Documentación Técnica de Ejercicios

Fecha	Versión	Descripción	Autor
18/08/2023	1.0	Documentación técnica de 4 ejercicios de programación.	Sergio Calle Misas

Índice

Contenido

Índice....1

Requisitos....2

Ejercicios....2

Ejercicio 1: Clasificación de Números....2

Requerimientos:....2

Ejemplo:....2

Código:....2

Ejercicio 2: Serie Tribonacci....3

Requerimientos:....3

Ejemplo:....3

Código:....3

Ejercicio 3: Transformación de lista....4

Requerimientos:....4

Código:....4

Ejercicio 4: Manipulación de listas....5

Requerimientos:....5

Código:....5

Requisitos

Asegúrate de tener Python instalado en tu sistema para poder ejecutar los scripts. Además, puedes utilizar un entorno de desarrollo como Visual Studio Code para una experiencia de codificación más cómoda.

Ejercicios

Ejercicio 1: Clasificación de Números

Requerimientos:

Dada una matriz de enteros, mantenga una puntuación total basada en lo siguiente:

1. Agregue 1 punto por cada número par en la matriz.
2. Agregue 3 puntos por cada número impar en la matriz, excepto por el número 5.
3. Sume 5 puntos cada vez que aparezca el número 5 en la matriz.

Tenga en cuenta que el 0 se considera par.

Ejemplo:

Input: [1,2,3,4,5] Output: 13

Input: [17,19,21] Output:9

Código:

```
# Definimos dos listas de números
listaNum = [1, 2, 3, 4, 5]
listaNum2 = [17, 19, 21]
total = 0  # Inicializamos la variable 'total' en 0 para llevar la cuenta de los
puntos

# Recorremos cada valor en 'listaNum2'
for value in listaNum2:
    if value % 2 == 0:  # Si el valor es par (su residuo al dividirlo por 2 es 0)
        print(f'El numero par ({value} da 1 punto)')  # Mostramos un mensaje
        total = total + 1  # Incrementamos 'total' en 1 punto
    elif value % 2 != 0 and value != 5:  # Si el valor es impar y no es 5
        print(f'El numero impar ({value} da 3 puntos)')  # Mostramos un mensaje
        total = total + 3  # Incrementamos 'total' en 3 puntos
    elif value == 5:  # Si el valor es 5
        print(f'El numero 5 da 5 puntos')  # Mostramos un mensaje
        total = total + 5  # Incrementamos 'total' en 5 puntos

print(f'El total es: {total}')  # Mostramos el total acumulado de puntos
```

Ejercicio 2: Serie Tribonacci

Requerimientos:

Crear una función que implemente la secuencia Tribonacci, esta es una variación de la secuencia de Fibonacci, en la que cada número siguiente se encuentra sumando los dos números anteriores. La secuencia Tribonacci es similar, pero en ligar de comenzar con los dos números predeterminados, la secuencia comienza con tres números predeterminados y cada número subsiguiente es la suma de los tres números anteriores. Lo más importante de esta variación es que los 3 números ingresados inicialmente deben ser validados bajo la Fibonacci original.

Ejemplo:

Input: [1,1,2] Output: [1,1,2,4,7,13]

Input: [3,5,8] output: [3,5,8,16,29,53]

Input: [1,2,3] Output: “La secuencia inicial no corresponde a una secuencia Fibonacci valida" - Revisar esta prueba

Código:

```
# Definimos una función llamada 'validarFibonacci' para verificar si una
secuencia es válida según la regla de Fibonacci
def validarFibonacci(dataArray):
    if dataArray[0] == dataArray[1] and dataArray[1] != 1:  # Si los
primeros dos elementos son iguales y el segundo no es 1
        return False  # No es una secuencia válida de Fibonacci
    if dataArray[0] + dataArray[1] == dataArray[2]:  # Si la suma de los
primeros dos elementos es igual al tercer elemento
        return True  # Es una secuencia válida de Fibonacci
    else:
        return False  # No es una secuencia válida de Fibonacci

# Definimos una función llamada 'tribonacci' para generar una secuencia
Tribonacci
def tribonacci(valor):
    tribon = []  # Creamos una lista vacía llamada 'tribon'
    if validarFibonacci(valor):  # Si la secuencia inicial es válida
según la regla de Fibonacci
        a, b, c = valor[0], valor[1], valor[2]  # Asignamos los primeros
tres valores a 'a', 'b' y 'c'
        for i in range(6):  # Generamos los siguientes 6 valores en la
secuencia Tribonacci
            tribon.append(a)  # Agregamos 'a' a la lista 'tribon'
```

```
            a, b, c = b, c, a + b + c  # Calculamos los nuevos valores de
'a', 'b' y 'c'
        return tribon  # Devolvemos la secuencia Tribonacci
    else:
        return 'La secuencia inicial no corresponde a una secuencia
Fibonacci válida'

# Llamamos a la función 'tribonacci' con diferentes secuencias iniciales
y mostramos los resultados
#print(f'Tribonacci: {tribonacci([1, 1, 2])}')  # Resultado: [1, 1, 2, 4,
7, 13]
print(f'Tribonacci: {tribonacci([1, 2, 3])}')  # Resultado: [1, 2, 3, 6,
11, 20]
#print(f'Tribonacci: {tribonacci([3, 5, 8])}')  # Resultado: [3, 5, 8,
16, 29, 53]
```

Ejercicio 3: Transformación de lista

Requerimientos:

A partir del siguiente arreglo [2,3,1,1,3,2,1,4] se debe crear una función que reciba y retorne lo siguiente: Input: { tipo: 1/1,2,3 números:[2,3,1,1]} Output según el tipo: 1. Un nuevo arreglo con los valores únicos previamente generados 2.Un nuevo arreglo con solo los valores 1 y 3 3.Un nuevo arreglo ordenado de forma ascendente incluyendo los nuevos números ingresados.

Código:

```
# Definimos un diccionario llamado 'dato' con dos campos: 'tipo' y
'numeros'
dato = {
    "tipo": 1,             # El tipo de transformación que se aplicará a
la lista
    "numeros": [2, 3, 1, 1]  # Lista de números que se utilizarán en la
transformación
}

# Definimos una función llamada 'transformarLista' que toma un
diccionario 'dato' como argumento
def transformarLista(dato):
    listaPredeterminada = [2, 3, 1, 1, 3, 2, 1, 4]  # Una lista
predeterminada
    tipo = dato["tipo"]  # Extraemos el valor del campo 'tipo' del
diccionario 'dato'
    lista = listaPredeterminada + dato["numeros"]  # Concatenamos
'listaPredeterminada' y los números de 'numeros'
    resultado = []  # Creamos una lista vacía llamada 'resultado'

    # Comprobamos el tipo de transformación
    if tipo == 1:
        # Si el tipo es 1, recorremos cada valor en 'lista'
```

```
        for valor in lista:
            if valor not in resultado:  # Si el valor no está en
'resultado'
                resultado.append(valor)  # Agregamos el valor a
'resultado'
    elif tipo == 2:
        # Si el tipo es 2, recorremos cada valor en 'lista'
        for valor in lista:
            if valor == 1 or valor == 3:  # Si el valor es 1 o 3
                resultado.append(valor)  # Agregamos el valor a
'resultado'
    elif tipo == 3:
        # Si el tipo es 3, ordenamos la lista y asignamos a 'resultado'
        resultado = sorted(lista)

    return resultado  # Devolvemos la lista 'resultado' transformada

# Llamamos a la función 'transformarLista' con el diccionario 'dato' y
mostramos el resultado
print(f'Salida: {transformarLista(dato)}')
```

Ejercicio 4: Manipulación de listas

Requerimientos:

Crear dos vectores de un máximo de 15 elementos, los cuales deben ser alimentados con información introducida por el teclado por el usuario y cumplir con los requisitos propuestos.

a. El tamaño del vector 1 es de máximo 15 elementos

b. El tamaño del vector 2 es de máximo 15 elementos

c. El usuario del software (Lenguaje elegido) mediante una constante N debe indicar cuantos elementos quiere leer de los 15 posibles

d. Los datos ingresados por teclado deben ser números enteros positivos entre 1 y 30

e. Los vectores deben ser llenados en cualquier orden (ASC o DESC)

f. Después de ser leídos los valores en ambos vectores, estos deben ser mostrados de forma ordenada ascendentemente (Menor a mayor)

g. Mostrar una lista con la suma de ambos vectores según su posición.

Código:

```
# Definimos una función llamada 'leerNumero' para solicitar un número
entre 1 y 30
def leerNumero():
```

```
    while True:
        try:
            numero = int(input("Ingrese un número entre 1 y 30: "))  # 
Solicitamos un número al usuario
            if 1 <= numero <= 30:  # Verificamos si el número está en el 
rango 1-30
                return numero  # Devolvemos el número válido
            else:
                print("El número debe estar entre 1 y 30.")
        except ValueError:
            print("Entrada inválida. Intente nuevamente.")  # Manejamos 
errores de entrada inválida

# Solicitamos al usuario cuántos datos desea ingresar (con un máximo de 
15)
n = int(input("Ingrese cuántos datos desea leer (máximo 15): "))
if n > 15:
    n = 15

lista1 = []  # Creamos una lista vacía llamada 'lista1'
lista2 = []  # Creamos una lista vacía llamada 'lista2'

print("\nIngrese los datos para la lista1:")
# Solicitamos al usuario ingresar 'n' números y los agregamos a 'lista1'
for _ in range(n):
    numero = leerNumero()  # Solicitamos un número al usuario utilizando 
la función 'leerNumero'
    lista1.append(numero)  # Agregamos el número a 'lista1'

print("\nIngrese los datos para la lista2:")
# Solicitamos al usuario ingresar 'n' números y los agregamos a 'lista2'
for _ in range(n):
    numero = leerNumero()  # Solicitamos un número al usuario utilizando 
la función 'leerNumero'
    lista2.append(numero)  # Agregamos el número a 'lista2'

lista1.sort()  # Ordenamos 'lista1'
lista2.sort()  # Ordenamos 'lista2'

print("\nLista1 ordenada:", lista1)  # Mostramos 'lista1' ordenada
print("Lista2 ordenada:", lista2)  # Mostramos 'lista2' ordenada

sumaListas = []  # Creamos una lista vacía llamada 'sumaListas'
# Calculamos la suma de elementos correspondientes en 'lista1' y 'lista2'
for i in range(n):
    sumaListas.append(lista1[i] + lista2[i])  # Agregamos la suma a 
'sumaListas'

print("\nSuma de listas:", sumaListas)  # Mostramos la suma de listas
```