resultado = regregar_valor_mitad(numeroslista)
        print(resultado)

except ValueError:
    print("None")

print(regregar_valor_mitad([1, 2, 3]) == 2)
print(regregar_valor_mitad([5]) == 5)
print(regregar_valor_mitad([]) == None)
print(regregar_valor_mitad([1, 2]) == 3)  #se corrige regregar en vez de regresar
print(regregar_valor_mitad([1, 2, 3, 4, 5, 6]) == 7)
print(regregar_valor_mitad([1, 'hola', 3]) == None)
print(regregar_valor_mitad([1, 2, 3, 4, 5, 'hola']) == None)
```

El código proporcionado originalmente se trataba solo de una función, por lo que por si solo el código no iba a hacer absolutamente nada al momento de ejecutarse, pues necesita de otro código que lo llame.
Este código procesa una lista de números (que el usuario introduce) y devuelve como valor el numero que esta en medio (o la suma de los dos números de en medio dependiendo del número de elementos se introdujeron) de la lista. Asimismo, se agregó aparte el manejo de errores en caso de introducirse un valor no numérico, así regresa un mensaje de error.
El primer error notado casi al instante esta en las pruebas. Una de ellas dice “regresar” en vez de “regregar” por lo que no reconoce la función y causa un error.
El otro error notado con facilidad es en “len(lista)/2”, pues el resultado de esta operación devuelve un numero decimal, un tipo de numero que no es valido en una lista, se corrigió usando “//” lo cual también es una división pero al devolver el resultado ignora los decimales y pone el numero como entero.
El error principal radica en la parte donde se suman las 2 “lista[mitad]” Este fue un poco mas tardado en detectar, pues hasta no crearse código donde se llamara la función, no se logro dar cuenta de que la lógica estaba mal, pues causaba error ya que se sumaba demasiado y el resultado excedía los límites de la lista. Los principales sospechosos fueron los cálculos que se realizaban dentro de la función, por lo que se probó cambiándolos y así se solucionó el problema. Para esto, se cambió la operación dentro de los corchetes a una resta. Así, en caso de tenerse una lista par, se sumaria sin problemas, y si fuese impar, simplemente se mostraría la mitad y no se realiza ningún cambio.